Geometry of Simplex Lab

Henry Robbins (Summer 2020)

Objectives

- Understand the geometry of a linear program's feasible region.
- Use isoprofit lines and planes to solve 2D and 3D LPs graphically.
- Identify the most limiting constraint in an iteration of simplex both algebraically and geometrically.
- Identify the geometric features corresponding to dictionaries.
- Describe the geometrical decision made at each iteration of simplex.

Review

Recall, linear programs (LPs) have three main components: decision variables, constraints, and an objective function. The goal of linear programming is to find a **feasible solution** (a solution satisfying every constraint) with the best objective value. The set of feasible solutions form a **feasible region**. In lecture, we learned about isoprofit lines. For every objective value, we can define an isoprofit line. Isoprofit lines have the property that two solutions on the same line have the same objective value and all isoprofit lines are parallel.

In the first part of the lab, we will use a Python package called GILP to solve linear programs graphically. We introduce the package now.

GILP

If you are running this file in a Google Colab Notebook, uncomment the following line and run it. Otherwise, you can ignore it.

In [1]:
```
#!pip install gilp
```

In [2]:
```
# Imports -- don't forget to run this cell
import gilp
import numpy as np
```

This lab uses default LPs built in to GILP. We import them below.

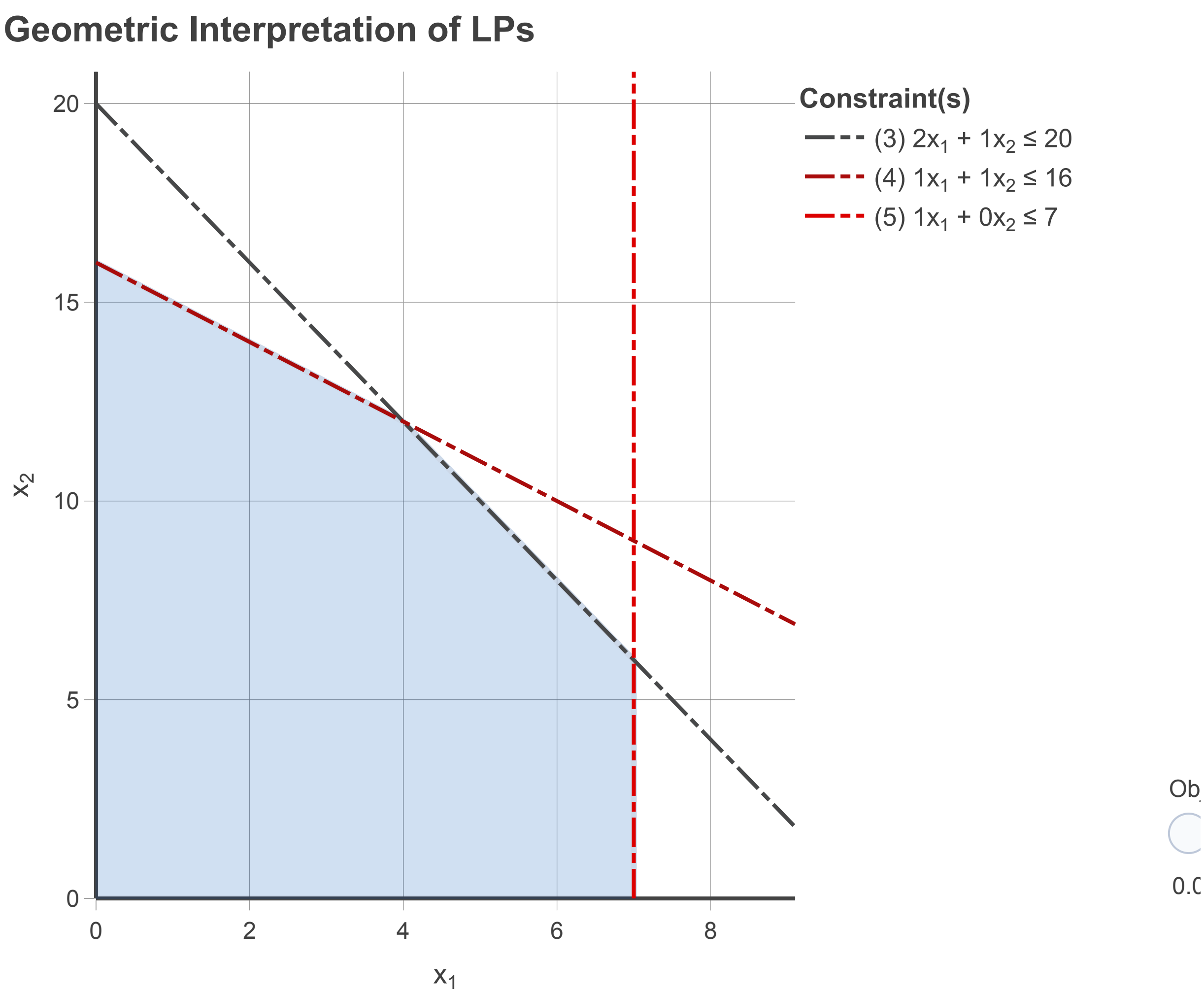
We access the LP examples using `gilp.examples.NAME` where `NAME` is the name of the example LP. For example, consider:
$$\begin{align*} \max \quad & 5x_1+3x_2\\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 20 \\ \quad & 1x_1 + 1x_2 \leq 16 \\ \quad & 1x_1 + 0x_2 \leq 7 \\ \quad & x_1, x_2 \geq 0 \\ \end{align*}$$

This example LP is called `ALL_INTEGER_2D_LP`. We assign this LP to the variable `lp` below.

