# 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 [2]:
```
#!pip install gilp
```

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

In [3]:
```
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:

$$\begin{aligned} \max \quad & 5x_1 + 3x_2 \\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 20 \\ & 1x_1 + 1x_2 \leq 16 \\ & 1x_1 + 0x_2 \leq 7 \\ & x_1, x_2 \geq 0 \end{aligned}$$

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

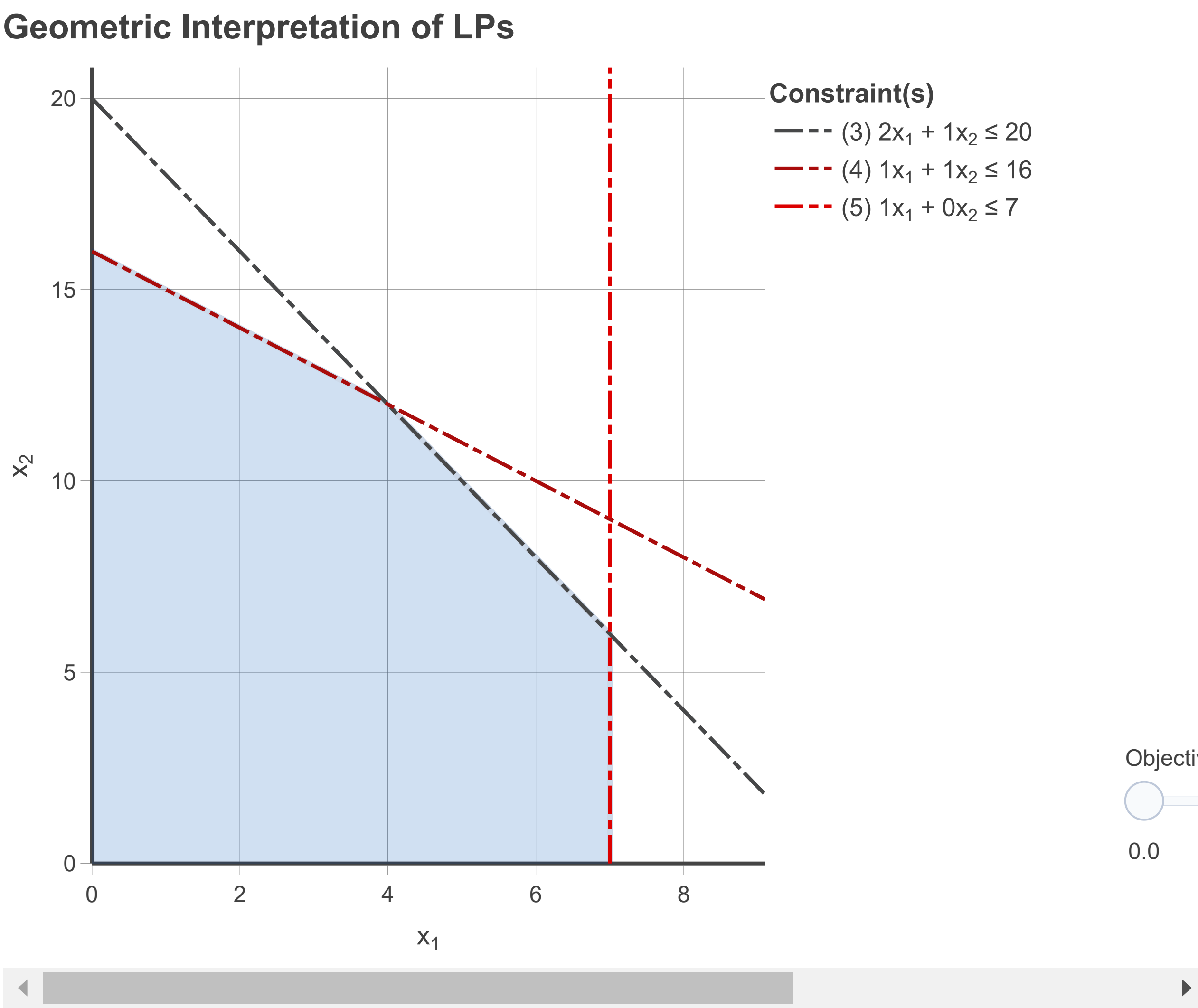
In [4]:
```
lp = ex.ALL_INTEGER_2D_LP
```

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

In [5]:
```
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 [6]:
```
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

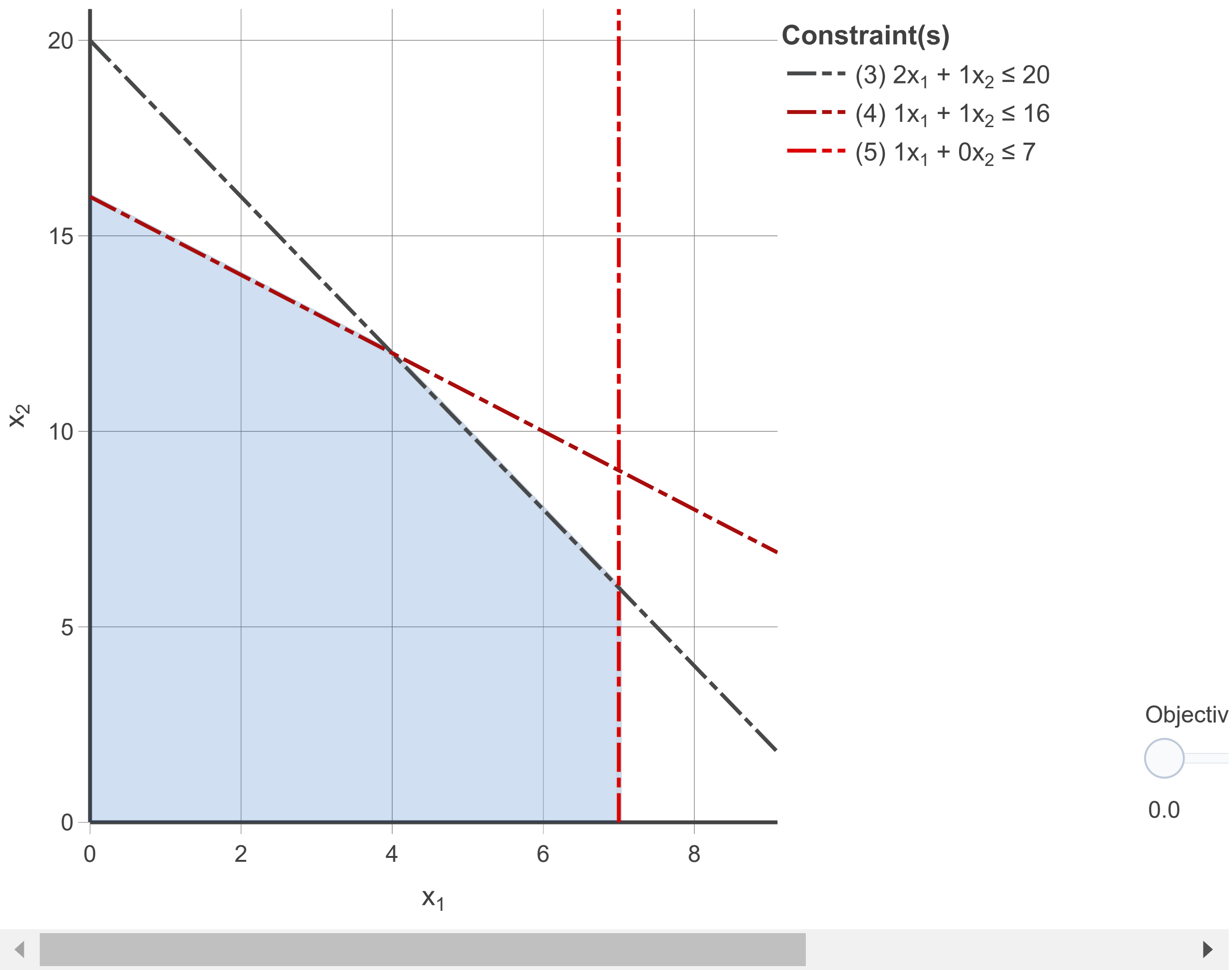
Let's use GILP to solve the following LP graphically:

$$\begin{aligned} \max \quad & 5x_1 + 3x_2 \\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 20 \\ & 1x_1 + 1x_2 \leq 16 \\ & 1x_1 + 0x_2 \leq 7 \\ & x_1, x_2 \geq 0 \end{aligned}$$

Recall, this LP is called `ALL_INTEGER_2D_LP` .

In [7]:
```
lp = ex.ALL_INTEGER_2D_LP # get LP example
lp_visual(lp).show() # visualize it
```

## Geometric Interpretation of LPs




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

