Instituto Superior Técnico

Applied Mathematics and Computation

Integrated Project

Author

Leandro Duarte - *93112*

Professors:

José Aguilar Madeira

Maria do Rosário Oliveira

Mentor:

Sabrina Masson

Contents

1 Preface

The project is the outcome of addressing a problem in the field of Linear and Integer Optimization within the course "1st Cycle Integrated Project in Applied Mathematics and Computation" in collaboration with BI4ALL.

BI4ALL ensures consulting services in the areas of Digital Transformation and Data Strategy, focusing on Analytics, Big Data, Data Science, Artificial Intelligence, Data Visualizations, CPM and Software Engineering.

The project aims to provide a solution to a problem raised by the AI & Data Science BI4ALL department while deepening my knowledge in Linear Programming tools and acquiring some work experience in the industry in order to facilitate my future integration on the BI4ALL team.

Throughout the process helpful meetings were held with BI4ALL's team and professor José Madeira in order to outline development options and debate problem solving strategies.

The main sources of information I used to study, practice and develop this project were the Google OR-Tools developers guide and the book "Linear and Integer Optimization, Theory and Practice" by Gerard Sierksma and Yori Zwols.

The project's anticipated output is a linear optimization model that reduces the sales workforce of a company necessary to meet its customers' requests with quality. The programming language used was *Python* and *Visual Studio Code* was used as code editor.

2 Background Theory

To work on this project it was necessary to have knowledge on *Linear Optimization Problem Solving*, on *Python* programming language and libraries such as *Google OR-Tools*, *Pulp* or *Scipy*.

Linear programming problems either maximize or minimize a linear objective function subject to a set of linear equality and/or inequality constraints.

An *integer linear optimization model* is a linear optimization model in which the values of the decision variables are restricted to be integers.
Binary variables, also called $\{0, 1\}$-variables, can be used to linearize a variety of complex logical forms, such as 'either-or' constraints or 'if-then' constraints.

The problem concerning this project is an *Integer Linear Optimization problem*, therefore the optimization model will have a structure similar to:

$$\begin{aligned} max \quad & c_1x_1 + ... + c_nx_n \\ s.t. \quad & a_{11}x_1 + ... + a_{1n}x_n \leq b_1 \\ & \quad . \qquad\qquad . \quad . \\ & \quad . \qquad\qquad . \quad . \\ & \quad . \qquad\qquad . \quad . \\ & a_{m1}x_1 + ... + a_{mn}x_n \leq b_m \\ & x_1, ..., x_n \geq 0 \quad , integers \end{aligned} \tag{2.1}$$

The form can also be written as

$$max\left\{c^Tx | Ax \leq b, x \geq 0, x \quad integer\right\}$$

A vector x that satisfies all of the constraints is called a *feasible solution*. A collection of all feasible solutions is called a *feasible set*. A feasible solution x that minimizes/maximizes the objective function is called an *optimal solution*.

It is not always possible to find an optimal solution because the Linear Optimization model may be infeasible (there is no solution concerning the given constraints) or unbounded(the objective function may be improved indefinitely without violating the constraints and bounds).
Another reason why a solver might fail is because there may be a limit on the amount of time that is used to find a solution.

3 Challenge Analysis

An American company has several customers spread across a certain region. To have a close relation with all these customers this company has a sales team with full and part-time job employees assisting customers with meetups, sales and mediating when issues arise.

The company believes it has too many sales representatives and wants to reduce costs while improving costumer experience. It was assumed that this employees work remotely in different locations.

There are a few guidelines to be respected for each client:

- The distance between a sales representative's office and the customers it serves must not exceed a pre-defined maximum driving time.
- Meetings with each customer must be held online at least once a month.
- Every three months, a face-to-face meeting must be planned.

All sales reps have an associated review score which will be the major factor that will differentiate them, followed by years of experience.

No data was made available so the problem was held in a way its solution could fit any company that has a random number of clients and sales reps in a certain geographical area of the world.

4 Solution Design

In this section, the reader will be guided through the process of decision making used to design the optimization model requested and the solution will be described in a non technical manner.

