SYMCOMP 2013
Lisbon, 9-10 September 2013

TWO-VARIABLE LINEAR PROGRAMMING: A GRAPHICAL TOOL WITH *MATHEMATICA*

José C. Pereira[1] and Susana Fernandes[2]

1: Departamento de Engenharia Electrónica e Informática
Faculdade de Ciências e Tecnologia
Universidade do Algarve
Campus de Gambelas, 8005-139 Faro, Portugal
e-mail: unidadeimaginaria@gmail.com

2: Departamento de Matemática
Faculdade de Ciências e Tecnologia
Universidade do Algarve
Campus de Gambelas, 8005-139 Faro, Portugal
e-mail: sfer@ualg.pt

Keywords: linear programming for two variables, graphical visualization, dynamic, interactive, teaching tool, Wolfram Mathematica, Computable Document Format

Abstract. *This paper presents the* GLP-Tool*, an interactive tool for graphical linear programming involving two variables. The* GLP-Tool *is designed to solve user-defined linear programming problems with two variables, up to a didactical limit of five constraints (plus the signal constraints). Implemented using the computer algebra system* Mathematica*, this interactive tool allows the user to dynamically explore different objective functions and constraint sets, and also perform post-optimal and sensitivity analysis. All the* GLP-Tool *functionalities are represented graphically and updated in real time. These interactive, dynamic, and graphical features make the* GLP-Tool *a powerful tool for teaching linear programming both in undergraduate and high school courses. After completing its development, we intend to make the* GLP-Tool *available at the Wolfram Demonstrations Project website.*

1 INTRODUCTION

When introducing the subject of Linear Programming (LP) it is rather useful to present the graphical method for solving a two-variable linear program as it provides valuable insights about the general nature of multivariable linear programming models. The graphical method illustrates numerous aspects of the more complex, algebra-based, solution algorithm - the simplex method. Specifically, graphically solving a linear program provides students with intuitive visual aids to facilitate their understanding of concepts such as the feasible region, basic feasible solutions, unbounded solutions, binding /nonbinding constraints, degeneracy, slack, and so on.

Nonetheless, without a dynamic tool, it is not easy to show students what happens in a (two-variable) LP problem as constraint boundary lines and objective-value lines move around on a graphic. To provide students with an effective learning environment a tool should show graphically and dynamically the construction of the feasible region of two-variable linear programs, and should allow to interactively experiment with the feasible solutions set and the objective function. This kind of tool is what is called an active learning tool.

Active learning is generally defined as any instructional method that engages students in the learning process [1]. The core elements of active learning are student activity and engagement in the learning process within the classroom. In short, active learning requires students to do meaningful learning activities and think about what they are doing. Prince [2] summarizes some of the most relevant literature in the field of active learning, which report strong evidences of the effectiveness of using active learning techniques. In particular, the work of Kydd [3] reports the effectiveness of using an active learning technical tool while teaching linear programming.

This paper presents the *GLP-Tool*, a good example of an active learning technical tool that engages students and provides them with an effective learning environment. Implemented using the computer algebra system *Mathematica*, this interactive tool allows the user to dynamically explore and solve two-variable linear programming problems with different objective functions and constraint sets, and also to perform post-optimal and sensitivity analysis. All the *GLP-Tool* functionalities are represented analytically and graphically and updated in real time.

The interactive, dynamic, analytical, and graphical features of the *GLP-Tool* make this application a powerful tool for teaching LP both in undergraduate and high school courses. It can be used both by teachers to graphically illustrate fundamental concepts and by students to experiment with changes within a graphical representation of a linear program, facilitating their understanding of numerous LP concepts.

When we set up to produce an innovative tool for graphical linear programming, the *GLP-Tool*, we chose to implement it using the computer algebra system *Mathematica* not only because it enables the production of sophisticated, dynamic, interactive visual applications but also due to the previous experience of one of the authors in producing

didactical tools with *Mathematica* [4].
We intend to make the *GLP-Tool* available online to the general public, via the Wolfram Demonstrations Project website. The corresponding source code will also be available at this website.
We believe that this new *GLP-Tool* is an important contribution to the improvement of the teaching-and-learning process of LP precisely by providing teachers and students alike with an active learning technical tool to explore fundamental concepts of the subject.

The rest of this paper is organized as follows. In Section 2 we present an overview of existing tools for graphical linear programming and also present the motivations for using the computer algebra system *Mathematica* to produce a new innovative tool. In Section 3 we describe the GLP-Tool concept and explain how to use its main features. In Section 4 we provide several usage examples of how to introduce LP concepts with the didactical GLP-Tool in a classroom environment. In Section 5 we make some final remarks about our current and future work related with the GLP-Tool concept.

