Geometry of Simplex Lab

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 highest 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
```

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

In [2]:
```
from gilp import examples as ex
```

We access the LP examples using `ex.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.

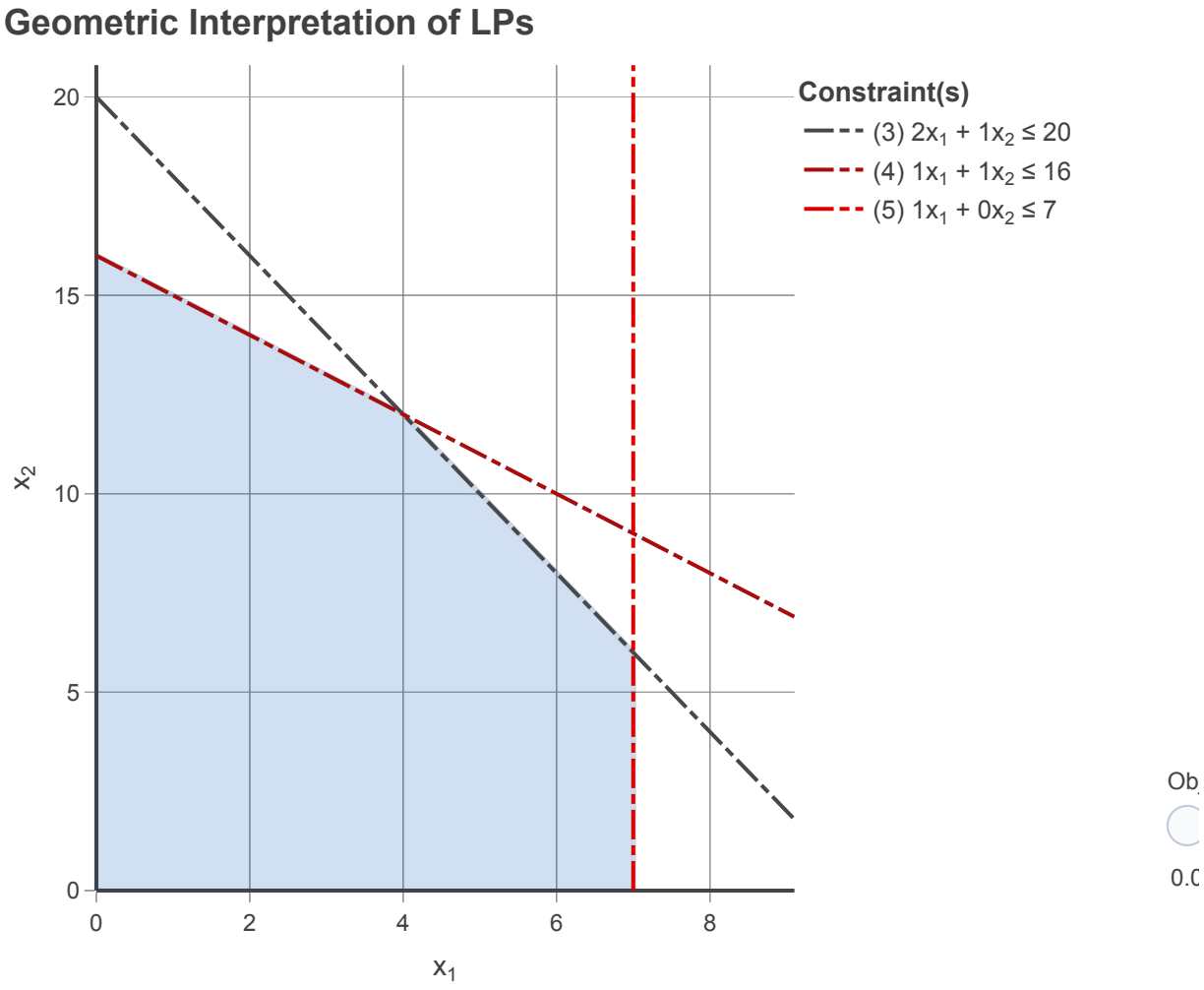
In [3]:
```
lp = ex.ALL_INTEGER_2D_LP
```

We can visualize this LP using a function called `lp_visual()`. First, we must import it.

In [4]:
```
from gilp.visualize import lp_visual
```

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

In [5]:
```
lp_visual(lp).show()
```



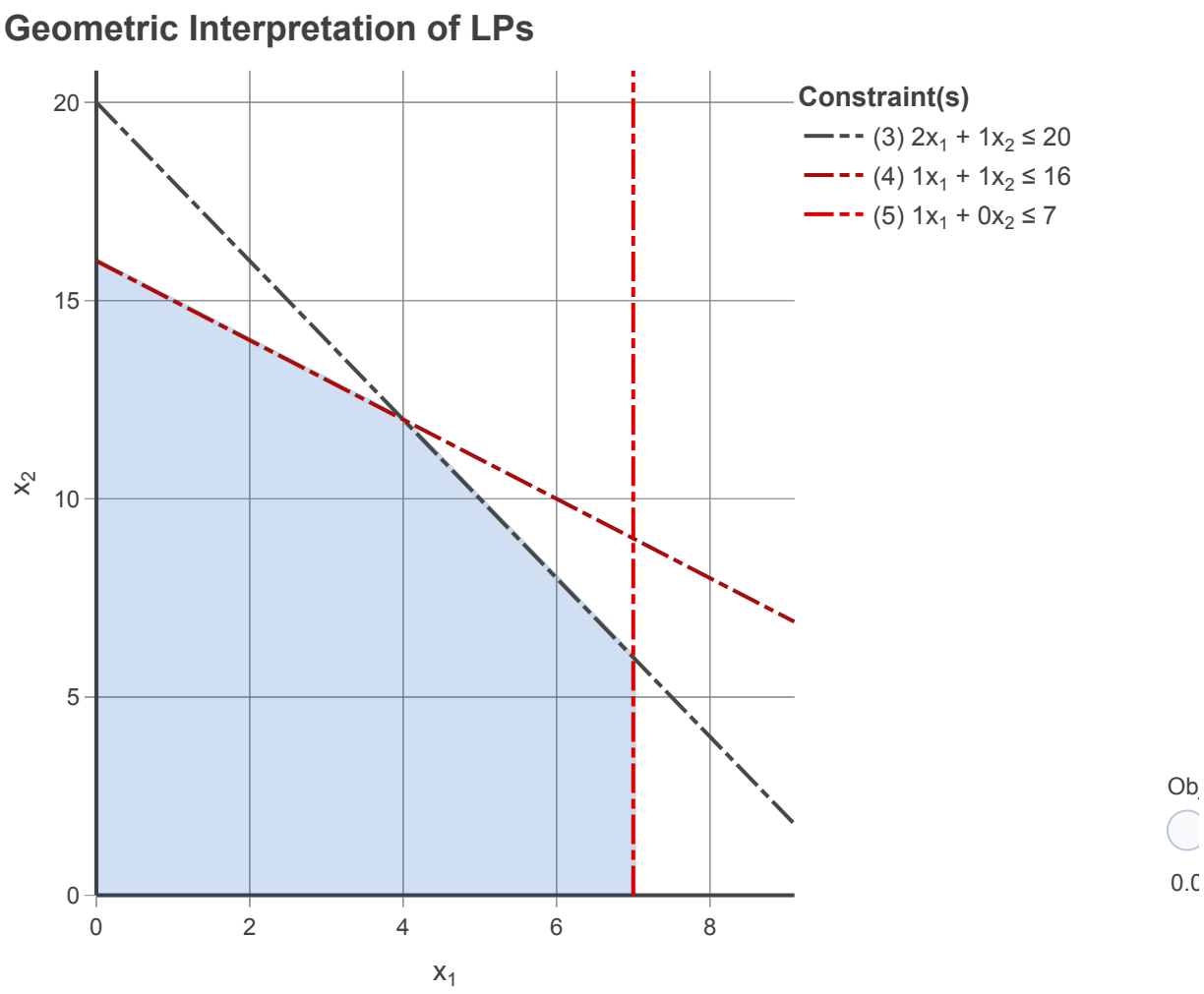
On the left, you can see a coordinate plane where the x-axis corresponds to the value of x_1 and the y-axis corresponds to the value of x_2. The region shaded blue is the feasible region. Along the perimeter of the feasible region, you can see points where two edges come to a "corner". You can hover 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 [6]:
```
lp = ex.ALL_INTEGER_2D_LP # get LP example