4.1 Generating Data

The first obstacle to overtake was gathering data for the model's development and testing. It was decided to randomly generate both the clients and sales representatives profiles and its location within a particular geographic area.

In order to do so, profiles' characteristics were defined, according to the problem's needs, as follows:

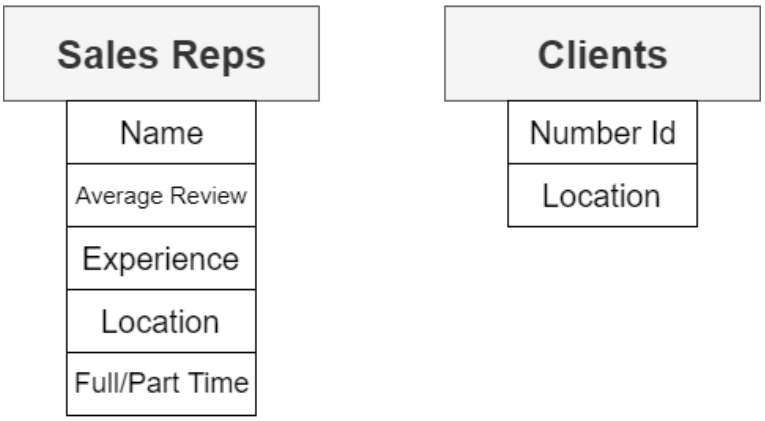


Figure 1: Data Specifications

The attributes *Name, Average Review, Experience* and *Full/Part Time* were defined using a random generator for every case.

Each feature will be considered subsequently to help determine whether or not a Sales Representative remains on the company's payroll.

- *Name* and *Number Id* are used to identify *Sales Reps* and *Clients* respectively, and do not affect the final output;
- *Average Review* is an integer between 1 and 5. It reflects the past reviews *Clients* have given to a *Sales Representative* and the higher the better in order to be chosen;
- *Experience* has a lower weight on the *Sales Reps* selection than reviews. It represents the years working in the industry from 0 to 25;
- *Locations* were used to calculate the driving distance between employee and its *Client* and are defined by latitude and longitude;

- *Full/Part Time* define the work regime and affect the maximum number of *Clients* possible to be assigned to a *Sales Rep*.

The geographical area affected has been defined manually and the state of South Carolina was chosen to proceed. This area can be easily changed in the program.

The *number of Sales Reps* and Clients, the Maximum Driving Distance and the importance of *Average Reviews* and Experience are options that can be defined in the configurations section and, obviously, affect this data generation and the problem solution.

After the development of a script that generates all this, a specific set of data concerning *Clients* and *Sales Representatives* was predefined in order to develop the model and quickly recognize and address possible issues.

4.2 Linear Optimization Model

Due to the nature of the problem, as the variables are required to be integers, this can be defined as a *Mixed Integer Program* and so it was declared a *MIP solver* to work with.

The first stage of solving is to **define the Variables** needed to work with. It was declared a matrix x of boolean variables that represent decisions with $0 - 1$ values.

In this *Assignment Problem* each variable refers to whether or not a *Client* is assigned to a *Sales Representative*.

Clients / Sales Reps	0	1	2	3
Albert	1	0	1	0
Doris	0	0	0	1
Sophia	0	1	0	0

Figure 2: Model variables representation.

For example, if the variable $x[Albert,\ 2] = 1$ it means that *Sales Rep* Albert is assigned to *Client* 2, and vice-versa.

At this point, only the variables are created. At the conclusion of the program, the solver will specify its values, which will provide the desired output.

After the variables definition, **three Constraints were set** to represent the problems limitations.

- Every *Client* must be assigned to exactly one *Sales Rep*;
- Each *Client* must be within the pre-defined maximum driving distance from its assigned *Sales Rep*;
- Every *Sales Representative* working full time (part time) can have at most 8 (4) *Clients* assigned.

