LPG Program Tutorial

This is a tutorial help file for the program named **LPG**. The letters **LPG** stand for **Linear Programming Graphs**. This tutorial will explain the basic features of **LPG**. This tutorial is intended to help new users get started using the program and is provided in addition to two other Help files that accompany the program. If you are a new user we recommend you print this entire tutorial so you can read these instructions from a paper copy while you view the output on your screen. This tutorial is 29 printed pages.

This tutorial assumes you are either a student or a teacher in a math class that is at the intermediate algebra level or beyond. You should be familiar with the slope-intercept form for the equation of a line and you should know the **Trichotomy Law** and be able to graph any system of linear inequalities in the two variables x and y. If you have already been using the program we suggest you close it down and then restart it to insure that all the program default values and inequalities will match what is described in this tutorial.

This program can be used to graph systems of linear inequalities in two variables and to solve maximum/minimum problems when an objective function is associated with the region of feasible solutions that represents the possible solutions to those inequalities. Even more significant is the ability of this program to help you create problems that have "nice" points of intersection.

Program initialization consists of two stages; the first of which creates the examples that are shown in this tutorial. Next, the program looks for a file named **`StartupLPG.txt`**. If that file is found then its contents are read and overwrite the tutorial examples. Since, if the **`StartupLPG.txt`** file exists, we cannot guarantee its contents will match the tutorial examples, we suggest you delete that file. Later, after you are familiar with this program, you can save any default examples you want by just naming your saved file **`StartupLPG.txt`**. The file **`StartupLPG.txt`** can thus be changed, or deleted or overwritten with impunity. To insure the tutorial examples appear as intended, no file named **`StartupLPG.txt`** is distributed as part of any new installation. If you have already been using the program we suggest you close it down and re-start it so what we describe in this tutorial will match what you see on your screen.

When the program is first started it should show the following graph.

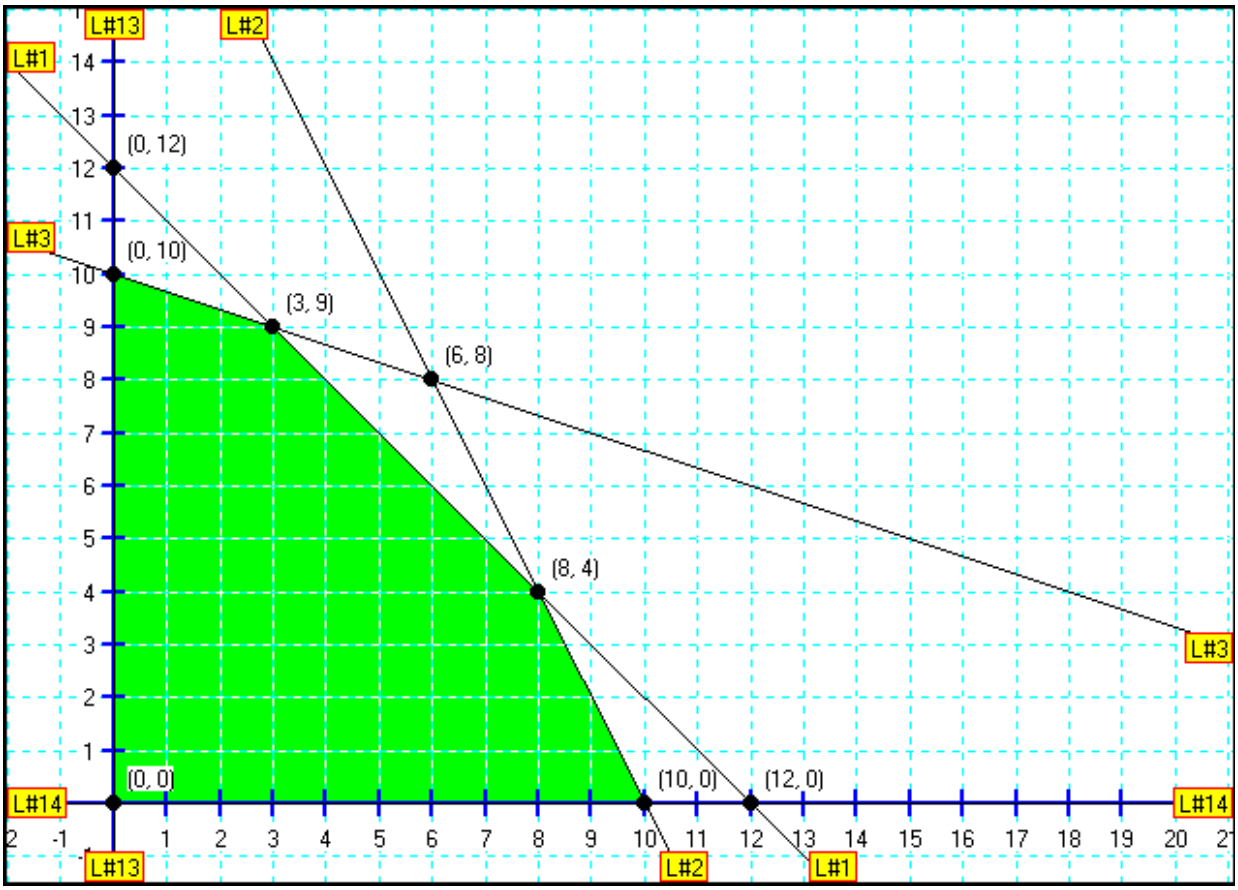


Figure 1. A Graph of a system of linear inequalities.

The particular inequalities that were used to make the above graph are:

$x + y \leq 12$	(line #1)
$2x + y \leq 20$	(line #2)
$x + 3y \leq 30$	(line #3)
$x \geq 0$	(line #13)
$y \geq 0$	(line #14)

You might note there are 5 sides in the polygon that represents the region of feasible solutions that is shown in the above figure. Each side or edge of the polygon is determined by each of these 5 constraints. You will see several little boxes with yellow backgrounds like L#3. These are just labels of the lines so you can tell which line number is associated with each edge of the region of feasible solutions.

The last two of the above five inequalities were automatically set by the program. They insure that the graph of the region of feasible solutions is contained within the first quadrant. The last two inequalities are called non-negativity conditions, and almost all practical applications of linear inequalities assume neither x nor y are negative. This program does not require these non-negativity conditions to always hold, but this program has options to automatically always set such conditions without requiring you to explicitly enter them.

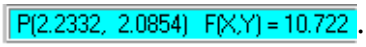
The objective function that is associated with the above inequalities is $2x + 3y$. You don't see a representation of this objective function anywhere in the above graph. Since this is a function of the two variables x and y we can also write it using the notation **F(X,Y)=2X+3Y**. This program also allows you to set one of three options associated with the objective function. You can try to minimize **F(X,Y)** or you can try to maximize **F(X,Y)**, or you can try to both maximize and minimize **F(X,Y)**. The default option is to try to both maximize and minimize **F(X,Y)**.

Using the Objective Function Trace Mode

There are two yellow-colored buttons in the tool bar. The first of these has the letter **F** on its face. This button 

controls the use of what is called the **Objective Function Trace Mode**. Click on this button with your mouse and you should see the button change to a white background that indicates the objective function trace mode is active. 

.

Now left-click your mouse somewhere in the middle of the graph and continue to hold down the left mouse button while you drag the mouse over the graph. You should see a point at the tip of the mouse cursor and if you read the information in the status line at the bottom of the screen you should temporarily see many displayed lines that look something like 

. This particular display information means when **X=2.2332** and **Y=2.0854** then **F(X,Y)=10.722**. In this example the point **P(2.2332,2.0854)** is called the trace point and it is determined by where you are currently pointing the mouse.

Of course the trace point and the function continually change values as you move the mouse over the graph. You might note that the program will calculate objective function values even when you move the mouse outside the green-colored part of the graph that represents the region of feasible solutions.

Let up on the left mouse button and then click down and hold the right mouse button down while you drag the mouse around. When dragging the right mouse button with the **Objective Function Mode** turned on you will see a line whose slope matches the objective function line slope. In this mode the y-intercept is particularly important, and you may note the status line at the bottom of the screen shows the value of the y-intercept. Continuing to hold down the right mouse button, move the trace point so it is over the point **(3,9)**. See the next figure below where the y-intercept point can be seen and the status line shows the approximate values of **F** and the y-intercept.

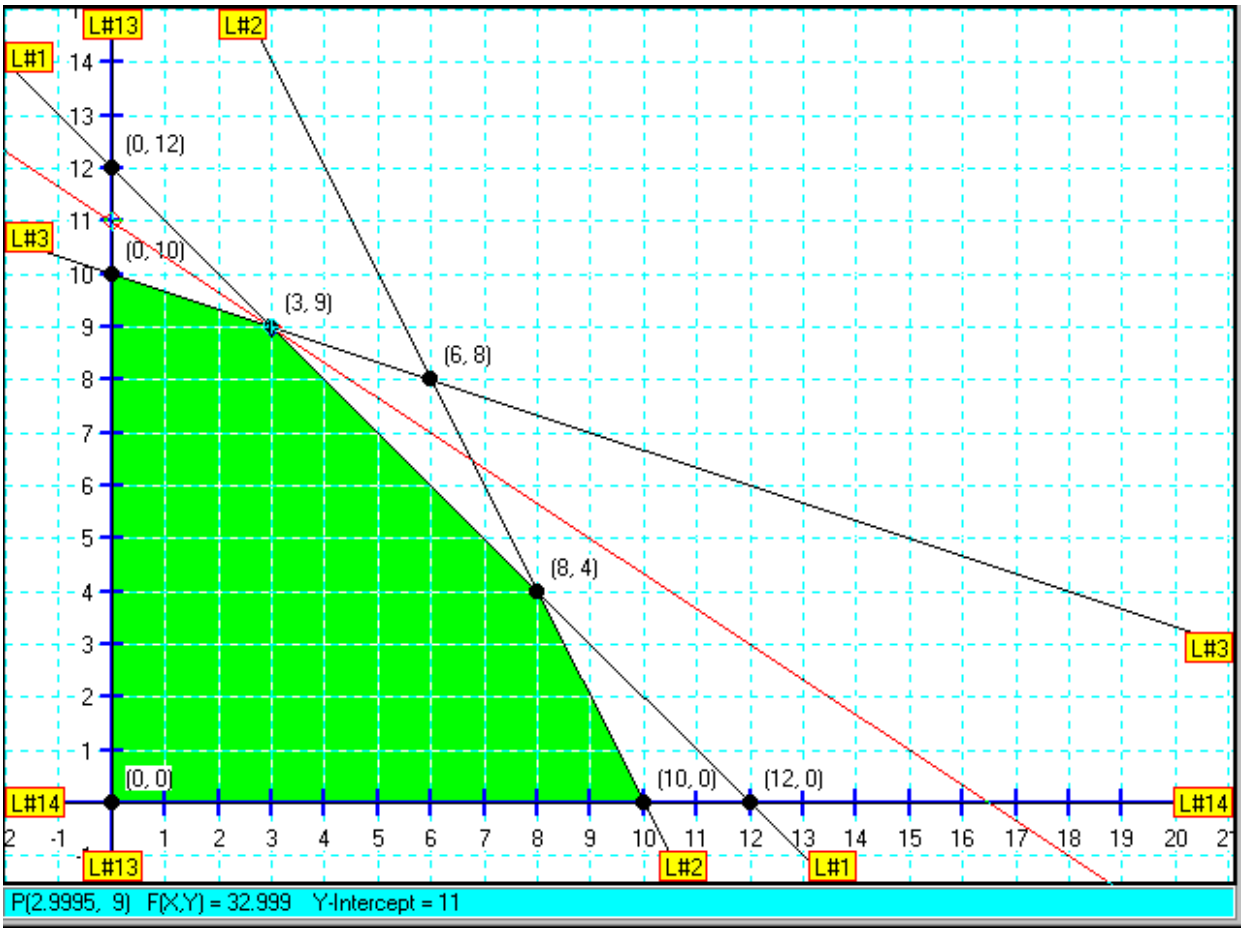


Figure 2. Objective Function Trace Mode Line & Y-Intercept.

