How to Solve a Staff Scheduling Problem with Python

Minimize the number of workers per shift while assigning enough workers for each time window

Khuyen Tran Jun 5 · 5 min read ★

Motivation

Imagine you are a manager of a coffee shop. Your coffee shop opens 24h daily. The daily schedule is divided into 8 **time windows** as shown in the table below. Each time window requires a different amount of staff.

Data obtained from Applied Integer Programming by Chen, D.-S., Batson, R. G., & Dang, Y. (2010)

Staff members need to be scheduled into 4 different **shifts** like below:

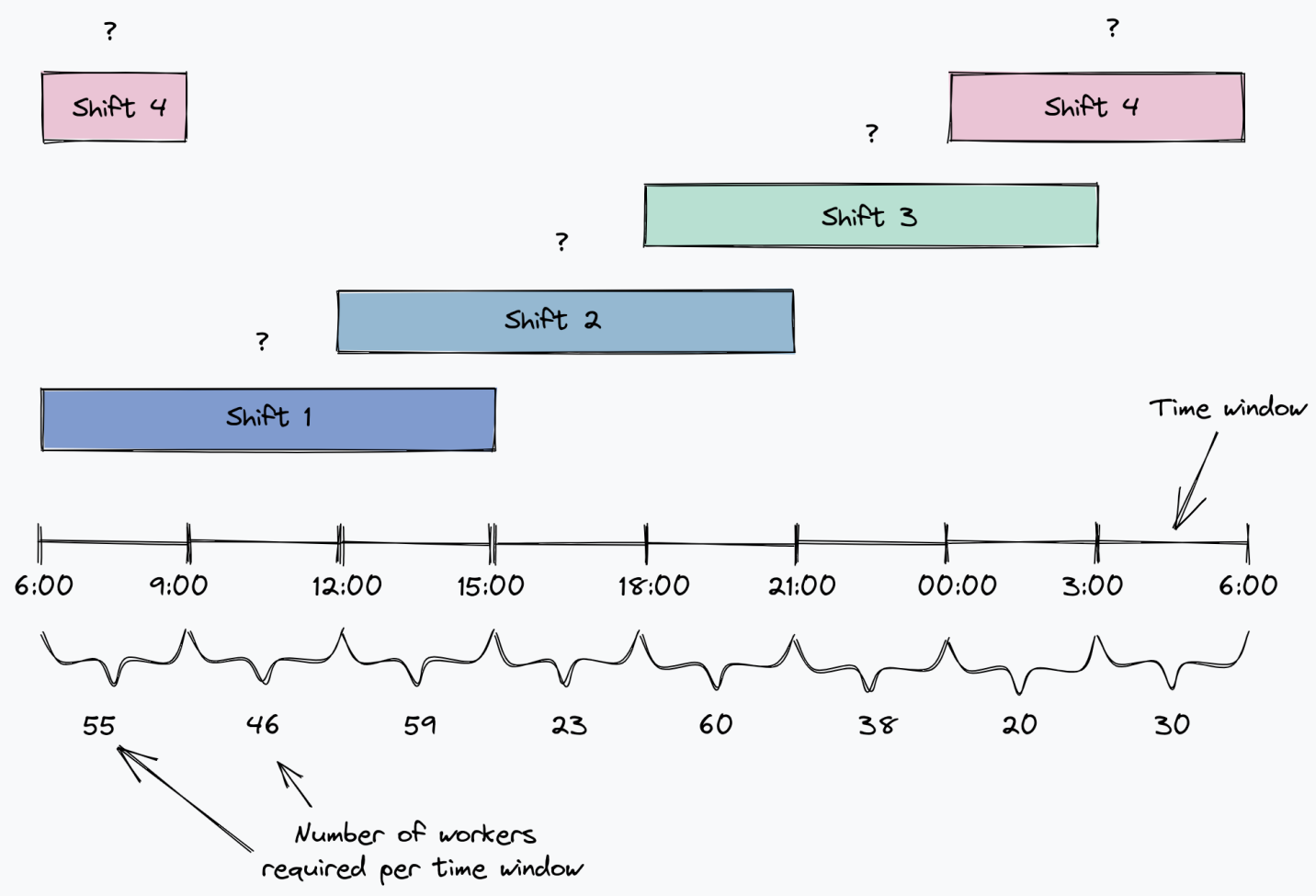

Image by Author

How can you decide how many staff workers needed per shift?