It is worth mentioning the reasoning behind the maximum number of *Clients* that is possible to assign to a *Sales Rep*:

- Because this must be a general model that could be applied to various businesses it is not possible to define a precise job description of a *Sales Rep*.
  It was assumed a *Sales Rep* spends at average 4 hours a week working on a *Client*, $1h30m$ a month with the online meeting and at most 7 hours every three months going to their face-to-face meeting.
  Assuming three months corresponds to 523 working hours, the maximum number of *Clients* (max) for a full time working *Sales Rep* is calculated as follows:

$$\begin{aligned} & 523 \geqslant 7max + 1.5 \times 3max + 4 \times 4.4 \times 3max \\ \Leftrightarrow\ & 523 \geqslant 65.3max \\ \Leftrightarrow\ & max \leqslant 8.01 \\ \Leftrightarrow\ & max \leqslant 8 \end{aligned} \tag{4.1}$$

The model structure ends with the **definition of its Objective Function** where it were included our variables multiplied by coefficients representing *Sales Reps Average Review* and Years of Experience.
The goal is to **Minimize** the number of *Sales Representatives* needed and in this selection, *Average Reviews* have an influence factor of 0.75 and *Experience* has a factor of 0.25.

- This means that if *Client* 3 can be assigned to *Sales Reps* Doris and Albert. Doris has an Average Review of 4 and 5 Years of Experience and Albert has an Average Review of 2 and 11 Years of Experience, than Doris will be the chosen Rep.

$$\begin{aligned} & x[Doris, 3] \times \Big(0.75(Avg.Review/5) + 0.25(Experience/25)\Big) \\ & = 1 \times \Big(0.75 \times (4/5) + 0.25 \times (5/25)\Big) \\ & = 1 \times (0.6 + 0.05) \\ & = 0.65 \end{aligned} \tag{4.2}$$

$$
\begin{aligned}
& x[Albert, 3] \times \Big(0.75(Avg.Review/5) + 0.25(Experience/25)\Big) \\
& = 1 \times \Big(0.75 \times (2/5) + 0.25 \times (11/25)\Big) \\
& = 1 \times (0.3 + 0.11) \\
& = 0.41
\end{aligned}
\tag{4.3}
$$

0.75 and 0.25 were set as the standard values, but those can be changed in the configurations sections.

Being *Variables*, *Constraints* and *Objective Function* all defined, the **Solver was called** to check if there was an **Optimal Solution**, using the static set of data predefined.

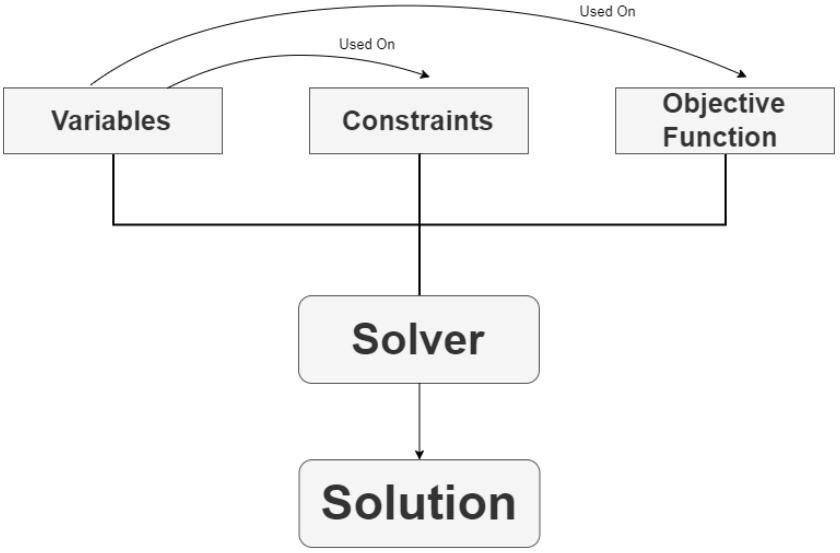


Figure 3: Optimization Model flow.

4.3 Testing and Iterating

Being all the *Integer Problem Optimization Model* components set, the testing was started with the fixed set of data and **a problem emerged**.
The model would not find a solution to the challenge even though our data was correctly generated.