lp_visual(lp).show() # visualize it
```



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

A: The isoprofit line will reveal the optimal solution to the input by approching the point on the graph that indicates the optimal solution as the objective value increases.

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: The objective value is 56 and the optimal solution is x1 = 4 and x2 = 12. This is because given all of the feasible solutions in our constraints, these are the values in which the isoprofit line is the largest while still maintaining a feasible solution, therefore it is optimal.

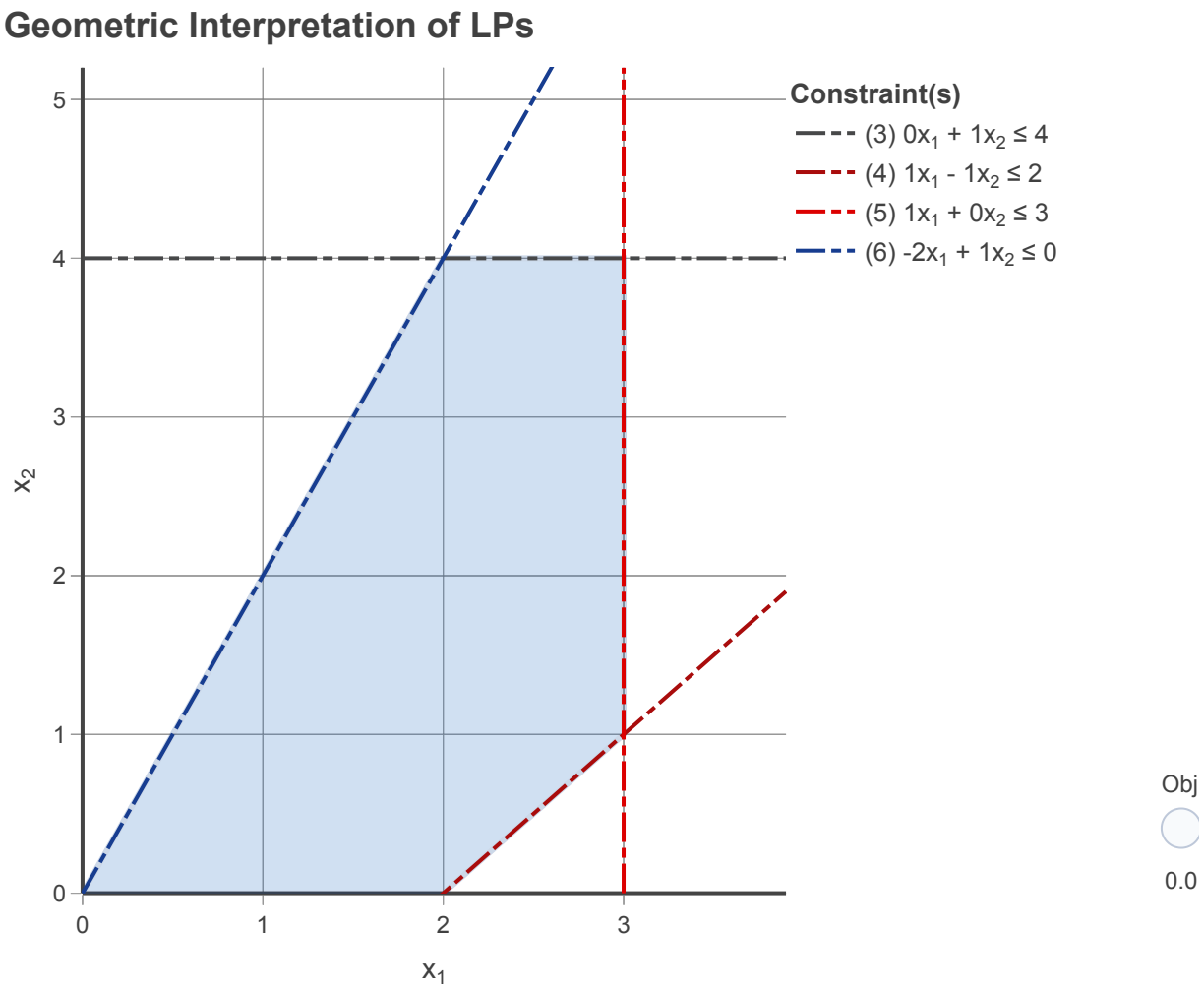
Q3: Plug your solution from **Q2** back into the LP and verify that each constraint is satisfied (don't forget non-negativity constraints!) and the objective value is as expected. Show your work.

A:

```
2(4) + 1(12) = 20 <= 20

1(4) + 1(12) = 16 <= 16

1(4) + 0(12) = 4 <= 7

4 > = 0

12 >= 0
```

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 [7]:
```
lp = ex.DEGENERATE_FIN_2D_LP # get LP example
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.)

