Assignment 3: Oil Production Optimization With Gurobi Report

Group Samkok

Present

Asst. Prof. Dr. Diew Koolpiruck

Member

Pattarapon Buathong	62070504012
Suchada Panyawiraphat	62070504016
Pawornrat Supavit Pranayanuntana	62070504030

This report is a part of INC364 Manufacturing Execution System

2nd Semester, Academic Year 2021

Department of Control System and Instrumentation Engineering

King Mongkut's University of Technology Thonburi

Contents

Assignment III

Assume that you are working in a Thai's well-known crude oil company. They ask you to develop a five-year operating plan for the gulf of Thailand which includes four offshore oil rigs in it.They are limited to operating a maximum of three rigs in this gulf each year. Even if a rig does not operate in a given year, the company is still required to pay royalties on it if there is a reasonable expectation that it will operate in the future. Otherwise, it may be permanently closed with no further royalties due.
The following table summarizes the annual royalties due on each active rig (whether operating or not):

Rig IDs	**Royalties (Baht)**
Rig A	13,085,600
Rig B	16,357,000
Rig C	13,085,600
Rig D	16,357,000

Each Rig is limited to drilling a certain amount of crude oil per year and the quality of it (measured in API gravity) is different. These restrictions include the following:

Rig IDs	**Max Production (barrel)**	**API Gravity**
Rig A	1.9 Million	21
Rig B	1.5 Million	30
Rig C	1.0 Million	45
Rig D	2.3 Million	15

Each year, the crude oil drill from each operating rate must be combined to produce a specified grade of crude oil. The combined crude oil's annual requirements are as follows:

Year	**Expected API Gravity**
1	27
2	24
3	36
4	18
5	30

The combined crude oil is sold at a price 75 baht per barrel.

Due to the financial economic crisis, the company's revenue from crude oil sales is forecast to be discounted at a rate of 5% per annum.The cost is assumed to be increased at the rate of 2% per annum due to inflation.

You are assigned to give a plan according to the requirements that which rigs should be operated annually. How much crude oil should be drilled from each rig? How much the expected revenue gain from your plan?

Linear Programming Model

Decision Variables

$$x_i = Oil\ Rig\ Usage\ ;\ 0 = Not\ being\ used,\ 1 = being\ used$$

$$y_i = Barrels\ Produced$$

Constraints

$$x_1 + x_2 + x_3 + x_4 \leq 3$$

$$x_1 y_1 \leq 1,900,000$$

$$x_2 y_2 \leq 1,500,000$$

$$x_3 y_3 \leq 1,000,000$$

$$x_4 y_4 \leq 2,300,000$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 27$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 24$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 36$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 18$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 30$$

Objective Function

$$Maximize:\ x_1 y_1 + x_2 y_2 + x_3 y_3 + x_4 y_4$$

Optimization Result using Gurobi

Best Objective

```
Solution count 5: 1.99417e+07 1.935e+07 1.90167e+07 ... -0

Optimal solution found (tolerance 1.00e-04)
Best objective 1.994166666667e+07, best bound 1.994166666667e+07, gap 0.0000%
Total produced = 19941666.666666664 barrels
```

Figure 1 Optimal solution (objective function) calculated by Gurobi

It means that the optimized summation of five years production is equal to roughly 1994166.67 barrels.

Oil Production

We exported the solution variable after Gurobi optimization as the CSV file. Thus, we can see the optimal solution of each decision variable easily.

In the first year, the expected API gravity is 27. The production planning for each rig are: RigA: 0 barrels (did not use the rig), RigB: 1,500,000 barrels, RigC: 1,000,000 barrels, and RigD: 1,875,000 barrels. The total oil production in this year is 4,375,000 barrels.

Crude Oil Production[2015/01/01,RigA]	0
Crude Oil Production[2015/01/01,RigB]	1500000
Crude Oil Production[2015/01/01,RigC]	1000000
Crude Oil Production[2015/01/01,RigD]	1875000

Figure 2 First year (2015) production planning for each rig

In the second year, the expected API gravity is 24. The production planning for each rig are: RigA: 1,900,000 barrels, RigB: 0 barrels (did not use the rig), RigC: 1,000,000 barrels, and RigD: 1,700,000 barrels. The total oil production in this year is 4,600,000 barrels.