2 Web-based tools for graphical linear programming

There are several Java applets freely available online which graphically solve linear programs. Some of them graphically illustrate the steps of the algebra-based simplex method, moving from one basic solution to the next until it reaches the optimal solution(s). Examples of these Java applets are the 'Linear Programming and Pivoting in 2D' [5], the 'LP Explorer 1.0' [6] or the 'Graphical Simplex Algorithm (2D)' [7]. While being very useful when trying to visually explain/understand the simplex method we believe they are not suited for introducing the subject of linear programming since they do not show very important features in understanding LP as, for instance, the mapping of the objective function value within the feasible solutions set.
Other Java applets exist that do allow the user to manipulate the constraints and/or the objective function in order to see the effect on the feasible region and on the optimal solution(s). Examples of theses applets are the 'Exploring linear programming' [8], the 'Linear programming applet' [9] or the 'Animated linear programming applet' [10]. Typically these applets allow the user to define a linear program with two variables with a total number of constraints up to 4 or 5.
All Java applets freely available online that graphically solve linear programs are not visually sophisticated, when compared to a commercial application. Wolfram's *Mathematica* is a powerful computer algebra system that enables the production of sophisticated, dynamic, interactive visual applications. Wolfram's *Mathematica* is used in scientific, engineering, and mathematical fields and in other areas of technical computing. Creating interactive visual models with *Mathematica* allows users to explore hard-to-understand concepts, test theories, and quickly gain a deeper understanding of the materials being introduced firsthand. The users can explore changes to text, functions, formulas, matrices, graphics, tables, or even data. In what concerns the work presented in this paper,

not only *Mathematica* algebraically solves LP programs using a single command as also its graphics are completely integrated into its dynamic interactive language. Any visualization can immediately be animated or made interactive using a single command and developed into sophisticated, dynamic visual applications.
We have found some *Mathematica* demonstrations on the website of the Wolfram Demonstrations Project which address the subject of linear programming, but none of them works with an user-defined linear program. An intentionally very simple application is the 'Graph of Inequalities' [11] which shows that the graph of a linear equation is a straight line, the graph of a linear inequality is an half-plane and that the graph of a system of linear inequalities is the intersection of the half-planes. There's a Wolfram *Mathematica* demonstration that graphically illustrates the steps of the (two-phase) simplex method on random generated linear programs; the 'Two-Phase Simplex Method' [12]. While being very useful when trying to visually explain/understand the simplex method we believe it is not suited for introducing the subject of linear programming.
Some *Mathematica* demonstrations focus on illustrating the weak version of the fundamental theorem of linear programming which states that the optimal solution to a linear program, if it exists, is attained at (at least) one vertex of the feasible solutions set. Examples of these are the 'Parametric Linear Programming' [13], the 'The Fundamental Theorem of Linear Programming' [14] or the 'Graphical Linear Programming for Two Variables' [15]. While being very useful for illustrating the weak version of the fundamental theorem of linear programming, they do not enable the visualization of very important concepts of LP. Other *Mathematica notebooks* exist that do allow the user to manipulate the constraints (and the objective function) in order to see the changes on the feasible region and on the optimal solution(s). Examples of these are the 'Oil Mallee Farming Optimization Problem' [16] or the 'A Simple, Standard Linear Programming Scenario' [17].
One word about the internet site WolframAlpha. WolframAlpha is a denominated computational knowledge engine which can be used to solve a user-defined linear program. When entering a linear program as - `Max` 'objective function' in 'inequality 1' and 'inequality 2' and .... - the engine outputs the optimal solutions(s), the corresponding optimal value and it also graphs the feasible solutions set, highlighting the feasible solution(s) where the optimal value is reached. WolframAlpha is a powerful engine but it is not an interactive didactical tool. For instance, it does not allow the user to drag the objective function line and see the way its value changes within the feasible region.
When looking for a graphical tool for introducing linear programming we found internet freely available tools to be either unfit for this purpose ([5], [6], [7], [12]), misdirected ([13], [14], [15]) or incomplete ([8], [11], [16], [17]). Only two of them ([9], [10]) were flexible enough to enable the visualization of linear programs that either: have an unbounded feasible region; have redundant constraints; have degenerated solutions; are unbounded; are unfeasible. Unfortunately we encountered some malfunctions of the 'Linear programming applet' [9] when defining totally different instances (namely the ones we will present

further on this article). The'Animated linear programming applet' [10] is the only application that actually computes the solution of the problem, but unfortunately it uses a numerical system with finite precision which in some cases leads to errors in the output. Another drawback is the fact that this application has a very poor graphical presentation. Anyone can work online with all these applications, Java applets, Wolfram's *Mathematica* demonstrations or WolframAlpha. Nowadays the internet is globally reachable, nonetheless, in some situations (some portuguese schools, for instance) an application running locally on a personal computer is necessary. Wolfram's *Mathematica* demonstrations have the advantage of being built into a single CDF file which runs as a stand-alone application. Anyone can download it, and just 'click' his way through the installation of the CDF player. For downloading a Java applet the user has to be able to read html or Java script code, and also to be able to find all files needed to build the application.
With our new tool for graphical linear programming, the *GLP-Tool*, we aim at bringing together the best of two worlds: the flexibility of Java applets and the sophistication and easy packaging of Wolfram's *Mathematica* demonstrations.