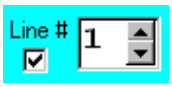
Multiplying the y-intercept by the **Objective Function's B** coefficient yields the value of the objective function. When the line is dragged so that it just touches the last vertex point of the region of feasible solutions then you will also have found the maximum or minimum of the objective function. The highest y-intercept determines the maximum and the lowest y-intercept determines the minimum. Don't forget to multiply the y-intercept value of the **B** coefficient of the objective function.

It is a mathematical fact that each linear inequality determines a half-plane. Each half-plane is an example of what is called a convex set. The intersection of two or more convex sets is again a convex set. In fact, the region of feasible solutions for any linear programming problem is always a convex set. When the region is bounded, the edges of the convex set form a closed polygon. Each edge of the polygon is associated with a linear inequality and we call those edges lines. We also refer to each linear inequality as a constraint. This program numbers each line or constraint, but under the Options menu you can turn off the numbering for those times when you don't need it.

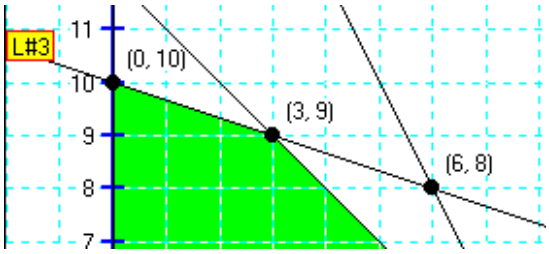
Now go back to the left mouse button and try to move the mouse so that it is near the point **(3,9)**. The exact objective function value at that point should be **2*3+3*9=6+27=33**. It may not be possible to point your mouse so that the status line shows the exact coordinates of the point **(3,9)**. So when you read the status line coordinates and even the objective function value at the bottom of **Figure 2** you have to use some common sense. Most computer screens have a dot or pixel resolution that is about 100 pixels per inch. So you should not expect any coordinate reading to be exact and you should expect a little round off error of about 1/100 of an inch.

Now try to move the mouse directly over the point **(8,4)**. It may also be impossible to precisely hit that point according to the coordinates shown in the status line, but when **X=8** and **Y=4** then **F(8,4)=2*8+3*4=16+12=28**.

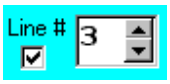
Using and Changing the Line Trace Mode

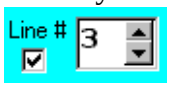
Because it is difficult to precisely move and keep the mouse over a vertex or intersection point, the program has another trace mode that is called the **Line Trace Mode**. Locate the yellow-colored button that appears as  . The reason for the letter **L** on this button is because this button controls the **Line Trace Mode**. Left-click this button once and you will notice two things happen. First, the **Objective Function Trace Mode** button pops back up and that mode is automatically turned off. Second, the line trace mode button now appears with a white background  and that means the line trace mode is now active. You might also note that there are two other controls that appear as  and these two controls are intimately related to the line trace mode. For now, left-click your mouse anywhere over the graph and continue to hold down the left mouse button and drag the mouse across the graph. The line trace mode is very similar to the objective function trace mode except that the mouse cursor point will now stay attached to line #1. Now you can more easily move the mouse so that it is near either point **(3,9)** or point **(8,4)**. You may not be able to make the status line hint show the exact integer coordinates because most computer screens have a dot or pixel resolution that is about 100 pixels per inch. So not being on the exact pixel means the program can be no more accurate than about 1 part in 100.

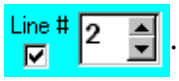
Now suppose you want to move the mouse across the edge of the region of feasible solutions that contains the points **(0,10)** and **(3,9)**. Note that this particular edge is on the line that is #3.



We know this point is determined by line #3 because we can see the little yellow box L#3 near that point. In fact, the point **(0,10)** is the intersection point of lines #3 and #13. The two points **(0,10)** and **(3,9)** are both on line #3. To change the **Line Trace Mode** to work along line #3 all you have to do is click the up-arrow of the control



until it indicates Line #3. Now if you continue to left-drag the mouse across the graph you will note that the trace point stays attached to line #3. You can move the trace point anywhere along line #3, even over the points **(6,8)** and **(18,4)** that are way outside the region of feasible solutions. To switch to line #2 just click the down-arrow of the control 

until it indicates line #2 is selected. 

.

Now you can move the trace point anywhere along line #2.

Now left-click the **Line Trace Mode** button and that should turn off the **Line Trace Mode** and you will see the button pops back up and shows its normal yellow color when it is turned off.

The two buttons 

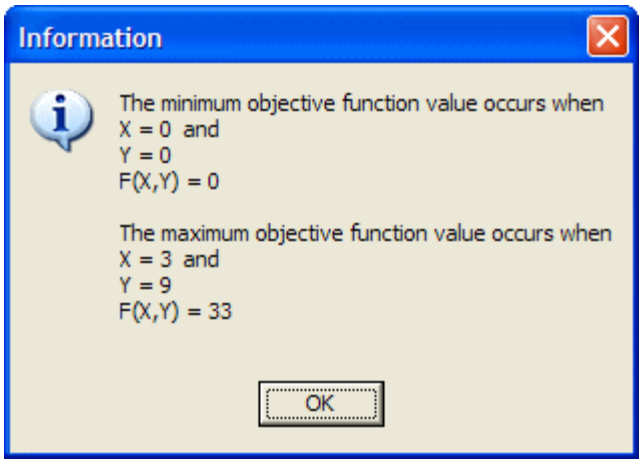
and 

act as a group like radio buttons in that only one can be turned on at a time. Left-clicking either button turns off the other one. Also, left-clicking either button when it is turned on just turns that button off.

Solving Any Linear Programming Problem

After a linear programming problem has been setup, it is easy to solve that problem. If you now left-click the button on the tool bar that appears as 

you should see the following message:



We have two answers to the default problem because the objective function option was set to try to find both minimum and maximum values.

So solving a linear programming problem is no problem. 99% of your time spent with the **LPG** program just has to do with entering and setting up your problem. So most of this tutorial will concentrate on how to enter and setup problems and make nice graphs.

Before continuing, a sometimes more instructive alternative to just pressing the 

button is to select the menu item **Action | Make A Table of Objective Function Values….** Select this menu item now and you should see a dialog box that contains the information as is shown in **Figure 3** below.

Source of the Point:	X	Y	2*X + 3*Y
Intersection Lines 1 & 2 feasible;	8	4	28
Intersection Lines 1 & 3 feasible; Maximum !	3	9	33
Intersection Lines 2 & 3 not feasible;	6	8	36
Intersection Lines 1 & 13 not feasible;	0	12	36
Intersection Lines 2 & 13 not feasible;	0	20	60
Intersection Lines 3 & 13 feasible;	0	10	30
Intersection Lines 1 & 14 not feasible;	12	0	24
Intersection Lines 2 & 14 feasible;	10	0	20
Intersection Lines 3 & 14 not feasible;	30	0	60
Intersection Lines 13 & 14 feasible; Minimum !	0	0	0

Figure 3. A Table of Values of the Objective Function.

Whenever you make a table of objective function values, the program makes a list of all intersection points of all pairs of constraint lines. It will mark each point as being feasible or not. Feasible points are those intersection points that are vertices of the polygon that represents the region of feasible solutions. Non-feasible points are intersection points that don't lie on an edge of the region of feasible solutions. For example, in Figure 1 above, the point **(6,8)** is not a feasible point.

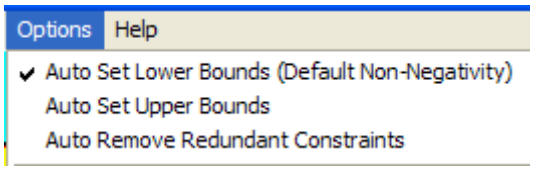
In addition to labeling each intersection point as being feasible or not, the program will also label points that qualify as absolute maximums or absolute minimums. The program will only label the first qualifying point in each category. Usually maximums and minimums occur only at vertices of the polygon. It is rare, but it can happen that the same maximum or minimum occurs at an infinite number of points that are all along one edge of the region of feasible solutions. When and if that happens, **LPG** will write an asterisk symbol following the **!**. So if you see **!*** following any maximum or minimum value then you know that value is not unique.

Click the **OK** button in the dialog box that shows the above table and that dialog box will disappear.

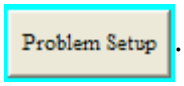
How To Enter A New Problem

As we have already stated, most of your time spent using this program will be involved with setting up a new problem. There are two phases to setting up a problem. The first phase is entering all the linear inequalities and entering the object function coefficients. As you are about to learn, that first phase is very easy. The second phase is making a good graph, and that phase can be more involved because of all the options you can set. In fact, this program goes to a good deal of trouble to check that any problem you give it is well-defined and consistent. If you should enter any inconsistent information or enter a problem that has no solution this program will give some indication of what is wrong.

To setup a new problem we recommend you turn off the second two **Auto** options that are under the main menu **Options**. You can leave the first option set if your problem assumes x and y should not be negative. The first three options under the **Options** menu appear as shown below.



Clicking on any one of these menu items just turns on or turns off a check mark in front of the item.

Next, click the button that appears on the tool bar as 

. You should then see the problem setup dialog box shown in **Figure 4** below.