Crude Oil Production[2016/01/01,RigA]	1900000
Crude Oil Production[2016/01/01,RigB]	0
Crude Oil Production[2016/01/01,RigC]	1000000
Crude Oil Production[2016/01/01,RigD]	1700000

Figure 3 Second year (2016) production planning for each rig

In the third year, the expected API gravity is 36. The production planning for each rig are: RigA: 0 barrels (did not use the rig), RigB: 1,500,000 barrels, RigC: 1,000,000 barrels, and RigD: 0 barrels (did not use the rig). The total oil production in this year is 2,500,000 barrels.

Crude Oil Production[2017/01/01,RigA]	0
Crude Oil Production[2017/01/01,RigB]	1500000
Crude Oil Production[2017/01/01,RigC]	1000000
Crude Oil Production[2017/01/01,RigD]	0

Figure 4 Third year (2017) production planning for each rig

In the fourth year, the expected API gravity is 18. The production planning for each rig are: RigA: 1,900,000 barrels, RigB: 100,000 barrels, RigC: 0 barrels (did not use the rig), and RigD: 2,300,000 barrels. The total oil production in this year is 4,300,000 barrels.

Crude Oil Production[2018/01/01,RigA]	1900000
Crude Oil Production[2018/01/01,RigB]	100000
Crude Oil Production[2018/01/01,RigC]	0
Crude Oil Production[2018/01/01,RigD]	2300000

Figure 5 Fourth year (2018) production planning for each rig

In the fifth year, the expected API gravity is 30. The production planning for each rig are: RigA: 1,666,666.67 barrels, RigB: 1,500,000 barrels, RigC: 1,000,000 barrels , and RigD: 0 barrels (did not use the rig). The total oil production in this year is 4,166,666.67 barrels.

Crude Oil Production[2019/01/01,RigA]	1666667
Crude Oil Production[2019/01/01,RigB]	1500000
Crude Oil Production[2019/01/01,RigC]	1000000
Crude Oil Production[2019/01/01,RigD]	0

Figure 6 Fifth year (2019) production planning for each rig

	Date	Royalties	Raw Rev	Pre adj Rev	Adjusted Rev
0	2015/01/01	45799600	328125000.0	3.281250e+08	2.823254e+08
1	2016/01/01	43378764	345000000.0	3.277500e+08	2.843712e+08
2	2017/01/01	30632080	187500000.0	1.692188e+08	1.385867e+08
3	2018/01/01	48602901	322500000.0	2.765034e+08	2.279005e+08
4	2019/01/01	46033890	312500000.0	2.545332e+08	2.084993e+08

Figure 7 Royalties, Raw Revenue, Pre-adjusted Revenue, and Adjusted Revenue in DataFrame.

Raw Revenue

The raw revenue is calculated by multiplying the barrels of oil produced in that year by the price of the oil per barrel (75 Baht).

$$Raw\ Revenue\ =\ (Oil\ barrels\ produced)(75)$$

First year: 328,125,000 Baht
Second year: 345,000,000 Baht
Third year: 187,500,000 Baht
Fourth year: 322,500,000 Baht
Fifth year: 312,500,000 Baht

Adjusted Revenue

The adjusted revenue is calculated by

$$Adjusted\ Revenue\ =\ [(Raw\ Revenue)(1\ -\ 0.05)^{year-1}]\ -\ (Royalties)(1\ +\ 0.02)^{year-1}$$

Year 1: 282,325,400 Baht
Year 2: 284,371,200 Baht
Year 3: 138,586,700 Baht
Year 4: 227,900,500 Baht
Year 5: 208,499,300 Baht

Datapane Graphs

We vertically concatenated four graphs from the Python program. From the top most, the first graph shows the production planning for each rig every year, if we follow this plan, we will produce the maximum oil barrel according to the optimizer by Gurobi. The second graph shows total oil production for each year. The third graph shows raw revenue for each year. The fourth graph shows the adjusted revenue calculated by adjusting the raw revenue to the inflation (5%) and subtracting the cost (royalties) that rise by 2% every year. The following link leads to the interactive graph:
https://datapane.com/u/heartnoxill/reports/mA2EMak/oil-production-model-4/