In [3]:
```
lp = gilp.examples.ALL_INTEGER_2D_LP
```

We can visualize this LP using a function called `lp_visual()`. The function `lp_visual()` takes an LP and returns a visualization. We then use the `.show()` function to display the visualiazation.

In [4]:
```
gilp.lp_visual(lp).show()
```



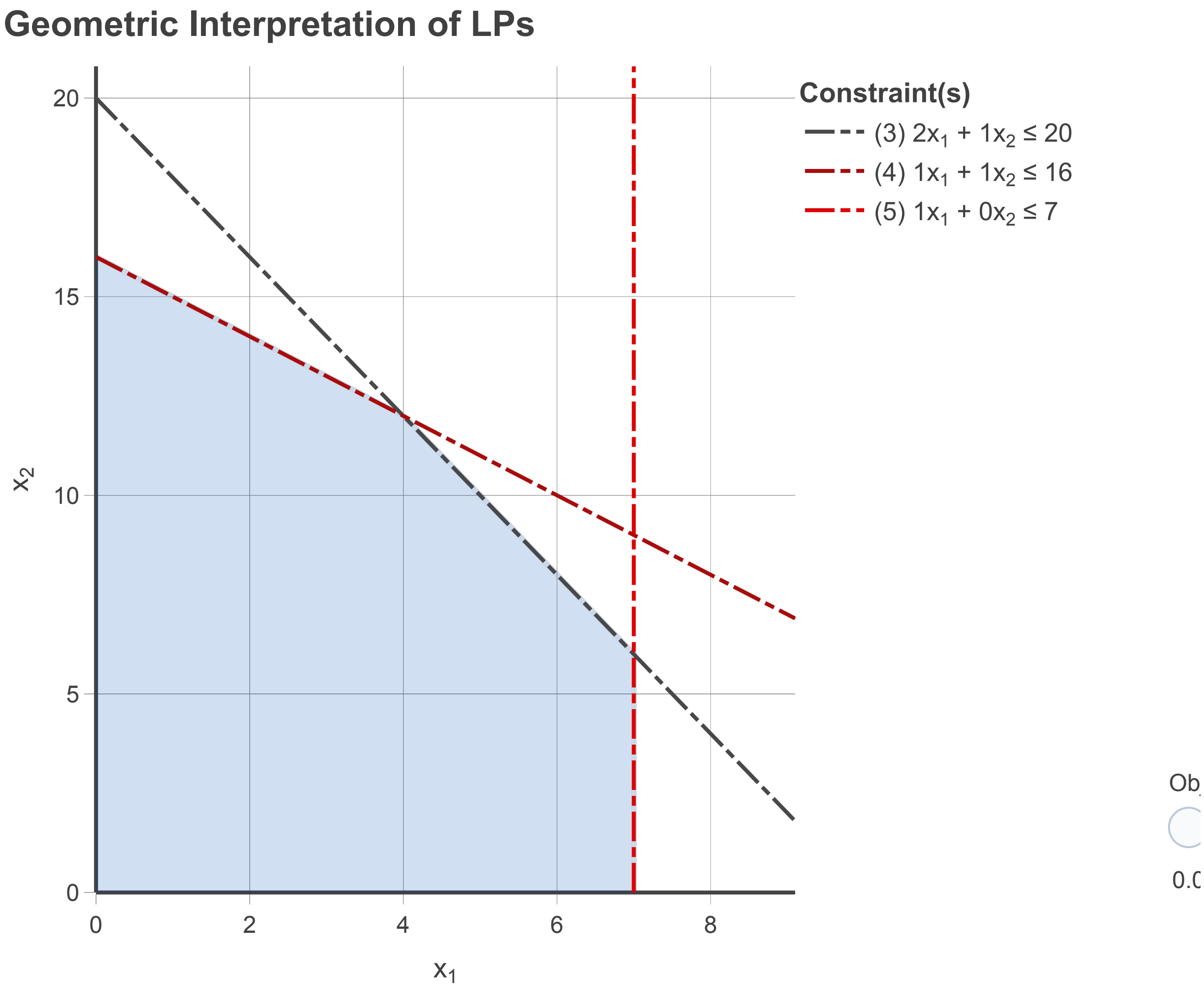
On the left, you can see a coordinate plane where the x-axis corresponds to the value of x_1 and the y-axis corresponds to the value of x_2. The region shaded blue is the feasible region. Along the perimeter of the feasible region, you can see points where two edges come to a "corner". You can hover over these **corner points** to see information about them. Only some of the information in the hover box will be relevant for Part I. The first two values of **BFS** represent the values of x_1 and x_2 respectively and **Obj** is the objective value. For example, the upper left corner point has solution $x_1 = 0$ and $x_2 = 16$ with objective value 48. The dashed lines represent the constraints. You can click on the constraints in the legend to mute and un-mute them. Note this does not alter the LP; it just changes visibility. Lastly, the objective slider allows you to see the isoprofit line for a range of objective values.