The data analysis revealed that it was possible for there to be a situation when no *Sales Rep* was within a *Client's* maximum driving distance, which would mean that the time limit constraint would never be satisfied.

In order to solve this case, it was developed a function that would check for any *Client* that was in such situation and remove it from the data set, raising a warning about the situation.

With this improvement, and after solving a number of code bugs, it was possible to **print a solution** out of the *Optimization Model*.

At that moment, it was decided to start **testing different random sets of data** but soon after a few tests a new problem was found.
A less frequent situation was the one where an x number of *Clients* were only within the driving time of a *Sales Rep A* but the number x was greater than the maximum capacity of *Clients* that *Sales Rep A* could be assigned. This would lead to some *Clients* having no option between *Sales Representatives*.
In order to solve this case, it was developed a new function to check for any *Sales Rep* that was in such situation and remove its furthest *Clients* of the data set, again raising a warning about the situation.

The Optimization Model was now working for different random generated data sets, and the final tasks were about improving the code to deliver the output in the best possible manner.

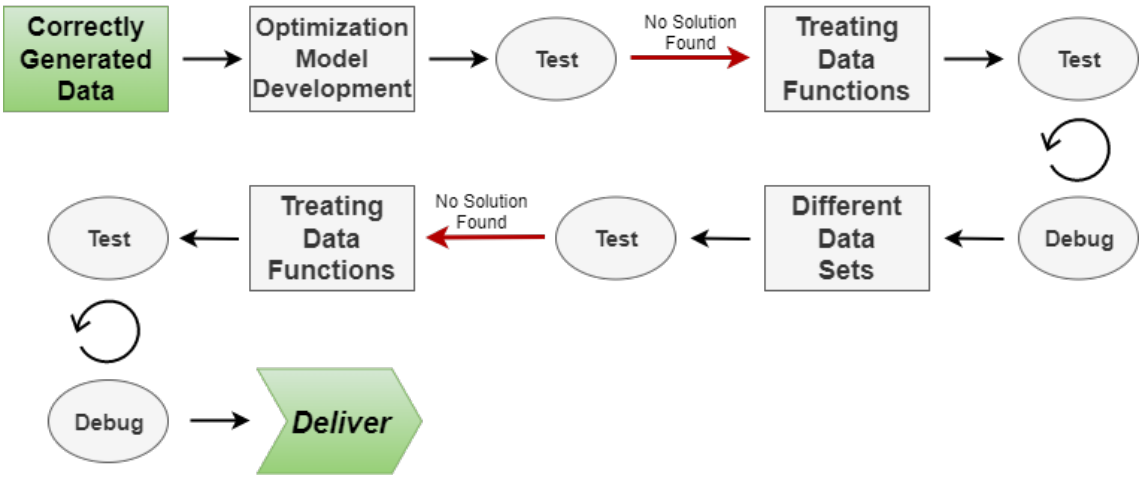


Figure 4: Developing process.

4.4 Mathematical Model

The optimization problem can be defined as follows:

$$
\begin{aligned}
Min \quad & \sum_{i=0}^{n-1}\sum_{j=0}^{m-1} x_{i,j} \times \Big(1 - \big(AvgReviewFactor \times (AvgReview/5)\big) - \\
& - \big(ExperienceFactor \times (Experience/25)\big)\Big) \\
s.t. \quad & \\
& \sum_{i=0}^{n-1} x_{i,j} = 1, \qquad \forall_j \\
& x_{i,j} \times distance(i,j) \leq MaxDrivingDistance \qquad \forall_i \quad \forall_j \\
& \sum_{j=0}^{m-1} x_{i,j} \leq 4 \quad , \qquad i \in \big\{i : \, Sales\ Rep\, i\ works\ part-time.\big\} \\
& \sum_{j=0}^{m-1} x_{i,j} \leq 8 \quad , \qquad i \in \big\{i : \, Sales\ Rep\, i\ works\ full-time.\big\} \\
& x_{i,j} \in \{0,1\} \qquad \forall_i \quad \forall_j
\end{aligned}
\tag{4.4}
$$

Where *AvgReviewFactor*, *ExperienceFactor* and *MaxDrivingDistance* are previously defined by the program user. And *AvgReview*, *Experience*, *distance(i, j)* and *full/part time* employees are defined with the data generator.

