This LP is called `ALL_INTEGER_3D_LP` :

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

In [ ]:
```
lp = gilp.examples.ALL_INTEGER_3D_LP # get LP example
gilp.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:**

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. For example, the constraints $2x_1 + 1x_2 \leq 20$ and $2x_1 + 1x_2 + x_3 = 20$ are the same constraint in terms of x1 and x2 (where all x are nonnegative).

**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 are the same constraint in terms of x1 and x2 (for both constraints, x1 must be <= 10 and x must be <= 20). In the second constraint, x3 must also be <= 20.

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

**A:** No.

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:** Optimal solution: x3 = 0, x4 = 0, x1 = 4, x2 = 12, x5 = 3. Objective value: 56. In the objective function, we are subtracting 2x3 and x4 from 56. Because x3 and x4 are both nonnegative, 56 must be the objective value of the optimal solution.

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!

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

**A:** max 5x1 + 3x2 s.t. x3 = 20 - 2x1 - x2, x4 = 16 - x1 - x2, x5 = 7 - x1; x1, x2, x3, x4, x5 >= 0

# 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:** max 5x1 + 3x2, s.t. x3 = 4-x1, x4 = 6-x2, x5 = 9-2x1-x2, x6=15-3x1-2x2; x1, x2, x3, x4, x5, x6>=0