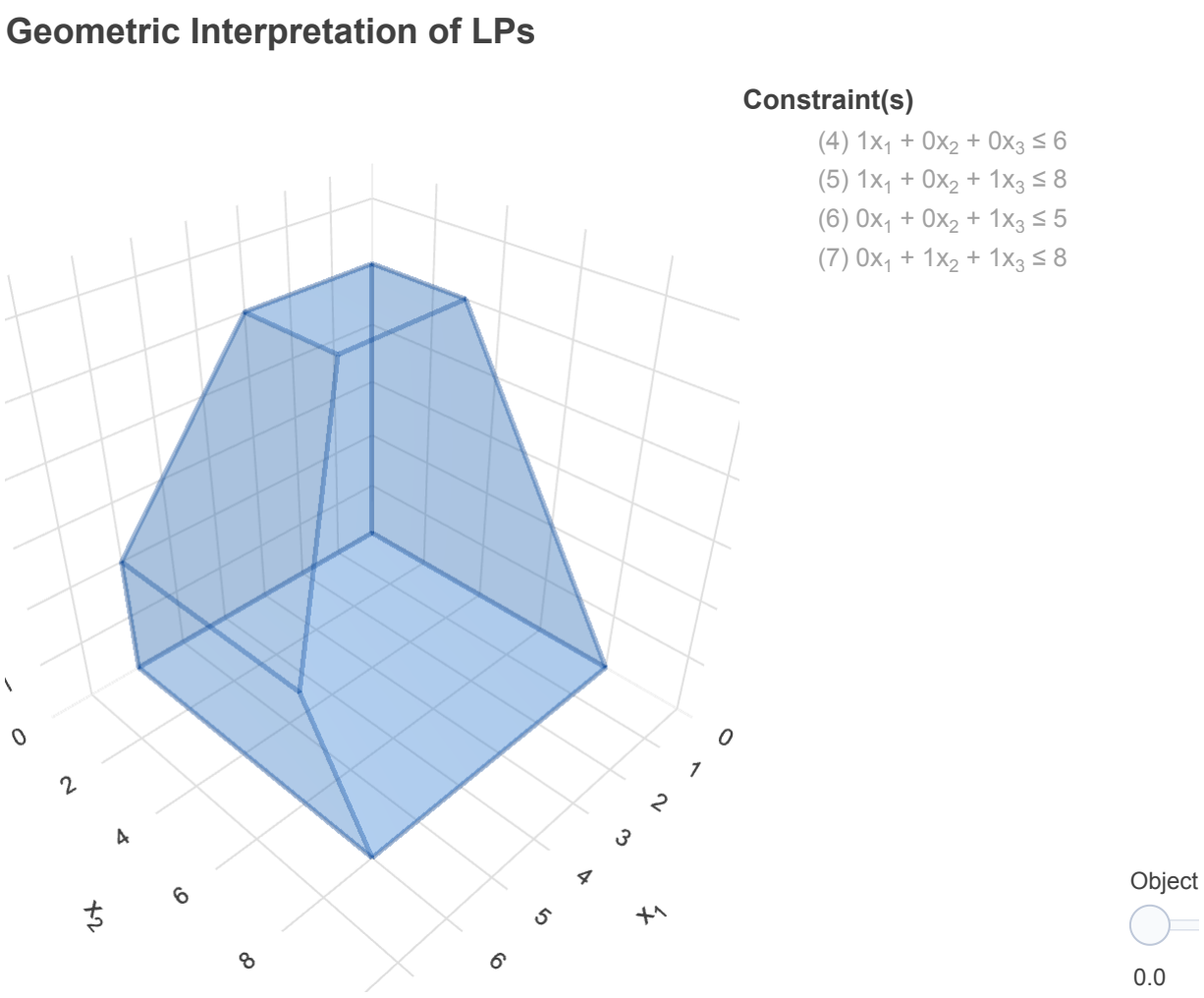
A: The optimal solution is x1 = 3 and x2 = 4 with an objective value of 11

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 [8]:
```
lp = ex.ALL_INTEGER_3D_LP # get LP example
lp_visual(lp).show() # visualize it
```



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: The objective value is 29 and x1 = 3, x2 = 3, and x3 = 5.

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: No, there are multiple ways to write the same inequality constraint. For example, x >= 0 can also be written as 2x >= 0. We can add coefficients without them changing the constraint because 2x >= 0 can be simlified to x >= 0.

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 constraints for x1 and x2 because the second constrain introduces a third variable that is added to 2x1 and x2, so 2x1 and x2 will still have to be <= 20 since we do not know the value of x3, which could be any vale >= 0.

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 exist many ways to represent the same linear program since each constraint an be written in a unique way.

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: Yes, the objective value is 56, and the corresponding optimal solution is (4, 12, 0, 0, 3). We can come to this solution quickly because all variables in the objective function have negative coefficients, so the maximum value occurs when these variables are sert to 0. We can then solve for the objective value and other variables based on this insight.

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:

0 <= 20 - 2x1 - x2
0 <= 16 - x1 - x2
0 <= 7 - x1

x3 = 20 - 2x1 - x2
x4 = 16 - x1 - x2
x5 = 7 - x1

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: No it is not.

z = 5x1 + 3x2

x3 = 4 - x1
x4 = 6 - x2
x5 = 9 - 2x1 - x2
x6 = 15 - 3x1 - 2x2

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: Here, using the same method as we did for question 9, by setting all variables in the abjective function to 0, we obtain a feasible solution of 0. This, however, is not optimal because both of the variables in the objective function have positive coefficients, which means that we must run the simplex method until these are negative to claim that a solution is optimal.

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 increaseing 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: We must make sure that all constraints are truthified after x1 is increased.

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: x1 can be increased to 4, which will decrease x3 to 0. This is the least value that x3 can be, meaning the most that x1 can be.