Part I: Solving Linear Programs Graphically

Let's use GILP to solve the following LP graphically:
$$\begin{align*} \max \quad & 5x_1+3x_2\\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 20 \\ \quad & 1x_1 + 1x_2 \leq 16 \\ \quad & 1x_1 + 0x_2 \leq 7 \\ \quad & x_1, x_2 \geq 0 \\ \end{align*}$$

Recall, this LP is called `ALL_INTEGER_2D_LP`.

In [5]:
```
lp = gilp.examples.ALL_INTEGER_2D_LP # get LP example
gilp.lp_visual(lp).show() # visualize it
```



Q1: How can you use isoprofit lines to solve LPs graphically?

A:

Q2: Use the objective slider to solve this LP graphically. Give an optimal solution and objective value. Argue why it is optimal. (Hint: The objective slider shows the isoprofit line (in red) for some objective value.)

A:

Q3: Plug your solution from **Q2** back into the LP and verify that each constraint is satisfied (don't forget non-negativity constraints!) and the objective value is as expected. Show your work.

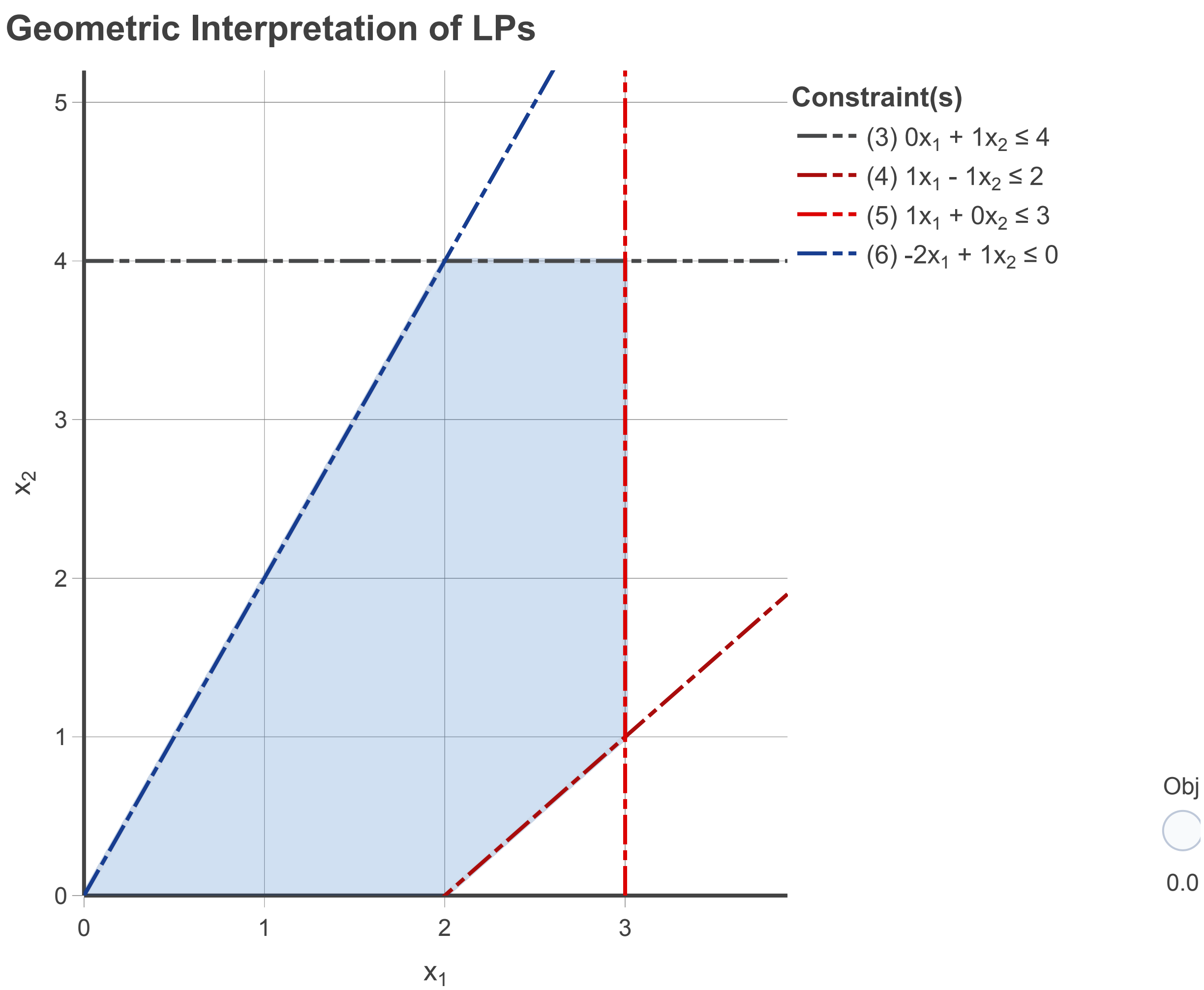
A:

Q3.5: Two of the constraints (lines) are satisfied with equality for the optimal solution. Which two are these? How would just knowing that fact allow you find the optimal solution?