3 THE *GLP-Tool* CONCEPT

The GLP-Tool is a dynamic, interactive, and visual tool, implemented with *Mathematica*, that allows to solve user-defined linear programming problems with two variables. In particular, the user can explore different objective functions and constraint sets, obtain graphical and numerical information on optimal solutions, and perform post-optimal and sensitivity analysis.
The GLP-Tool explores the class of linear programming problems that can be written in the general form

$$\begin{array}{lll} Max & z = ax + by & \\ s.t. & a_i x + b_i y \leq t_i & , i = 1, \ldots, 5 \\ & x, y \geq 0 & \end{array} \tag{1}$$

where the non-negativity constraints are optional.
The number of proper constraints is set to a didactical limit of five constraints, plus the non-negativity constraints. This limitation was based on the fact that the GLP-Tool is not a mere calculator but rather an active learning tool designed to engage students and provide them with an effective Linear Programming learning environment. With that in mind, we found that the five constraints limits was more than enough to illustrate all the main LP concepts, without cluttering the visual interface of the GLP-Tool. In addition, we could not find in the main literature any instances of graphical linear programming problems that would exceed the five constraints limit.
All the functionalities of the GLP-Tool, along with all the displayed information, are designed to conform with the way Linear Programming is used in research and presented

in the classroom, both at the high school and undergraduate levels. All the GLP-Tool features are represented graphically and updated in real time.
Figure 1 presents a usage example of the GLP-Tool. Visually, this active learning tool is divided in two main panels.

Left Panel: This panel contains all the controls that allow the user to define the linear programming problem instances. In addition, it includes some options related with the visual display of the feasible region and the objective function. The user can also choose to visualize the formalization of the problem instance. More importantly, the feature *Panel Mode*, offers three different ways to interact with the right panel, `Explore`, `Max`, and `Min`.

Right Panel: In this panel it is presented all the graphical information concerning the linear programming problem instance. The feasible region is depicted along with the constraint boundary lines. The objective function line is graphed for a fixed value of z. The feasible solutions corresponding to this value of z are also presented, when they exist. In addition, the problem formalization and the optimal solution information are displayed in two insets, when the corresponding options are chosen.

The GLP-Tool has a very intuitive interface that allows even the most inexperienced user, with no previous knowledge in educational software, to use all the GLP-Tool features in an efficient and autonomous way, right from the start.

3.1 How to use the GLP-Tool

As mentioned before, all the graphical and analytical information presented in the left and right panels is displayed in real time, thanks to the symbolic and numerical computation capabilities of *Mathematica*. This renders the GLP-Tool as an eminently dynamic tool designed for active learning.
The user can set the values of all the coefficients of a linear problem using the respective sliders. The values can be changed one by one in a discrete or continuous way. In particular, this kind of control allows the automatic continuous change of the coefficient values by clicking the play button (▸) depicted, for example, bellow the a-slider in Figure 1. When choosing this option, the user will immediately see an animation of the corresponding graphical information in the right panel. For instance, one may rotate or translate one of the constraint boundary lines and observe how the feasible region changes accordingly. In addition, the corresponding problem formalization can be seen continuously changing, when the respective light green inset is displayed in the right panel (see Figure 1). Note that, this inset is not a static display but rather a dynamic object in which all the contained information is updated in real time.
The problem formalization can also be seen in the graphic itself. When the mouse pointer is over each constraint boundary line a small yellow tag is displayed, identifying the line