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

A: Constraint 1: 4
Constraint 2: no limit
Constraint 3: 4.5
Constraint 4: 5

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: x1 = 4 - x3

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:

z = 5(4 - x3) + 3x2 = 20 - 5x3 + 3x2

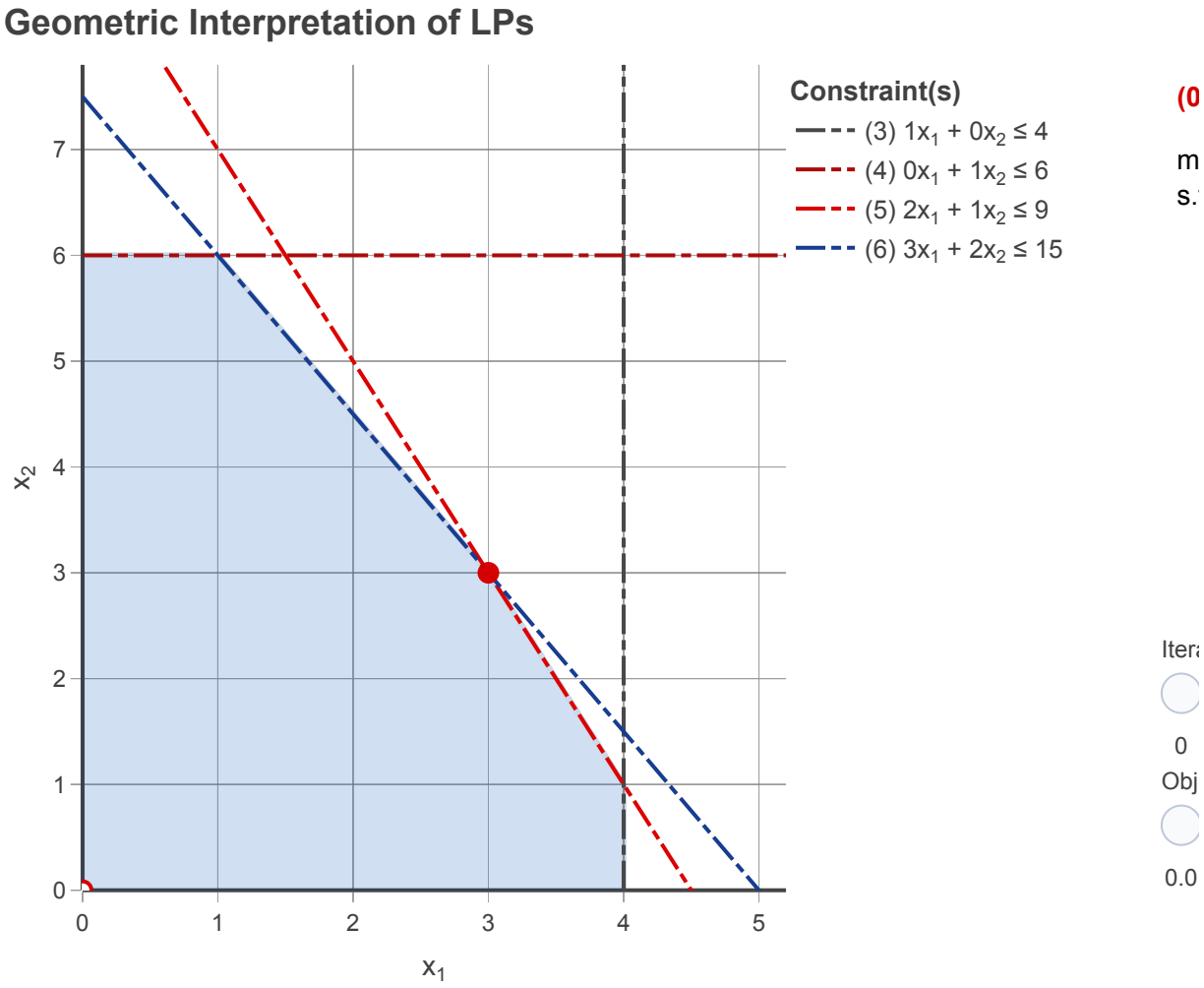
x1 = 4 - x3
x4 = 6 - x2
x5 = 9 - 2(4 - x3) - x2 = 1 + 2x3 - x2
x6 = 15 - 3(4 - x3) - 2x2 = 3 + x3 - 2x2