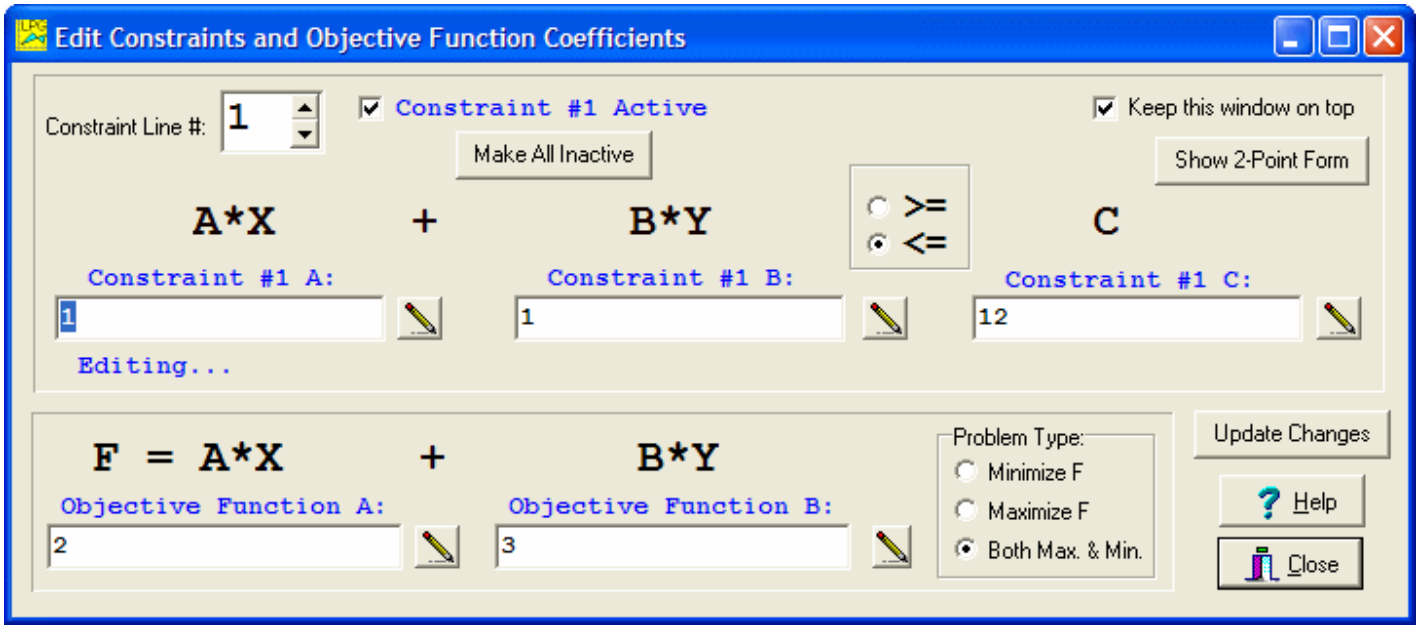


Figure 4. The Setup Dialog Box for Editing Constraints.

To enter a new problem you can first click the button 

. This turns off all constraints from any problem that may be currently entered in the program.

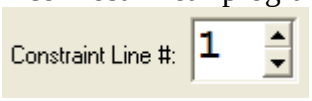
We usually enter the two objective function coefficients first as well as the **Problem Type**. After that we enter each constraint one by one.

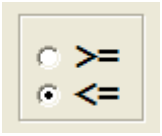
You can have only one objective function and it is an expression of the form **A*X+B*Y**. You enter the coefficients for **A** and **B** in the two edit boxes at the bottom left of the above dialog box. You also select one of the **Problem Type** options by clicking an item in the radio group 

.

After entering the two coefficients and selecting the problem type you can press the button 

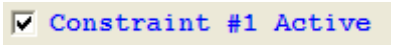
to store those entries internally in the program.

Next you can start entering your constraints one by one. Since most linear programming problems have more than one constraint, you can use the control that appears as 

to first select the number of the next constraint you wish to enter or edit. Each constraint must be in one of two forms: **A*X+B*Y<=C** or **A*X+B*Y>=C**. You enter the three coefficients for **A**, **B** and **C** in the three edit boxes and you select one radio button in the control 

that corresponds to the type of inequality contained in your constraint. After entering each constraint you should press the button that appears as 

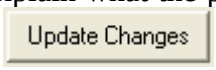
to internally register the **A**, **B**, **C**, coefficients and the type of constraint. This is important to do before you change the constraint line number to enter the next constraint.

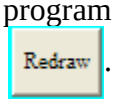
The control that appears as 

is used to disable a constraint without removing its coefficients from the program. When this check box is unmarked then the corresponding constraint is ignored by the program. However, a simple click on this check box will make the constraint active again. Sometimes it useful to temporarily turn off a constraint. By employing this checkbox you can quickly bring back any constraint you have temporarily removed.

Most practical linear programming problems have only a few constraints, typically three or less. However, the **LPG** program allows you to enter up to 10 constraints if you need that many. **LPG** also has an additional four constraints that are special. So your constraint lines will normally be numbered between 1 and 10, but constraint numbers 11, 12, 13, and 14 are also possible.

How the Program Works After You Setup Your Problem

It will be helpful to explain what the program does when you are entering and editing any problem. Each time you press the button 

in the above dialog box, the program resets itself, tries to resolve the changes you are making and it rebuilds the current problem. The program also rebuilds the current problem anytime you press the **Redraw** button on the tool bar that appears as 

.

The rebuilding of a problem involves eight stages that will be described next.

In the first stage, the program turns off the four special constraints numbered 11, 12, 13 and 14.

Second, the program looks to see if the menu item to **Auto Set Lower Bounds** is checked. If that menu item is checked then the program will automatically get the two lower bounds constants and completely setup constraint numbers 13 and 14 for you as described below.

Third, the program looks to see if the menu item **Auto Set Upper Bounds** is checked. If that menu item is checked then the program will automatically get the two upper bounds constants and completely setup constraint numbers 11 and 12 for you as described below.

Fourth, the program finds all intersection points between all pairs of constraint lines.

Fifth, the program builds a set of feasible intersection points, by keeping only those intersection points that satisfy the current set of constraints.

Sixth, the program checks to see if the set of feasible intersection points makes a closed polygon.

Seventh, the program looks to see if the menu item **Remove Redundant Constraints** is checked or not. If this menu item is checked then the program will look at the current set of constraints and try to remove any and all constraints that are not actually used. It is during this stage that the program may automatically make some of the constraints you entered inactive. It can also make any automatic constraints that it entered inactive. In fact, the program repeats these first seven stages until all redundant constraints have been removed from the problem.

Finally, at the eighth stage, if the final set of constraints still makes a closed polygon then the program will graph and shade the region of feasible solutions. If the final set of constraints determines an unbounded region the program will still graph that part of the region of feasible solutions that lies in the current window.

The Automatic Bounds Constraints

There are four special constraints the program sometimes uses. What makes these constraints special is that the program will automatically turn them on or automatically turn them off, as the program deems fit. The last two of the four special constraints are numbered #13 and #14 and these correspond to the lower bounds in your problem. When the **Option** menu item that appears as ✔ Auto Set Lower Bounds (Default Non-Negativity) is checked the program will automatically create constraints numbered #13 and #14 for you.

You don't need to open the **Automatic Bounds** dialog box now, but the constants used in constraints #13 and #14 are taken from the dialog box that you can open under the **Edit** menu. When you select the menu item that appears under **Edit** as Edit Automatic Lower/Upper Bounds... the program will open the following dialog box that we call the **Automatic Bounds** dialog box.

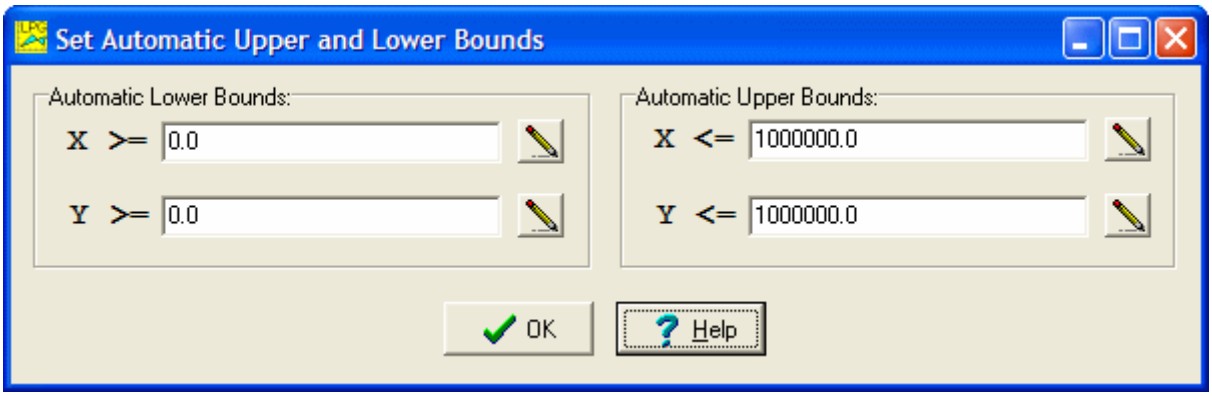


Figure 5. The Automatic Bounds dialog box.

The **Automatic Lower Bounds** are the two numbers that appear in the two edit boxes on the left. The two default lower bound values are 0.0 and for this reason constraint #13 and constraint #14 are sometimes called the non-negativity constraints. Of course if you set two lower bound constants other than **`0.0`**, say **`P`** and **`Q`**, then the program will enter constraint #13 as **`1*X+0*Y>=P`** and it will enter constraint #14 as **`0*X+1*Y>+Q`**. In this case when **`P`** and **`Q`** are not zero, they are considered lower bounds. These constants could even be negative numbers themselves. Rarely should you need to enter numbers other than 0's for lower bounds in a typical problem.

Similarly, the **Automatic Upper Bounds** are the two numbers that appear in the two edit boxes on the right. The two default upper bound constant values are positive one million. Of course if you set two other upper bounds, say **`P`** and **`Q`**, then the program will enter constraint #12 as **`1*X+0*Y<=P`** and it will enter constraint #11 as **`0*X+1*Y<=Q`**. In this case **`P`** and **`Q`** can even be negative numbers, but most problems will have solutions that lie within the default bounds ranges.

You don't have to set or even use these automatic bounds. The program only uses these values when you turn on either of the first two menu items under the main **Options** menu. When the first two menu items are turned off you can just enter your own constraints as lower or upper bounds if you want or need such bounds, using any Line #1 through Line #10.

Other Program Checking Options

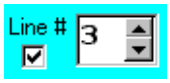
The **LPG** program checks for inconsistent as well as duplicate and redundant constraints. It is certainly possible to enter constraints that yield either an empty region of feasible solutions or even a region that is unbounded and extends to infinity in some direction. An empty region of feasible solutions is usually considered to be the result of an inconsistent system of constraints.

You are not required to enter a problem in which the region of feasible solutions is a finite closed polygon. But when that is the case, the program should make an accurate graph of that region. If you enter a system of constraints that does not determine a closed polygon, the program has to work a little harder to make the graph. In some cases you may to wish to enter enough constraints to make a closed polygon.

Using the Special Line Definition Button

At this point you should be able to enter any system of constraints and a get a solution to that system. After graphing a system you may wish to slightly alter one of the lines in that system. The red-colored button that appears as 

can be used to perform two special operations that can redefine a given line.

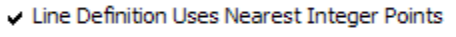
Before using this button, you should set the conrol 

to select the line you want to alter. Then left-click the **Line Definition** button and that button will appear pushed down 

.

For the first special operation, move the mouse anywhere over the graph and then push down the right mouse button and while continuing to hold down the right mouse button drag the mouse across the graph. You should see a line with a fixed slope move with the mouse. Position the line where you want it to remain and let up on the right mouse button. The line that was selected for alterations will now appear where you left it and the entire region of feasible solutions and all intersection points will be recalculated taking into account the new position of the altered line. You will also note that the **Line Definition** button gets turned off automatically (it pops back up) when you let up on the right mouse button.