**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:** Solving the system of equations 2x1 + x2 = 9 and 3x1 + 2x2 = 15, we get x1 = 3 and x2 = 3 (we want the maximum possible values of x1 and x2, because these will yield the maximum possible value of the objective function). x3 = 1, x4 = 3, x5 = 0, x6 = 0. The objective value of this solution is 24. This 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 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:** x1 has constraints (some in relation to x2).

**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 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:** x1 can be at most 4, N/A, 4.5, and 5 -> x1 can be at most 4 (this is the first constraint).

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)

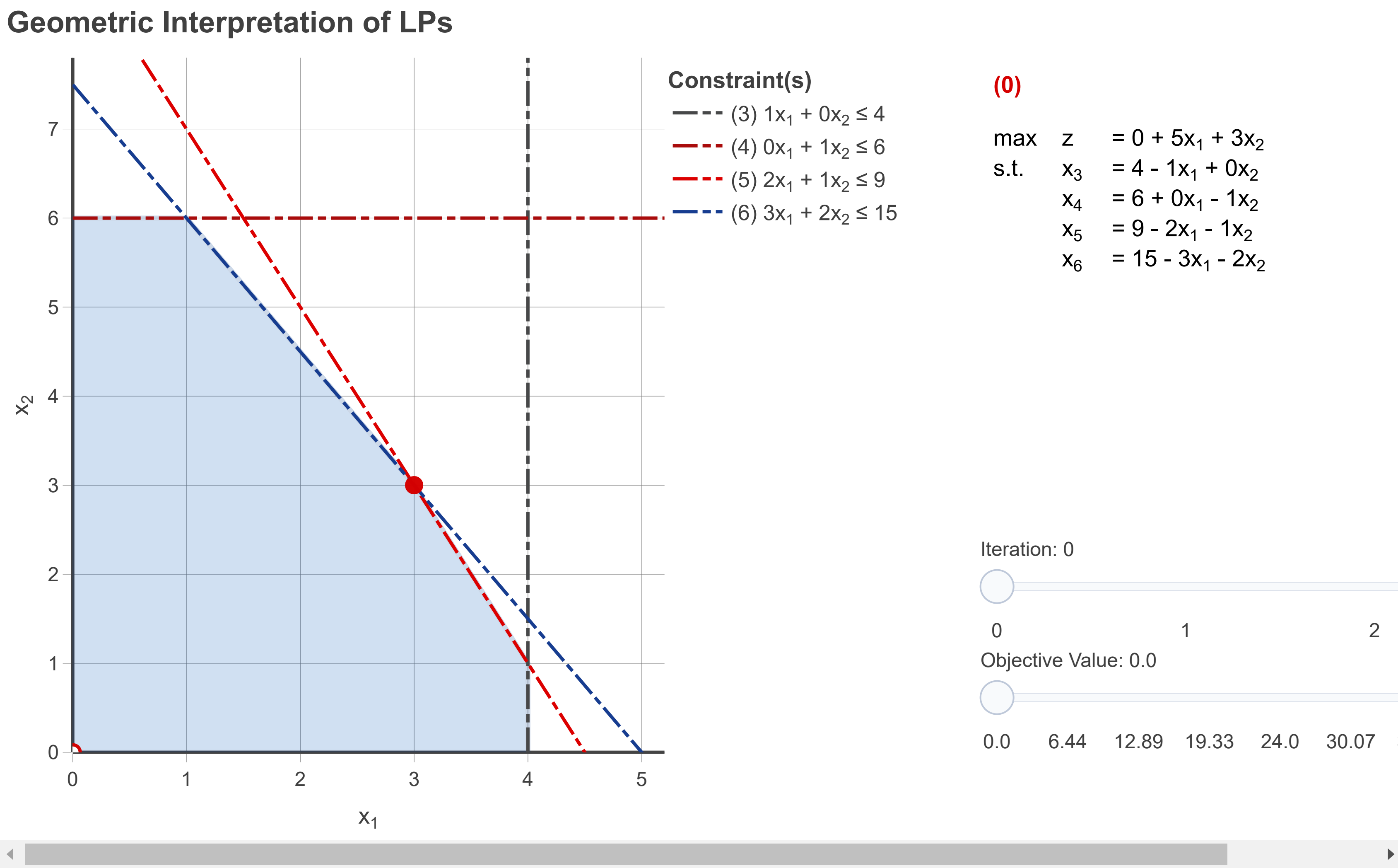
**A:** max 20- 5x3 + 3x2, s.t. x1 = 4-x3, x4 = 6-x2, x5 = 9-2(4-x3)-x2, x6=15-3(4-x3)-2x2; x1, x2, x3, x4, x5, x6>=0

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

**A:** x1 = 4, x2 = 1, x3 = 0, x4 = 5, x5 = 0, x6 = 1. Objective value: 23.

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 [4]:
```
lp = gilp.examples.LIMITING_CONSTRAINT_2D_LP # get the LP example
gilp.simplex_visual(lp, initial_solution=np.array([[0],[0]])).show() # show 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:** x cannot be greater than 4. This matches what I found algebraicallly.