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

A: The feasible solution is now 20, with x values (4, 0, 0, 6, 1, 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]:
```
from gilp.visualize import simplex_visual # import the function
import numpy as np
lp = ex.LIMITING_CONSTRAINT_2D_LP # get the LP example
simplex_visual(lp, initial_solution=np.array([[0],[0]])).show() # show t
he 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: We can increase x1 by at most 4, because we hit constraint x1 <= 4. This mirrors our algebra since we found that x1 can increase at most 4, and the limiting constraint was the transformed version of x1 <= 4.

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

A: x2 will be increased since it is the only positive variable in the objective function.

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: The most limiting constraint for x2 will be 2x1 + x2 <= 9. This constraint will only allow x2 to increase by 1. The corresponding objective value is 23 and the feasible solution is (4, 1, 0, 5, 0, 1).

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, x2 increased by 1

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: x3 is the only positive varaible in the objective function. The most limiting constraint algebraically is constraint 4, which is x6 = 1 - x3 + 2x5:

Constraint 1: 4
Constraint 2: no limit
Constraint 3: 2.5
Constraint 4: 1

Graphically, we can see that constraint 4 is limiting since increasing x3 will decrease x1 and increase x2.

The next feasible solution will now be 24 with corresponding solution (3, 3, 1, 3, 0, 0).

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

A: Yes, this soluton is optimal since we aim to maximize 24 - x5 - x6, and since all the variables are negative we know that setting x5 and x6 to 0 will result in the optimal solution of 24. We cannot obtain a solution higher than 24, therefore we know that this solution is the maximum (optimal) solution.

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

A: Graphically we see that when starting from 0,0 and increasing x1, we can travel all the way up until x1=4. There is a limiting constraint for x1 here, so we cannot go any further. We can then do the same for x2 starting at 4,0. We travel upwards until we hit the limiting constraint for x2, which is at x2 = 1. Finally, we go diagonal from 4,1 in a way such that x1 will decrease and x2 will increase until we get to the point 3,3. We hit the limiting constraint such that x1, x2 = 3. From here we know that this solution is optimal since all variables are negative.

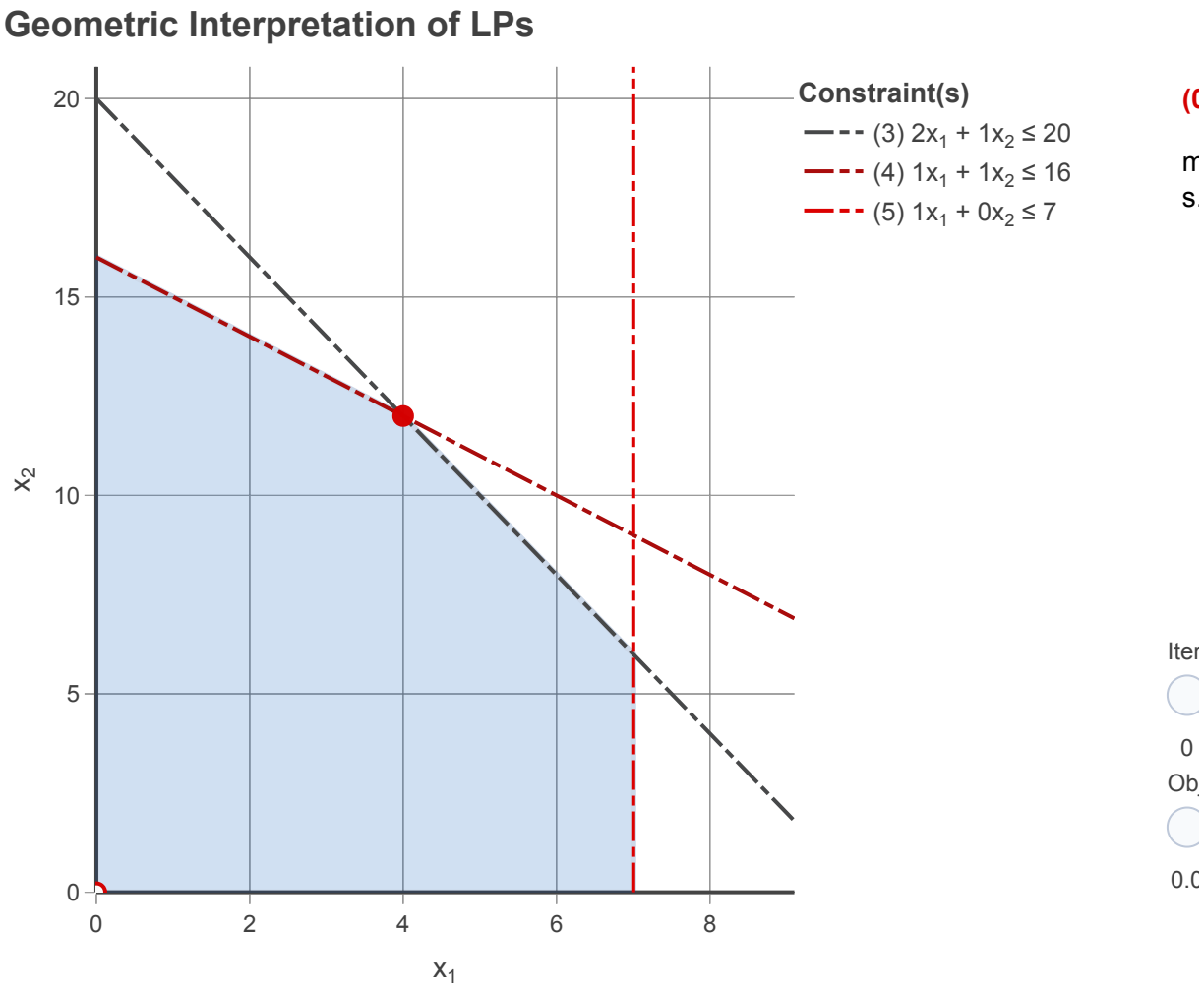
(BONUS): In 2D, we can increase a variable until we hit a 2D line representing the most limiting constraint. What would be the analagous situation in 3D?

A:

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 = ex.ALL_INTEGER_2D_LP # get LP example
simplex_visual(lp, initial_solution=np.array([[0],[0]])).show() # visual
ize 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 in the dictionary. For example, (1,3,4) means that $x_1, x_3,$ 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: The objective value at this point is 53 with corresponding solution (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: The slack is 16 - 6 - 7 = 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: x3 is a LHS variable that is defined by the RHS variables that were given to us. x3 and other new variables will allow us to create new dictionaries when using the simplex method to find optimal solutions for linear programs.

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

A: The feasible solution is 53 with corresponding solution(7,6,0,3,0) and x1 has a slack of 7, x2 has a slack of 6, x3 has a slack of 0, x4 has a slack of 3, and x5 has a slack of 0.

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: This means that the constraint is limiting

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: Variables x3 and x5 are not in the dictionary. They represent the slack of the constraint x1 = 7 - x5 as 0, and the constraint x2 = 6 - x3 + 2x5 as 0. The third constraint is not 0.

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 non dictionary variables are equal to 0, which tells us that 2 of the constraints are limiting, however not all of the constraints are, so it is not yet optimal.

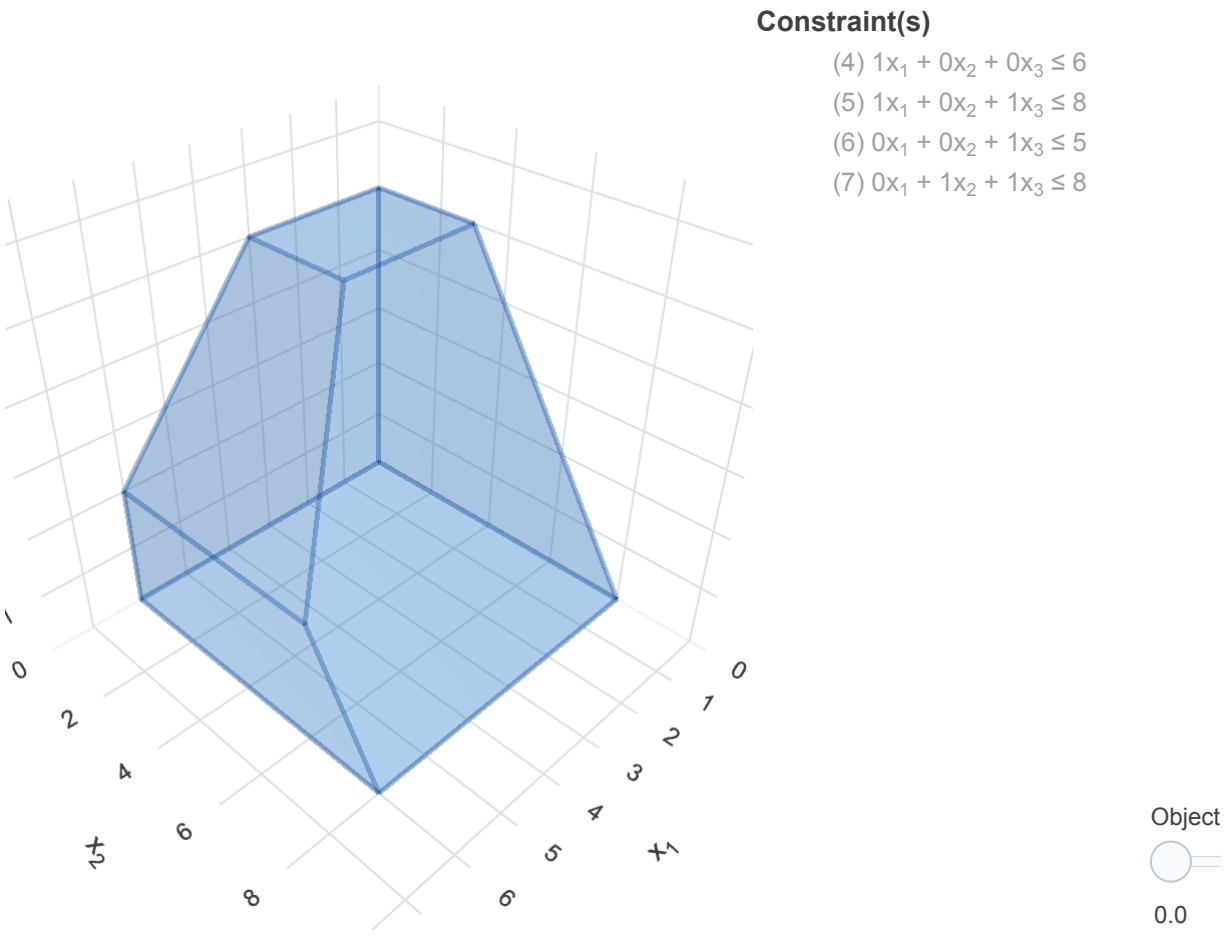
Q33: Look at some other corner points with this in mind. What do you find?

A: Nondictionary values are always equal to zero, and if the slack of every dictionary values is represented by these zeroes, all constraints will be limiting and the solution will be optimal.

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

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

Geometric Interpretation of LPs



Q34: Hover over the point (6,6,2) where $x_1 = 6, x_2 = 6,$ and $x_3 = 2$. Note which variables are not in the dictionary. Toggle the corresponding constraints on. What do you notice?

A: Variables 4, 5, and 7 are not in the dictionary. Constraints (4),(5) and (7) constrain this point, and these are the same constraints that are shown to define the pount (6,6,2) in the graph

Q35: Look at some other corner points and do as you did in Q34. Do you see a similar pattern? Combining what you learned in Q33, what can you say about the relationship between the variables not in the dictionary at some corner point, and the corresponding constraints?

A: Looking at other points, it is clear that all variables not in the dictionary, and therefore are equal to 0, will correspond to the constraints that graphically define each point