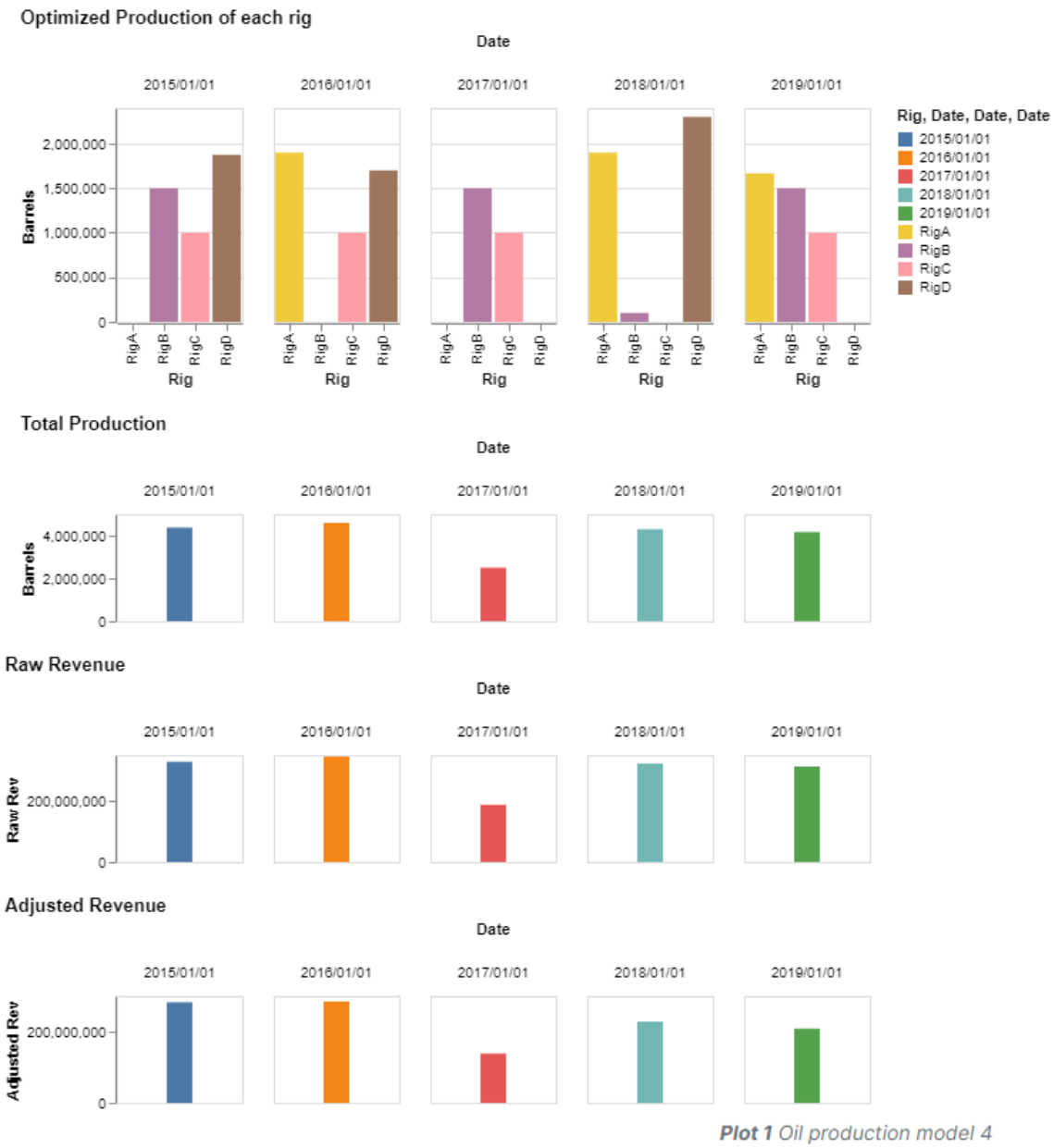

Figure 8 Bar graphs rendered using Altair and uploaded through Datapane

Gantt Chart

*PB = Pattarapon Buathong, SP = Suchada Panyawiraphat, PSP = Pawornrat Supavit Pranayanuntana