The variables $x_{i,j}$ represent the association between the n *Sales Reps* and the m *clients*.

The constraints refer to, in order:

- A *Client* can only be assigned to one *Sales Rep*;
- The Maximum Driving Distance between *Client* and *Sales Rep* has to be respected;
- Every *Sales Representative* has a maximum number of *Clients* it can handle, depending on its work regime.

5 Solution Implementation

In this section, the reader will be guided by the code files more in depth in order to better understand how the model works. (Files names are clickable links)

The complete files and code developed on this project can be consulted on the following GitHub repository:
github.com/Leandr0Duar7e/Optimization_Project

run_optimization_model.py is the file that executes the optimization when called on the terminal.
In file config.ini, it is possible to define the number of Sales Reps, Clients, the maximum driving distance between them and choose the weight of Average Reviews and Experience factors.

Data.py generate_data function generates random sets of data concerning Sales Reps and Clients.

The returned variables *sales_rep* and *clients* are both a list of lists where each list represents a person.
The attributes *name, number id, review, experience* and *Full/Part time* are all generated randomly, using the libraries *names* and *random*.

For the attribution of a *location*, the function *coordinate_generator* is called from the file adittional_functions.py.

- The function receives a polygon (*area*) and an integer (*nr*) concerning the number of locations already created, as arguments. The area is divided in four subareas, to achieve a more uniform distribution and simulate a more realistic situation.
  Randomly creates a point which coordinates represent latitude and longitude and returns the dictionary {*"lat": latitude, "lon": longitude*}, after checking that this point is within the chosen geographical area.

```
    # Generating Data
    sales, clients = generate_data(7, 20, south_carolina)
    print(sales)
    print(clients)
```

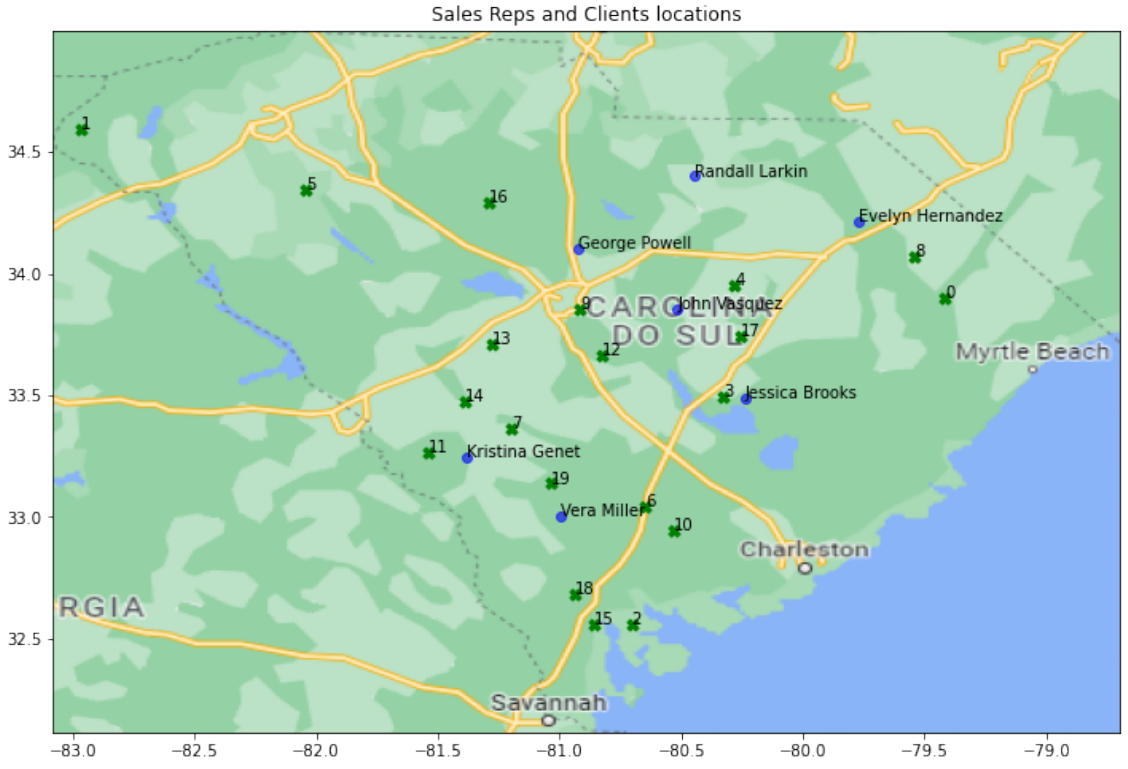
The presented piece of code generates the following output:

```
 [["Randall Larkin",2, 16,{"lat": 34.401102, "lon": -80.442628},1],
["George Powell", 3, 12, {"lat": 34.104101, "lon": -80.924947}, 1],
["Kristina Genet", 4, 9, {"lat": 33.245303, "lon": -81.378793}, 0],
["Jessica Brooks", 2, 2, {"lat": 33.487323, "lon": -80.236988}, 1],
```

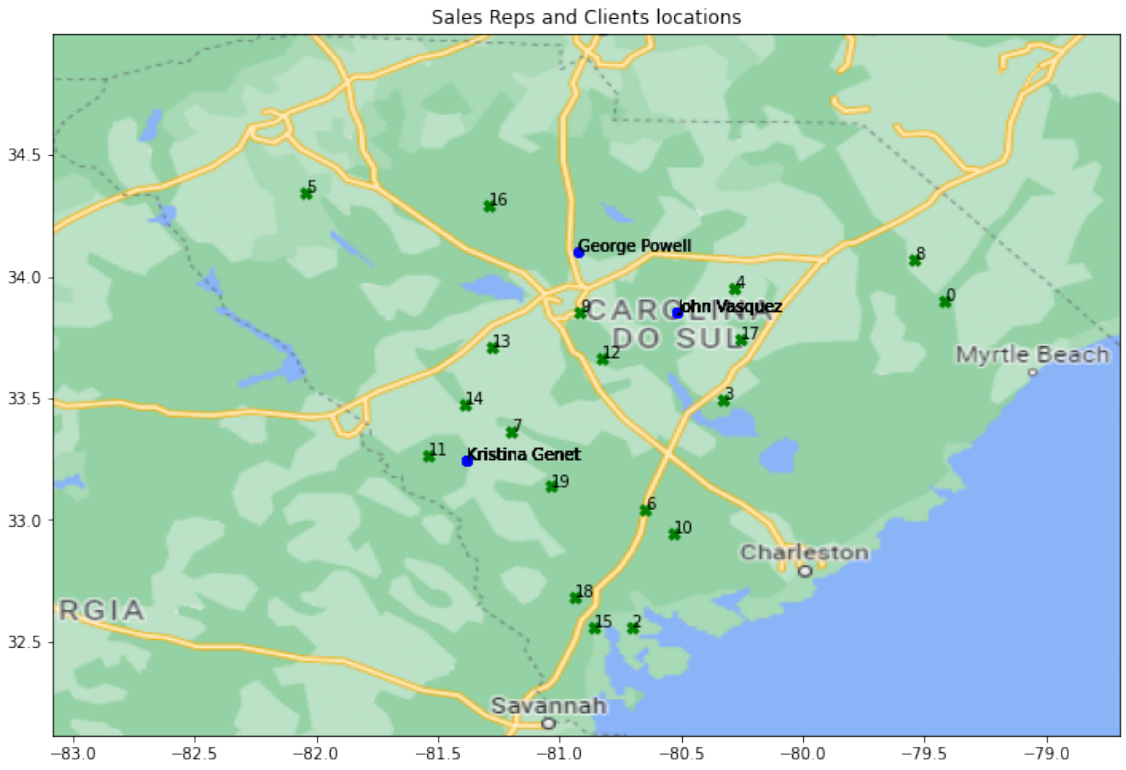
```
["John Vasquez", 3, 15, {"lat": 33.852873, "lon": -80.514586}, 0],
["Evelyn Hernandez",1,13,{"lat": 34.213974, "lon": -79.768324}, 0],
["Vera Miller", 2, 17, {"lat": 33.000601, "lon": -80.993143}, 0],]

[[0, {"lat": 33.897938, "lon": -79.418141}], [1, {"lat": 34.592222,
    "lon": -82.968292}], [2, {"lat": 32.558635, "lon":
   -80.703884}], [3, {"lat": 33.49271, "lon": -80.326073}], [4, {"
   lat": 33.955242, "lon": -80.280544}], [5, {"lat": 34.342664, "
   lon": -82.040896}], [6, {"lat": 33.043448, "lon": -80.645763}],
   [7, {"lat": 33.366307, "lon": -81.197821}], [8, {"lat":
   34.068536, "lon": -79.540536}], [9, {"lat": 33.856141, "lon":
   -80.917737}], [10, {"lat": 32.943726, "lon": -80.53213}], [11, {
   "lat": 33.26562, "lon": -81.540869}], [12, {"lat": 33.667415, "
   lon": -80.823989}], [13, {"lat": 33.707715, "lon": -81.275005}],
    [14, {"lat": 33.476923, "lon": -81.385967}], [15, {"lat":
   32.557737, "lon": -80.856303}], [16, {"lat": 34.290586, "lon":
   -81.291135}], [17, {"lat": 33.745449, "lon": -80.256602}], [18,
   {"lat": 32.68058, "lon": -80.936057}], [19, {"lat": 33.137256, "
   lon": -81.036352}],]
```