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

**A:** x2: it has a positive coefficient 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:** x2 cannot be greater than 1. x1 = 4, x2 = 1, x3 = 0, x4 = 5, x5 = 0 x6 = 1. Objective value: 23.

**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.

**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 cannot be greater than 1. x1 = 3, x2 = 3, x3 = 1, x4 = 3, x5 = 0, x6 = 0. Objective value: 24.

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

**A:** Yes, both variables on the RHS of the objective value have negative coefficients.

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

**A:** Graph the constraints in terms of the two variables in the objective function.

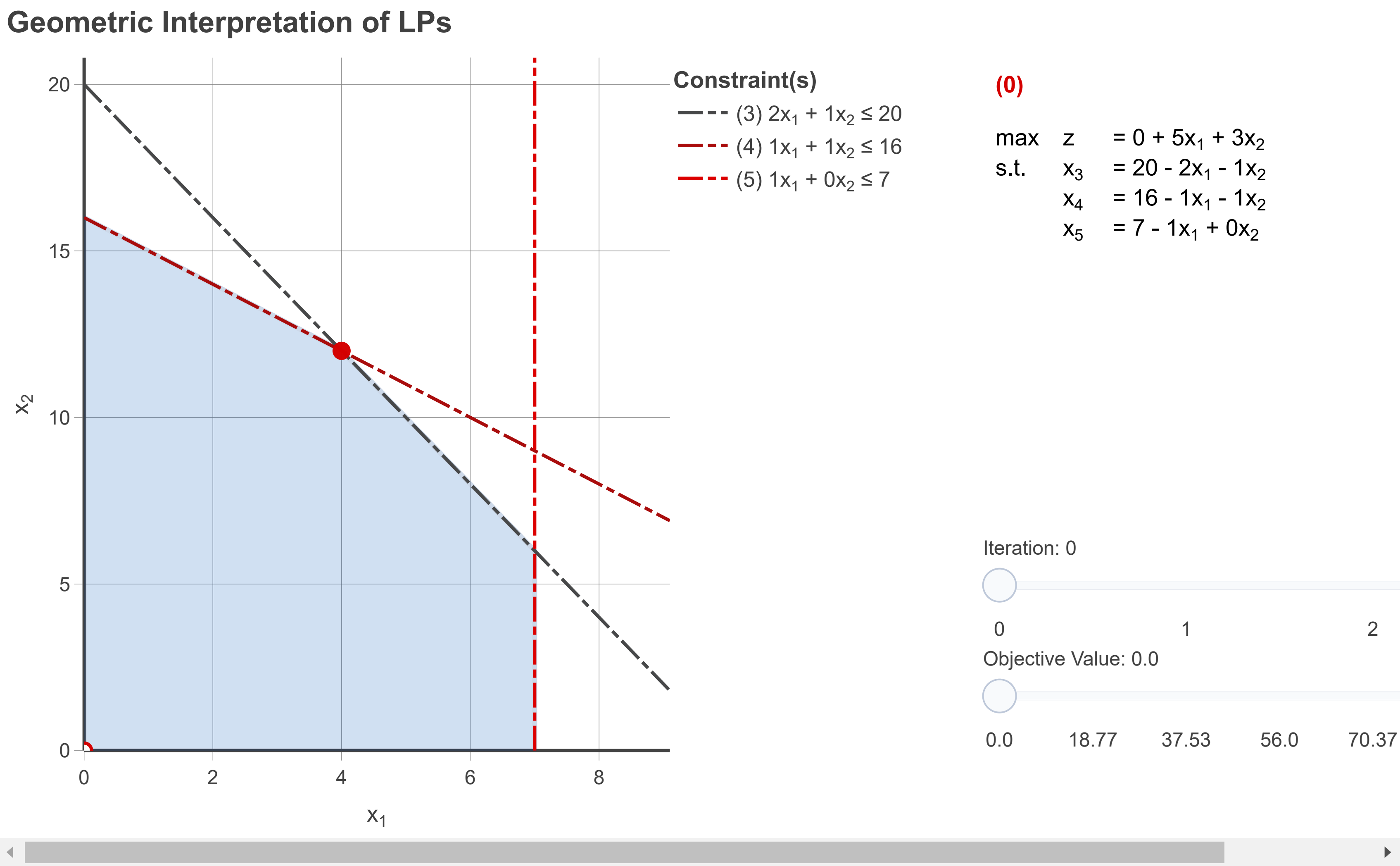
**(BONUS):** In 2D, we can increase a variable until we hit a 2D line representing the most limiting constraint. What would be the analagous situation in 3D?