Q36: What geometric feature do feasible solutions for a dictionary correspond to?

A: Feasible solutions correspond to corners of a graph.

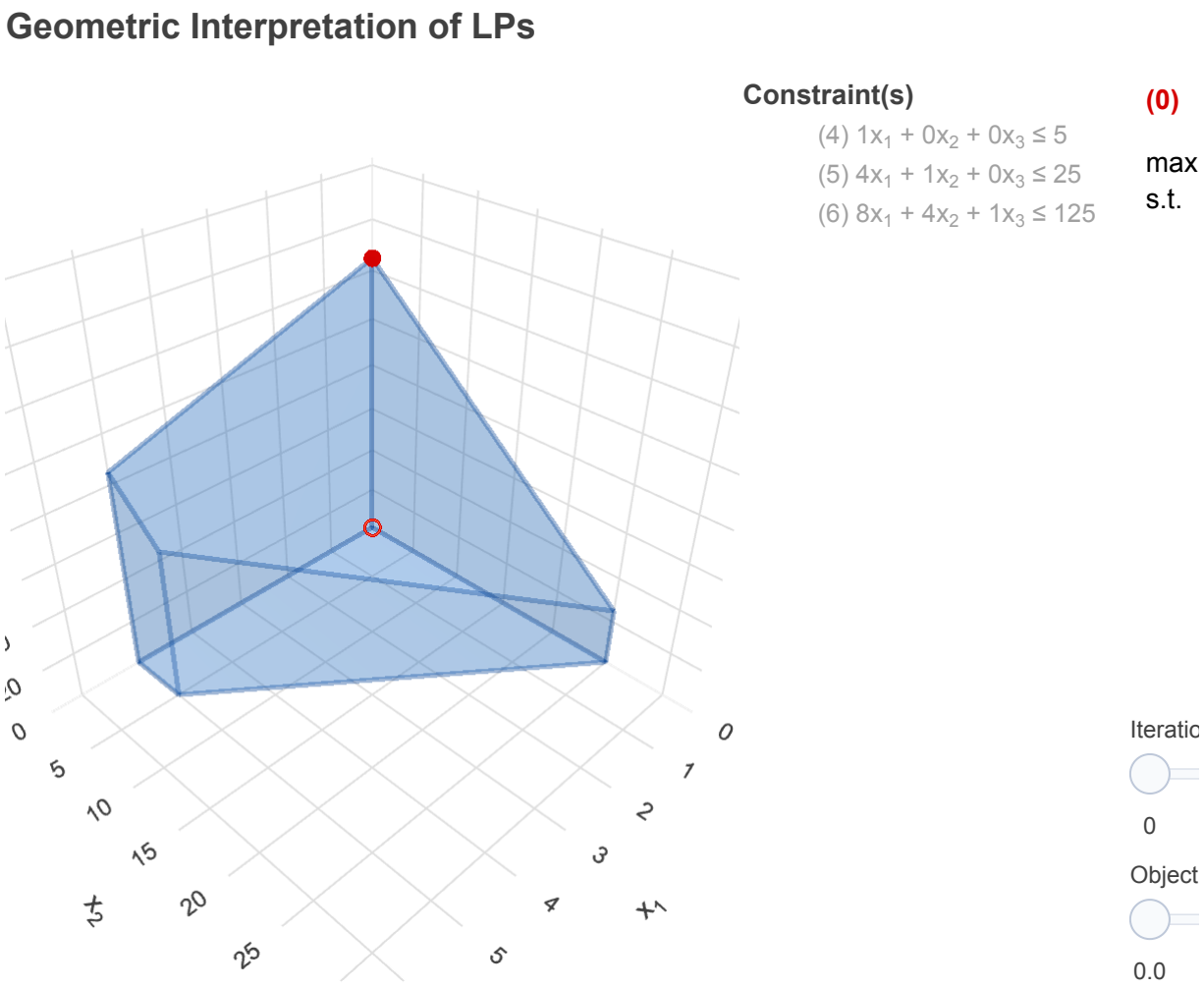
Part IV: Pivot Rules

The first step in an iteration of simplex is to choose an increasing variable. Sometimes, there are multiple options since multiple variables have a positive coefficent in the objective function. Here, we will explore what this decison translates to geometrically.

In this section, we will use a special LP commonly referred to as the Klee-Minty Cube.
$$\begin{align*} \max \quad & 4x_1+ 2x_2+ x_3\\ \text{s.t.} \quad\ & x_1 \leq 5 \\ & 4x_1 + x_2 \leq 25 \\ & 8x_1 + 4x_2 + x_3 \leq 125 \\ & x_1, x_2, x_3 \geq 0. \end{align*}$$

Furthermore, we will use an optional parameter called `rule` for the `simplex_visual()` function. This rule tells simplex which varaible to choose as an increasing variable when there are multiple options.

In [13]:
```
simplex_visual(ex.KLEE_MINTY_3D_LP, rule='dantzig', initial_solution=np.
array([[0],[0],[0]])).show()
```



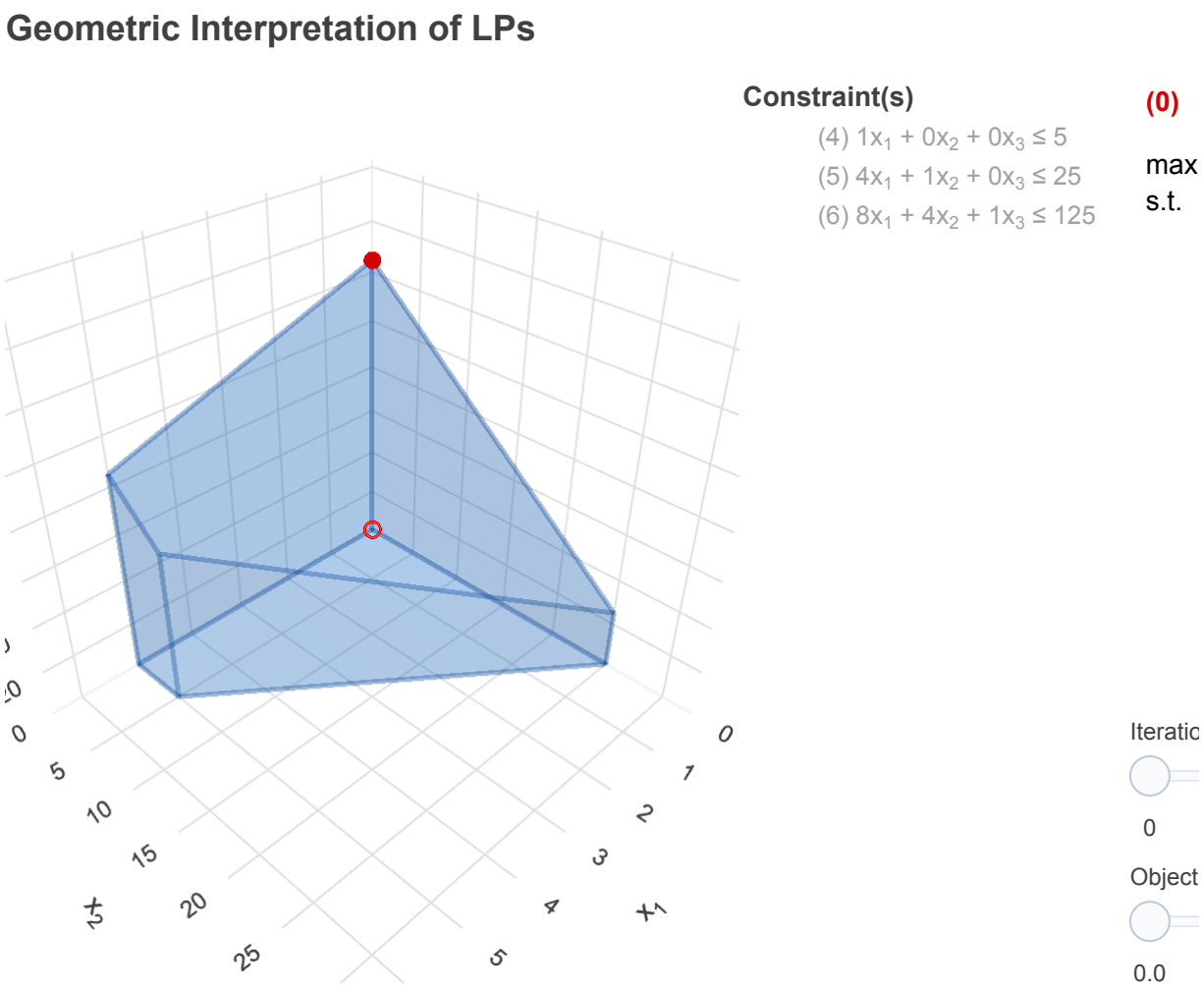
Q37: Use the iteration slider to examine the path of simplex on this LP. What do you notice?

A: I notice how the iterations get closer and closer to the optial value, and that each iteration goes across a given edge of the figure and stops once it hits a constraint. I also noticed how the only variable that increased at the end was x3.

Above, we used a pivot rule proposed by Dantzig. In this rule, the variable with the largest positive coefficient in the objective function enters the dictionary. Go through the iterations again to verify this.

Let us consider another pivot rule proposed by Bland, a professor here at Cornell. In his rule, of the variables with positive coefficents in the objective function, the one with the smallest index enters. Let us examine the path of simplex using this pivot rule! Again, look at the dictionary form LP at every iteration.

In [14]:
```
simplex_visual(ex.KLEE_MINTY_3D_LP,rule='bland', initial_solution=np.arr
ay([[0],[0],[0]])).show()
```



Q38: What is the difference between the path of simplex using Dantzig's rule and Bland's rule?

A: Bland's rule does only uses 5 iterations whereas Dantzig's uses 7.

Can you do any better? By setting `rule='manual_select'` , you can choose the entering variable explicitly at each simplex iteration.

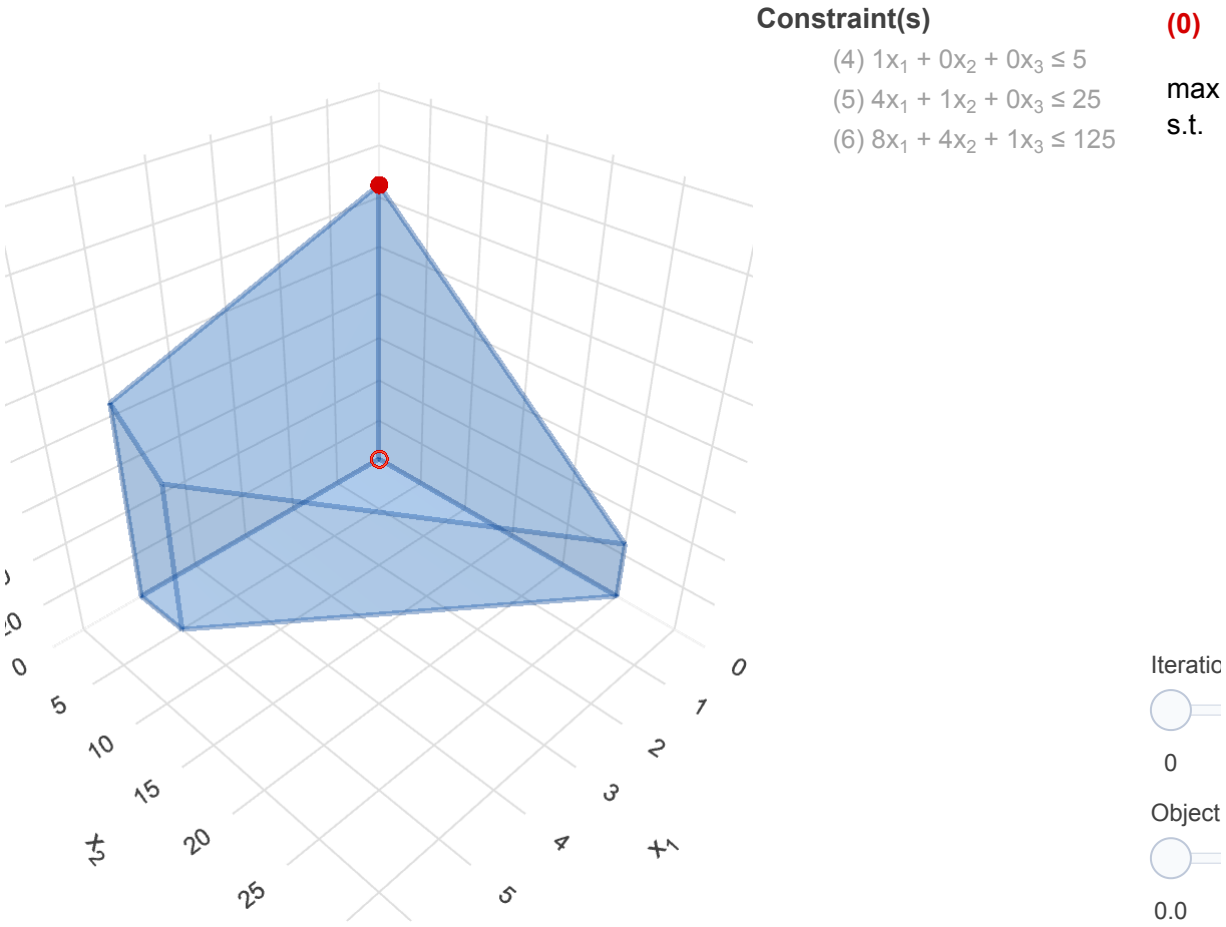
Q39: Can you do better than 5 iterations? How many paths can you find? (By my count, there are 7)

A: Yes, the least iterations you need is 1, by increasing x3 first. I also found 7 paths.

In [17]:
```
simplex_visual(ex.KLEE_MINTY_3D_LP,rule='manual_select', initial_solution=np.array([[0],[0],[0]])).show()
```