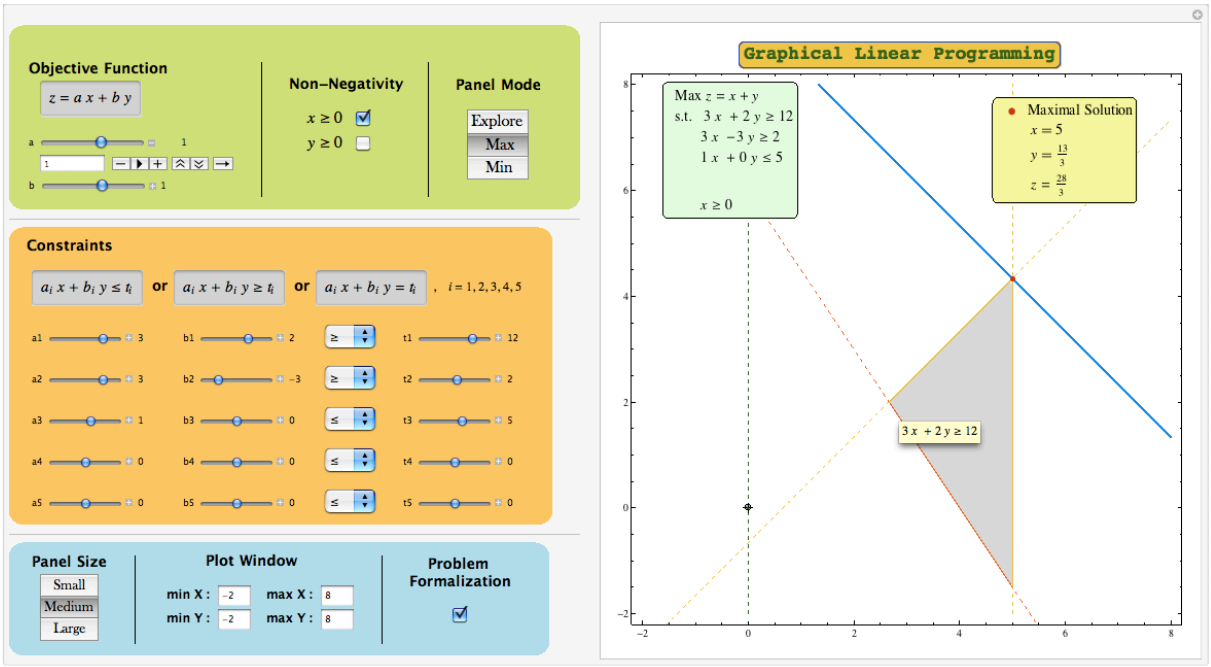


Figure 1: A general usage example of the GLP-Tool. The chosen panel mode is `Max`.

with the corresponding constraint (see Figure 1). The same is true for the objective function line when graphed for the optimal z.

The *Panel Mode* is one of the main features of the GLP-Tool. This functionality includes two optimize modes `Max` and `Min` that allow the user to obtain the optimal solution(s) of the respective optimization problems. The optimal information is then displayed both graphically and numerically in the right panel (see Figure 1). Note that, these modes are able to compute all kinds of optimal sets, wether they may be empty, with a single element or with multiple solutions, both bounded or unbounded.

A good usage example of the GPL-Tool is to choose one of the two optimize modes, with a fixed feasible region and click the play button for the objective function coefficients a and b. The corresponding line will start rotating and will eventually "visit" several vertices of the feasible region. The user can then observe the optimal solution information change accordingly. This visualization is, for instance, a vey good way to introduce the concept of basic feasible solutions. In addition it also promotes a better understanding of the role of the objective function coefficients in post-optimal and sensitivity analysis.

The *Panel Mode* includes a third mode, `Explore`. As its name indicates, this option allows the exploration of the set of feasible solutions for any given set of constraints. While in this mode, the user can *click-and-drag* the objective function line across the visible region of the graphic in the right panel, with the mouse pointer. When this line intersects the feasible region, the feasible solutions will be highlighted in red (see Figure 2). At the same

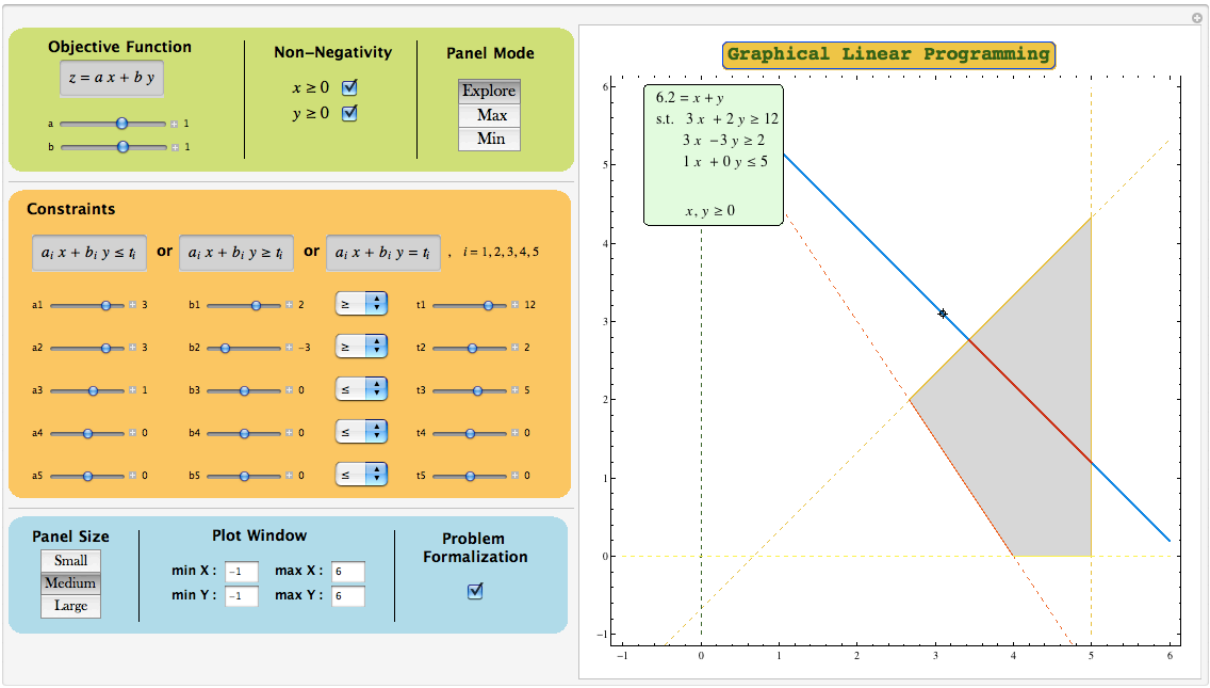


Figure 2: Using the GLP-Tool to explore the set of feasible solutions. The chosen panel mode is `Explore`.

time, the corresponding value of z is updated within the problem formalization display. Another useful feature of the GLP-Tool is the ability to resize the *Plot Window* (see Figures 1 and 2). This functionality is crucial to the visualization of any problem instance, no matter how odd are the coefficient values. Up to our knowledge there is no other graphical linear programming application with such feature, which seriously reduces their working scope.