Running the file run_optimization_model.py with this data will produce the following outcome:

```
    >>> python3 run_optimization_model.py
```



```
No solution found.
Treating Data ...
An error was solved!
Client 1 location is too far to be adressed.
Done! Let's try to solve this problem again...

Sales rep: George Powell   ||  Average review: 3   ||  Years of Experience: 12  || Working part time.
List of clients assigned to George Powell:

Client 2 -> Driving distance: 2h 48m 0s
Client 5 -> Driving distance: 1h 49m 54s
Client 8 -> Driving distance: 2h 7m 9s

Sales rep: Kristina Genet   ||  Average review: 4   ||  Years of Experience: 9  || Working full time.
List of clients assigned to Kristina Genet:

Client 6 -> Driving distance: 1h 34m 23s
Client 7 -> Driving distance: 0h 34m 12s
Client 9 -> Driving distance: 2h 7m 3s
Client 10 -> Driving distance: 1h 42m 56s
Client 12 -> Driving distance: 1h 20m 9s
Client 13 -> Driving distance: 1h 9m 16s
Client 16 -> Driving distance: 2h 16m 32s
Client 17 -> Driving distance: 2h 4m 15s

Sales rep: John Vasquez   ||  Average review: 3   ||  Years of Experience: 15  || Working full time.
List of clients assigned to John Vasquez:

Client 0 -> Driving distance: 1h 37m 25s
Client 3 -> Driving distance: 1h 11m 3s
Client 4 -> Driving distance: 0h 33m 2s
Client 11 -> Driving distance: 2h 29m 17s
Client 14 -> Driving distance: 1h 59m 26s
Client 15 -> Driving distance: 2h 18m 44s
Client 18 -> Driving distance: 2h 21m 22s
Client 19 -> Driving distance: 2h 2m 3s


!!! The following clients do not satisfy the given constraints. An alternative strategy must be developed to addressed them !!!
Client 1

Workforce optimization from 7 to 3 sales representatives.
```



Looking at the remaining files will help to understand how these results were produced.
The graphics are produced by *visualize_data* function in *Data.py* file, using the matplotlib library to overlay our map image with the locations scatter plot. In the last graphic only the chosen Sales Reps and addressable Clients are shown.