A:

Let's try another! This LP is called `DEGENERATE_FIN_2D_LP` .
$$\begin{align*} \max \quad & 1x_1+2x_2\\ \text{s.t.} \quad & 0x_1 + 1x_2 \leq 4 \\ \quad & 1x_1 - 1x_2 \leq 2 \\ \quad & 1x_1 + 0x_2 \leq 3 \\ \quad & -2x_1 + 1x_2 \leq 0 \\ \quad & x_1, x_2 \geq 0 \\ \end{align*}$$

In [6]:
```
lp = gilp.examples.DEGENERATE_FIN_2D_LP # get LP example
gilp.lp_visual(lp).show() # visualize it
```



Q4: Use the objective slider to solve the `DEGENERATE_FIN_2D_LP` LP graphically. Give an optimal solution and objective value. (Hint: The objective slider shows the isoprofit line (in red) for some objective value.)

A:

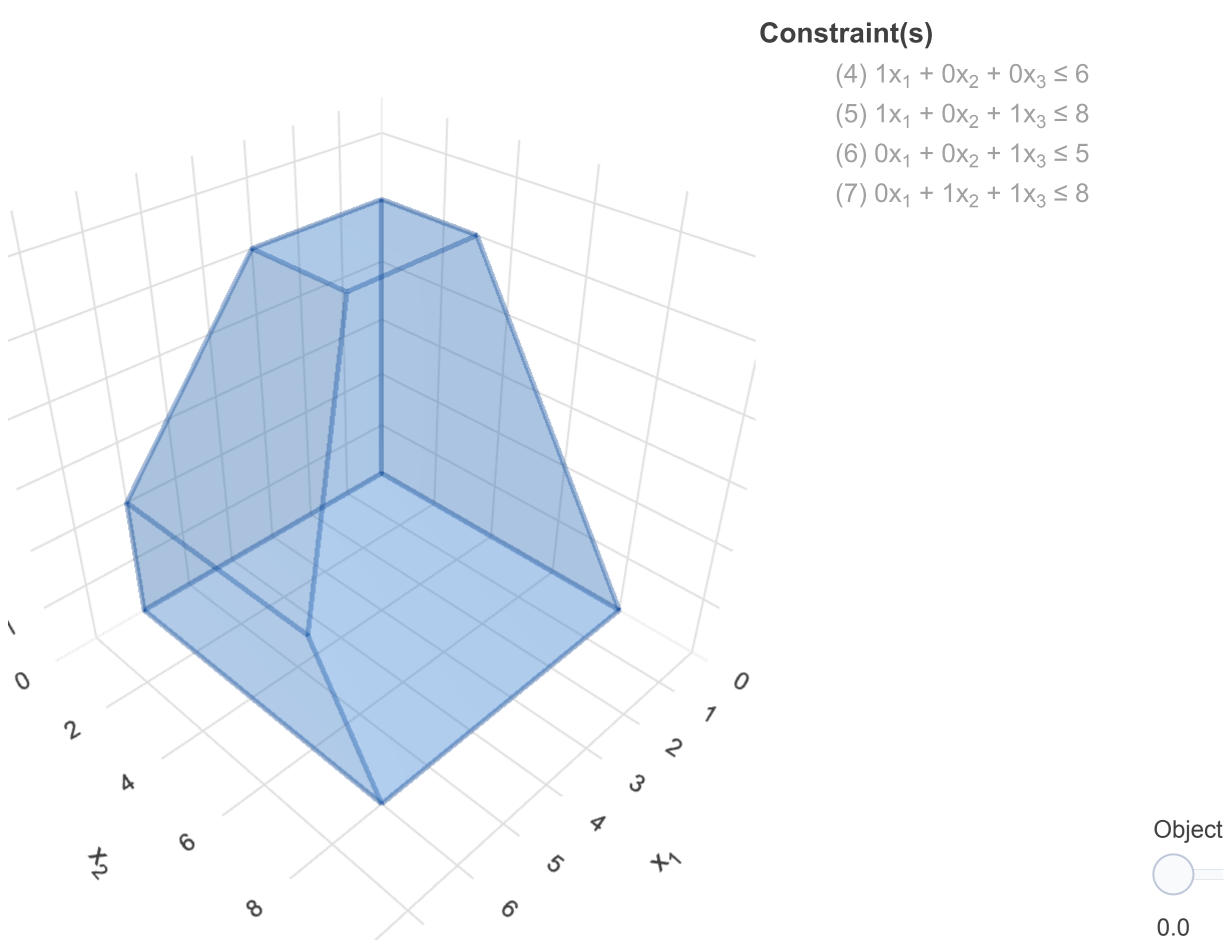
You should now be comfortable solving linear programs with two decision variables graphically. In this case, each constraint is a line representing an inequality. These inequalites define a shaded region in the coordinate plane which is our feasible region. Lastly, the isoprofits are parallel lines. To find an optimal solution, we just increase the objective value while the corresponding isoprofit line still intersects the 2D feasible region.