**A:** Isoprofit lines represent the objective value which is either maximized or minimized. In this case, 5x(1)+3x(2) is maximized. Graphing these lines and altering what it equals alters the y-intercept and we want to get this as large as possible while still having the line be in the feasible range. This only alters the y-intercept and not the slope. We can graph the isoprofit lines and maximize c to find the best objective value maximized. Essentially the y-intercept is the objective value.

**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 optimal solution is x(1)=4 and x(2)= 12. The objective value is 56. This is optimal because it is the furthest to the right parallel line that still exists in the feasible range of the linear program. We can verify that it is correct beacuse this value occurs at a edge point in the feasibility range.

**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.

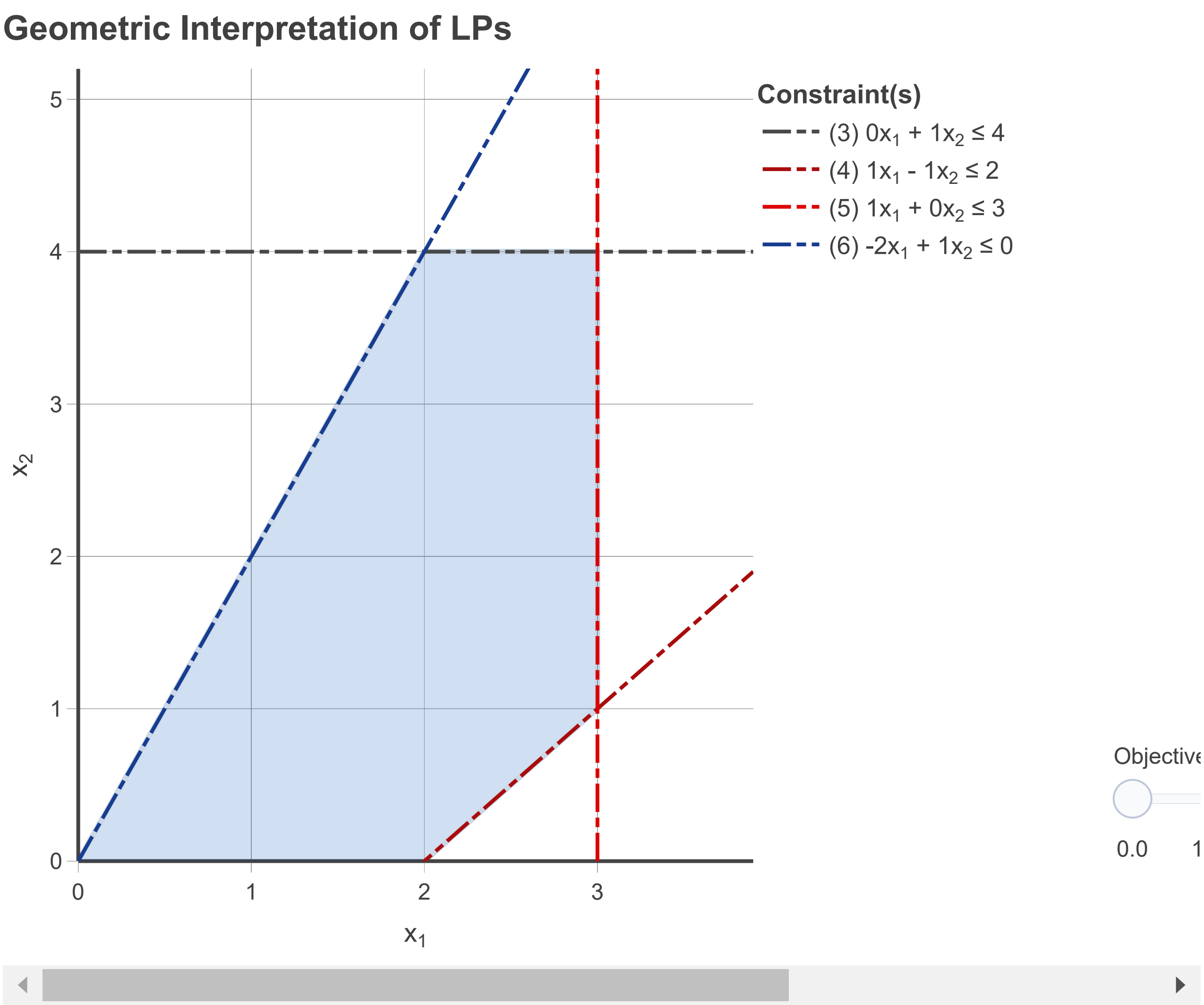
**A:** 5(4) + 3(12) = 56 2(4) + 1(12) <= 20 20 <=20 1(4) + 1(12) <= 16 16 <=16 1(4) + 0(12) <= 7 4 <=7 4,12 >= 0

Let's try another! This LP is called `DEGENERATE_FIN_2D_LP` .

$$\begin{aligned} \max \quad & 1x_1 + 2x_2 \\ \text{s.t.} \quad & 0x_1 + 1x_2 \leq 4 \\ & 1x_1 - 1x_2 \leq 2 \\ & 1x_1 + 0x_2 \leq 3 \\ & -2x_1 + 1x_2 \leq 0 \\ & x_1, x_2 \geq 0 \end{aligned}$$

In [8]:

```
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:** optimal solution is x(1)=3 and x(2)=4 and objective value is 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.