It is easy, isn't it?

You might say "That is easy! I will pick the highest number of demands among the three time windows in one shift. For example, since my coffee shop needs 55 workers from 6:00 to 9:00, 46 workers from 9:00 to 12:00, and 59 workers from 12:00 to 15:00, I will assign 59 workers from 6:00 to 15:00."

This solution works, but it is **not optimal**. Since there are some time windows that **workers from different shifts work together**, you might need fewer workers per shift

than you think.

Image by Author

But what is an optimal solution? From a manager's point of view, an optimal solution is to **minimize the number of workers per shift** to save money while still **assigning enough workers for each time window**.

Instead of spending hours trying to figure this out, let's utilize your Python skill to find the optimal solutions for this problem.

Introduction to PuLP

Linear programming (LP) is one of the best methods to find optimal solutions for problems with constraints like the above. PuLP is a Python library that makes it easy to apply linear programming using Python.

To install PuLP, type:

```
pip install pulp
```

Now's let download the data mentioned at the beginning of the article from Google Drive using gdown:

```
pip install gdown
```

Data obtained from Applied Integer Programming by Chen, D.-S., Batson, R. G., & Dang, Y. (2010)

Problem Statements

Input Parameters

First, let's create a matrix to show which shift each time window is associated with.

Image by Author

Some other helpful information to write down:

Image by Author

Decision Variables

Decision variables are unknown quantities that we want to solve for. In our example, the decision variable is the number of workers per shift.

Image by Author

To specify the decision variables in PuLP, use `LpVariable.dicts(name, list_of_variables, lowBound, upBound, cat)` .

- `name` : The name of the variable
- `lowBound` : The lower bound on this variable's range. Default is negative infinity
- `upBound` : The upper bound on this variable's range. Default is positive infinity
- `cat` : The category this variable is in, `Integer` , `Binary` or `Continuous` (default)

For example, since the number of workers per work shift needs to be an integer and needs to be greater than 0 we write:

Objective

Linear programming aims to either minimize or maximize some numerical values such as costs, profit, etc. In this problem, we want to minimize the cost of wages paid to all workers.

To create a PuLP problem with the objective to minimize, use `LpProblem(name, LpMinimize)` .

Use `LPMaximize` if the objective is to maximize.

Formulation

We want to minimize the amount of money spent on all workers in a day. Note that workers in different shifts get paid at different rates (.i.e, night-time workers often get paid more than day-time workers).

Image by Author

The demand within each time window t also needs to be satisfied.

Image by Author

Cool! Now that we have written the constraints and the objectives, we are ready to solve the problem!

Solve

```
Status: Optimal
```

Yay! The status is `optimal` when running `prob.solve()` ! This means that the solver found the optimal solutions!

Let's find out what those optimal solutions are:

Image by Author

Cool!

Interpretation

Let's visualize our results to see if they make sense.