```
        INSTRUCTIONS

        At each iteration of simplex, choose one of the variables with a
        positive coefficent in the objective function. The list of indices
        for possible variables (also called entering variables) is given.

Pick one of [1, 2, 3]3
```

Geometric Interpretation of LPs



Q40: What does the choice of increasing variable correspond to geometrically?

A: Which axis you will traverse

Q41: Are there any paths you could visualize taking to the optimal solution that `rule='manual_select'` prevented you from taking? If yes, give an example and explain why it is not a valid path for simplex to take. (Hint: Look at the objective value after each simplex iteration.)

A: Yes, for example, I could increase x1, then x2, then x3, then go diagnally by decreasing x1 and x3 while increasing x2, then decrease x3, then decrease x2 (which gets you bakc to the start) and then finally increase x3 to get to the optimal solution. This is not valid because you cannot do a cycle when traversing a graph. It is not possible to increase a variable that will not make progress towards termination, as defined by the simplex method.

Part V: Creating LPs in GILP (Optional)

We can also create our own LPs! First, we must import the `LP` class.

In [ ]:
```
from gilp.simplex import LP
```

Let us create the following LP.
$$\begin{align*} \max \quad & 3x_1+2x_2\\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 6 \\ \quad & 0x_1 + 1x_2 \leq 2 \\ \quad & x_1, x_2 \geq 0 \\ \end{align*}$$

We will create this LP by specifying 3 arrays of coefficents. We define the NumPy arrays `A`, `b`, and `c` and then pass them to the `LP` class to create the LP.

In [ ]:
```
A = np.array([[2,1],  # LHS constraint coefficents
              [0,1]])
b = np.array([6,2])  # RHS constraint coefficents
c = np.array([3,2])  # objective function coefficents
lp = LP(A,b,c)
```

Let's visualize it!

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

... and solve it!

In [ ]:
```
simplex_visual(lp, initial_solution=np.array([[0],[0]])).show()
```