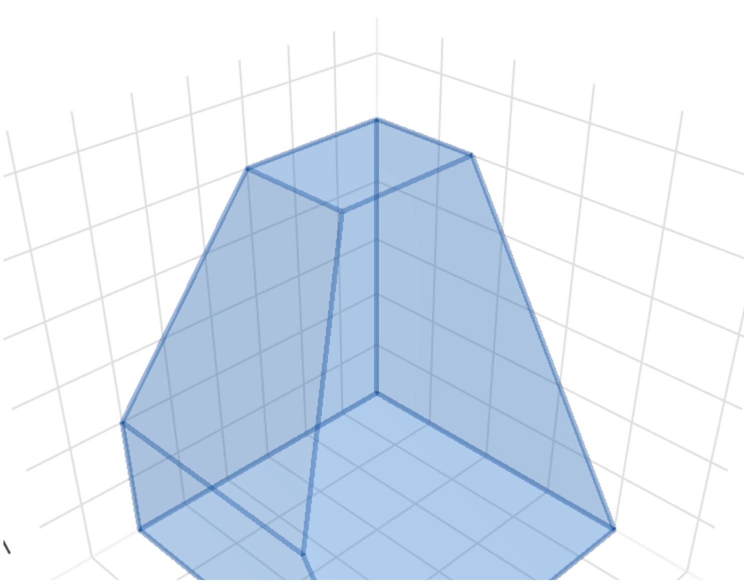
Now, we will try to wrap our head around an LP with three decision variables! Similar to before, we can plot solutions to a 3D LP on a plot with 3 axes. Here, the $x$-axis corresponds to the value of $x_1$ and the $y$-axis corresponds to the value of $x_2$ as before. Furthermore, the $z$-axis corresponds 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` :

$$
\begin{aligned}
\max \quad & 1x_1 + 2x_2 + 4x_3 \\
\text{s.t.} \quad & 1x_1 + 0x_2 + 0x_3 \leq 6 \\
& 1x_1 + 0x_2 + 1x_3 \leq 8 \\
& 0x_1 + 0x_2 + 1x_3 \leq 5 \\
& 0x_1 + 1x_2 + 1x_3 \leq 8 \\
& x_1, x_2 \geq 0
\end{aligned}
$$

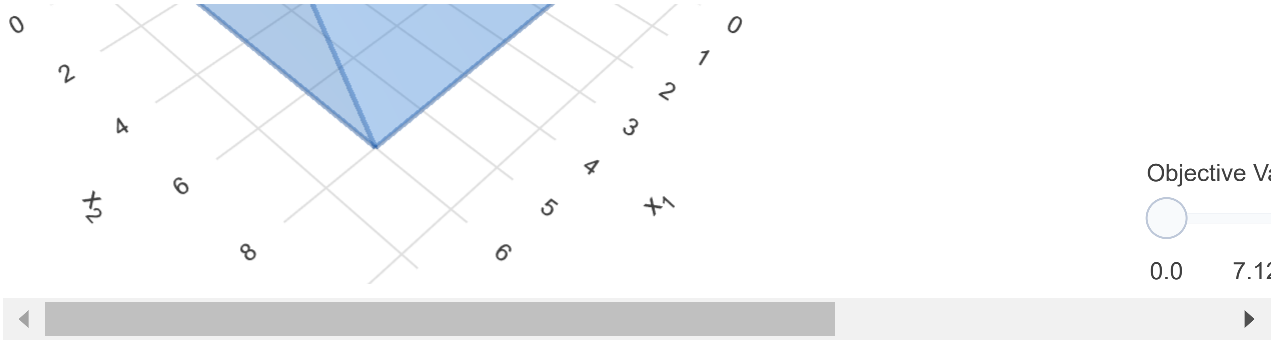
In [9]:
```
lp = ex.ALL_INTEGER_3D_LP # get LP example
lp_visual(lp).show() # visualize it
```

## Geometric Interpretation of LPs



**Constraint(s)**

(4) $1x_1 + 0x_2 + 0x_3 \leq 6$
(5) $1x_1 + 0x_2 + 1x_3 \leq 8$
(6) $0x_1 + 0x_2 + 1x_3 \leq 5$
(7) $0x_1 + 1x_2 + 1x_3 \leq 8$




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:** optimal solution is x(1)=3, x(2)=3, and x(3)=5 and objective value is 29

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:** There are multiple ways of writing the same inequality constant. For example, x(1) <= 8 + x(3) and x(3) >= x(1) - 8 are equivalent constraints.

**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:** These two are not necessairly the same constraint but in some manner could act as one. For example there might exist x(1) and X(2) and x(3) values that allow both constraints to be true, where they would be the same value. Basically in the second inequality, we have 20 - x(3) equals the first inequality. It can be true that one constraint is satisified with high values in x(1) and x(2) in the first constraint that make the 2nd constraint invalid, but it could be valid in some circumstances.

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

**A:** I do not think there is a unique way to write a given LP. There are numerous ways to write the dictionary method, especially when you have a lot of variables. A unique LP might be possible in the most basic 1 variable LP, but not in the circumstance we are placed in now, solving 3 or more variables.

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:

$$\begin{aligned} \max \quad & 56 - 2x_3 - 1x_4 \\ \text{s.t.} \quad & x_1 = 4 - 1x_3 + 1x_4 \\ & x_2 = 12 + 1x_3 - 2x_4 \\ & x_5 = 3 + 1x_3 - 1x_4 \\ & x_1, x_2, x_3, x_4, x_5 \geq 0 \end{aligned}$$

**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 optimal solution is x(3)=0 and x(4)=0, the objective value is 56. This is because the x values are all constrained to be non-negative and a value of 0 would be the least assignment you can give to all of these values and since all the coefficients for x values in the objective function are negative and a greater than 0 vlaue would further minimized this objective value, an assignment of 0 to these x values and an objective value of 56 is optimal.