Now, we will try to wrap our head around an LP with three decision variables! Similar to before, we can plot solutions to a 3D LP on a plot with 3 axes. Here, the x-axis corresponds to the value of x_1 and the y-axis corresponds to the value of x_2 as before. Furthermore, the z-axis corresponds to the value of x_3. Now, constraints are *planes* representing an inequality. These inequality planes define a 3D shaded region which is our feasible region. The isoprofits are isoprofit *planes* which are parallel. To find an optimal solution, we just increase the objective value while the corresponding isoprofit plane still intersects the 3D feasible region. Let us look at an example.

This LP is called `ALL_INTEGER_3D_LP` :
$$\begin{align*} \max \quad & 1x_1+2x_2+4x_3\\ \text{s.t.} \quad & 1x_1 + 0x_2 + 0x_3 \leq 6 \\ \quad & 1x_1 + 0x_2 + 1x_3 \leq 8 \\ \quad & 0x_1 + 0x_2 + 1x_3 \leq 5 \\ \quad & 0x_1 + 1x_2 + 1x_3 \leq 8 \\ \quad & x_1, x_2 \geq 0 \\ \end{align*}$$

In [7]:
```
lp = gilp.examples.ALL_INTEGER_3D_LP # get LP example
gilp.lp_visual(lp).show() # visualize it
```

Geometric Interpretation of LPs



The 3D feasible region is shown on the left. Hold and drag the mouse to examine it from different angles. Next, click on a constraint to un-mute it. Each constraint is a gray plane in 3D space. Un-mute the constraints one by one to see how they define the 3D feasible region. Move the objective slider to see the isoprofit planes. The isoprofit plane is light gray and the intersection with the feasible region is shown in red. Like the 2D visualization, you can hover over corner points to see information about that point.

Q5: Use the objective slider to solve this LP graphically. Give an optimal solution and objective value. (Hint: The objective slider shows the isoprofit plane for some objective value in light gray and the intersection with the feasible region in red.)

A:

When it comes to LPs with 4 or more decision variables, our graphical approaches fail. We need to find a different way to solve linear programs of this size.

Part II: The Simplex Algorithm for Solving LPs

Dictionary Form LP

First, let's answer some guiding questions that will help to motivate the simplex algorithm.

Q6: Does there exist a unique way to write any given inequality constraint? If so, explain why each constraint can only be written one way. Otherwise, give 2 ways of writing the same inequality constraint.

A: $x_1 + 4 \leq x_2$ can be rewritten as $x_2 - 4 \geq x_1$

Q7: Consider the following two constraints: $2x_1 + 1x_2 \leq 20$ and $2x_1 + 1x_2 + x_3 = 20$ where all x are nonnegative. Are these the same constraint? Why? (This question is tricky!)

A: Yes, these are the same constraint. This is because if $2x_1 + 1x_2 \leq 20$ implies that there's some nonnegative constant y that needs to be added in order for the LHS and RHS to be equal. In this case, x_3 is our y.

Q8: Based on your answers to **Q6** and **Q7**, do you think there exists a unique way to write any given LP?

A: No, there should always be another way to rewrite a given LP.

You should have found that there are many ways to write some LP. This begs a new question: are some ways of writing an LP harder or easier to solve than others? Consider the following LP:
$$\begin{align*} \max \quad & 56 - 2x_3 - 1x_4\\ \text{s.t.} \quad & x_1 = 4 - 1x_3 + 1x_4 \\ \quad & x_2 = 12 + 1x_3 - 2x_4 \\ \quad & x_5 = 3 + 1x_3 - 1x_4 \\ \quad & x_1, x_2, x_3, x_4, x_5 \geq 0 \\ \end{align*}$$

Q9: Just by looking at this LP, can you give an optimal solution and its objective value. If so, explain what property of the LP allows you to do this. (Hint: Look at the objective function)

A: The coefficients of the decision variables are all negative, but the decision variables themselves are all nonnegative. Therefore, the decision variables can't contribute to the constant in the objective function; they can only take away or do nothing. Clearly the optimal solution requires the decision variables to not take away from 56, and if they can't add (due to negative coefficients), they must only be able to do nothing. Thus, the optimal solution is to make x_3 and x_4 = 0. Going down the constraints, $x_1 = 4-1*0+1*0 = 4, x_2 = 12, x_5 = 3$. The final objective value is 56.

The LP above is the same as `ALL_INTEGER_2D_LP` just rewritten in a different way! This rewitten form (which we found is easier to solve) was found using the simplex algorithm. At its core, the simplex algorithm strategically rewrites an LP until it is in a form that is "easy" to solve.