**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 [5]:
```
lp = gilp.examples.ALL_INTEGER_2D_LP # get LP example
gilp.simplex_visual(lp, initial_solution=np.array([[0],[0]])).show() # visualize 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:** x1 = 7, x2 = 6, x4 = 3. Objective value: 53.

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:** x3 represents the slack in the initial constraint.

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

**A:** 0, 3, 0 corresponds to the slacks in the constraints represented by x3, x4, x5.

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 is on the line that represents the 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:** x3 and x5 are not in the dictionary. They represent the slacks of the constraints 20 >= 2x1 + x2 and 16 >= x1 + x2.

**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:** x3 and x5 = 0. In the feasible solution at (7, 6), there is no slack in these constraints.

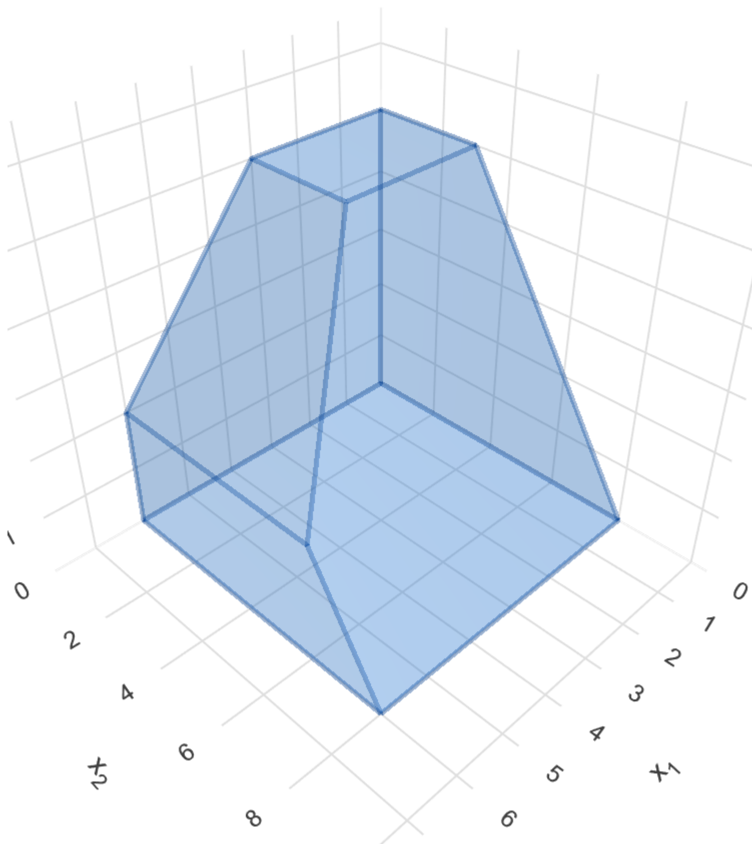
**Q33:** Look at some other corner points with this in mind. What do you find?

**A:** The same thing holds true for other corner points.

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

In [6]:
```
lp = gilp.examples.ALL_INTEGER_3D_LP # get LP example
gilp.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) $0x_1 + 1x_2 + 1x_3 \leq 8$

Objective Value: 0.0

0.0 7.12 14.24 21.37 28.49 33.23 4(

**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:** The point is on the planes representing these constraints.

**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 constraints that correspond to nondictionary variables are planes that include the faces of the solid around the point.

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

**A:** Corner points.

# Part IV: Choosing an Increasing Variable

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.

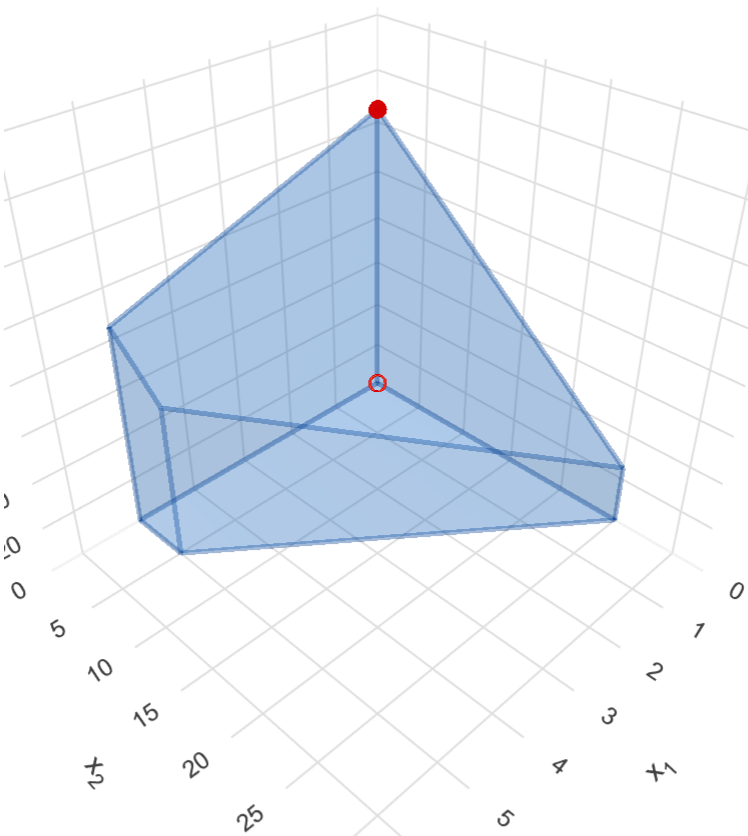
$$
\begin{array}{ll}
\max & 4x_1 + 2x_2 + x_3 \\
\text{s.t.} & x_1 \leq 5 \\
& 4x_1 + x_2 \leq 25 \\
& 8x_1 + 4x_2 + x_3 \leq 125 \\
& x_1, x_2, x_3 \geq 0.
\end{array}
$$