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!

$$\begin{aligned} \max \quad & 5x_1 + 3x_2 \\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 20 \\ & 1x_1 + 1x_2 \leq 16 \\ & 1x_1 + 0x_2 \leq 7 \\ & x_1, x_2 \geq 0 \end{aligned}$$

**A:**

$$\begin{aligned} \max \quad & 5x_1 + 3x_2 \\ \text{s.t.} \quad & 0 \leq 20 - 2x_1 - 1x_2 \\ & 0 \leq 16 - 1x_1 - 1x_2 \\ & 0 \leq 7 - 1x_1 - 0x_2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

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

# 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.

$$\begin{aligned} \max \quad & 5x_1 + 3x_2 \\ \text{s.t.} \quad & 1x_1 + 0x_2 \leq 4 \\ & 0x_1 + 1x_2 \leq 6 \\ & 2x_1 + 1x_2 \leq 9 \\ & 3x_1 + 2x_2 \leq 15 \\ & x_1, x_2 \geq 0 \end{aligned}$$

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

**A:** No

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

**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:** Let's give a feasible solution where x_1 = 0 and x_2 = 0. This would yeild an objective value of 0. This is a feasible solution because all x values need to be nonnegative and assigning a 0 is permissible in this case (makes feasibility). This is not the optimal solution because there are still positive coefficients in the objective solution. While there are still positive coefficeints in this maximizing objective function, the simplex method states that the value of x values can be increased, further optimized. This means that another iteration can be run in the algorithm's case.

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:** There are certain constraints that need to be followed to ensure feasibility. Since all x values can be only non-negative, we need to ensure that the LHS variables don't drop below 0. We take the most limiting value and use that as our maximum maximizing value.

**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 increase by 4 at most to remain feasible. x_3 must be >= 0

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

**A:** Constraints: (1) 4 (most limiting) (2) no limit (3) 9/2 (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:** $x_1 = 4 - 0x_2 - 1x_3$

**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:**

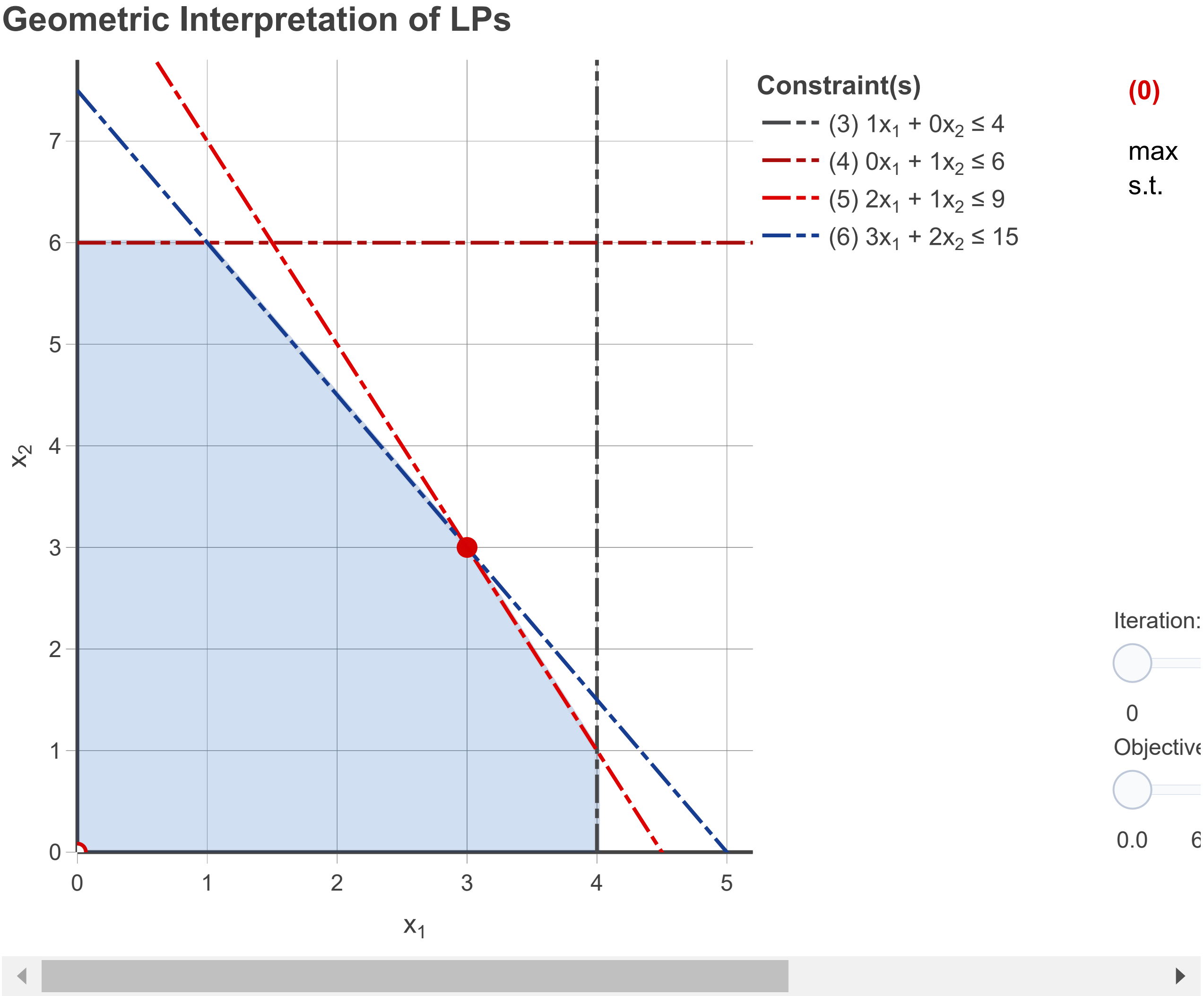
$$\begin{aligned} \max \quad & z = 20 - 5x_3 + 3x_2 \\ \text{s.t.} \quad & x_1 = 4 - 1x_3 - 0x_2 \\ & x_4 = 6 + 0x_3 - 1x_2 \\ & x_5 = 1 + 2x_3 - 1x_2 \\ & x_6 = 3 + 3x_3 - 2x_2 \\ & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0 \end{aligned}$$

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

**A:** {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 [11]:
```
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 the simplex visu
```




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:** Increase by 4. You hit constraint (3) first. This does match the constraint that I mentioned earlier algebraically.

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

**A:** We will be increasing vaariable $x_2$ next because it is the only other variable in the dictionary objective function that is still positive.

**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 the most limiting constraint is constraint (5). $x_2$ can increase by 1. The objective value would be 23 and the corresponding feasible solution would be {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:** Guess 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:** $x_3$ is the increasing variable.

Constraint (algebraically): (3) 4 (4) no limit (5) 2.5 (6) 1 (most limiting)

Graphically, constraint 6 is the most limiting because it will create a roof on $x_3$ expansion before any other constraint.

Same constraint

The next objective value is 24. The feasible, corresponding solution it yields is {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 the dictionary form for the objective function contains all negative coefficients, so is optimal, like explained earlier.

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

**A:** I looked in the respective axis directions and increased in that direction from the previous solution until a constraint on the graph was hit. This would be the most limiting constraint.