The simplex algorithm relies on an LP being in **dictionary form**. Recall the following properties of an LP in dictionary form:

- All constraints are equality constraints
- All variables are constrained to be nonnegative
- Each variable only appears on the left-hand side (LHS) or the right-hand side (RHS) of the constraints (not both)
- Each constraint has a unique variable on the LHS
- The objective function is in terms of the variables that appear on the RHS of the constraints only.
- All constants on the RHS of the constraints are nonnegative

Q10: Rewrite the example LP `ALL_INTEGER_2D_LP` in dictionary form. Show your steps!
$$\begin{align*} \max \quad & 5x_1+3x_2\\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 20 \\ \quad & 1x_1 + 1x_2 \leq 16 \\ \quad & 1x_1 + 0x_2 \leq 7 \\ \quad & x_1, x_2 \geq 0 \\ \end{align*}$$

A:
$$\begin{align*} \max \quad & 5x_1+3x_2\\ \text{s.t.} \quad & 2x_1 + 1x_2 + x_3 = 20 \\ \quad & 1x_1 + 1x_2 + x_4 = 16 \\ \quad & 1x_1 + 0x_2 + x_5 = 7 \\ \quad & x_1, x_2, x_3, x_4, x_5 \geq 0 \\ \end{align*}$$$$\begin{align*} \max \quad & 5x_1+3x_2\\ \text{s.t.} \quad & x_3 = 20 - 2x_1 - 1x_2\\ \quad & x_4 = 16 - 1x_1 - 1x_2\\ \quad & x_5 = 7 - 1x_1 - 0x_2 \\ \quad & x_1, x_2, x_3, x_4, x_5 \geq 0 \\ \end{align*}$$

Most Limiting Constraint

Once our LP is in dictionary form, we can run the simplex algorithm! In every iteration of the simplex algorithm, we will take an LP in dictionary form and strategically rewrite it in a new dictionary form. Note: it is important to realize that rewriting the LP **does not** change the LP's feasible region. Let us examine an iteration of simplex on a new LP.
$$\begin{align*} \max \quad & 5x_1+3x_2\\ \text{s.t.} \quad & 1x_1 + 0x_2 \leq 4 \\ \quad & 0x_1 + 1x_2 \leq 6 \\ \quad & 2x_1 + 1x_2 \leq 9 \\ \quad & 3x_1 + 2x_2 \leq 15 \\ \quad & x_1, x_2 \geq 0 \\ \end{align*}$$

Q11: Is this LP in dictionary form? If not, rewrite this LP in dictionary form.

A:
$$\begin{align*} \max \quad & 5x_1+3x_2\\ \text{s.t.} \quad & x_3 = 4 - 1x_1 - 0x_2\\ \quad & x_4 = 6 - 0x_1 - 1x_2 \\ \quad & x_5 = 9 - 2x_1 - 1x_2 \\ \quad & x_6 = 15 - 3x_1 - 2x_2 \\ \quad & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \\ \end{align*}$$

Q12: Recall from **Q9** how you found a feasible solution (which we argued to be optimal) just by looking at the LP. Using this same stratagy, look at the LP above and give a feasible solution and its objective value for this LP. Describe how you found this feasible solution. Is it optimal? Why?

A: Need to set $x_1 = 4$ and $x_2 = 1$. This solution is optimal because you can't increase x_1 anymore than 4 without making x_3, x_4, x_5, x_6 all nonnegative, and you want to maximize x_1 first because you get the most bang for your buck (coefficient of 5 vs 3). Then, because of the x_5 constraint the most you can maximize x_2 is to set it equal to 1.

From **Q12** we see that every dictionary form LP has a corresponding feasible solution. Furthermore, there are positive coefficents in the objective function. Hence, we can increase the objective value by increasing the corresponding variable. In our example, both x_1 and x_2 have positive coefficents in the objective function. Let us choose to increase x_1.

Q13: What do we have to be careful about when increasing x_1?

A: x_1 can't exceed 4 because of the x_3 constraint and the x_5 constraint, so we have to make sure that it doesn't make the constraints it's a part of negative.

Q14: After choosing a variable to increase, we must determine the most limiting constraint. Let us look at the first constraint $x_3 = 4 - 1x_1 - 0x_2$. How much can x_1 increase? (Hint: what does a dictionary form LP require about the constant on the RHS of constraints?)

A: x_1 can only increase to at most 4.

Q15: Like in **Q14**, determine how much each constraint limits the increase in x_1 and identify the most limiting constraint.

A: 1st constraint: x_1 can only increase to at most 4. 2nd constraint: x_1 has no limit. 3rd constraint: x_1 can only increase to at most 4.5. 4th constraint: x_1 can only increase to at most 5.

Thus, the most limiting constraint is the first one.

If we increase x_1 to 4, note that x_3 will become zero. Earlier, we identified that each dictionary form has a corresponding feasible solution acheived by setting variables on the RHS (and in the objective function) to zero. Hence, since x_3 will become zero, we want to rewrite our LP such that x_3 appears on the RHS. Furthermore, since x_1 is no longer zero, it should now appear on the LHS.

Q16: Rewrite the most limiting constraint $x_3 = 4 - 1x_1 - 0x_2$ such that x_1 appears on the left and x_3 appears on the right.