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 [7]:
```
klee_minty = gilp.examples.KLEE_MINTY_3D_LP
```

In [8]:
```
gilp.simplex_visual(klee_minty, rule='dantzig', initial_solution=np.array([[0],[0],[0]])).show()
```

## 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)**

$$
\begin{array}{lll}
\max & z & = 0 + 4x_1 + 2x_2 + 1x_3 \\
\text{s.t.} & x_4 & = 5 - 1x_1 + 0x_2 + 0x_3 \\
 & x_5 & = 25 - 4x_1 - 1x_2 + 0x_3 \\
 & x_6 & = 125 - 8x_1 - 4x_2 - 1x_3
\end{array}
$$

Iteration: 0

0 1 2 3 4 5

Objective Value: 0.0

0.0 44.09 88.17 125.0 165.33

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

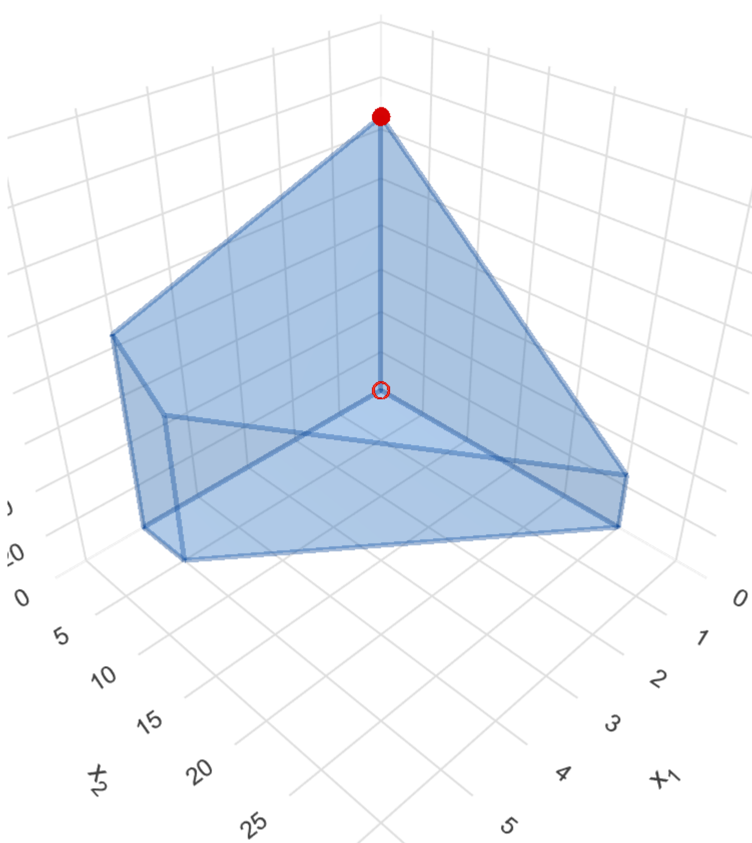
**A:** It zig-zags across the edges to the corner point representing the optimal solution.

Above, we used a 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 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 rule! Again, look at the dictionary form LP at every iteration.