We note that the usage descriptions contained in this section are far from being comprehensive. All the GLP-Tool features can be combined to produce a large variety of different useful interactions. As the user becomes more acquainted with this tool, he/she will naturally find those interactions that fit best his/her purpose.

It is also through this kind of dynamic interaction that the GLP-Tool becomes a fully active learning technical tool that engages students and teachers and provides them with an effective teaching/learning environment.

4 TEACHING GRAPHICAL LINEAR PROGRAMMING USING THE *GLP-Tool*

In this section we present a didactical use of the *GLP-Tool*.

We assume that the formulation of linear programs was already discussed and now we

want to introduce the graphical method for solving a two-variable linear program. Consider the following LP problem instance:

$$\begin{array}{ll} min & Z = x + y \\ s.t. & 3x + 2y \geqslant 12 \\ & 3x - 3y \geqslant 2 \\ & x \leqslant 5 \\ & x, y \geqslant 0 \end{array} \tag{2}$$

The GLP-Tool starts by default with a feasible region defined only by the non-negativity constraints. The tool shows the construction of the feasible region as new constraints are added.

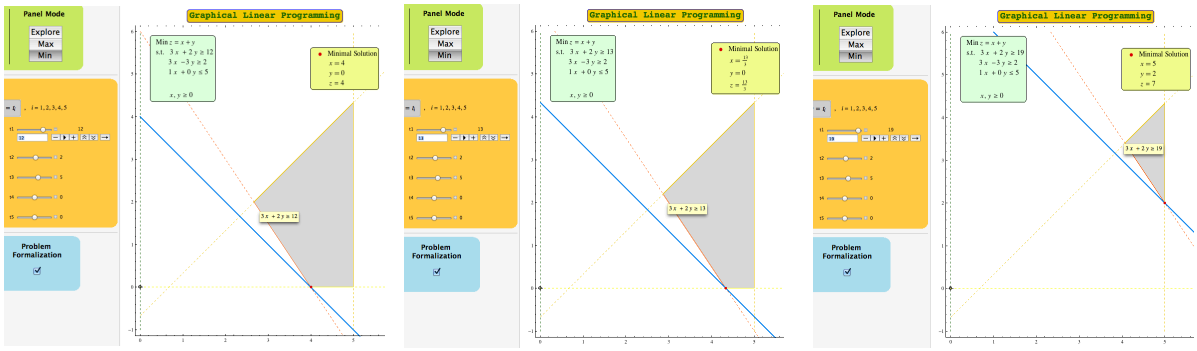
Working with instance (2), after introducing all the constraints the teacher should point out that the constraint $x \geqslant 0$ is actually redundant (the green dashed vertical line does not bound the feasible region in Figure 2). The students can easily verify this fact by checking and unchecking the corresponding non-negativity option in the left panel and observe that the feasible region in the right panel remains unchanged.

Next, it is necessary to set the values of the coefficients of the objective function. Afterwards, the students can use the `Explore` mode to observe the change in the value of z as the objective function line is dragged across the right panel (see Figure 2). The teacher can question which feasible solution(s) achieve(s) the optimal value of the problem. By using the `Min` and `Max` modes it is possible to actually compute the optimal solutions $(4, 0)$ and $(5, \frac{13}{3})$ for the minimum value $z = 4$ and the maximum value $z = \frac{28}{3}$, respectively. This information is displayed in the light yellow inset in the right panel while the optimal solution is highlighted in the graphic (see Figure 1)

While in the `Max` mode, we can also introduce the concept of binding and nonbinding constraints. The students can observe that any change in the independent terms of the inequalities $3x - 3y \geqslant 2$ and $x \leqslant 5$ lead to a different optimal solution. Therefore, these inequalities are binding ones. As for the other constraints, some variation of the independent terms is allowed before the optimal solution changes (how much variation?) and so, they are considered nonbinding. In particular, the case of the $x \geqslant 0$ constraint shows that all redundant constraints are also nonbinding.

The teacher should have shown up to this point how it is possible to change the coefficients values either manually, by dragging the slider or by direct input, or automatically, by clicking the play button.

Returning to the initial problem instance (2), it is possible to introduce post-optimal analysis by questioning the students if the optimal solution would remain the same when the constraint $x \leqslant 5$ changes to $x \leqslant 6$. And what about when the constraint $3x + 2y \geqslant 12$ changes to $3x + 2y \geqslant 13$? After observing the effects of such changes, the teacher should encourage the students to try and modify other independent terms, one at a time. Figure 3 presents one possible example of this kind of interaction with the GLP-Tool.



(a) LP instance with $t_1 = 12$. (b) LP instance with $t_1 = 13$. (c) LP instance with $t_1 = 19$.

Figure 3: Three similar LP problem instances only with different t_1 values inputed directly. This kind of interaction with the GLP-Tool is very useful to introduce many concepts of post-optimal and sensitivity analysis.

Similarly, sensitivity analysis can be introduced by questioning the students within which interval may the independent term of the inequality $3x + 2y \geqslant 12$ vary without changing the optimal solution. This concept can then be generalized to include all other coefficients. As before, the students should modify the values of several coefficients, of binding and nonbinding constraints, and try to determine within which limits does the optimal solution remains unchanged.
Notice that, when changing the coefficients/independent terms of the inequalities some times we will get a feasible region with only one solution while other times we will get an empty feasible region. In this case, an inset is displayed in the right panel with the information that the problem is unfeasible, as depicted in Figure 4.

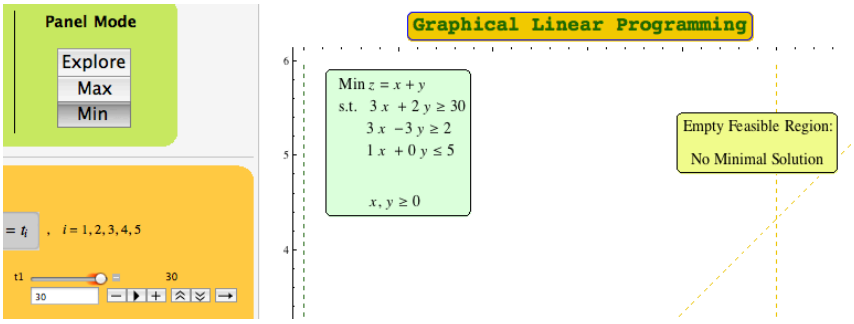
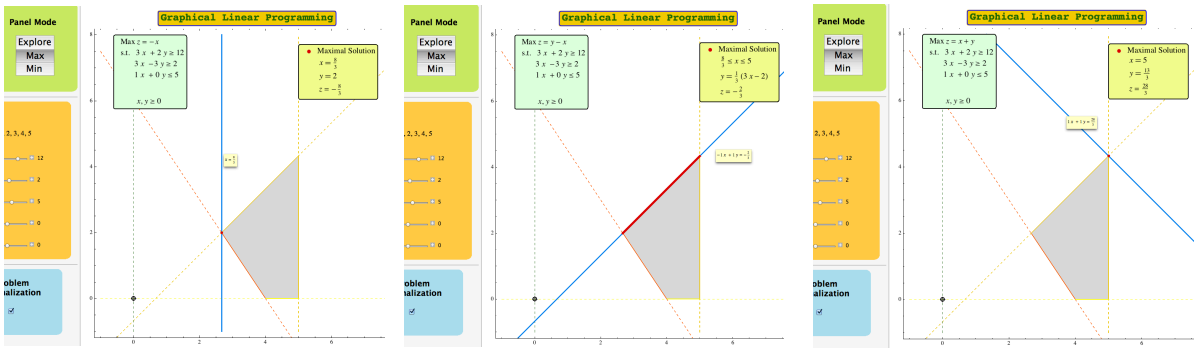


Figure 4: Example of an unfeasible LP problem. In this case, the GLP-Tool simply displays that information in an inset in the right panel.

Considering again the initial problem instance (2), the students can now perform the sensitivity analysis of the coefficients of the objective function. An interesting usage



(a) LP instance with one maximal solution.

(b) LP instance with multiple maximal solutions.

(c) LP instance with another single maximal solution.

Figure 5: Three similar LP problem instances only with different objective function coefficients. As the values of coefficients a and b change, so does the optimal solution. This example illustrates the weak version of the fundamental theorem of LP. It can also be used to show why and when do multiple optimal solutions occur.

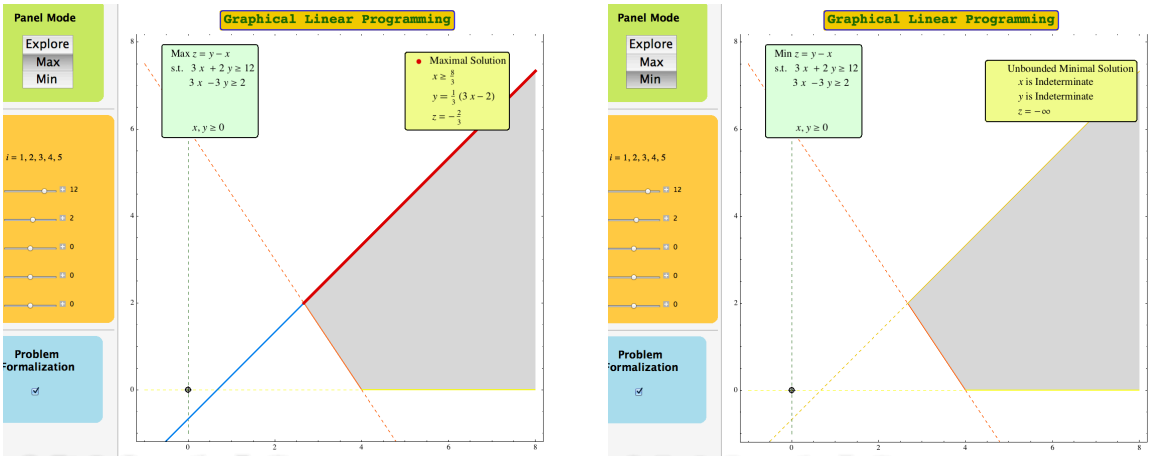
example in this analysis is to modify the value of the coefficients a and b using the corresponding play button, while on the `Max` mode. As the values of the coefficients change, the objective function line can be seen rotating in the right panel, moving from one vertex to another adjacent vertex of the feasible region. Also, as depicted in Figures 5(a), 5(b), and 5(c), the information concerning the current maximal solution is displayed and updated in the corresponding inset. The teacher should note that this example clearly illustrates the weak version of the fundamental theorem of LP, which states that the optimal solution to a linear program, if it exists, is attained at at least one vertex of the feasible region.

In addition, the same example can also be used to introduce the concept of multiple optimal solutions. After observing the motion of the objective function line and looking at Figures 5(a), 5(b), and 5(c), the students should be asked questions such as When do multiple optimal solutions occur? How many are there? What do they have in common? What makes the values of coefficients a and b in Figure 5(b) special?

Let us now consider the following LP problem instance

$$\begin{array}{ll} max & Z = -x + y \\ s.t. & 3x + 2y \geqslant 12 \\ & 3x - 3y \geqslant 2 \\ & x, y \geqslant 0 \end{array} \tag{3}$$

As depicted in Figure 6, LP problem (3) has an unbounded feasible region. However, the teacher should note that the maximization problem is in fact bounded but, in turn, the set of maximal solutions is unlimited and corresponds to a ray of the boundary of the feasible region (see Figure 6(a)). On the other hand, using the `Explore` mode the



(a) LP instance with ilimited multiple maximal solutions.

(b) Unbounded minimization LP problem instance.

Figure 6: One LP problem instance with an unbounded feasible region. Nonetheless, the maximization problem is bounded and the set of maximal solutions corresponds to a ray of the boundary of the feasible region. The minimization problem is unbounded.

students can verify that there is no lower limit for the optimal value. Shifting to the `Min` mode they can then concluded that indeed, the minimization problem is unbounded and no minimal solution can be attained (see Figure 6(b)).
When introduced for the first time to the notion of unbounded feasible solutions sets, students tend to believe that this always implies that the LP problem itself is unbounded as in Figure 6(b) or that, at the very least, the set of optimal solutions must be unlimited (or maybe just with infinite multiple elements?) as in Figure 6(a). In any case, to help clarify the relations between these various notions, it is useful to consider the objective function $min\ z = x + y$ for the LP instance (3). In this case, as depicted in Figure 7, the LP problem has one single optimal solution $(x, y) = (4, 0)$, corresponding to the minimal value $z = 4$. The teacher should promote a discussion about the relations between this case and the two previous cases, depicted in Figure 6.

5 FINAL REMARKS

This paper presents an innovative active learning tool for introducing graphical linear programming, the GLP-Tool. This dynamic, interactive, visual tool was implemented using the computer algebra system *Mathematica*. When available in the CDF format, the GLP-Tool can be used for free as a standalone application by anyone with access to a computer.

- When introducing the subject of Linear Programming it is rather useful to present the graphical method for solving a two-variable linear program as this method pro-

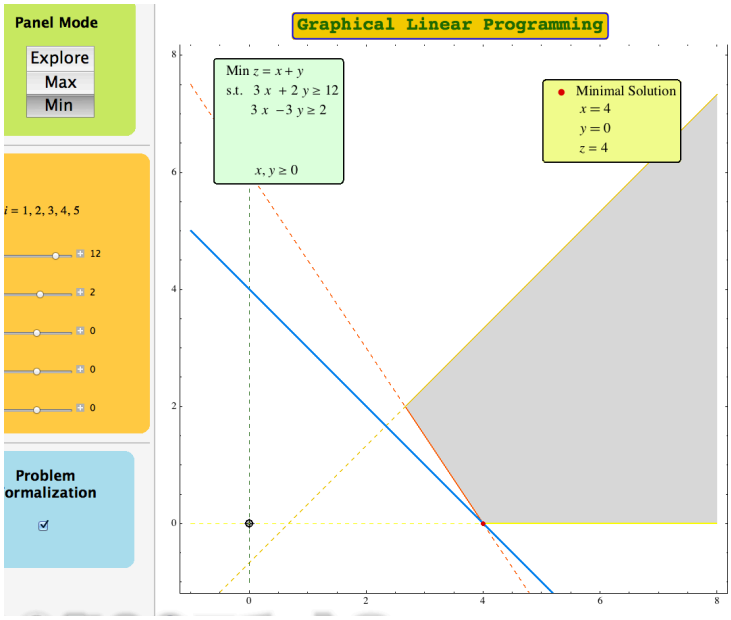


Figure 7: Example of an LP problem with an unbounded feasible solution but with only one optimal solution.

vides valuable insights about the general nature of multivariable linear programming models. Nonetheless, without a dynamic tool, it is not easy to show students what happens in a LP problem as constraint boundary lines and objective-value lines move around on a graphic.