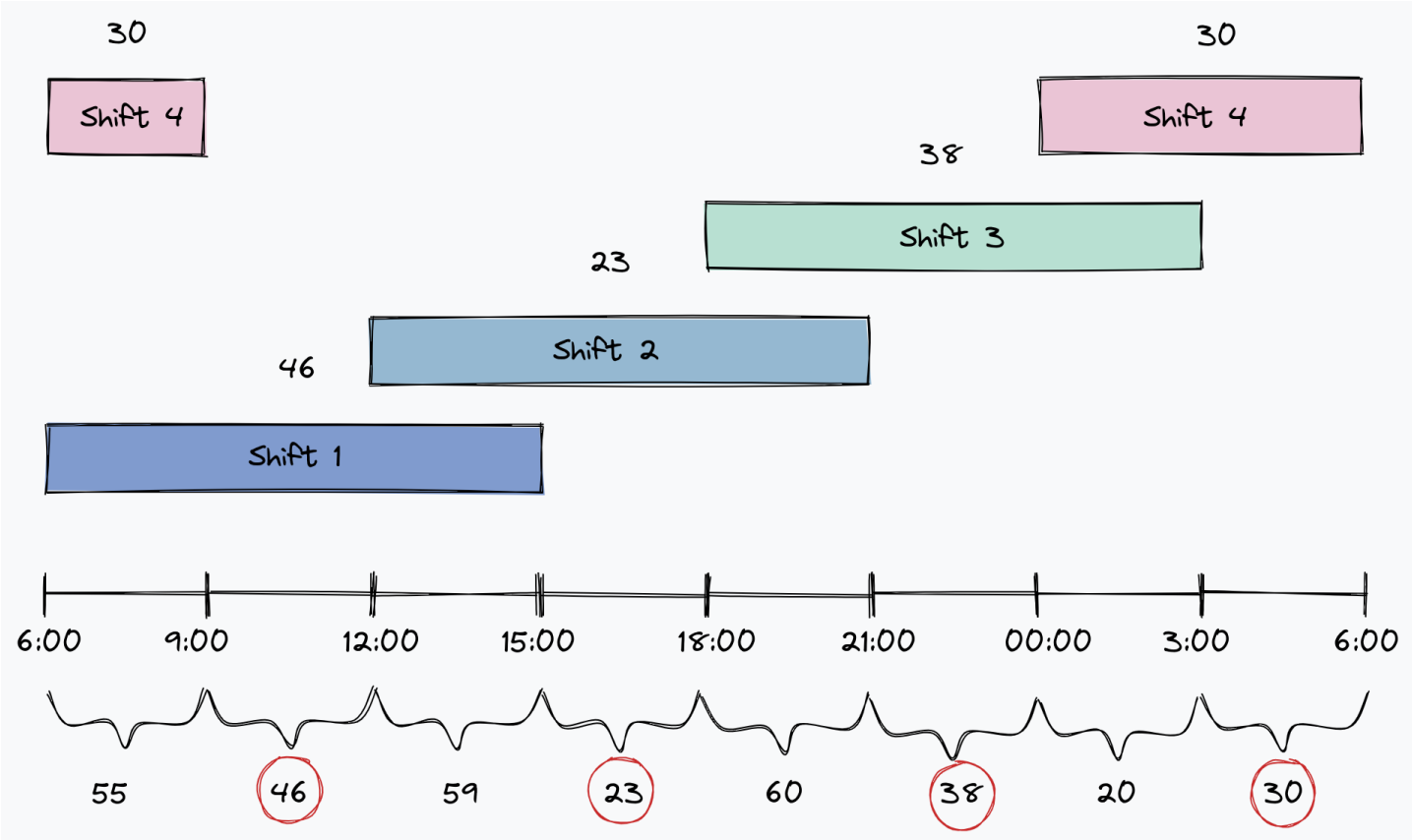

Image by Author

The coffee shop needs 55 workers from 6:00 to 9:00 and 59 workers from 12:00 to

15:00. How does the coffee shop meet these demands if the solver only assigns 46 workers to shift 1 (from 6:00 to 15:00)?

It is because there are some time windows that workers from **different shifts work together**. Let's calculate exactly how many workers are in each time window.