**(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?

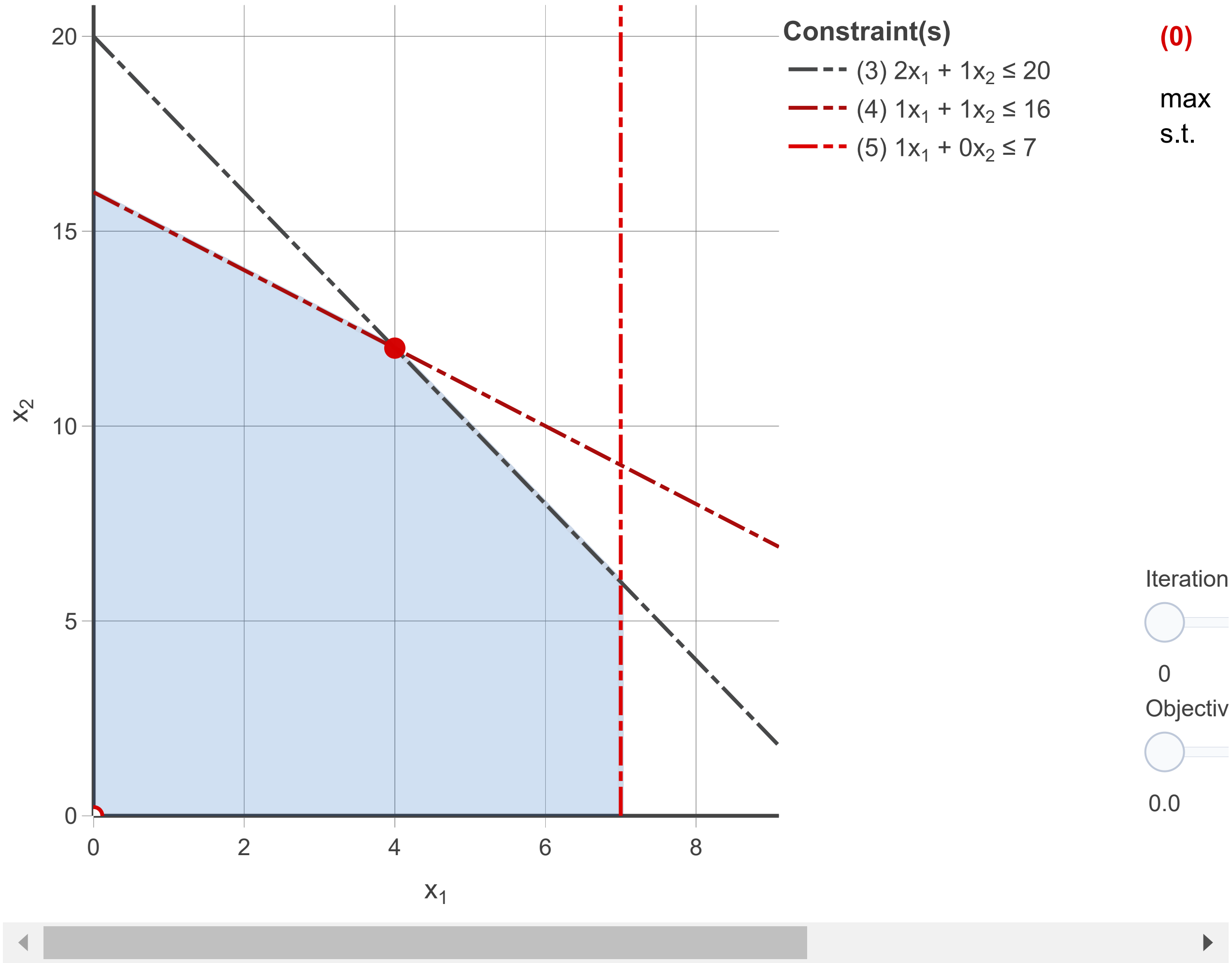
**A:** 3D plane

# 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 [12]:
```
lp = ex.ALL_INTEGER_2D_LP # get LP example
simplex_visual(lp, initial_solution=np.array([[0],[0]])).show() # visualize it
```

## Geometric Interpretation of LPs




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:** {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 value in this constraint

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

**A:** Each slack is 0 except for the currently redundant (not in future) constraint, constraint 4.

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:** They both exhibit the same value. The feasible solution occurs at the constraint and it is optimal at that constraint, can not increase. This occurs at corner points. They occur at the same point (corner points).

**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 3 and 5 are not in the dictionary. They represent the slack in constraints 3 and 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 non-dictionary variables are of value are 0. This means that the feasible solution at this point meets at the same constraints and those constraints are at max value, and cannot be increased further. These values are RHS variables as well.

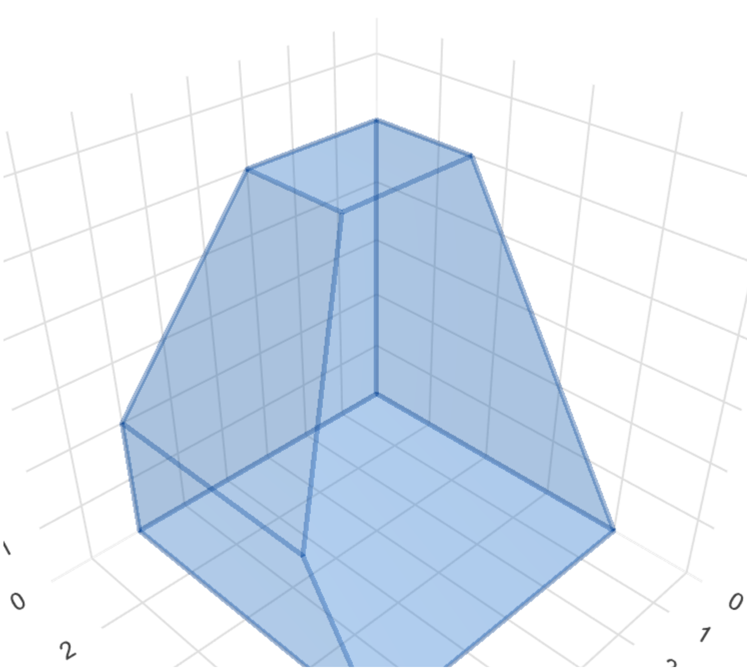
**Q33:** Look at some other corner points with this in mind. What do you find?

**A:** The x-values not included in the dictionary are at value 0. The values included are at a value other than 0.

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

In [13]:
```
lp = ex.ALL_INTEGER_3D_LP # get LP example
lp_visual(lp).show() # visualize it
```

## Geometric Interpretation of LPs




**Constraint(s)**

(4) $1x_1 + 0x_2 + 0x_3 \leq 6$

(5) $1x_1 + 0x_2 + 1x_3 \leq 8$

(6) $0x_1 + 0x_2 + 1x_3 \leq 5$

(7) $0x_1 + 1x_2 + 1x_3 \leq 8$

**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:** 4, 5, and 7. As mentioned before they limit the feasible range at that point and all 3 planes form to that one point (on the same location).