A: $x_1 = 4-x_3-0x_2$

Q17: Using substitution, rewrite the LP such that x_3 appears on the RHS and x_1 appears on the LHS. (Hint: Don't forget the rule about which variables can appear in the objective function)

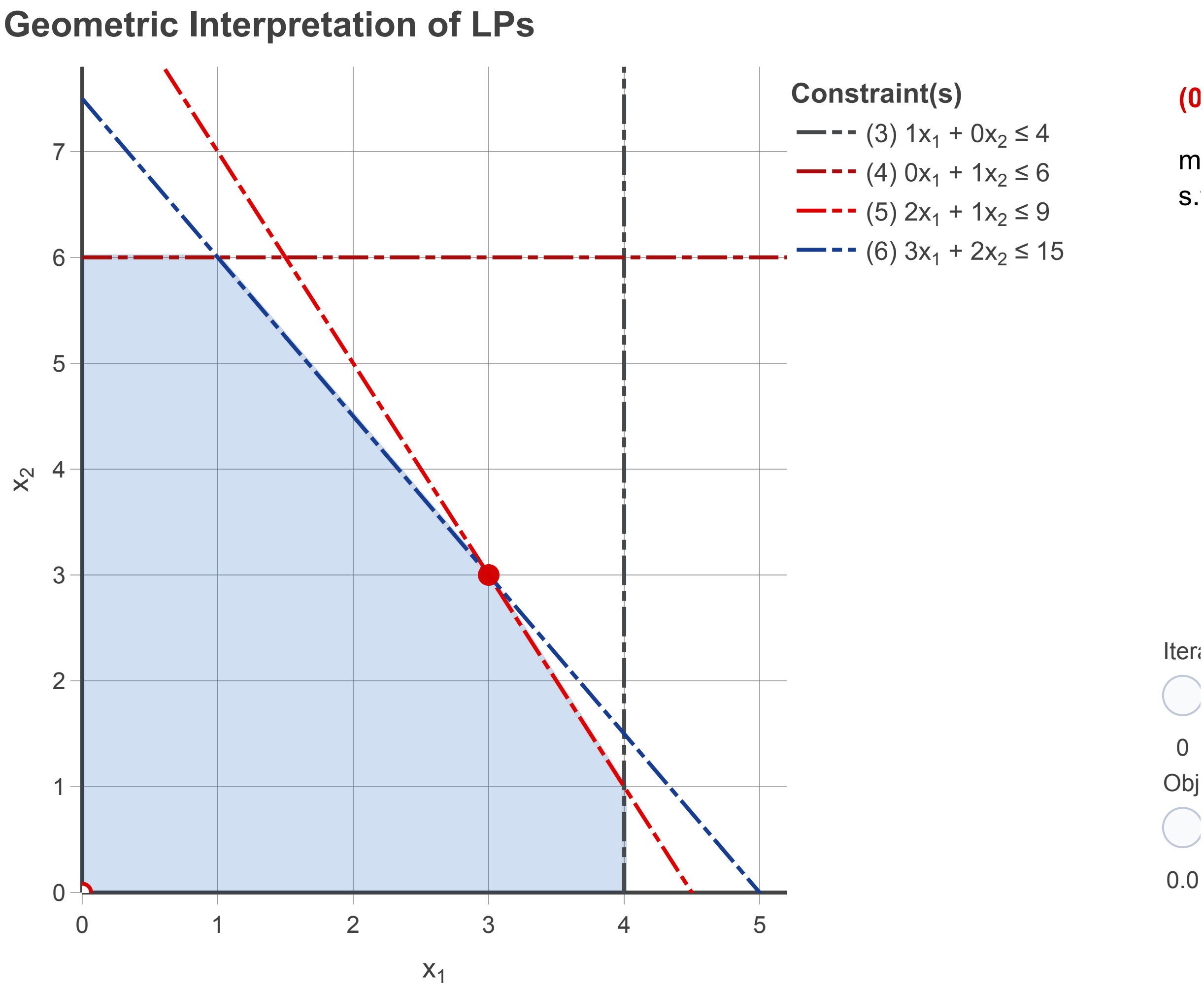
A: $$\begin{align*} \max \quad & 5(4-x_3)+3x_2 = 20-5x_3+3x_2\\ \text{s.t.} \quad & x_1 = 4-x_3-0x_2\\ \quad & x_4 = 6 - 0x_1 - 1x_2 \\ \quad & x_5 = 9 - 2x_1 - 1x_2 \\ \quad & x_6 = 15 - 3x_1 - 2x_2 \\ \quad & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \\ \end{align*}$$

Q18: We have now completed an iteration of simplex! What is the corresponding feasible solution of the new LP?

A: $$x_1 = 4, x_2 = 0, x_3 = 0, x_4 = 6, x_5 = 1, x_6 = 3$$

Now that we have seen an iteration of simplex algebraically, let's use GILP to visualize it! The LP example we have been using is called `LIMITING_CONSTRAINT_2D_LP` . To visualize simplex, we must import a function called `simplex_visual()` .

In [9]:
```
lp = gilp.examples.LIMITING_CONSTRAINT_2D_LP # get the LP example
gilp.simplex_visual(lp, initial_solution=np.array([[0],[0]])).show() # s
how the simplex visualization
```



This visualization is much the same as the previous one but we now have an addtional slider which allows you to toggle through iterations of simplex. Furthermore, the corresponding dictionary at every iteration of simplex is shown in the top right. If you toggle between two iterations, you can see the dictionary form for both the previous and next LP at the same time.

Q19: Starting from point (0,0), by how much can you increase x_1 before the point is no longer feasible? Which constraint do you *hit* first? Does this match what you found algebraically?

A: You can increase x_1 to at most 4; it hits the constraint $x_1=4$. This matches what we found algebraically.

Q20: Which variable will be the next increasing variable and why? (Hint: Look at the dictionary form LP at iteration 1)

A: x_2 would be the next increasing variable.

Q21: Visually, which constraint do you think is the most limiting constraint? How much can x_2 increase? Give the corresponding feasible solution and its objective value of the next dictionary form LP. (Hint: hover over the feasible points to see information about them.)

A: I think that the constraint (5) is the most limiting constraint because it limits how much x_2 can increase when $x_1=4$. x_2 can increase to at most 1 in this case, leaving a feasible solution of $x_1=4, x_2=1$ and an objective value of 23. This is evident in the LP, where one would set x_3 and x_5 = 0.

Q22: Move the slider to see the next iteration of simplex. Was your guess from **Q21** correct? If not, describe how your guess was wrong.

A: Yes, it was correct.

Q23: Look at the dictionary form LP after the second iteration of simplex. What is the increasing variable? Identify the most limiting constraint graphically and algebraically. Show your work and verify they are the same constraint. In addition, give the next feasible solution and its objective value.

A: Here, x_3 is the increasing variable, and the last constraint is the most limiting. This is because that's the line the objective function travels on.

Algebraically:

1st constraint caps x_3 at 4. 2nd constraint doesn't give x_3 a limit. 3rd constraint caps x_3 at 2. 4th constraint caps x_3 at 1.

Q24: Is the new feasible solution you found in **Q23** optimal? (Hint: Look at the dictionary form LP)

A: Yes, it's optimal.

Q25: In **Q21** and **Q23**, how did you determine the most limiting constraint graphically?

A: The most limiting constraint is the one that caps the increase of a certain variable.

(BONUS): In 2D, we can increase a variable until we hit a 2D line representing the most limiting constraint. What would be the analagous situation in 3D?

A: Increase the line until you hit a plane that blocks off the feasible region.

Part III: Geometrical Interpretation of the Dictionary

We have seen how the simplex algorithm transforms an LP from one dictionary form to another. Each dictionary form has a corresponding dictionary defined by the variables on the LHS of the constraints. Furthermore, each dictionary form has a corresponding feasible solution obtained by setting all non-dictionary variables to 0 and the dictionary variables to the constants on the RHS. In this section, we will explore the geometric interpretation of a dictionary.

In [11]:
```
lp = gilp.examples.ALL_INTEGER_2D_LP # get LP example
gilp.simplex_visual(lp, initial_solution=np.array([[0],[0]])).show() # v
isualize it
```

Recall, we can hover over the corner points of the feasible region. **BFS** indicates the feasible solution corresponding to that point. For example, (7,0,6,9,0) means $x_1 = 7, x_2 = 0, x_3 = 6, x_4 = 9$, and $x_5 = 0$. **B** gives the indices of the variables "being defined" in that dictionary – that is, the variables that are on the LHS of the constraints. For simplicity, we will just say these variables are *in the dictionary*. For example, if **B** = $(1,3,4)$, then x_1, x_3, and x_4 are in the dictionary. Lastly, the objective value at that point is given.

Q26: Hover over the point (7,6) where $x_1 = 7$ and $x_2 = 6$. What is the feasible solution at that point ?

A: (7, 6, 0, 3, 0)

We have a notion of *slack* for an inequality constraint. Consider the constraint $x_1 \geq 0$. A feasible solution where $x_1 = 7$ has a slack of 7 in this constraint. Consider the constraint $2x_1 + 1x_2 \leq 20$. The feasible solution with $x_1 = 7$ and $x_2 = 6$ has a slack of 0 in this constraint.

Q27: What is the slack in constraint $1x_1 + 1x_2 \leq 16$ when $x_1 = 7$ and $x_2 = 6$?

A: 3

Q28: Look at the constraint $2x_1 + 1x_2 \leq 20$. After rewriting in dictionary form, the constraint is $x_3 = 20 - 2x_1 - 1x_2$. What does x_3 represent?

A: x_3 represents the slack in the inequality.

Q29: What do you notice about the feasible solution at point (7,6) and the slack in each constraint?

A: The only constraint without slack is (5). (3) has a slack of 5, (4) has a slack of 3.

It turns out that each decision variable is really a measure of slack in some corresponding constraint!

Q30: If the slack between a constraint and a feasible solution is 0, what does that tell you about the relationship between the feasible solution and constraint geometrically?

A: The feasible solution exists on the border of that constraint.

Q31: For (7,6), which variables are **not** in the dictionary? For which constraints do they represent the slack? (Hint: The **B** in the hover box gives the indices of the variables in the dicitonary)

A: x_3 and x_5 are not in the dictionary. x_3 represents the slack for (3), x_5 represents the slack for (5).

Q32: For (7,6), what are the values of the non-dictionary variables? Using what you learned from **Q30**, what does their value tell you about the feasible solution at (7,6)?

A: The values for the nondictionary variables is 0. Their value indicates that the feasible solution at (7, 6) is optimal on that constraint line.

Q33: Look at some other corner points with this in mind. What do you find?

A: All of them are the maximum value on their constraint lines.

Now, let's look at a 3 dimensional LP!

In [12]:
```
lp = gilp.examples.ALL_INTEGER_3D_LP # get LP example
gilp.lp_visual(lp).show() # visualize it
```