In [12]:
```
gilp.simplex_visual(klee_minty, rule='bland', initial_solution=np.array([[0],[0],[0]])).show()
```

## 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)**

$$\begin{array}{lll} \text{max} & z & = 0 + 4x_1 + 2x_2 + 1x_3 \\ \text{s.t.} & x_4 & = 5 - 1x_1 + 0x_2 + 0x_3 \\ & x_5 & = 25 - 4x_1 - 1x_2 + 0x_3 \\ & x_6 & = 125 - 8x_1 - 4x_2 - 1x_3 \end{array}$$

Iteration: 0

0 1 2 3

Objective Value: 0.0

0.0 44.09 88.17 125.0 165.33

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

**A:** Using Bland's rule, the simplex moves more directly across the edges to the corner point representing the optimal solution.

Can you do any better? By setting `rule='manual'` , 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:** I can do 1 iteration (pick x3). I can find 7 paths.

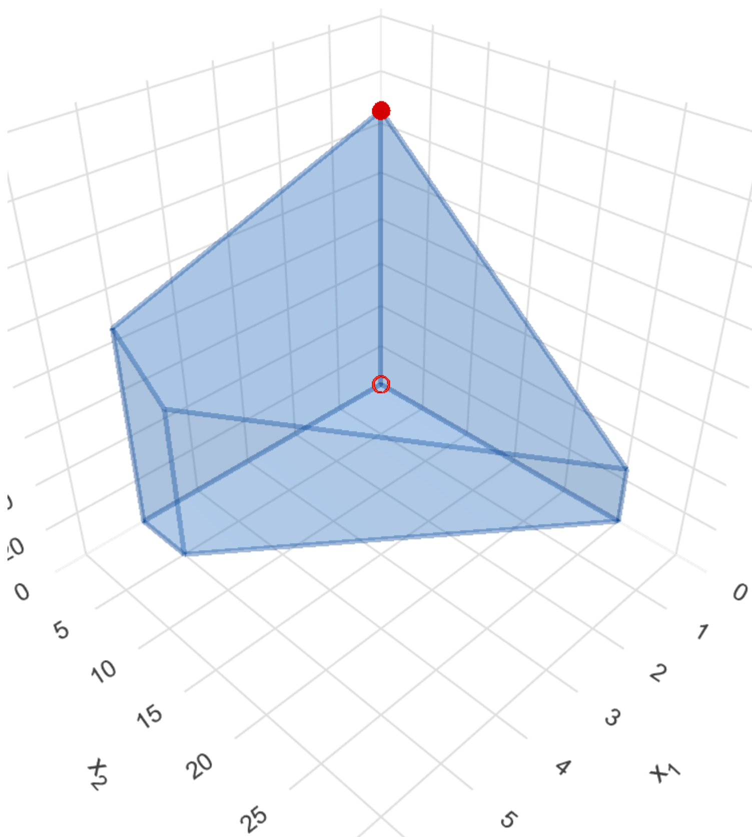
In [15]:
```
gilp.simplex_visual(klee_minty,rule='manual', initial_solution=np.array([[0],[0],[0]])).show()
```

```
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]1
Pick one of [2, 3]3
Pick one of [4]4
```

## 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 | $z$ | $= 0 + 4x_1 + 2x_2 + 1x_3$ |
| s.t. | $x_4$ | $= 5 - 1x_1 + 0x_2 + 0x_3$ |
| | $x_5$ | $= 25 - 4x_1 - 1x_2 + 0x_3$ |
| | $x_6$ | $= 125 - 8x_1 - 4x_2 - 1x_3$ |

Iteration: 0

0 1 2

Objective Value: 0.0

0.0 44.09 88.17 125.0 165.33

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

**A:** The choice of increasing variable corresponds to a plane representing a constraint.

**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:** After increasing x1 and x3, 'rule='manual_select'' only let me increase x4. I could not increase x2 or x6 because they had negative coefficients in the objective function.

# Part V: Creating LPs in GILP (Optional)

We can also create our own LPs! Let us create the following LP.

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

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 = gilp.LP(A,b,c)
```

Let's visualize it!

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