A second special operation is used to change both the slope and position of a line. To perform this operation you start again by left-clicking the **Line Definition** button. That button will appear pushed down. Now choose any point in the graph, move the mouse to that point, and begin by holding down the left mouse button at that point. Continue to hold down the left mouse button and move the mouse in any direction until it is over another point that should be a second point on your desired line. Then when you let up on the left mouse button the program will re-define the selected line so that it goes through the point you first clicked and it goes through the last point where you let up the left mouse button. The **Line Definition** button will automatically pop back up when you let up the left mouse button at the second point.

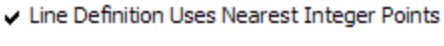
Under the **Options** menu you will find an option like 

. When this option is turned on, the program will determine the two points with the nearest integer coordinates. So using the steps in this example it is very easy to just draw the line where you want it. You may note the program will fix the position of the first point you left-click (this is called an anchor point) and then as you drag the mouse you can see the other point through which the line will be drawn. The line you see drawn when you let up on the left mouse button is basically the newly defined line, except the line will start and end with points with integer coordinates.

Building a Problem By Just Drawing Lines

You can create a problem with "nice" intersection points by making judicious use of the **Line Definition** button. 

. First, make sure the option to use integer point coordinates is checked under the **Options** menu.



.

To illustrate how to setup a nice problem from nothing, click the Setup button Problem Setup, and once the dialog box is up click the button Make All Inactive. This turns off all constraints. Click the Close button to close down the setup dialog box.

You should now be looking at a blank graph. If necessary, click on the control Line # 1 to make sure line #1 is the currently chosen line, even though it is inactive.

Then to create the first nice line, click the **Line Definition** button Line Def.. As soon as you press this button it appears as Line Def.. Now move your mouse near the point on the y-axis that is **`(0,12)`**. Click and hold down the left mouse button then move the mouse near the point that is **`(6,7)`**. While you are moving the mouse the program will anchor the current line at the first point **`(0,12)`** and when you let up on the left mouse button after moving the mouse near **`(6,7)`** the program will then turn on constraint #1 and it will draw a line through the two points **`(0,12)`** and **`(6,7)`**. Note that you only had to move the mouse near these two points and the program automatically selected the nearest integer coordinates. Also note that the **Line Definition** button automatically switched itself off after you determined the second point on the line. You should now see the graph shown below.

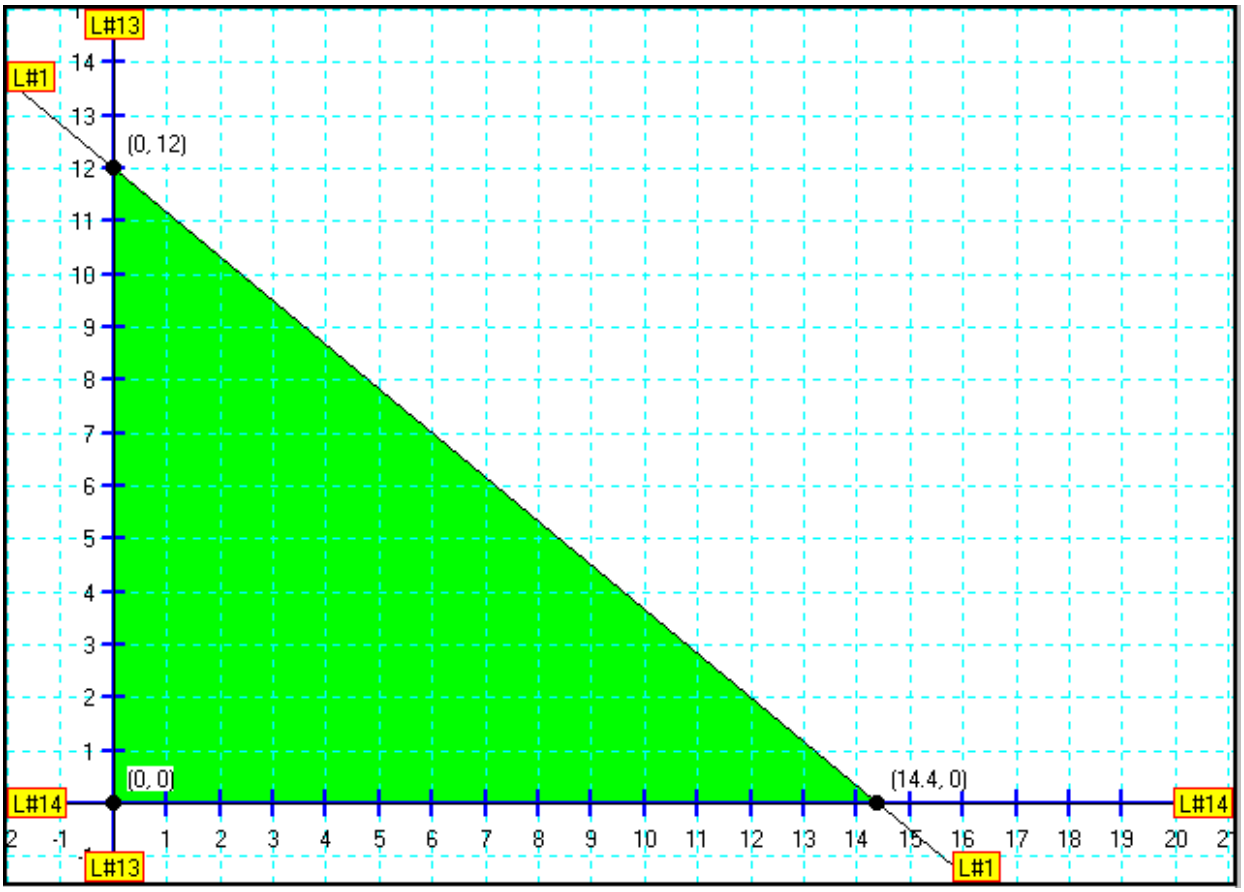
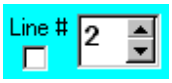


Figure 6. The Graph of One Constraint.

To begin the setup for the second line, first click the control 

to make line #2 the currently chosen line. It is inactive because the checkbox is not checked, but that is **OK**. Next, left-click the Line Definition button 

and note that this button appears in the down position again 

and the program has erased the green shading so you can see all the points under line #1.

Move the move the mouse near the point **(6,7)**. Click and continue to hold down the left-mouse button and move the mouse until it is near the point **(0,3)** and then let up on the left mouse button. You should see the **Line Definition** button pops back up, Line #2 is now active and line #2 goes through the two points **(6,7)** and **(0,3)**.

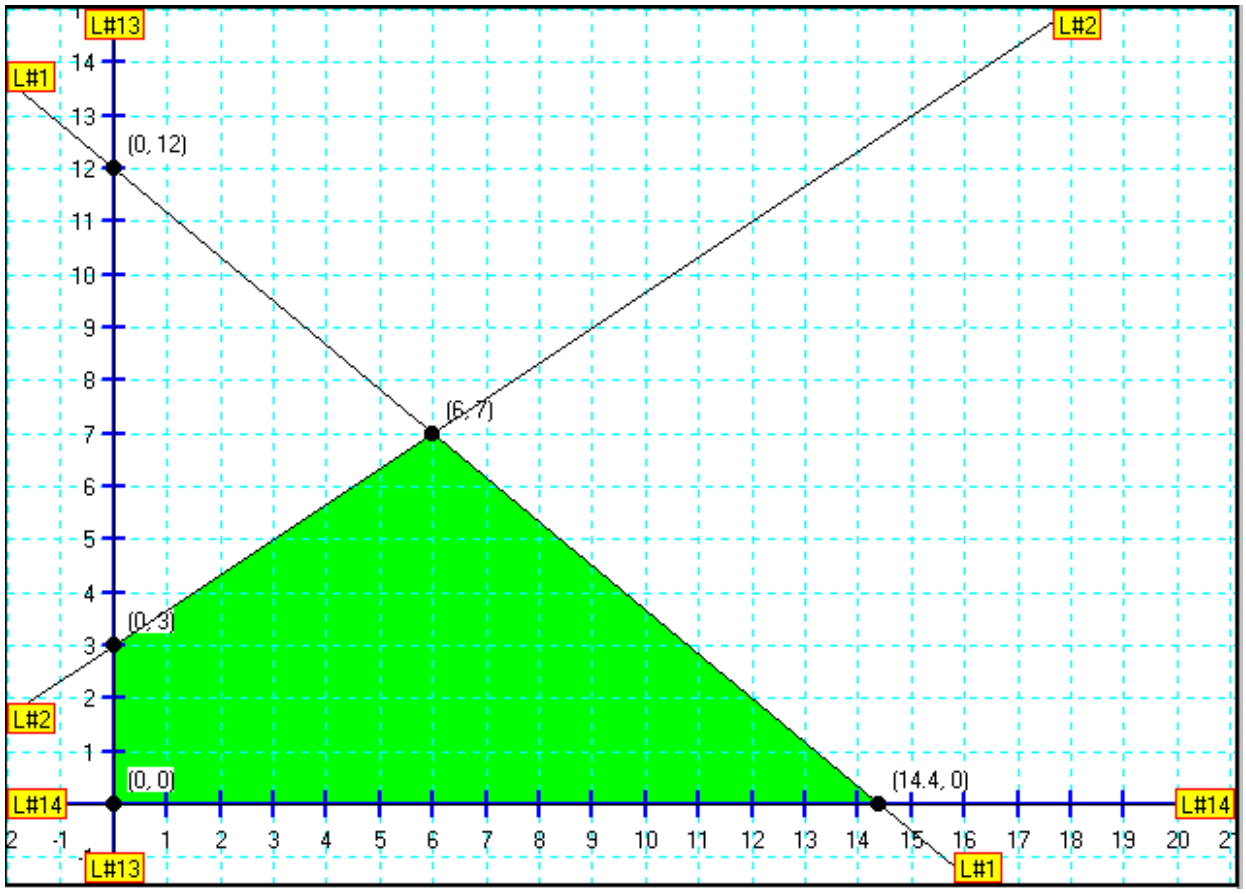
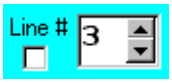


Figure 7. The Graph of Two Constraints.

Note that we have two lines with a nice intersection point between them, and both lines also have integer intercepts on the y-axis. Line #1 has **14.4** for an x-axis intercept.

We continue by building a third constraint by next selecting line #3, even though it is inactive. 

We start re-defining line #3 by first clicking the **Line Definition** button. 

. While this button is down the program is expecting you to click the mouse down at the first point, continue to hold the mouse down and move it to the second point, and then let up on the mouse at the second point.

While looking at the current graph we can we see line #2 appears to go through the point **(3,5)** that has integer coordinates. We can also see that line #1 appears to go through the point **(12,2)** that also has integer coordinates. So we move our mouse near the point **(3,5)**, left-click and continue to hold down the left-mouse button until we move the mouse near the point **(12,2)**. After letting up on the left mouse button we see the new graph shown below.

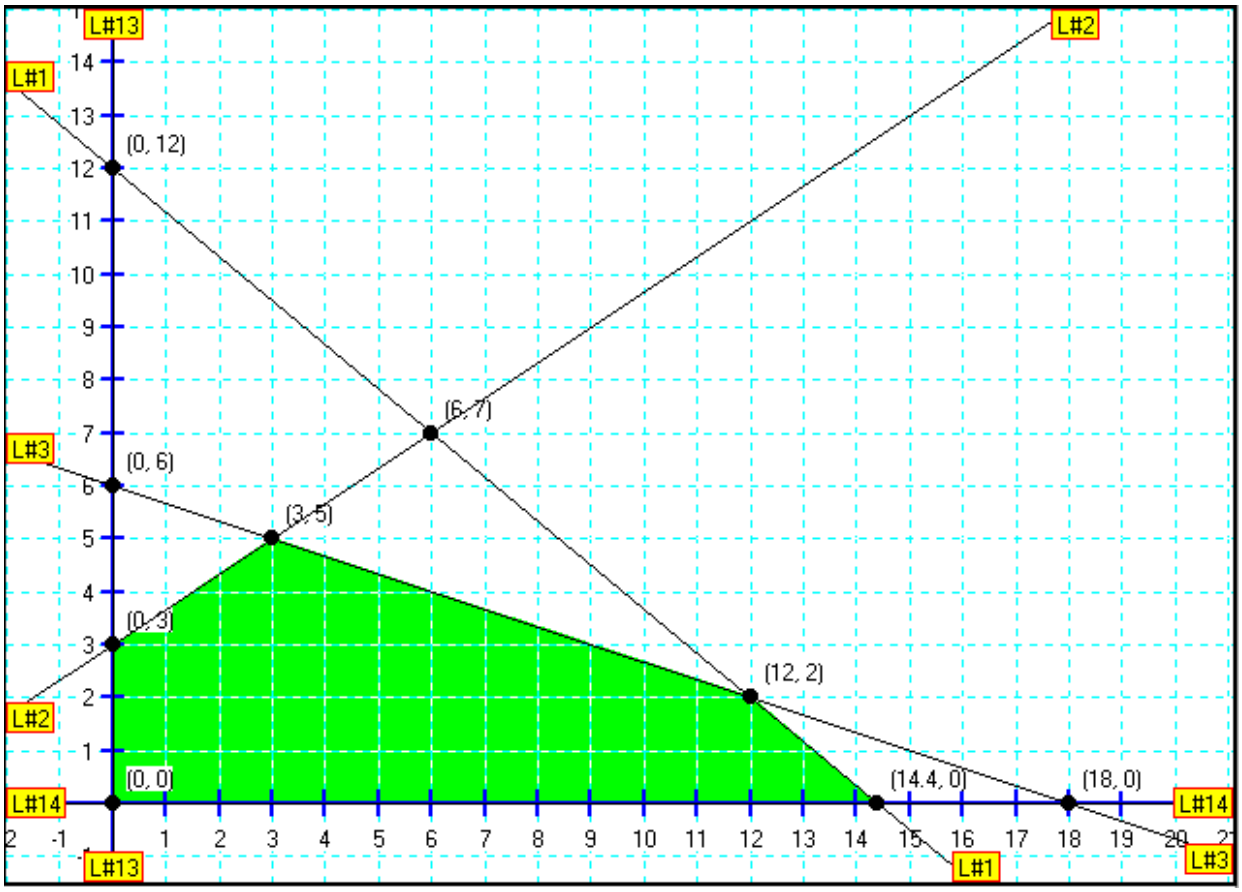


Figure 8. The Graph of Three Constraints.

Note that we now have three nice intersection points between our three lines.

In some cases after defining a line, you may need to bring up the **Setup** dialog box and change the inequality for that line. In the above example all three lines had inequalities that were $\leq$ and we just happened to get a non-empty region of feasible solutions.

For example, press the button Problem Setup and select line #3 in the **Setup** dialog box and change its inequality to $\geq$. Update the changes and you should see the following graph.

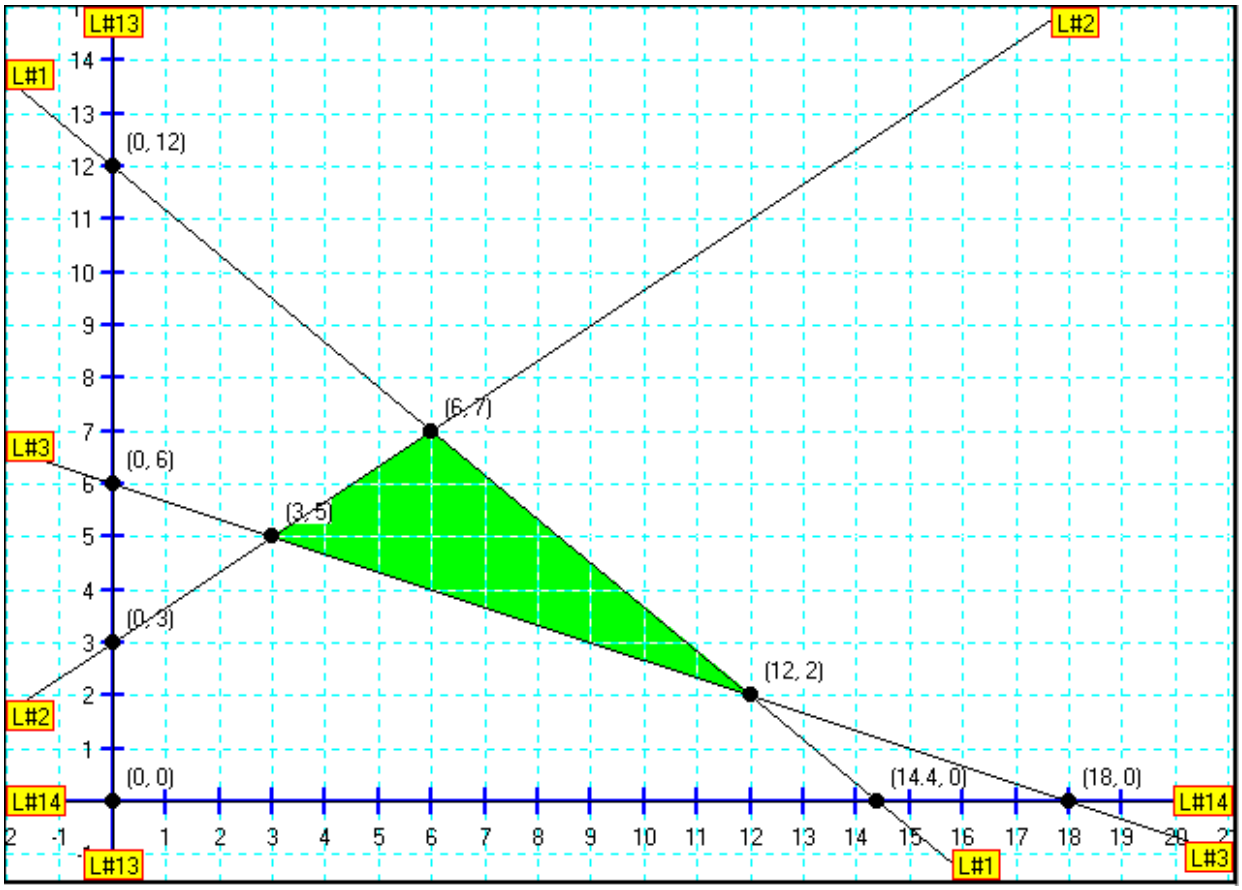


Figure 9. A graph showing mixed constraints.

Positioning Intersection Point Labels

Although you may think the graph in Figure 9 is complete, the graph can be improved. If you are very picky about graphs, you might notice that the labels for all three points on Line #2 overlap that line and are not clearly readable. Wouldn't it be nice if we could change the positions of the point labels? Well, it is not only nice, it is also possible. In fact, the final touch for any graph may be to slightly alter the positions of some of the labels.

Under the **Options** menu select **Point Label Placement/Detail Positioning…** and you should see the following dialog box.

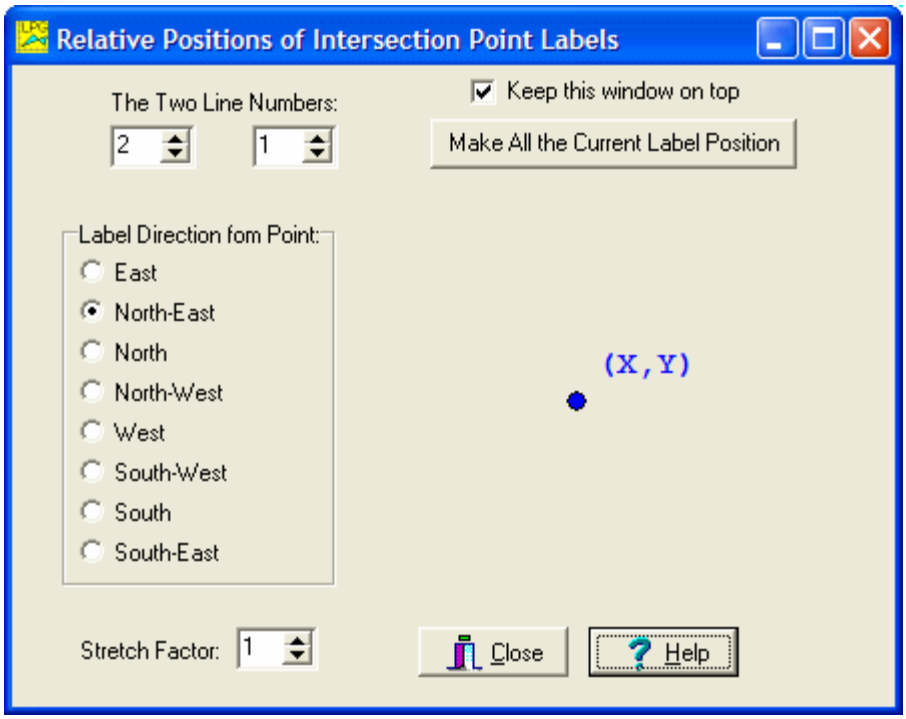
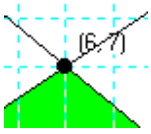
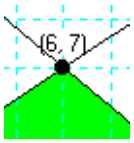


Figure 10. The Point Label Positioning Dialog Box.

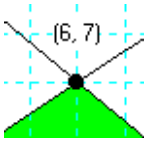
If you can you should move this dialog box so it is out of the way and you can see most of the graph of the region of feasible solutions. In particular, look at the position of the label for the point **(6,7)** in the main graph. This is the point of intersection of Line #1 and Line #2.



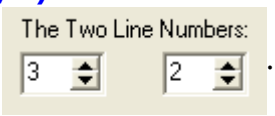
Now back to the dialog box we just opened. There are two line numbers that are always selected and they are shown in the two controls in the upper-left corner. The ordering of the two line numbers is not important. Since Line #2 and Line #1 are already selected and since we want to alter the position of the point **(6,7)** we can do so immediately by just clicking a different label position in the dialog box. Click the **North** position radio button in the dialog box and then look at the point in the graph. Notice how the label is now above the point.



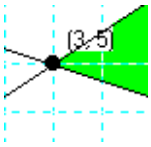
In the **Point Positioning** dialog box increase the stretch factor from 1 to 2. Notice how the point now moves further north.



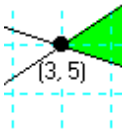
In this position the point looks fairly good, so we will leave it there. Each intersection point can be given a direction and a stretch factor. By making judicious use of both values you should always be able to place a point label at a position where it looks good on the graph.

Ok, now lets move to the point **(3,5)** that is the intersection of Line #2 and Line #3. In the dialog box change the line numbers so they look like: 

. Note that the label is hard to read in its current North-East position.

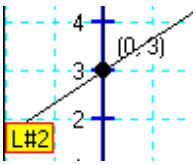


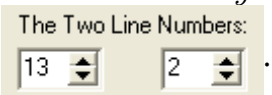
Select the **South** position for this label.



Now that label is much more readable.

The final label we want to alter is the one for the point **(0,3)**.



This point is the y-intercept of Line #2. Since the default constraint associated with Line #13 is $x \geq 0$ this point is an intersection point of Line #2 and Line #13. We might note that Line #13 is the y-axis and Line #14 is the x-axis. In any case, change the two line numbers so they appear as: 

. Then select the **South-East** position for this point of intersection. OK, you can close the point positioning dialog box. Now compare the current graph shown below in **Figure 11** with the graph in **Figure 9**. The **Figure 11** graph has better labels.

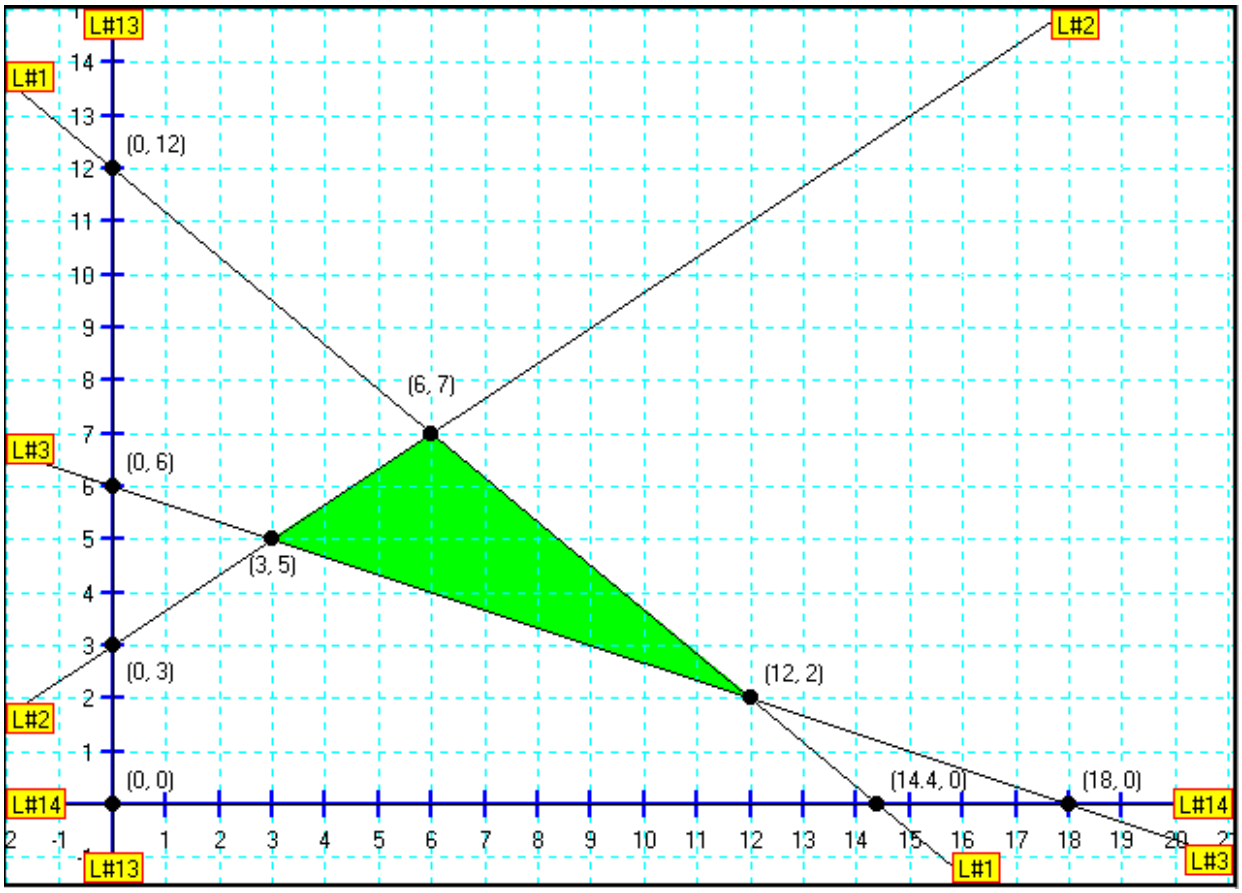


Figure 11. A better-labeled graph than Figure 9.

Performing Zooming Operations

There are times when you will want to zoom in or zoom out from the current graph. **LPG** has several different options for zooming. The simplest of these is to just select the menu item under **Action** that says **Auto-Scale to Show All Intersection Points**. If you do this now you will see the next figure.

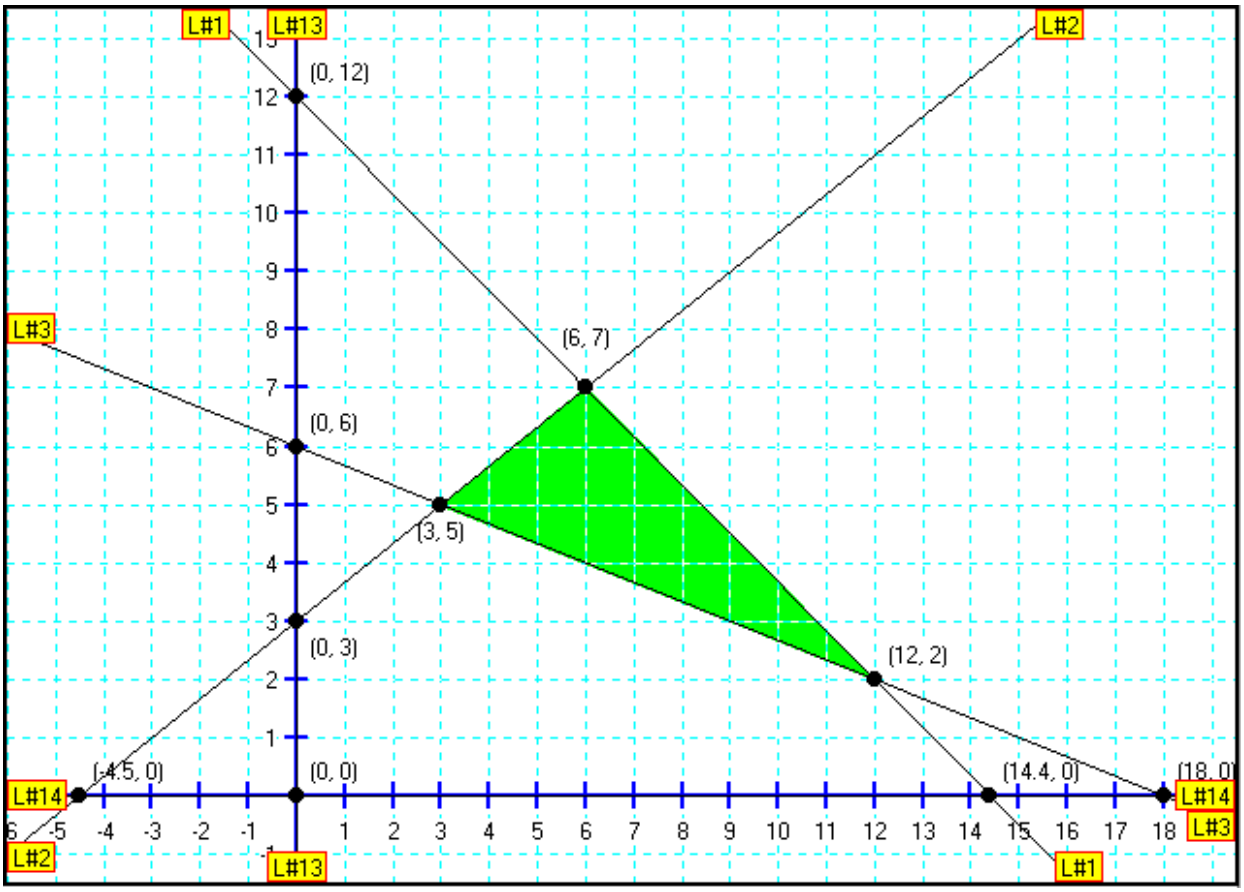
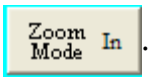


Figure 12. The mixed constaints graph after zooming.

The above graph has changed the window extents from the previous graph so that we now see the point **`(-4.5,0)`** on the negative x-axis.

Another way to zoom is to first set the **Zoom Mode**. In the tool bar you will see a button that looks like 

. Pushing this button does NOT perform zooming. All that pushing this button does is switch the zoom mode between **In** and **Out**. You will always know which mode is currently active because the caption on this button always gives the current mode.

After setting the **Zoom Mode** all you have to do is point the mouse at the center of the region of interest and double left-click the mouse. You can practice this now. Zooming in or out either halves or doubles the XY extent values.

After zooming either way you sometimes will want to return to the original graph. You can do this most easily by right-clicking the **Zoom Mode** button and you will see the following dialog box.

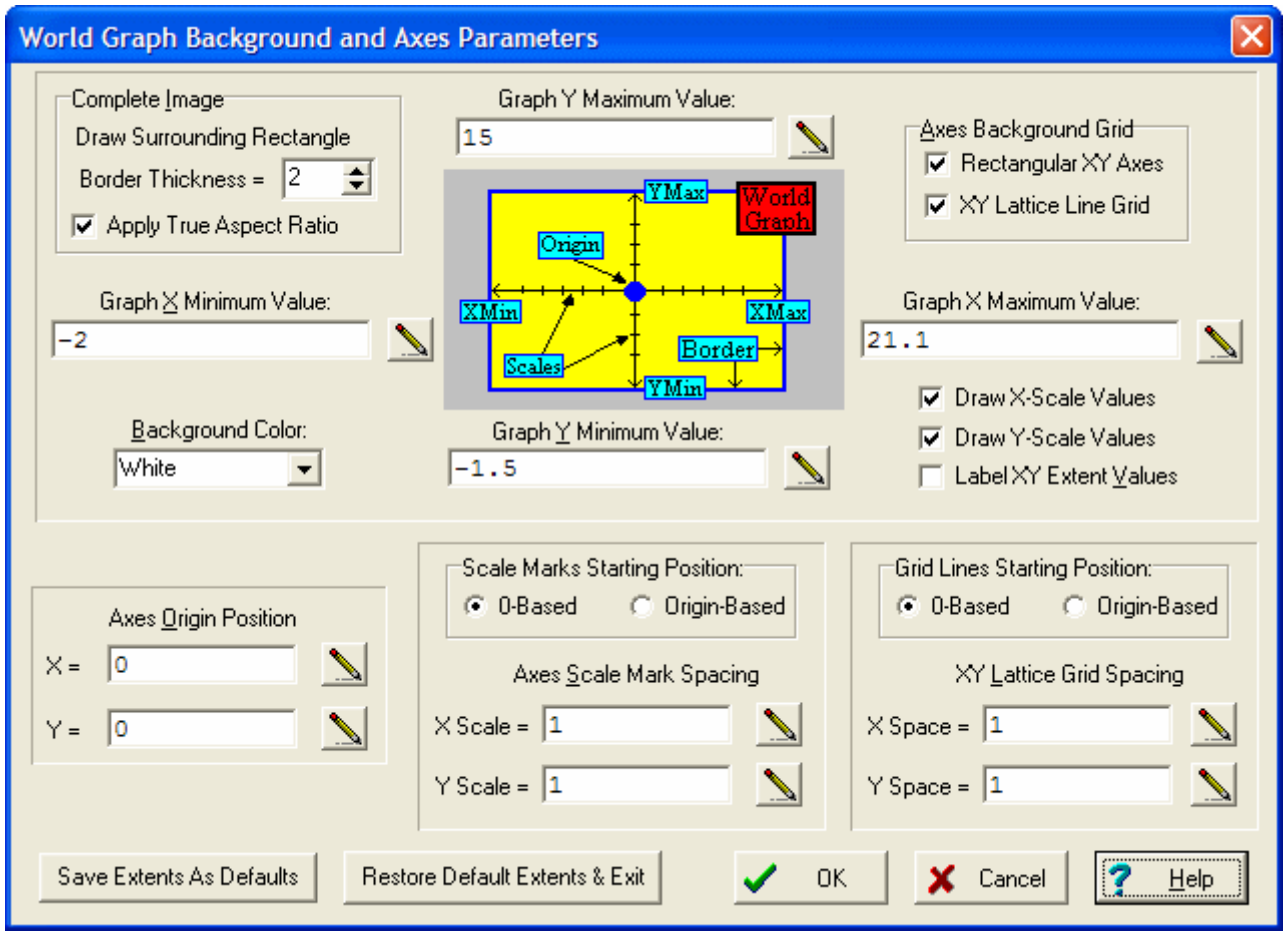


Figure 13. The World Graph Parameters dialog box.

The four most important numbers in this dialog box are the X-Minimum, X-maximum, Y-Minimum and Y-Maximum values. If you left-click the button with the caption **Restore Default Extents & Exit** you should see the graph shown above in **Figure 11**.

There is an especially important way of zooming in. You can do this by using the mouse to draw a rectangle about the region you wish to make the new window. First, hold down the Control key (**Ctrl**) on your keyboard, and then right-click the mouse at the upper-left corner and while continuing to hold down the right mouse button, move the mouse to the lower-right corner of the zoom region. Then let up on the right mouse button and then let up on the Control key on your keyboard. The rectangular region you see drawn on the screen when you let up on the right mouse button becomes the new graph window.

This special way of zooming in is very easy to use. We usually zoom out much more than necessary, and then we zoom back in a couple of times by using the rectangle drawing method.

Performing Small Graph Translations

In addition to zooming, sometimes you just need to move a graph a little up or down or left or right. We call such movements graph translations. These are easy to perform. Just hold down the Control key (**Ctrl**) on your keyboard and then left-click your mouse in the middle of the graph, and while continuing to hold down both the control key and the left mouse button, you can drag the mouse around a little in any direction. You should notice a hand appears on the screen and the graph should appear to move with the hand.

Let up on the left mouse button and then the control key to fix the graph in its current position. Just remember **Left Mouse + (Ctrl key)** means translate the graph while **Right Mouse + (Ctrl key)** means zoom in on the graph.

Changing Colors

There are two menu items that you can use to change colors in the program. There are line colors and there are fill colors. See the **Options** menu items labeled **Line Colors and Style** and **Fill Area Colors**.

Labeling Graphs and the Drawing Extras

LPG is not a graphics editor. However, it does have a feature that allows you to add graphic elements to any graph to make that graph more presentable. As an example, the graph below is rather plain as it only shows the region of feasible solutions to a problem.

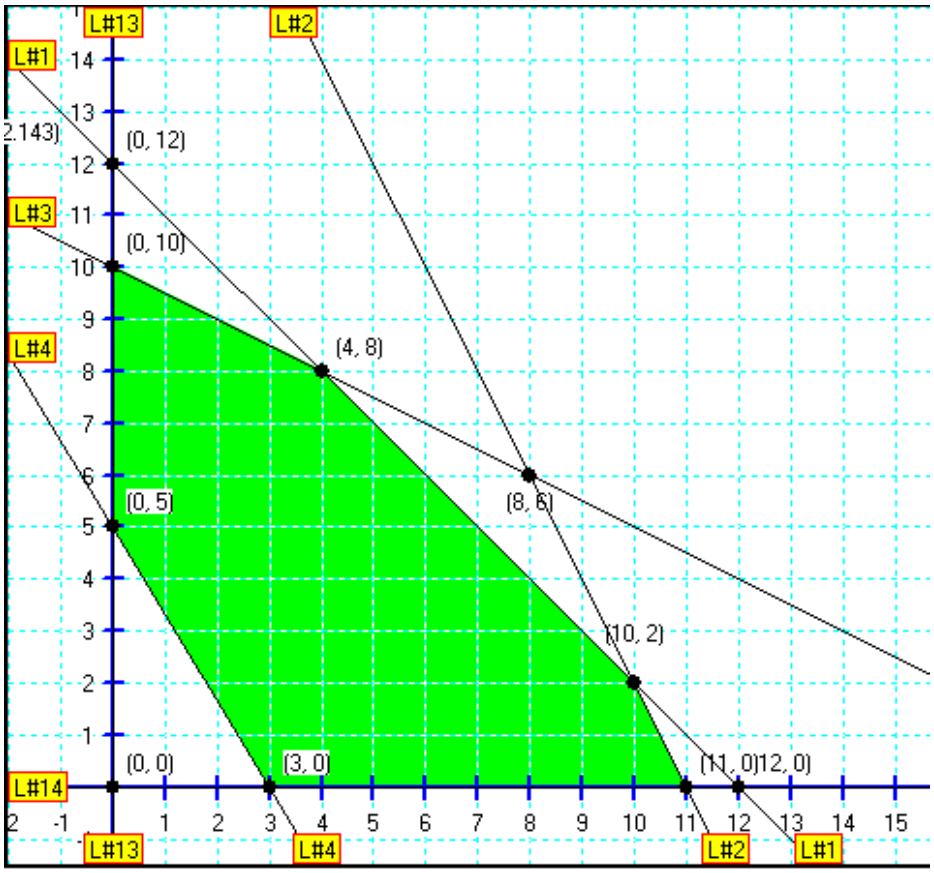


Figure 14. A plain graph showing the region of feasible solutions.

If we wanted to add comments about the maximum and minimum points or otherwise label other parts of the graph we could show the following enhancements to the graph.

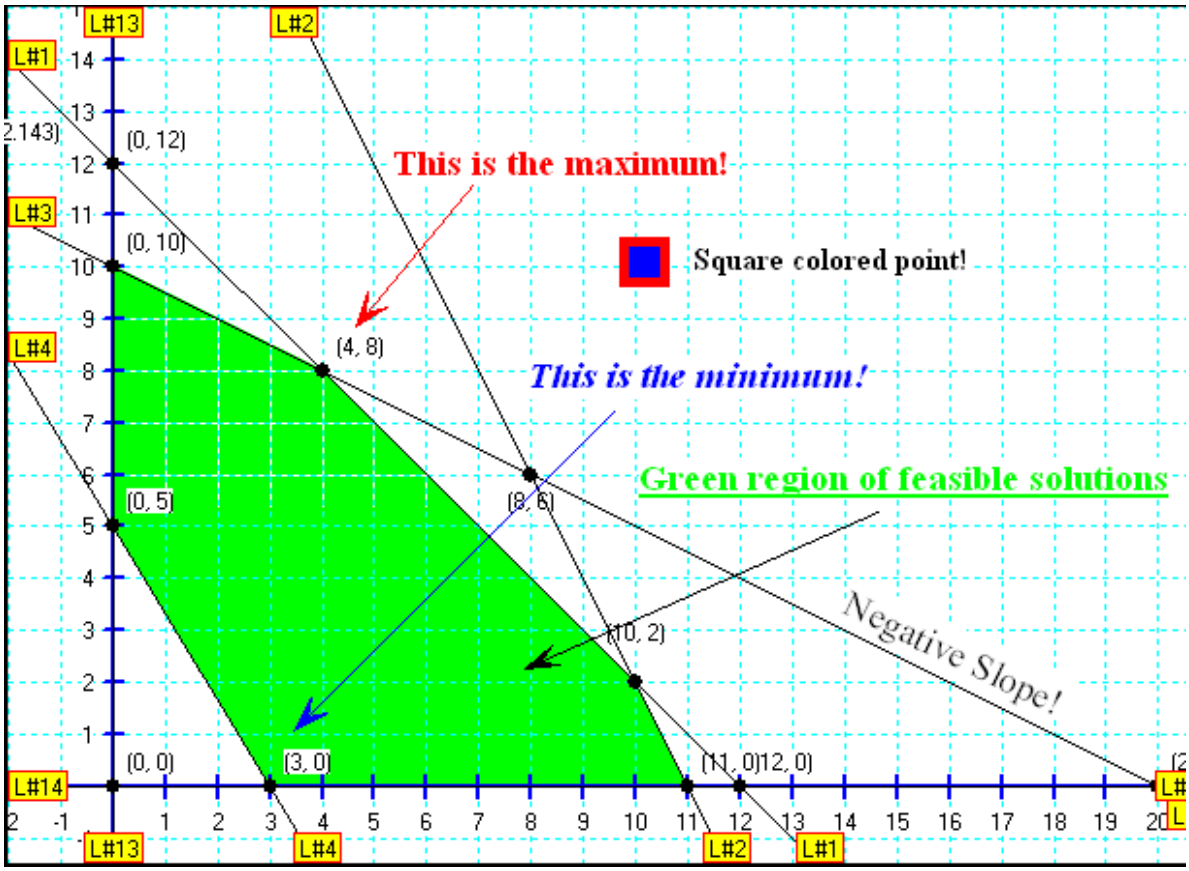


Figure 15. A graph with extra labels and extra information.

These extra graph parts were added by choosing the menu item **Options | Drawing Extras…**. This opens a dialog box that has three tabbed pages for defining extra points, and extra lines and extra text labels that can be added to a graph. In the **Figure 15** above we have added colored lines with arrows that label the maximum and minimum points and we added some text labels and a single square isolated point. The regular help file contains the technical details of how to add graphic elements like this. You just need to know that **LPG** has the ability to add titles and legends and labels to any graph and those elements can all be saved in a separate text file. Later, you can load these elements to make them appear in your graph.

Figure 16 below shows several possible ways of drawing text at different angles, and it shows various line and point styles using several colors. So if you really want to add extra parts to a graph it is possible to do so in many different ways.

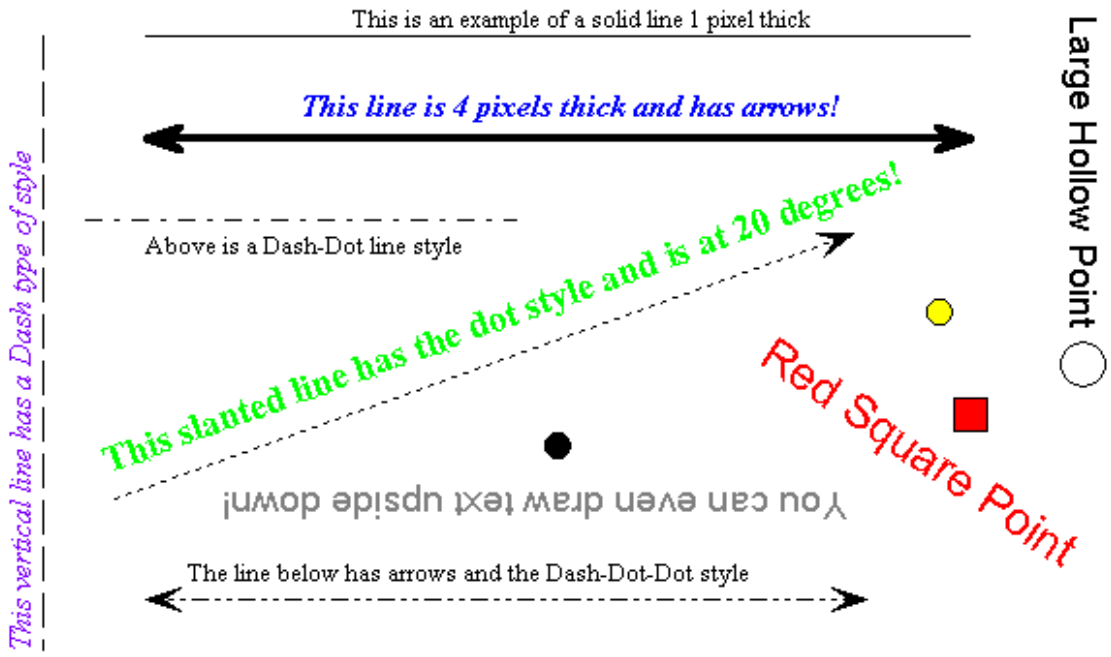


Figure 16. A sampling of the extra kinds of graphic labels, points, and lines.

Saving The Current Graph and Problem

At any time you can use the **File** menu item to save the current graph. When you save the current linear programming problem you save it in a text file. As an example, when the graph in **Figure 11** is saved the following shows the contents of the saved file. There are over 120 different values that get saved as is shown in the list below.

Most users will never need to know the contents of this file. But if you are so inclined, you can edit the right side of each = in the file using any word processor that works with ASCII text files. The NOTEPAD editor is one example. Once you understand the file format you can even create example graphs by just editing text. In fact, you don't have to edit or use all the lines that are shown below. You only need to enter the constraints you use, enter the two objective function coefficients and the max/min objective values and you may want to set the four **World Graph** XY max/min values.

```
LP.Constraint[1].A = 5
LP.Constraint[1].B = 6
LP.Constraint[1].C = 72
LP.Constraint[1].Less Than = True
LP.Constraint[1].Active = True
LP.Constraint[2].A = -2
LP.Constraint[2].B = 3
LP.Constraint[2].C = 9
LP.Constraint[2].Less Than = True
LP.Constraint[2].Active = True
```

```
LP.Constraint[3].A = 1
LP.Constraint[3].B = 3
LP.Constraint[3].C = 18
LP.Constraint[3].Less Than = False
LP.Constraint[3].Active = True
LP.Constraint[4].A = 1
LP.Constraint[4].B = 1
LP.Constraint[4].C = 1
LP.Constraint[4].Less Than = True
LP.Constraint[4].Active = False
LP.Constraint[5].A = 1
LP.Constraint[5].B = 1
LP.Constraint[5].C = 1
LP.Constraint[5].Less Than = True
LP.Constraint[5].Active = False
LP.Constraint[6].A = 1
LP.Constraint[6].B = 1
LP.Constraint[6].C = 1
LP.Constraint[6].Less Than = True
LP.Constraint[6].Active = False
LP.Constraint[7].A = 1
LP.Constraint[7].B = 1
LP.Constraint[7].C = 1
LP.Constraint[7].Less Than = True
LP.Constraint[7].Active = False
LP.Constraint[8].A = 1
LP.Constraint[8].B = 1
LP.Constraint[8].C = 1
LP.Constraint[8].Less Than = True
LP.Constraint[8].Active = False
LP.Constraint[9].A = 1
LP.Constraint[9].B = 1
LP.Constraint[9].C = 1
LP.Constraint[9].Less Than = True
LP.Constraint[9].Active = False
LP.Constraint[10].A = 1
LP.Constraint[10].B = 1
LP.Constraint[10].C = 1
LP.Constraint[10].Less Than = True
LP.Constraint[10].Active = False
LP.Constraint[11].A = 1
LP.Constraint[11].B = 1
LP.Constraint[11].C = 1
LP.Constraint[11].Less Than = True
LP.Constraint[11].Active = False
LP.Constraint[12].A = 1
LP.Constraint[12].B = 1
LP.Constraint[12].C = 1
```

```
LP.Constraint[12].Less Than = True
LP.Constraint[12].Active = False
LP.Constraint[13].A = 1
LP.Constraint[13].B = 0
LP.Constraint[13].C = 0
LP.Constraint[13].Less Than = False
LP.Constraint[13].Active = True
LP.Constraint[14].A = 0
LP.Constraint[14].B = 1
LP.Constraint[14].C = 0
LP.Constraint[14].Less Than = False
LP.Constraint[14].Active = True
LP.ObjectiveFunction.A = 2
LP.ObjectiveFunction.B = 3
LP.Maximize Objective Function = True
LP.Minimize Objective Function = True
WorldGraph.WorldXMinimum = -2
WorldGraph.WorldXMaximum = 21.1
WorldGraph.WorldYMinimum = -1.5
WorldGraph.WorldYMaximum = 15
WorldGraph.WorldOriginX = 0
WorldGraph.WorldOriginY = 0
WorldGraph.WorldXScale = 1
WorldGraph.WorldYScale = 1
WorldGraph.PrintXScaleValues = TRUE
WorldGraph.PrintYScaleValues = TRUE
WorldGraph.ZeroBasedScales = TRUE
WorldGraph.ZeroBasedGrid = TRUE
WorldGraph.HorizontalGridSpacing = 1
WorldGraph.VerticalGridSpacing = 1
WorldGraph.GridIndex = 5
WorldGraph.TrueAspect = TRUE
WorldGraph.BorderThickness = 2
WorldGraph.BackgroundColor = White
WorldGraph.FeasibleInteriorColor = Lime
WorldGraph.LineBorderColor = Red
WorldGraph.LineLabelColor = Yellow
WorldGraph.ConnectedCurve = TRUE
WorldGraph.SignificantDigits = 5
WorldGraph.DecimalPlaces = 5
WorldGraph.Numeric Format = 0
WorldGraph.Scale SignificantDigits = 3
WorldGraph.Scale DecimalPlaces = 1
WorldGraph.Scale Numeric Format = 0
WorldGraph.PrintXYExtentValues = FALSE
WorldGraph.CurvePen.PenColor = Black
WorldGraph.CurvePen.PenWidth = 1
WorldGraph.XYAxesPen.PenColor = Blue
```

```
WorldGraph.XYAxesPen.PenWidth = 2
WorldGraph.HorzVertGridPen.PenColor = Aqua
WorldGraph.HorzVertGridPen.PenWidth = 1
WorldGraph.Feasible Brush Style = 0
WorldGraph.Font Name = MS Sans Serif
WorldGraph.Font Point Size = 8
WorldGraph.Font Color = Black
WorldGraph.Font Bold Style = FALSE
WorldGraph.Font Italic Style = FALSE
WorldGraph.Font Underline Style = FALSE
ALIGN.[2,1] @ North; Stretch = 2
ALIGN.[3,1] @ North-East; Stretch = 1
ALIGN.[3,2] @ South; Stretch = 1
ALIGN.[13,1] @ North-East; Stretch = 1
ALIGN.[13,2] @ South-East; Stretch = 1
ALIGN.[13,3] @ North-East; Stretch = 1
ALIGN.[14,1] @ North-East; Stretch = 1
ALIGN.[14,2] @ North-East; Stretch = 1
ALIGN.[14,3] @ North-East; Stretch = 1
ALIGN.[14,13] @ North-East; Stretch = 1
```

Re-loading A Saved Graph

After saving any graph in a text file that you name, you can later re-load that same file. In fact, the **LPG** program extends the **File** menu to show the last six files you either saved or loaded. So you can quickly re-load a file by selecting the filename from the bottom of the **File** menu. Otherwise you can use the regular **File Open** function to re-load a graph.

Setting Up the Help System

The **LPG** program uses **Adobe Acrobat PDF files** for its help system. **PDF** stands for Portable Document Format. The Adobe company makes a free program called **Adobe Acrobat Reader** that can be used to display and print PDF files. Adobe also makes a program called **Acrobat Professional** that is not free that can be used to edit, display, and print PDF files. You must at least install the free Adobe Acrobat Reader on your computer to have the help system of **LPG** properly open any PDF help file. Just go to **www.adobe.com** to locate and download the free **Acrobat Reader** program.

When **LPG** is installed on your computer one of the special files it has is named **AdobeReader.ini**. This is just a text file that you can edit with the Windows text editor program named **NotePad**. In a new installation the contents of this file is two lines that might appear as:

```
C:\Program Files\Adobe\Reader 10.0\Reader\AcroRd32.exe
C:\Program Files (x86)\Adobe\Reader 10.0\Reader\AcroRd32.exe
```

These two lines just represent the two most probable locations for the program known as **Adobe Acrobat Reader**. You can insert more than just these two lines in the file **AdobeReader.ini** and the program will try all such lines to determine if any line is valid and refers to an Adobe product that will read PDF files.

Once you have the program **Adobe Acrobat Reader** installed on your computer, you can edit the contents of the **AdobeReader.ini** file to make it contain a path to the executable program file that is the Acrobat Reader program. The above two lines are just example paths. However, the true path that is needed depends on your individual computer system and how and where the **Acrobat Reader** program is installed. If you have **Acrobat Professional**, you could also make a path to that program.

As long as either **Acrobat Reader** or **Acrobat Professional** is installed on your system you will be able to display and print PDF files. If you don't have a correct path in **AdobeReader.ini** then the program will still be able to open the help files, but it won't automatically jump to a designated help topic in a help file until you make the correct path to one of the Adobe programs in the **AdobeReader.ini** file.

Conclusion

This concludes the tutorial for the **LPG** program. There is more information contained in two other help files, but by now you should be able to enter any problem and create a reasonable graph for that problem.

You should also know there is a whole suite of other programs related to math and computer science at the author's homepage. See **https://sites.google.com/site/johnkennedyshome/home**