**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:** Yes, the only thing that differs is the number of constraints that limit each corner, due to the building of the feasibility range and the nature of how it needs to be constructed. At each corner, there will be a certain set of defined values in that specific dictionary, the values that are not included are at value 0 and some of these correspond to a graphical and analytical constraint that limits the feasibility range. This represents the slack and the slack is 0 at these points because they exhibbit the same location and value in space at the corner points.

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

**A:** They correspond to a point in space and they sometimes have more than one plane that hits thid point. This point is always a corner as well.

# 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.

$$\begin{array}{ll} \max & 4x_1 + 2x_2 + x_3 \\ \text{s.t.} & x_1 \leq 5 \\ & 4x_1 + x_2 \leq 25 \\ & 8x_1 + 4x_2 + x_3 \leq 125 \\ & x_1, x_2, x_3 \geq 0. \end{array}$$

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 [14]:
```
simplex_visual(ex.KLEE_MINTY_3D_LP, rule='dantzig', initial_solution=np.array([[0],[0],
```

## Geometric Interpretation of LPs

**Constraint(s)**

(4) $1x_1 + 0x_2 + 0x_3 \le 5$

(5) $4x_1 + 1x_2 + 0x_3 \le 25$

(6) $8x_1 + 4x_2 + 1x_3 \le 125$

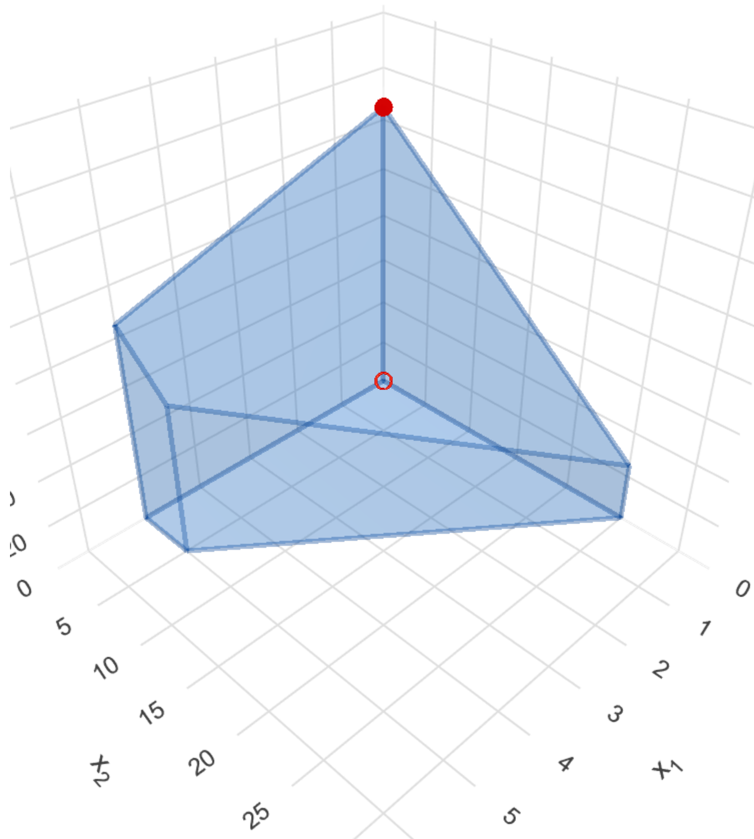
**(0)**

max z

s.t. x

x

x




Iteration: 0

0 1

Objective V

0.0 33.0

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

**A:** It essentially creates a path along the edges of a geometric figure and eventually finds its way to a maximum best solution corner (at the top)

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 [15]:
```
simplex_visual(ex.KLEE_MINTY_3D_LP,rule='bland', initial_solution=np.array([[0],[0],[0]
```

## Geometric Interpretation of LPs

**Constraint(s)**

(4) $1x_1 + 0x_2 + 0x_3 \le 5$

(5) $4x_1 + 1x_2 + 0x_3 \le 25$

(6) $8x_1 + 4x_2 + 1x_3 \le 125$

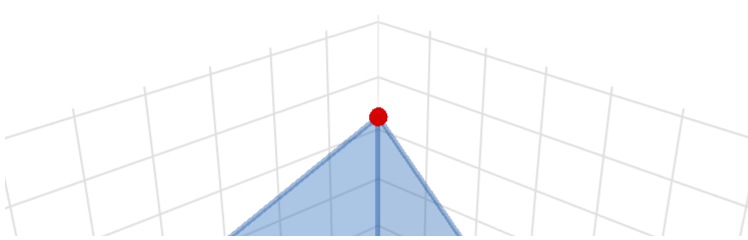
**(0)**

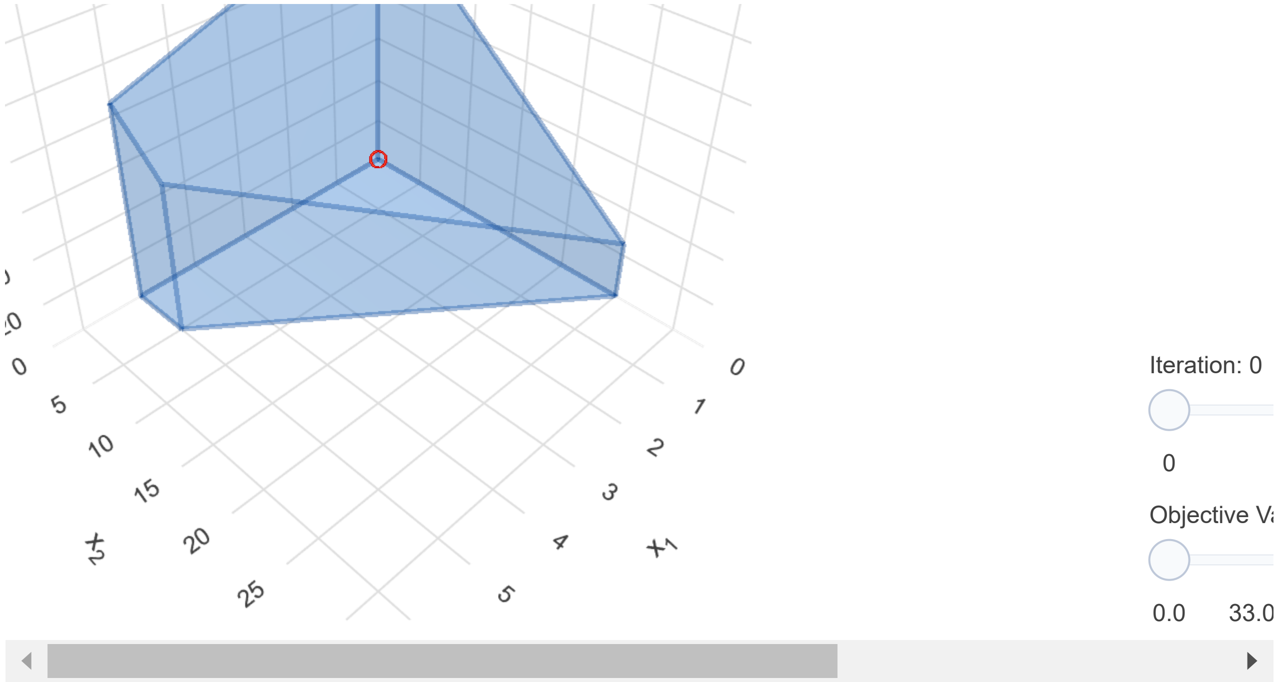
max z

s.t. x

x

x






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

**A:** Bland's rule requires only 5 iterations in this specific case, while Danzing's rule requires 7. Bland's first, second, fourth and fifth iteration match Danzing's first, second, sixth, and seventh interation