- The GLP-Tool is a visual, interactive, dynamic tool where all the analytical and graphical information is update in real time. These features render the GLP-Tool as a fully active learning technical tool that engages students and teachers and provides them with an effective teaching/learning environment.

- All the functionalities of the GLP-Tool, along with all the displayed information are designed to conform with the way Linear Programming is presented in the classrooms both at the high school and undergraduate levels.

- Implemented with *Mathematica* the GLP-Tool brings together the best of two worlds: the flexibility of Java visual applets and the sophistication and easy packaging of Wolfram's Mathematica demonstrations.

- The GLP-Tool has a very intuitive interface that allows even the most inexperienced user, with no previous knowledge in educational software, to use all its features in an efficient and autonomous way, right from the start.

- In our future work, we plan to continue improving the GLP, including new features such as the graphical visualization of the steps of the simplex method or the computation of the sensitivity analysis intervals for each coefficient of a problem instance.

REFERENCES

[1] Bonwell, C. C., Eison, J. A., 'Active learning: Creating excitement in the classroom', *ASHEERIC Higher Education* Report No. **1**, George Washington University, Washington, DC, 1991.

[2] Prince, M., 'Does active learning work? A review of the research', *Journal of Engineering Education*, ASEE, Vol. **93(3)** , pp. 1-9, 2004.

[3] Kydd, C., 'The Effectiveness of Using a Web-Based Applet to Teach Concepts of Linear Programming: An Experiment in Active Learning', *Transactions on Education*, INFORMS, Vol. **12(2)** , pp. 78-88, 2012.

[4] Conceição, A. C., Pereira, J. C., Silva C. M., Simão, C. R. 'Mathematica in the Class-Room: New Tools for Exploring Precalculus and Differential Calculus', *CSEI2012 - 1st National Conference on Symbolic Computation in Education and Research*, Lisbon, Portugal, 2012. http://hdl.handle.net/10400.1/1105

[5] Shepard, B., 'Linear Programming and Pivoting in 2D', *The Computational Geometry Lab at McGill*, 2010. http://cgm.cs.mcgill.ca/ beezer/cs601/main.htm Accessed May 20, 2013.

[6] Hall, J., Baird, M., 'LP Explorer 1.0', *University of Edinburgh*, 2002. http://www.maths.ed.ac.uk/LP-Explorer/ Accessed May 22, 2013.

[7] Zhang, Y., 'Graphical Simplex Algorithmv(2D)',*UCMERCED*, 2010. https://eng.ucmerced.edu/people/yzhang/ projects/clientsideLP Accessed May 21, 2013.

[8] Green, L., 'Exploring linear programming', *Lake Tahoe Community College*, 2010. http://www.ltcconline.net/greenl/java/IntermedCollegeAlgebra/LinearProgramming/LinearProgramming.html Accessed May 21, 2013.

[9] Kydd, C., 'Linear programming applet', *University of Delaware*, 2010. http://www.udel.edu/present/tools/lpapplet/lpapplet.html Accessed May 14, 2013.

[10] Wright, D., 'Animated linear programming applet', *St. Edward's University Computer Sciences Advanced Computing Lab*, 2010. http://www.cs.stedwards.edu/∼wright/linprog/AnimaLP.html Accessed May 22, 2013.

[11] Pegg, Jr. E., 'Graph of Inequalities', *Wolfram Demonstrations Project*. http://demonstrations.wolfram.com/GraphOfInequalities/ Accessed May 22, 2013.

[12] Mukherjee, S., 'Two-Phase Simplex Method', *Wolfram Demonstrations Project*. http://demonstrations.wolfram.com/TwoPhaseSimplexMethod/ Accessed May 22, 2013.

[13] Bunduchi E., Mandric I., 'Parametric Linear Programming', *Wolfram Demonstrations Project*. http://demonstrations.wolfram.com/ParametricLinearProgramming/ Accessed May 22, 2013.

[14] Boucher, C., 'The Fundamental Theorem of Linear Programming', *Wolfram Demonstrations Project*. http://demonstrations.wolfram.com/TheFundamentalTheoremOfLinearProgramming/ Accessed May 22, 2013.

[15] Carducci, O. M., 'Graphical Linear Programming for Two Variables', *Wolfram Demonstrations Project*. http://demonstrations.wolfram.com/GraphicalLinearProgrammingForTwoVariables/ Accessed May 22, 2013.

[16] Kragt, M., Jiang, Z., 'Oil Mallee Farming Optimization Problem', *Wolfram Demonstrations Project*. http://demonstrations.wolfram.com/OilMalleeFarmingOptimizationProblem/ Accessed May 22, 2013.

[17] Boucher, C., 'A Simple, Standard Linear Programming Scenario', *Wolfram Demonstrations Project*. http://demonstrations.wolfram.com/ASimpleStandardLinearProgrammingScenario/ Accessed May 22, 2013.