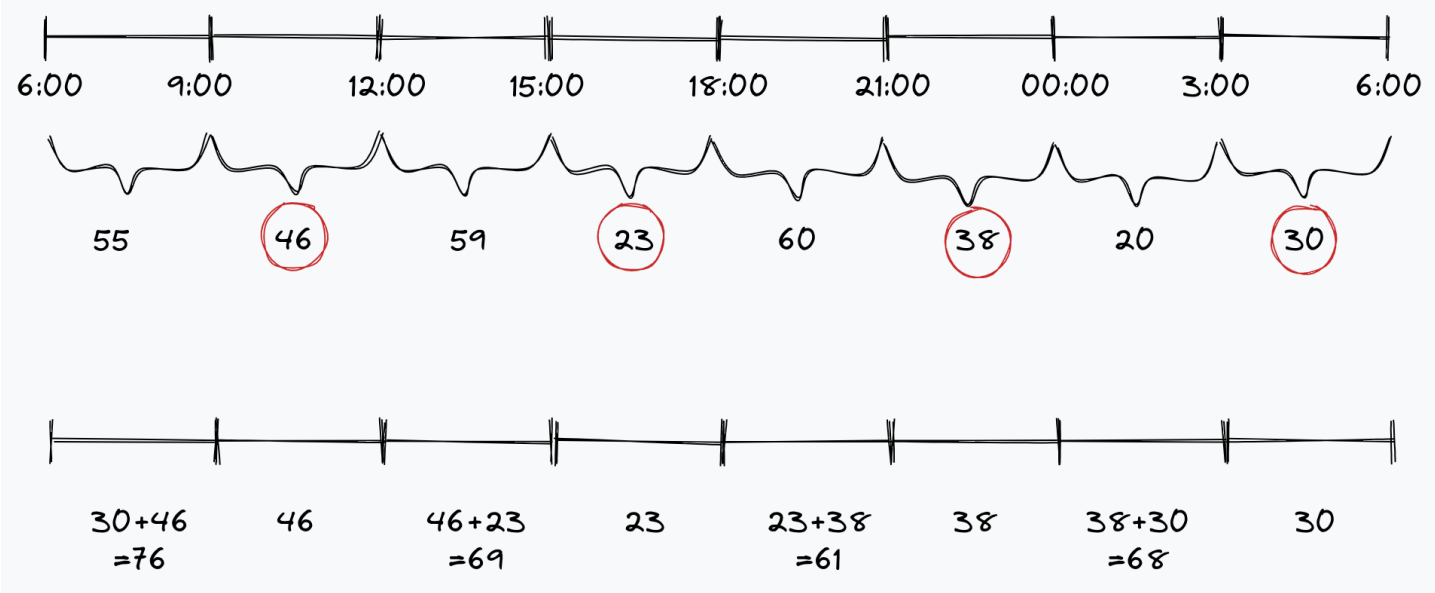

Image by Author

Aha! From the calculation above, we can see that there are enough workers to meet the demand in each time window. How cool is that?

Conclusion

Congratulations! You have just learned how to solve an optimization problem using PuLP. I hope this article will give you the motivation to utilize your Python skills to solve similar problems.

This might seem intuitive, but when your problem gets bigger, it will be much easier to solve the problem using tools like PuLP.

Feel free to fork and play with the code for this article in this Github repo:

khuyentran1401/Data-science

Collection of useful data science topics along with code and articles - khuyentran1401/Data-science

github.com

I like to write about basic data science concepts and play with different algorithms and data science tools. You could connect with me on LinkedIn and Twitter.

Star this repo if you want to check out the codes for all of the articles I have written. Follow me on Medium to stay informed with my latest data science articles like these:

Simulate Real-life Events in Python Using SimPy

towardsdatascience.com

Maximize your Productivity with Python

You create a to-do list to be productive but end up wasting your time on non-important tasks. What if you could create...

towardsdatascience.com

How to Find a Best Match with Python

Provided Individual Preferences, how to Match so that the Total Preference is Maximized?

towardsdatascience.com

Python Clean Code: 6 Best Practices to Make your Python Functions more Readable

Stop Writing Python Functions that Take more than 3 Minutes to Understand

towardsdatascience.com

Reference

Chen, D.-S., Batson, R. G., & Dang, Y. (2010). *Applied integer programming: modeling and solution*. J. Wiley & Sons.