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:** 1 iterations. I can find 7 paths that get equal to or less than 5 iterations

In [24]:
```
simplex_visual(ex.KLEE_MINTY_3D_LP,rule='manual_select', initial_solution=np.array([[0]
```

```
        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
```

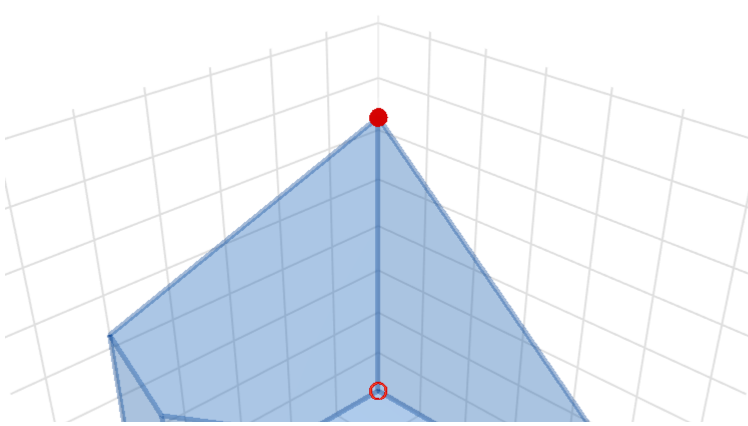
## Geometric Interpretation of LPs

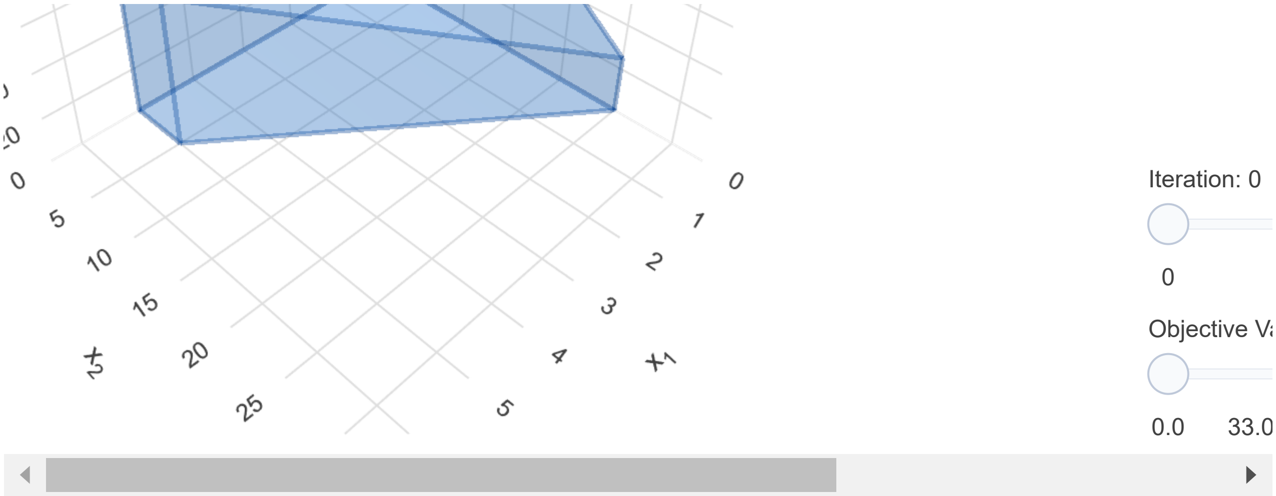
**Constraint(s)**

(4) $1x_1 + 0x_2 + 0x_3 \leq 5$
(5) $4x_1 + 1x_2 + 0x_3 \leq 25$
(6) $8x_1 + 4x_2 + 1x_3 \leq 125$

**(0)**

max z
s.t. $x_4$
$x_5$
$x_6$






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

**A:** It means moving in the axial direction of that specific variable and increasing that value. It shows up as a red path and it means that you are following and edge until you hit a corner that restrains the amount you can increase (as mentioned previously).

**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:** There are no paths in the paths that I did construct that were prevented by this algorithm. After every simplex iteration, it gave me the choice to increase all values that still had positive coeffiecients. All the values I was not allowed to increase were negative coefficients, and increasing those would ruin feasibility (all x must be nonnegative).

# Part V: Creating LPs in GILP (Optional)

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

In [19]:
```
from gilp.simplex import LP
```

Let us create the following LP.

$$\begin{aligned} \max \quad & 3x_1 + 2x_2 \\ \text{s.t.} \quad & 2x_1 + 1x_2 \leq 6 \\ & 0x_1 + 1x_2 \leq 2 \\ & x_1, x_2 \geq 0 \end{aligned}$$