After having a set of data produced, the function *solve_problem(sales_rep_fixed, clients_fixed, max_driving_dst, index)* in file Problem_MIP.py is called.
The argument *index* is a list of Clients that are impossible to address and will be removed in the following code block:

```
    if index != []:
        update = 0
        for i in index:
            clients.pop(i - update)
            update += 1
            )
```

The solver is created in the code line :

```
solver = pywraplp.Solver.CreateSolver("SCIP")
```

It was used the third party backend solver *SCIP*(Solving Constraint Integer Programs) which is currently one of the fastest non-commercial solvers for mixed integer programming.

On constraints definition it is reasonable to explain the method used on the second one. Function *driving_time(lat_1, lon_1, lat_2, lon_2)*, from adittional_functions.py file, calculates the driving distance between both given Sales Rep and Client.
Using *requests* and *json* libraries, the API of *OpenStreetMap* free service is called and loads of content are saved with information about routes between two given locations, of which we choose the driving duration(in seconds) to return.

```
r = requests.get(f"""http://router.project-osrm.org/route/v1/
car/{lon_1},{lat_1};{lon_2},{lat_2}?overview=false""")
```

After constraints definition, the solver minimizes the objective function which is the sum of all variables multiplied by the Average Review and Experience factors defined in *config.ini* file.

In order to **print a Solution** it was checked if an *Optimal* or *Feasible* one was found by the solver. When this is the case, a sequence of prints shows the selected Sales Rep and their associated clients.

As the reader can see in the example above, **when there is no solution**, the data is analysed and treated and the solver is called once again.
Data corrections are ensured by both functions in data_correction.py file.
Using the *driving_time()* function, brute force is used to see which Clients do not comply with restrictions considering any Sales Rep.
The model alerts the company about these clients in an effort to encourage the development of an alternative strategy.

The output ends with the program execution time.

6 Overall Evaluation

Having set the goal of creating a model suitable for any business with this kind of problem, I believe that was achieved developing this adaptable solution.
All the specifications are easily changeable to support a different area, number of Sales Reps or Clients, maximum driving distance and factors of employees classification. So a company only needs to adapt its workforce data to run the model.

In order to validate the model results, wider and smaller geographical areas (such as the state of Utah and the country of Portugal), large numbers of Clients and Sales Reps, different driving distances and factors of performance evaluation, and various random generated sets were tested.
Although some tests took a longer time to run, a solution was found in every case with correct specifications.

Even though everything is working, the project presents some weaknesses. The clients distribution is not uniform among chosen Sales Reps, the maximum number of Clients is a speculative figure and Clients to Clients distance is not taken into account.
Having said that, some future work on the project could include the following improvements:

- Define a minimum number of Clients each Sales Rep have to be addressed;
- Create a uniform distribution of clients after choosing the sales workforce;
- Getting data about Sales Reps job description and designing a function that calculates the maximum number of clients each sales rep can address;
- Experiment different solvers and compare results and performances.
- Run a post-optimality analysis understanding the effects of parameter changes on an optimal solution of the model and taking further conclusions about business development.
- Getting data about Clients in order to establish an importance factor, taking into account, for example, the volume of invoices.

7 Conclusion

The development of this project strongly improved my programming skills using VSCode editor and Python language. I was introduced, by my mentors, to good practices that I was not used to apply.

I had the opportunity to put into practice and expand my knowledge in the Optimization area, learning how to use multiple libraries that may be useful in my near future.

It was also an opportunity for me to make some contacts in the business world, get to know BI4ALL, and collaborate with my mentor there.

Although I have met the project's objectives, I have left comments for its future improvement that I hope me or another colleague can deal with.

8 Sources

- Google OR-Tools Guides
- Quantitative Economics with Python -Linear Programming (Thomas J. Sargent John Stachurski)
- Generate random coordinate points in the contiguous United States - (Micheal Beatty)
- Driving Distance between two or more places in Python - (Vaclav Dekanovsky)
- A Short Guide to Writing Your Final Year Project Report - (Cardiff University School of Computer Science and Informatics)
- LINEAR AND INTEGER OPTIMIZATION - Theory and Practice - (Gerard Sierksma and Yori Zwols)
- Linear Optimization Model Solver - (Gerard Sierksma and Yori Zwols)
- SCIP manual