Level 1	Level 2	Level 3	Responsibility	Week 1							Week 2							Week 3
				21 Feb	22 Feb	23 Feb	24 Feb	25 Feb	26 Feb	27 Feb	28 Feb	1 Mar	2 Mar	3 Mar	4 Mar	5 Mar	6 Mar	7 Mar
Oil Production Optimization	Understand and define problems	Define problem	PB, SP, PSP															
		Define scope	PB, SP, PSP															
		Define programming design	PB, SP, PSP															
		Brainstorming	PB, SP, PSP															
	Define Linear Programming Mathematic Model	Define decision variable	PB															
		Define constriants	SP, PSP															
		Define objective function	SP, PSP															
	Program Developing	Learning programming examples	PB, SP, PSP															
		Define program flow	PB, SP, PSP															
		Programming	PB															
		Debug and evaluate	SP, PSP															
	Report and presentiation making	Summarizing results	PB, SP, PSP															
		Create report	SP, PSP															
		Create presentation	SP, PSP															
		Submit report and prepare for presentation	PB, SP, PSP															

Figure 9 Gantt chart

Discussion

Many problems occurred during the assignment which are part of two categories: the linear programming mathematical model and the coding. We had no idea what the decision variables should be when we first read the problem. It took some research to realize the variable for the rigs could be binary (0 and 1) as well, and it can be used to determine the availability of something. Hence, we use binary as one of our decision variables. For the coding part, many problems occurred during the development. First, some of our constraints are below:

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 27$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 24$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 36$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 18$$

$$\frac{(21)x_1y_1 + (30)x_2y_2 + (45)x_3y_3 + (15)x_4y_4}{\Sigma x_i y_i} \leq 30$$

We have to do the sum of the x and y variables and divide the whole numerator term. Surprisingly, Gurobi cannot perform a divide operation. Therefore, we fixed the problem by multiplying both sides of the constraints by the denominator term and changed the Gurobi coding part from dividing to multiplying instead. To sum multiple numerator terms above, we used the gurobipy.quicksum method, but if we only want to sum single term, e.g., x_1y_1 (Rig 1), we will use the .sum method. Second, after getting the optimization solution and wanting to print out the solution. If we use the ordinary print() method, it will only print the head and the tail of the DataFrame. We fixed this problem by using the Pandas package to export the DataFrame as the CSV file, so we can look at it in Excel, which we think is easier than showing the whole DataFrame in JupyterNotebook. Third, after getting and understanding the solutions, we will use Altair to render the graph and publish it through Datapane. The problem we faced is that Altair won't receive the same DataFrame for multiple bar graphs, it might have happened because we did some DataFrame modifications for each graph. To fix this, we just separated the source DataFrame into 4 sources for our 4 graphs. You can see from the code that there are sources for Oil Each Rig Produced, Total Oil Produced, Raw Revenue, and Adjusted Revenue graph, respectively. By doing this, we think it might slow down our program since it has to handle four different DataFrames. Although we did not measure the time and complexity, it is obvious that this should be improved if possible. Lastly, this is related to production planning. In the fifth year, our plan for Rig A is to produce 1,666,666.67 barrels, which may be impossible to produce accurately in real life. Therefore, this could be concerned as one of the problems we can research further.

Conclusion

Oil production optimization with Gurobi assignment involves finding a linear programming mathematical model, coding in Python, and summarizing the optimized solution. We started with the linear programming mathematical model including decision variables, constraints, and objective function. All of these are created according to the problem information, for example, constraints are given as the limits of the rig and the demands of the oil API gravity, decision variables are in the problem as we could not operate every oil rigs at the same time, and objective function is obviously to maximize the production, so that it will maximize our profit. Next is the coding in Python part, we divided the code into three large parts: Gurobi Optimizations, Altair and Datapane Graph Plotting, and Executing Requirements and Informations. The first part of the code, Gurobi Optimizations, we transformed all of our decision variables, constraints, and objective function to the term that Gurobi accepts. The DEFINE VARIABLES part consists of the variables in decision variables and constraints. The CONSTRAINTS part consists of the constraints, and the multiplied term that we looked at as one bigger term. The DEFINE MODEL part consists of our goal that is to maximize the objective function, that is to maximize the oil production in five years. You can see from the “objective” variable that we added up the five years of oil produced and fed it to the Gurobi. The SOLVE MODEL receives the “objective” variable and will auto-optimize for the solution. Lastly, we return the solution (the optimized variables), and model (optimized objective function) variables for this Python function. The second part of the code, Altair and Datapane Graph Plotting, we pulled the solution DataFrame, and did some modifications of the columns. The Altair package is used for rendering the graph, in this case, there are four bar graphs that will be vertically concatenated. After that, the graphs will be published through the Datapane package. Therefore, our graph will be on the Datapane website and it is also interactive. The last part of the code, Executing Requirements and Informations, we declared the constants from the problem in there. Then we proceed to call both Gurobi Optimizations, and Altair and Datapane Graph Plotting. Consequently, we will receive all of our requirements for the production planning and report making. We can prepare the production planning by looking at the CSV file or the Datapane graph, both will show the optimal value for each rig, and whether this rig will be operated this year or not.

Appendix (Code)

```
import gurobipy
import numpy as np
import pandas as pd
import altair as alt
import datapane as dp
from typing import List, Dict
def optimize_planning(timeline: List[str],
                      rigs: List[str],
                      yearly_requirements: Dict[str, int],
                      production_per_rig: Dict[str, int],
                      API_cap: Dict[str, int]) -> pd.DataFrame:

    model = gurobipy.Model("Optimize production planning")

 ###################################################################

    # DEFINE VARIABLES
    line_opening = model.addVars(
        timeline, rigs, vtype=gurobipy.GRB.BINARY, name="Use status"
    )

    oil_produced = model.addVars(
        timeline,
        rigs,
        vtype=gurobipy.GRB.CONTINUOUS,
        name="Crude Oil Production",
    )

    sum_of_oil_produced = model.addVars(
        timeline,
        vtype=gurobipy.GRB.CONTINUOUS,
        name="Crude Oil Divide",
    )

    API_Mul_Oil = model.addVars(
        timeline,
        rigs,
        vtype=gurobipy.GRB.CONTINUOUS,
        name="API Mul Oil Produced",
    )


###################################################################

    # CONSTRAINTS
    model.addConstrs(
        (
            oil_produced[(date, rig)] <= production_per_rig[rig] *
line_opening[(date, rig)]
```

```
            for date in timeline
            for rig in rigs
        ),
        name="Oil produced - Barrels per year",
    )

    model.addConstrs(
        (
            line_opening.sum(date, "*") <= 3
            for date in timeline
        ),
        name="Restricted to 3 rigs per year",
    )

    model.addConstrs(
        (
            oil_produced.sum(date, '*') == sum_of_oil_produced[(date)]
            for date in timeline
        ),
        name="Divide of oil produced",
    )

    model.addConstrs(
        (
            API_Mul_Oil[(date, rig)] == API_cap[rig]*oil_produced[(date, rig)]
            for date in timeline
            for rig in rigs
        ),
        name="API Mul Oil",
    )

    model.addConstrs(
        (
            gurobipy.quicksum(API_Mul_Oil[(date, rig)] for rig in rigs)
            == yearly_requirements[date] * sum_of_oil_produced[(date)]
            for date in timeline

        ),
        name="Oil produced API - Requirement",
    )

##################################################################

    # DEFINE MODEL
    # Objective : maximize a function
    model.ModelSense = gurobipy.GRB.MAXIMIZE

    # Function to maximize [maximize the sum]
    optimization_var = gurobipy.quicksum(
        oil_produced[(date, rig)] for date in timeline for rig in rigs
    )
    objective = 0
```

```
    objective += optimization_var

################################################################

    # SOLVE MODEL
    model.setObjective(objective)
    model.optimize()
    sol = pd.DataFrame(data={"Solution": model.X}, index=model.VarName)

    print("Total produced = " + str(model.ObjVal) + " barrels")

    return sol, model
def plot_planning(planning, timeline):

    # Plot graph - Oil each rig produced
    source = planning.filter(like="Crude Oil Production",axis=0).copy()
    source["Date"] = list(source.index.values)
    source = source.rename(columns={"Solution": "Barrels"}).reset_index()
    source[["Date", "Rig"]] = source["Date"].str.split(",", expand=True)
    source["Date"] = source["Date"].str.split("[").str[1]
    source["Rig"] = source["Rig"].str.split("]").str[0]
    bars = (
        alt.Chart(source)
        .mark_bar()
        .encode(
            x="Rig:N",
            y="Barrels:Q",
            column=alt.Column("Date:N"),
            color="Rig:N",
            tooltip=["Date", "Rig", "Barrels"],
        )
        .interactive()
        .properties(
            width=550 / len(timeline) - 22,
            height=150,
            title="Optimized Production of each rig"
        )
    )

    # Plot graph - Total oil produced
    source2 = source.groupby("Date")["Barrels"].sum().reset_index()
    bars_sum = (
        alt.Chart(source2)
        .mark_bar()
        .encode(
            y=alt.Y("Barrels", axis=alt.Axis(grid=False)),
            column=alt.Column("Date:N"),
            color="Date:N",
            tooltip=["Date", "Barrels"],
        )
        .interactive()
        .properties(
```

```
            width=550 / len(timeline) - 22,
            height=75,
            title="Total Production"
        )
    )

    # Plot graph - Raw revenue
    source3 = source.groupby("Date")["Barrels"].sum().reset_index()
    source3["Raw Rev"] = source3["Barrels"]*75

    bars_rawrev = (
        alt.Chart(source3)
        .mark_bar()
        .encode(
            y=alt.Y("Raw Rev", axis=alt.Axis(grid=False)),
            column=alt.Column("Date:N"),
            color="Date:N",
            tooltip=["Date", "Raw Rev"],
        )
        .interactive()
        .properties(
            width=550 / len(timeline) - 22,
            height=75,
            title="Raw Revenue"
        )
    )

    # Plot graph - Adjusted revenue
    # raw_rev*(1-0.05)^year-1 - royalties*(1+0.02)^year-1
    source4 = planning.filter(like="Use status",axis=0).copy()
    source4["Date"] = list(source4.index.values)
    source4 = source4.rename(columns={"Solution": "Usage"}).reset_index()
    source4[["Date", "Rig"]] = source4["Date"].str.split(",", expand=True)
    source4["Date"] = source4["Date"].str.split("[").str[1]
    source4["Rig"] = source4["Rig"].str.split("]").str[0]
    source4["Year"] = source4["Date"].str.split("/").str[0]
    source4["Year"] = pd.to_numeric(source4["Year"])
    source4["Royalties"] = source4["Year"] # Dummy

    # Calculate royalities by rig and year
    for index, row in source4.iterrows():
        if(source4["Rig"][index] == "RigA"):
            source4["Royalties"][index] =
source4["Usage"][index]*(13085600)*((1+0.02)**(source4["Year"][index]-2015))
        elif(source4["Rig"][index] == "RigB"):
            source4["Royalties"][index] =
source4["Usage"][index]*(16357000)*((1+0.02)**(source4["Year"][index]-2015))
        elif(source4["Rig"][index] == "RigC"):
            source4["Royalties"][index] =
source4["Usage"][index]*(13085600)*((1+0.02)**(source4["Year"][index]-2015))
        elif(source4["Rig"][index] == "RigD"):
            source4["Royalties"][index] =
```

```
source4["Usage"][index]*(16357000)*((1+0.02)**(source4["Year"][index]-2015))

    source4 = source4.groupby("Date")["Royalties"].sum().reset_index()
    source4["Raw Rev"] = source3["Raw Rev"]
    source4["Pre adj Rev"] = source4["Raw Rev"] # Dummy

    for index, row in source4.iterrows():
        source4["Pre adj Rev"][index] *= (1-0.05)**index

    source4["Adjusted Rev"] = source4["Pre adj Rev"]-source4["Royalties"]
    print(source4)


    bars_adjustrev = (
        alt.Chart(source4)
        .mark_bar()
        .encode(
            y=alt.Y("Adjusted Rev", axis=alt.Axis(grid=False)),
            column=alt.Column("Date:N"),
            color="Date:N",
            tooltip=["Date", "Adjusted Rev"],
        )
        .interactive()
        .properties(
            width=550 / len(timeline) - 22,
            height=75,
            title="Adjusted Revenue"
        )
    )


    # Vertically Concatenate all graphs
    chart = alt.vconcat(bars, bars_sum, bars_rawrev, bars_adjustrev)
    chart.save("Oil_plan_model_4.html")

    dp.Report(dp.Plot(chart, caption="Oil production model 4")).upload(
        name="Oil production model 4",
        description="Oil production model 4 by Samkok",
        open=True,
        visibily="PUBLIC",
    )
# To get maximum oil production each year, comment other years away

# Year 1 (2015)
yearly_requirements: Dict[str, int] = {
    "2015/01/01": 27,
    "2016/01/01": 24,
    "2017/01/01": 36,
    "2018/01/01": 18,
    "2019/01/01": 30
}
```

```
yearly_requirements_df = pd.DataFrame.from_dict(yearly_requirements,
orient="index")

calendar: List[str] = list(yearly_requirements.keys())

production_per_rig = {"RigA": 1900000, "RigB": 1500000, "RigC": 1000000, "RigD":
2300000}

rigs: List[str] = list(production_per_rig.keys())

API_cap = {"RigA": 21, "RigB": 30, "RigC": 45, "RigD": 15}

solution, model = optimize_planning(
    calendar,
    rigs,
    yearly_requirements,
    production_per_rig,
    API_cap
)
# print(solution)
# solution.to_csv('Optimized_test.csv', encoding='utf-8')
plot_planning(solution, calendar)
# print(model.ObjVal)
```

Reference

Baptiste, S. (2020, October 10). *Optimization of a weekly production plan with Python and Gurobi — Part 1*. Retrieved from https://towardsdatascience.com/optimization-of-a-weekly-production-plan-with-python-and-gurobi-part-1-d1257ad29a9.

Emmanuel, J. (2016, August 10). *Integer Linear Programming | 0-1 Binary Constraints | Examples - Part 1*. Retrieved from https://youtu.be/B3biWsBLeCw.

Gurobi Optimization. (n.d). *Build Your Optimization Skills with Python Jupyter Notebook Modeling Examples*. Retrieved from https://www.gurobi.com/resource/modeling-examples-using-the-gurobi-python-api-in-jupyter-notebook/.

Najman, J. (2021, December). *How do I divide by a variable in Gurobi?*. Retrieved from https://support.gurobi.com/hc/en-us/articles/360053259371-How-do-I-divide-by-a-variable-in-Gurobi-?fbclid=IwAR2OKtr3SdZQ9txSvl_Z7Yc-ahTtdZQhtAjE iGHczRDDBRWpZd1 KGYq3Ao8.

Roman. (2013, May 10). *How to iterate over rows in a DataFrame in Pandas*. Retrieved from https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas.

ShanZhengYang. (2016, February 10). *How to sum in pandas by unique index in several columns?*. Retrieved from https://stackoverflow.com/questions/35307732/how-to-sum-in-pandas-by-unique-index-in-several-columns.

user7289. (2013, June 4). *Writing a pandas DataFrame to CSV file*. Retrieved from https://stackoverflow.com/questions/16923281/writing-a-pandas-dataframe-to-csv-file.

Zhou, Y. (2019). *How can I retrieve the values for a single variable? The variable is something like d(b,t)*. Retrieved from https://support. gurobi.com/hc/en-us/community/posts/360050125432-how-can-I-retrieve-the-values-for-a-single-variable-The-variable-is-something-like-d-b-t-.