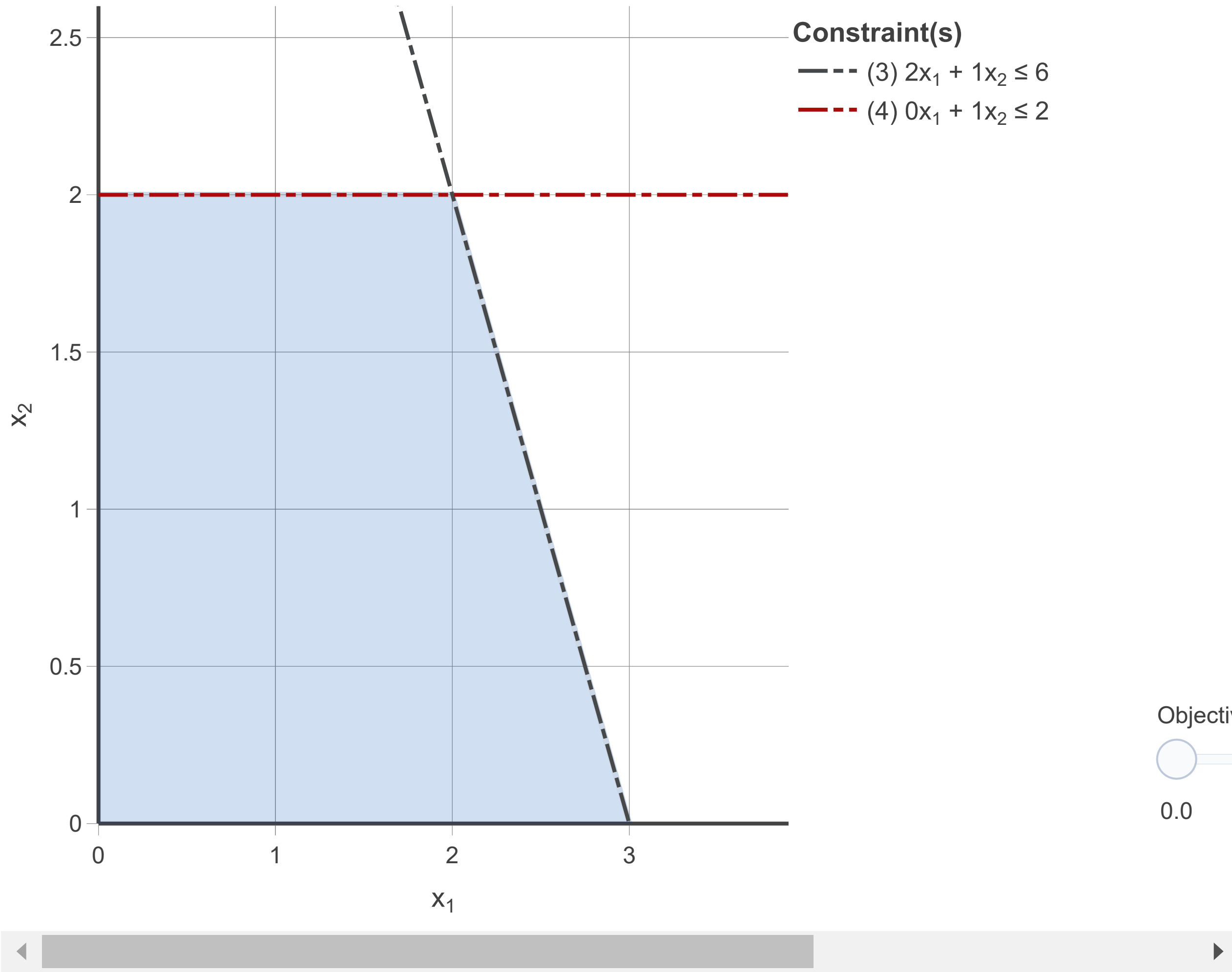
We will create this LP by specifying 3 arrays of coefficents. We define the NumPy arrays `A`, `b`, and `c` and then pass them to the `LP` class to create the LP.

In [20]:
```
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 [21]:
```
lp_visual(lp).show()
```

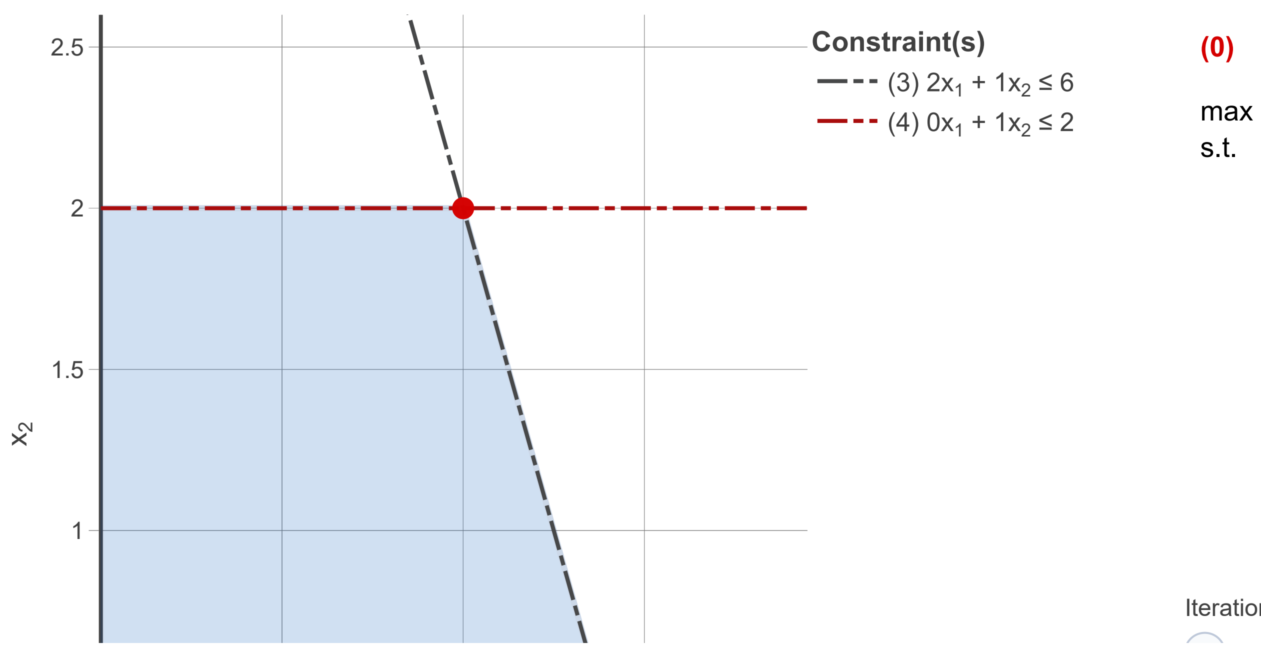
## Geometric Interpretation of LPs

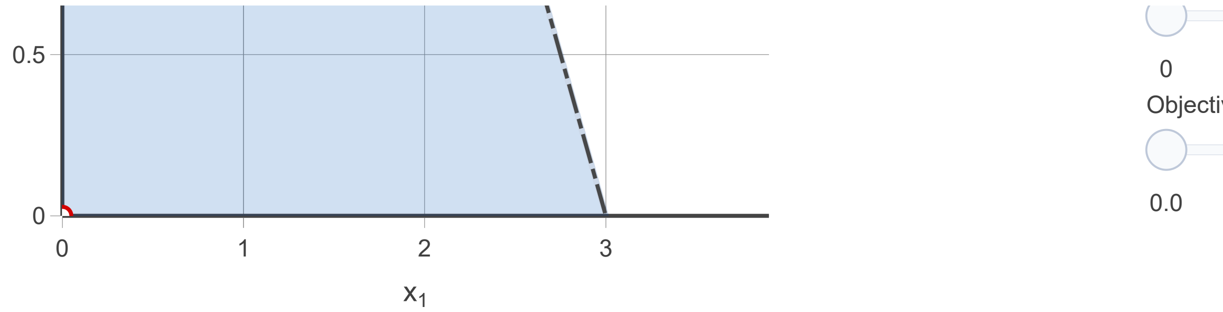



... and solve it!

In [22]:
```
simplex_visual(lp, initial_solution=np.array([[0],[0]])).show()
```

## Geometric Interpretation of LPs






In [ ]: