UNIVERSIDAD DE MURCIA
FACULTY OF MATHEMATICS

Modeling Laboratory

Increasing the irrigation area on a farm

Advisor: José Fernández, Manuel Pulido
Authors: Ana Clemente Pérez, Jorge Ibáñez Puertas, Nicolás Colchero Truniger

MURCIA, APRIL 2023

Contents

1 Introduction

1.1 Looking for a company

Mathematical modeling involves describing real-world phenomena using mathematical terms, in what are known as models. Optimization, on the other hand, seeks to obtain optimal solutions to problems posed by mathematical models.

We initially contacted two companies: Grupo Buitrago, which provides agricultural services, and Baïa, a small start-up in the food and supplements industry. We chose Baïa for its potential and the opportunity to support them with our resources. However, being it a new company, we did not have enough information to formulate a problem to solve that would benefit both them and us. Furthermore, the problem posed was not one of optimization, but of forecasting. As a result, we decided to reach out to a company through an acquaintance of one of our team members.

1.2 Explotación Agraria Puertas SL

Explotación Agraria Puertas SL, an agricultural company based in Albacete, is considering a structural change in the irrigation system on one of its farms in La Grajuela, due to the age and high current maintenance costs of its irrigation machinery.

The goal is to determine whether we can find a better way to distribute and locate the irrigation systems compared to the current irrigation system distribution, which uses pivot central irrigation and sprinkler coverage irrigation, installed four decades ago, covering the most area possible at the least cost. In rearranging the machinery, some of the current parts may be used. This is however a complex task that requires a series of constraints, primarily to cover an irregular surface with circles of different radii and sizes. In our particular case, we will focus on the use of two specific irrigation systems: sprinkler coverage irrigation and center pivot irrigation. Center pivot systems will cover the largest area possible of the farm, and the remaining area will be covered by sprinkler coverage irrigation, since the first system is more cost-effective for large surfaces, and the second system does not necessarily need to cover regular areas, making its installation in areas not covered by pivots simpler.

Through a series of constraints, we will be able to determine where to place the pivots within the farm and what size to install them; with their size depending on the number and length of the sections that compose them. Additionally, we will need to consider a constraint that prevents the circles covering the pivots from overlapping, as otherwise, the pivots would collide when rotating.

Summarizing, **the problem involves** finding the best arrangement of pivots on an irregular surface using circular shapes, considering installation and purchase costs of these systems.

To tackle this problem, we will now explain it in more detail, as well as the technical results to consider for its resolution. Next, we will present a formulation of the problem and seek a solution using AMPL software, aiming to obtain the coordinates of the pivot centers within the farm and the sections (also known as spans) that should compose them, which will indicate the total pivot size.
We believe this is an interesting project because the formulation could also be useful for solving similar problems

on any given farm, potentially making the work and planning of machinery installers easier.

Figure 1: A pivot in *La Gineta* whith its respective spans/sections.

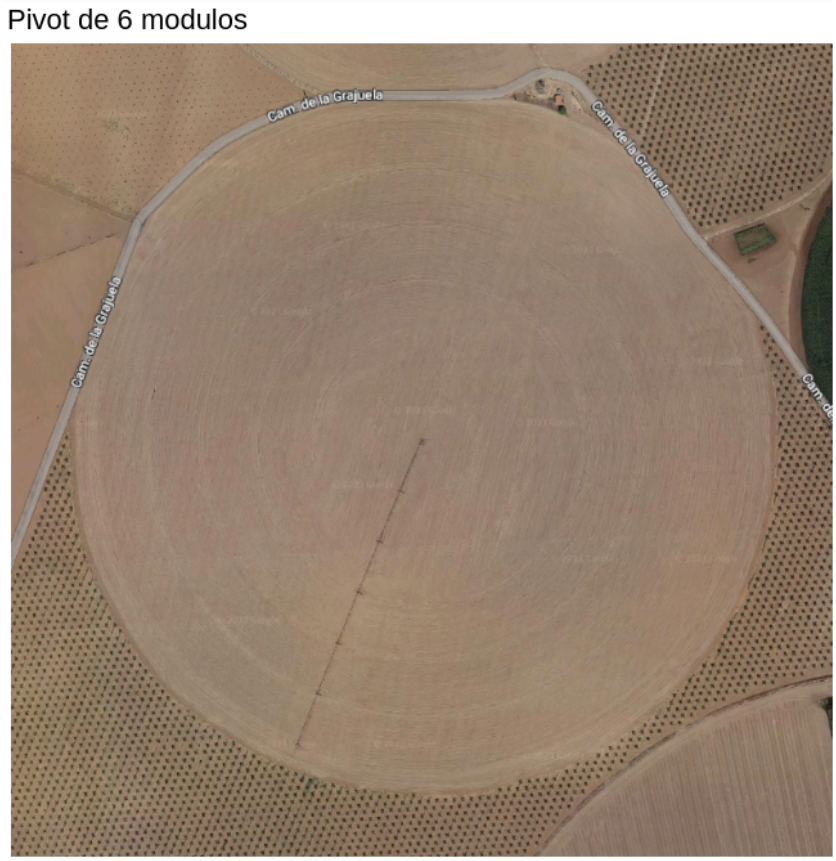

Figure 2: We can see the full area covered by a pivot upon its rotation, this one being formed by six sections that can be slightly distinguished in the image, making six concentric circles in its rotation. The rest of the surface is covered by sprinkler coverage.

2 Problem

2.1 Describing the problem

As we have mentioned before, the idea is to achieve the **maximum profitability of the land** by placing the irrigation pivots on the farm's terrain, that is, we want to obtain the optimal distribution of pivots on the surface and how many sections they would be comprised of so that the maximum possible area is covered with the minimum cost. In order to formulate the problem, we must take into account some considerations such as data on the farm's area and dimensions, and information about the operation of this irrigation system. But first, let's clarify exactly **what a pivot is**, as it is an uncommon irrigation structure for the average person, who might be more familiar with sprinkler irrigation, and not likely to be familiar with pivot irrigation. We already saw in the introduction a couple of images, one of their structure, and another of the surface they span. Let's now look at the following explanatory diagram:

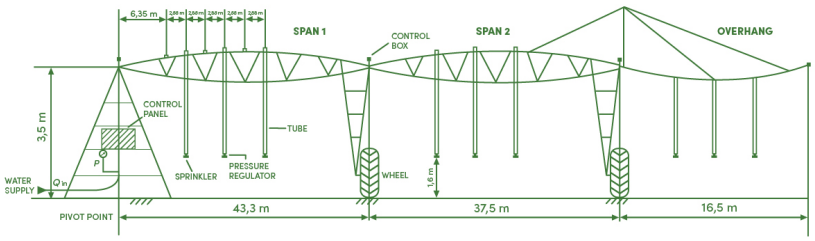

Figure 3: Schematic side view of a pivot. Source: https://www.agrivi.com/blog/center-pivot-system-an-efficient-and-economical-solution-for-irrigation/

This irrigation system was created by Frank Zybach in the United States in 1949 under the name "Zybach Self-Propelled Sprinkling Apparatus." On the left, we see the device located at the center of the pivot, which is a control panel on a central tower and the water intake for the system. Next, to the right, we see the "span 1," which is the first section that makes up the pivot. This is equipped with irrigation systems, pipes, plumbing, and all structures necessary for irrigation. The next structure, separating section 1 and section 2, is known as a "tower" and consists of a control box, wheels, and motors. It is responsible for the movement and rotation of the section. In our work, we consider tower i as a part of section i, considering them as one single entity. The "overhang" is a non-essential structure that we do not take into account in this work.

Now that we understand the basics about the technical concept behind this irrigation system, let's summarize what we need to formulate and solve the problem.

- **Farm's surface**: we have an irregular surface as a farm. The idea is to interpret the terrain as a union of convex sets. For this, we have used the tool https://sigpac.jccm.es/ from the Government of Castilla La Mancha, to take measurements of the edges that define the farm's boundaries.
- **Pivots**: as we have already mentioned, pivots are devices that supply water and fertilizers (dissolved in water) to the crop in the form of sprinklers. We must take into account that they can be made up of different sections and that these can have different lengths.

Figure 4: In black, representation of the farm's surface using polygons.

Figure 5: Points describing the farm.

- **Pivot sections**. The different components that will complete the total radius of the pivot. Pivots are also known as "compasses".
- **Knowledge of the installation price and machinery for sections and complete pivots according to size, and about the installation price and machinery for the sprinkler coverage irrigation system**.

Let's now take a representation of the farm we will work on, with a sketch of the polygons it can be divided into, and with the points of the corners of the complete polygon that we will choose to work with and to define the boundaries of the surface. We begin with a table of distances between points.

Regarding prices, there is no established fixed list of lengths and prices from any of the brands dedicated to the manufacturing of pivots: Valley, Lindsay... since the installation depends on the terrain of the farm, materials to be used, pipes, pivot and section specifications, the type of motors used, the form of sprinkling, and many more specs. However, in consultation with an installation technician, we reviewed historical data and current data from farms in the area to establish a list of the most commonly used sizes in the area, as well as

Table 1: Distance between points

Distance between points	meters
p1 to p2	921.35
p2 to p3	408.3
p3 to p4	205.48
p4 to p5	127.82
p5 to p6	236.19
p6 to p7	74.85
p7 to p8	460
p8 to p9	207.63
p9 to p10	119.72
p10 to p11	512.24
p11 to p12	2000
p12 to p1	590.14

the prices for these. This data is as follows:

Possible lengths for each section: 37, 39, 47, 49, 52, 54 m.

Another piece of information to take into account is that a pivot is usually made up of sections of equal length, but they can have **up to two different lengths** as long as they are arranged in a way so that the longer sections are placed on the inside and the shorter ones at the end of their radius.

Total length of a pivot (the radius of the circle that will be formed upon pivot rotation) must not exceed 800 meters, as doing so would make the structure unstable. However, we don't have a lower bound as such, because although the technician told us that they usually aren't shorter than 200-150m, this is more of an economic limit, not a limit of infeasibility. That is, it is considered that installing a pivot smaller than that size is not a good idea, since it's more expensive compared to overhead irrigation, but in principle, it should be our program that determines whether it is worth it or not to install one of this size. This is because coverage sprinkling irrigation is cheaper for small areas, just as pivot irrigation is cheaper for large areas.
As we already mentioned, the **prices** are not fixed, so we will discuss this topic in more depth in the following section.

For our problem, before moving on to the formulation and implementation, we will first need knowledge and estimation about the installation costs of the irrigation systems, and above all, we need to know the boundaries of the farm in order to recreate it in some way within AMPL.

2.2 Price estimation of irrigation systems

The idea is to cover the farm with **up to two different irrigation systems**:

1. Coverage sprinkler irrigation. Fortunately for this type of irrigation we have a relatively simple situation. An associated installer estimates the total cost of installing this type of irrigation to lie around 5000 euros per hectare, that is, 5000 euros for 10000 m^2.

2. Center pivot irrigation (compass). This presents a greater difficulty; let's discuss why.

In order to try to solve this inconvenience, we have installation data for six pivots in the area in recent years, plus data about the sections that compose them, and the total cost and installation prices. With this, we will try to make a price estimate based on the type of sections our formulation indicates we need to cover the surface; but ultimately, it is important to remark the real installation price will depend on the supplier and the installer, it's not fixed and it is fully customized.

As for the prices of the sections that make up the pivots, **there is no fixed rate** for the installation of these machines. This price is given by a third-party company dedicated to their installation. The total installation price is highly variable and depends on factors such as: size and capacity of the system, terrain characteristics (in particular, this terrain is a plain since we are on the Manchegan plateau), transportation, materials, towers and wheels installed, the specifications of the chosen motor, labor, and above all and most importantly, the length of the pipe that goes from the water extraction wells to the place where we locate the center of each compass. This depends on the solution given by the problem. For this reason, we cannot know exactly the installation cost of the results we obtain, but based on six previous installations known in the Grajuela area, we will try to obtain an estimate of the price of each section.

We hereby provide a **table with data from previous pivot installations** we have. Remember that sections or modules are the segments that, when joined, generate the full radius of a pivot/compass; and also, different section lengths can be used when making the same compass. These sections must be of the same brand, and when joined together to create the final pivot, it is common to start from the center with the longest sections and place the shorter sections at the end of the pivot. The order of the sections will be irrelevant when solving the problem since our solution will tell us the total radius, and what and how many sections to use. That is, once the solution is found, it will be enough to place the largest sections near the center, and that's it. In some cases, another irrigation structure called "overhang" may be placed at the end of the pivot, which we will not take into account.

This is the installation information for pivots that we have, in which the sections we are going to use to solve our problem are used.

Pivot	Radius	Length of section 1	Number of sections of this kind	Length of section 2	Number of sections of this kind	Machinery price	Installation price
Pivot 1	245	49	5	0	0	67500	4000
Pivot 2	381	54	6	49	1	84000	7500
Pivot 3	339	52	2	47	5	70000	6000
Pivot 4	313	37	7	54	1	68000	6800
Pivot 5	324	54	6	0	0	63000	5000
Pivot 6	468	39	12	0	0	104000	8000

Table 2: Information on configurations of previously installed center pivot irrigation systems.

Based on these data, we shall **estimate the price** of machinery and installation per section. So, if our solution says "install a 200-meter pivot with this type of section and this type of section," we can have an idea

of the cost, so that the program will be able to choose the necessary combination of sections for each pivot in the optimal solution, and the price will simply be a base cost plus a certain price per section installed.

The following table represents the section configurations of each pivot and their total cost:

Pivot	Span/Section Configuration	Machinery Price (€)	Installation Price (€)	Total Price (€)	Total Radius (m)
1	5 x 49m	67.500	4.000	71.500	245
2	6 x 54m + 1 x 49m	84.000	7.500	91.500	381
3	2 x 52m + 5 x 47m	70.000	6.000	76.000	339
4	7 x 37m + 1 x 54m	68.000	6.800	74.800	313
5	6 x 54m	63.000	5.000	68.000	324
6	12 x 39m	104.000	8.000	112.000	468

Now, according to these data, we shall now make a price estimate for each span/section, taking into account both their installation price and market cost.

Price estimation for different sections:

1. **Sections of length 49m:** pivot 1 has 5 sections of 49m with a total machinery cost of 67500€. Estimated price per section of 49m:
$$\frac{67.500\,€}{5} = 13.500\,€$$
2. **Sections of length 54m:** pivot 2 has 6 sections of 54m and 1 section of 49m with a total machinery cost of 84,000 €. We can subtract the cost of a 49m section from the total cost of pivot 2 to estimate the cost of the 54m sections. 84,000 €- 13,500 €(cost of a 49m section) = 70,500 €for 6 54m sections.
Estimated price per section of 54m:
$$\frac{70,500\,€}{6} = 11.750\,€$$
3. **Sections of length 52m and 47m:** pivot 3 has 2 sections of 52m and 5 sections of 47m with a total cost of machinery from 70,000 €. Here, the pivot has in total a radius of length 339. 2 sections of 52m +5 sections of 47m = 104m (sections of 52m) + 235m (sections of 47m) = 339m (total length of the radius). Now, let's calculate the ratio of the pivot that each section of 47m implies in relation to the radius length total: Proportion of 47m sections =
$$\frac{\text{Total length of 47m sections}}{\text{Total radius length}} = \frac{235}{339} \approx 0.6932$$
Since the total cost of the machinery for this pivot is 70,000 €, we can use the proportion to estimate the price of the 47m sections: Cost of 47m sections =
$$70,000\,€ \cdot 0.6932 \approx 48,524\,€$$
and we then divide the estimated cost of the 47m spans by the number of spans to obtain the estimted

price from these: Estimated price per 47m section =

$$\frac{48.524\,€}{5} \approx 9.705\,€$$

Therefore, the estimated price of each 47m stretch is approximately 9.705 €. Now, the price of 52 m sections according to this should be:

$$\frac{(70,000 - 9,705 \cdot 5)}{2} = 10.737{,}5\,€$$

.

4. **Sections of length 37m:** pivot 4 has 7 sections of 37m and 1 section of 54m with a total machinery cost of 68,000 €. We can subtract the cost of a 54m section from the total cost of pivot 4 to estimate the cost of the 37m sections. 68.000 €- 11.750 €(cost of a 54m section) = 56.250 €for 7 37m sections. Estimated price per 37m section:
$$\frac{56.250\,€}{7} = 8.036\,€$$
(rounded)

5. **Sections of length 39m:** pivot 6 has 12 sections of 39m with a total machinery cost of 104,000€. Estimated price per section of 39m:
$$\frac{104,000\,€}{12} = 8.667$$

Thus, this is the final **price table**(without taking into account installation account, which we will always consider to be 6000 €, and we will also consider the price for the central tower in the pivot to be 6000 €). This means buying and installing a pivot will cost 12000 € plus the sections we will use:

Table 3: Section length and estimated price in €

Section length (m)	Estimated price (€)
37	8.036
39	8.667
47	9.705
49	13.500
52	10.737,5
54	11.750

Additionally, the cost per hectare of sprinkler irrigation coverage is 5,000 €. These are the prices we will be working with.
Let's observe that if, for example, we want to approximately cover about 7 hectares, we would need a 150 meter pivot. $7 \cdot 5,000 = 35,000$ € is what it would cost to cover this area with sprinkler coverage. It can be seen that if three 52-meter sections are chosen, a larger area of the farm is covered in hectares, and the price is roughly the same. This is why sprinkler irrigation coverage is preferred for covering small areas, but for larger areas, it

is less expensive to use pivots. It is important to keep in mind that these estimates of the price per meter of the pivots **are based on limited information and may not be accurate**.

2.3 Delimiting the boundaries of the plot to work on

To establish the lines that will demarcate the boundaries of the convex polygons defining the plot where we intend to work, we will need pairs of points: one point to indicate the beginning of the segment, and one point to indicate the end of the segment. This is a problem that we have decided to solve in the following way:

1. We start Python, and use an image with the edges of the plot already demarcated; specifically, Figure 4 on page 6.

2. We create a code that opens the image and overlays it on a coordinate grid. In addition, we wish that when we click on different points of the image, (on the corners), the script will print the coordinates of the clicked point back. The following code does it:

```
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

# Load the map of the settlement
img = mpimg.imread('Medidasv3.png')

# Create a figure and an axis
fig, ax = plt.subplots()

# Show the image on the axis
ax.imshow(img)

# Add a 2D grid to manually find the points
ax.grid(True)

# Set the axis limits to match the image dimensions
ax.set_xlim(0, img.shape[1])
ax.set_ylim(img.shape[0], 0)

# Define the event handler to get the coordinates of a clicked point
def onclick(event):
    x = event.xdata
    y = event.ydata
    if x is not None and y is not None:
        print(f"Point clicked at ({x:.2f}, {y:.2f})")

# Connect the event handler to the plot
cid = fig.canvas.mpl_connect('button_press_event', onclick)

# Display the plot
plt.show()
```

We now show this code running:

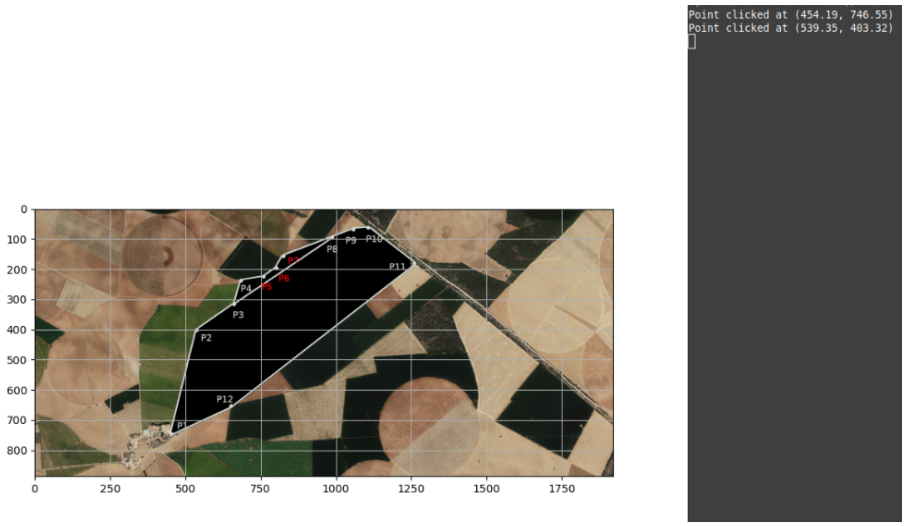

Figure 6: Running coordinate finder code, showing the farm, the coordinate grid on it, and printing points that are clicked. This will help us to find the corners of the farm that define the segments of its boundaries.

3. With this, we have just found the set of points that define the corners of the farm! These are the following:

Table 4: Points on the plane

Point	Coordinate X	Coordinate Y
p1	603.87	818.61
p2	683.87	479.26
p3	810.32	394.1
p4	834.84	319.26
p5	886.45	314.1
p6	956.13	267.65
p7	965.16	244.42
p8	1113.55	187.65
p9	1187.1	159.26
p10	1236.13	150.23
p11	1385.81	265.06
p12	800	717.97

4. Now, we have a problem with this. These coordinates are according to the pixels of the image, and are not true to the actual dimensions of the farm.

5. Using the dimensions measured in Table 1, on page 6, by using the joint tool of the Ministry of Agriculture and the Junta de Castilla La Mancha; we are going to modify the points so that we preserve the shape

of the farm, and the distance between the points is the actual physical distance. This is a problem that consists of calculating vectors, normalizing them, multiplying the components of the normalized vector by the actual physical distance, and then from a starting point, adding vectors. We have done this using a spreadsheet with the following name: 'PointsAchievedInGridPictureandScaledPoints.xlsx'.

Point	x	y	Vector	x	y	Vector Norm
1	603.87	818.61	P1P2	80	-339.35	348.6522946
2	683.87	479.26	P2P3	126.45	-85.16	152.4527077
3	810.32	394.1	P3P4	24.52	-74.84	78.75440305
4	834.84	319.26	P4P5	51.61	-5.16	51.86730859
5	886.45	314.1	P5P6	69.68	-46.45	83.74308867
6	956.13	267.65	P6P7	9.03	-23.23	24.92335852
7	965.16	244.42	P7P8	148.39	-56.77	158.8786487
8	1113.55	187.65	P8P9	73.55	-28.39	78.83904236
9	1187.1	159.26	P9P10	49.03	-9.03	49.85460661
10	1236.13	150.23	P10P11	149.68	114.83	188.6532038
11	1385.81	265.06	P11P12	-585.81	452.91	740.4733785
12	800	717.97	P12P1	-196.13	100.64	220.443613

Normalized Vectors	x	y	Vector Norm		Vector norms they ought to	Nuevos vect	x	y	New Vector Norms
P1P2	0.229454965	-0.97331928	P1P2	1	921.35	P1P2	211.4083319	-896.767718	921.35
P2P3	0.829437547	-0.55859946	P2P3	1	408.3	P2P3	338.6593506	-228.076159	408.3
P3P4	0.311347671	-0.95029607	P3P4	1	205.48	P3P4	63.97571951	-195.266837	205.48
P4P5	0.995039099	-0.09948463	P4P5	1	127.82	P4P5	127.1858976	-12.7161254	127.82
P5P6	0.832068665	-0.55467264	P5P6	1	236.19	P5P6	196.526298	-131.008131	236.19
P6P7	0.362310721	-0.93205737	P6P7	1	74.85	P6P7	27.11895748	-69.7644942	74.85
P7P8	0.933983271	-0.35731673	P7P8	1	460	P7P8	429.6323048	-164.365698	460
P8P9	0.932913412	-0.36010077	P8P9	1	207.63	P8P9	193.7008117	-74.7677232	207.63
P9P10	0.983459771	-0.18112669	P9P10	1	119.72	P9P10	117.7398038	-21.6844876	119.72
P10P11	0.793413507	0.60868301	P10P11	1	512.24	P10P11	406.4181348	311.7917852	512.24
P11P12	-0.79112905	0.611649268	P11P12	1	2000	P11P12	-1582.2581	1223.298536	2000
P12P1	-0.88970598	0.45653398	P12P1	1	590.14	P12P1	-525.051085	269.418963	590.14

Figure 7: Resizing of points using a spreadsheet.

New point coordinates			Vectors			Norms
	x	y				
1	603.87	818.61	P1P2	211.4083319	-896.767718	921.35
2	815.2783319	-78.1577178	P2P3	338.6593506	-228.076159	408.3
3	1153.937682	-306.233877	P3P4	63.97571951	-195.266837	205.48
4	1217.913402	-501.500714	P4P5	127.1858976	-12.7161254	127.82
5	1345.0993	-514.216839	P5P6	196.526298	-131.008131	236.19
6	1541.625598	-645.22497	P6P7	27.11895748	-69.7644942	74.85
7	1568.744555	-714.989464	P7P8	429.6323048	-164.365698	460
8	1998.37686	-879.355162	P8P9	193.7008117	-74.7677232	207.63
9	2192.077672	-954.122885	P9P10	117.7398038	-21.6844876	119.72
10	2309.817475	-975.807373	P10P11	406.4181348	311.7917852	512.24
11	2716.23561	-664.015588	P11P12	-1582.2581	1223.298536	2000
12	1133.977511	559.2829484	P12P1	-530.107511	259.3270516	590.1393846

Figure 8: Resizing of points using a spreadsheet.

6. This returns the points scaled to the real distance perfectly, which is as follows:

Table 5: Points on the plane

Point	Coordinate X	Coordinate Y
p1	603.87	818.61
p2	815.2783319	-78.15771781
p3	1153.937682	-306.2338767
p4	1217.913402	-501.5007139
p5	1345.0993	-514.2168393
p6	1541.625598	-645.22497
p7	1568.744555	-714.9894641
p8	1998.37686	-879.3551622
p9	2192.077671	-954.1228855
p10	2309.817475	-975.8073731
p11	2716.235610	-664.0155879
p12	1133.977511	559.2829484

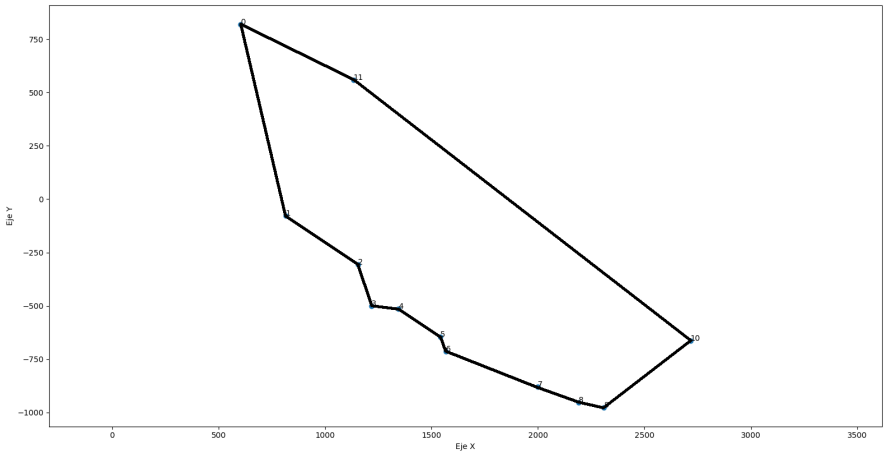

Figure 9: Figure of representation of the farm according to the points (still unclimbed). Returns the symmetric of the rotated farm shape, but this will not matter as long as the dimensions are correct after scaling.

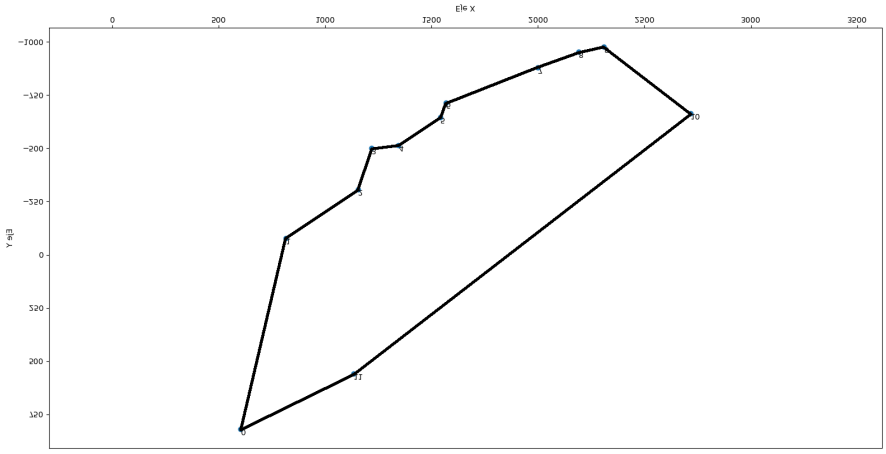

Figure 10: Polygon representing the farm, this time as it actually is.

3 Farm area and graphical representation for our work

Now, from the scaled points, we can determine the total area of the farm using the following script, in square meters, and it will also provide us with a graphical representation of the farm's surface:

```
import matplotlib.pyplot as plt

def polygon_area(points):
    n = len(points)
    area = 0.0
    for i in range(n):
        j = (i + 1) % n
        area += points[i][0] * points[j][1]
        area -= points[j][0] * points[i][1]
    area = abs(area) / 2.0
    return area

# Defining polygon corner points for establishing boundary lines
points = [
    (603.87, 818.61),
    (815.278331884383, -78.1577178120667),
    (1153.93768246152, -306.233876729812),
    (1217.91340197405, -501.500713936887),
    (1345.09929959985, -514.216839334081),
    (1541.62559760689, -645.224969965924),
    (1568.74455508656, -714.989464124598),
    (1998.37685984082, -879.35516222352),
    (2192.07767156003, -954.122885469045),
    (2309.81747534986, -975.807373093402),
    (2716.23561010679, -664.015587857317),
    (1133.97751131438, 559.282948398585)
]

# Generating the whole figure and using axes
fig, ax = plt.subplots()

# Graph every line using beginning and ending points
n = len(points)
for i in range(n):
    start_point = points[i]
    end_point = points[(i + 1) % n]
    x_values = [start_point[0], end_point[0]]
    y_values = [start_point[1], end_point[1]]
    ax.plot(x_values, y_values, label=f"L{i+1}")

# Find the polygon's total area
area = polygon_area(points)
print(f" rea  del pol gono: {area} metros cuadrados")

# Add a legend and show the graph plot
```

```
ax.legend()
plt.show()
```

The representation of the farm has already been discussed in the previous section. The new information is related to the **farm's surface area**: "Polygon area: 1458888.8282936495 square meters," or in other words, 145.88 hectares. This is the area we will use for the problem, since it is congruent with the points we are going to use; and it is a better option than the actual size, given that using this surface area, we avoid problems after finding the coordinates with the click method, which can be imprecise, and the actual size may not match the area in between our calculated points. For this reason, we use the area defined by the points we previously found and scaled.

4 Model Formulation

Now we will explain the AMPL model we used to optimize and solve our problem. First we will explain the different steps by which we created the model, explaining all variables and parameters used as our model has been evolving towards the final model. We will then show and discuss the final model used.

4.1 Initial simplied model approach

The first model we created was quite simple, we considered a **convex polygon** defined by its borderlines (each line is defined by the 2 vertices of the polygon that are in it). In this model we considered a **single length for pivot sections**. Our objective was to cover the maximum surface possible with pivots.

It is also important to discuss the number of pivots that our model should consider. To let the model choose the number of pivots was not viable, so the number should be manually introduced through a parameter. Two outputs could be received:

1. In the case that any pivot was to be installed but not used (result length of pivot = 0) the solution would be optimal.
2. In case that all pivots to be installed were used, there could be a better solution using more pivots.

We will now show the different sets, parameters, variables and constraints used in this model. When defining coordinates, they are put together for simplicity, although they actually are 2 different parameters/variables. Lastly, when the notation {SET} is used, it means the same as in AMPL, the parameter, variable or constraint applies to all members of the set.

4.1.1 Sets

1. **PIVOTS** : This sets defines the pivots that will be placed in the parcel. This set is defined through the parameter "num_pivots".
2. **LINES** : This set defines the lines that bound the polygon.

4.1.2 Parameters

1. **num_pivots** : Number of pivots that the model will use.
2. **(x1,y1){LINES}** : They define the coordinates of the first point of a given line.
3. **(x2,y2){LINES}** : They define the coordinates of the second point of a given line.
4. **lenSec** : It defines the only length of the pivot section used in this early model.

4.1.3 Variables

1. **(x,y){PIVOTS}** : Defines the position of a given pivot.
2. **numSec{PIVOTS}** : Defines the number of pivot sections that will be used for a given pivot.

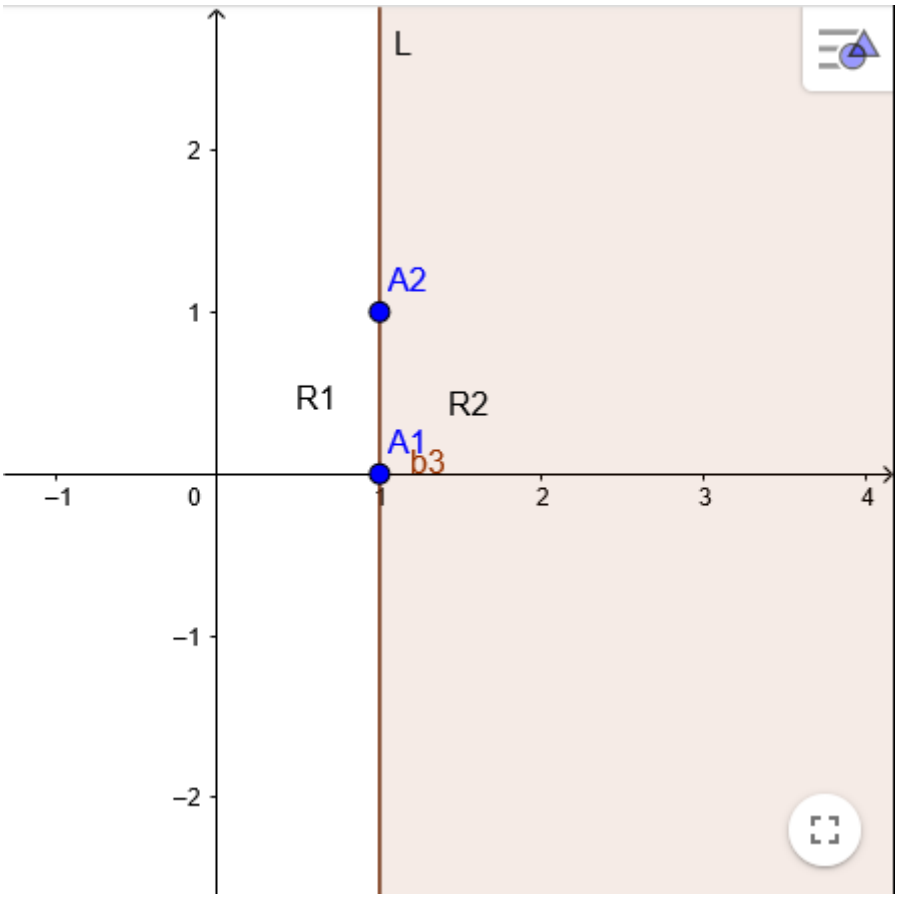

Figure 11: Half-plane that satisfies the restriction

4.1.4 constraints

1. **Inside_Polygon** Given a line defined by the points $(x_1, y_1), (x_2, y_2)$ and the center of a pivot, (x, y). The constraint

$$(x_2 - x_1) \cdot (y_1 - y) - (x_1 - x) \cdot (y_2 - y_1) \geq 0$$

forces the center of each pivot to be located within the polygon if it's satisfied for every borderline. It is however, somewhat difficult to explain.

It can be shown that the constraint holds when the point (x, y) is located in the right half-plane defined by the line if we look from the point (x_1, y_1) to the point (x_2, y_2).

Let's define $(x_1, y_1) = A1$ and $(x_2, y_2) = A2$.

In the image we can see the half-plane "R2" in which the condition would be satisfied if we introduced the line as (A1,A2), since it is the right half-plane if we were to look from point A1 to point A2.

If we introduce the line as (A2,A1), the condition would be met in the opposite half plane, "R1".

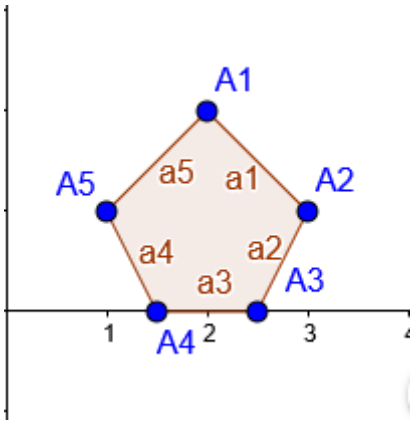

Figure 12: An example on how points should be introduced

If the points are properly introduced, this constraint forces the center of the pivot to be placed inside of the polygon, since it will be on the half-plane of every half-plane whose intersection is the polygon.

In order to reliably **introduce the polygon**, starting at any point, we first introduce the line that passes through that point and the next point **clock-wise**, and then we repeat this process on the next point until we have completely closed the polygon.

Let us show an example of how this should be done: given an example polygon, i.e the one in the picture above, we should start by introducing the line that connects the point A1 to the point A2 (the next point clock-wise):

L1 A1 A2

And then, we can continue the process on A2, etc., until point A1 is finally reached once again:

L2 A2 A3

L3 A3 A4

L4 A4 A5

L5 A5 A1

In AMPL each point A will be defined by the parameters (x_1,y_1) in case they are the first point in the line or (x_2,y_2) if they are the second.

2. **Far_Borders**

   Given a line defined by the points $(x_1, y_1), (x_2, y_2)$ and a pivot defined by its center (x, y) and the number of sections it has, $numSec$, the constraint:

$$((x_2 - x - 1) \cdot (y_1 - y) - (x_1 - x) \cdot (y_2 - y_1))^2 / ((x_2 - x_1)^2 + (y_2 - y_1)^2) >= (lenSec \cdot numSec)^2$$

   ensures that the center of the pivot is further away from the line than the length of the pivot.

This is the case because the left-hand side of the inequality computes the square of the distance between a line and a point.

It is clear that this constraint, together with the previous one, ensures that the pivot will be wholely placed inside the parcel.

3. **Far_Between**

   Given two pivots with centers (x_1, y_1) and (x_2, y_2) and number of sections $numSec_1$ and $numSec_2$. The constraint

   $$(x_1 - x_2)^2 + (y_1 - y_2)^2 \geq (lenSec \cdot (numSec_1 + numSec_2))^2$$

   ensures that the distance between the centers is greater than the sum of the length of the pivots. Therefore, the area covered by the pivots won't intersect.

4.1.5 Objective

maximize: $\sum_{i\ in\ PIVOTS} \pi \cdot (lenSec \cdot numSec[i])^2$

It is easy to see that the objective function in this early model is to simply cover as much area as possible with pivots.

4.2 Adding different lengths for pivot sections

The previous model was a decent first attempt at implementing the model, but we now need to eliminate the simplifications, starting with adding different sizes for the pivot sections.

As it was already stated, all pivot sections have the same length except for the last one. The last one can be of a different size, provided that this size is significantly lower than the size of the rest of the sections. The fact that the different pivot section is the last one is of no interest to us (we care about the length of the pivot, not the distribution of its sections). The way we have modeled this is through two new sets: the set **SIZES** and its subset **EXTRA_SIZES**.

SIZES stores the different lengths a section can have, while **EXTRA_SIZES** stores the sizes that are considered small enough to be the last section of a pivot of different section length. In order to ensure the last pivot section is smaller than the rest, only pivots whose section length is not in the subset **EXTRA_SIZES** can have its last section of a different length (this different length will be in the set EXTRA_SIZES).

We now specify the new sets, parameters, variables and constraints necessary for the model.

4.2.1 New sets

1. **SIZES** : Stores the different sizes a pivot section can have.
2. **EXTRA_SIZES** : Stores the possible sizes of the last section (it is a subset of SIZES).

4.2.2 New parameters

1. We now will stop using the parameter lenSec, since we now use multiple lengths stored in the SIZES set.
2. **UB** : Defines an upper bound to the number of sections a pivot can have. It will be used in the constraint 3.
3. **max_length** : Defines the maximum possible length of a pivot (800 in our case).

4.2.3 New variables

1. **numSec{PIVOTS,SIZES}** : We need to generalize numSec so that it counts the number of sections of each size. It will be 0 for each size but one.
2. **hasSize{PIVOTS,SIZES}** : These binary variables mark whether a certain size is used in a pivot ($numSec > 0$).
3. **extraSize{PIVOTS,EXTRA_SIZES}** : It marks whether a pivot has extra size (this is, whether it has its last section of a different size). It is a binary variable.
4. **length{PIVOT}** : It stores the total size of the pivot.
5. **length2{PIVOT}** : It stores the square of the total size of the pivot.
6. **exist{PIVOT}** : It's a binary variable that marks whether a pivot exists, it will be important when we change the objective function to include prices.

4.2.4 New constraints

1. Some of the previous constraints need to be rewriten using the new variables length and length2, in particular, in the constraint 2 the right hand size of the inequality must be changed for length2 (of the given pivot) and in the constraint 3 the right hand size of the inequality must be replaced by $(length_1 + length_2)^2$ (length of both pivots).
2. Given a pivot, the following constraints simply calculates its size and the square of its size:
   Calculating_Length:

$$\sum_{j\ in\ EXTRA\_SIZES} j \cdot extraSize[i,j] + \sum_{k\ in\ SIZES} k \cdot numSec[i,k] = length[i]$$

   Calculating_Length2:

$$length2[i] = (length[i])^2$$

   where i is a pivot.

3. **Has_Size**

   The following constraint makes sure that the variable **hasSize has the correct value**, it will be set to 1 if numSec if greater than 0.

   $$numSec[i,j] \leq hasSize[i,j] \cdot UB$$

   where i is a pivot, j is a length of section. UB is an upper bound to the number of section a pivot can have.

4. We need constraints to make sure **no more than 1 section length** is used and, in case this length is not in the set EXTRA_SIZES allow up to 1 extraSize to be used. This is easily done through the constraints:
   Single_Pivot_Section

   $$\sum_{j \ in \ SIZES} hasSec[i,j] \leq 1$$

   Single_Extra_Size

   $$\sum_{j \ in \ EXTRA\_SIZES} extraSize[i,j] \leq \sum_{k \ in \ SIZES \ diff \ EXTRA\_SIZES} hasSize[i,k]$$

   where i is a pivot. Note that $\sum_{j \ in \ EXTRA\_SIZES} extraSize[i,j]$ will be 1 if the pivot length used is not in the set **EXTRA_SIZES** and 0 otherwise.

5. **Pivot_Exists**

   Lastly, we have a constraint to give meaning to the variable exist and force a maximum length for each pivot.

   $$length[i] \leq max\_length \cdot exist[i]$$

4.2.5 New objective

We need to rewrite the objective to make use of the new variable length2:
maximize: $\sum_{i \ in \ PIVOTS} \pi \cdot length2[i]$

4.3 Including prices

As it was already stated, our objective is to **minimize the cost of covering the parcel**. In order to do this, we need to add information about prices through parameters.

We will also rewrite our objective, since now we care about costs and not area covered.

4.3.1 New parameters

1. **cover_cost** : As it was already stated, the area not covered by pivots will be irrigated by 'coverage' sprinkler irrigation. This parameter tells us the cost of using this type of irrigation by square meter.

2. **(initial_cost, cost{SIZES}** : The cost of a pivot is assumed to be: $initial_cost + \sum_{i\ in\ SIZES} numSec_i * cost_i + \sum_{i\ in\ EXTRA\_SIZES} extraSize_i * cost_i$

3. **parcel_area** : The total area of the parcel (in square meters). The optimal solution (length and position of pivots) does not depend on this parameter. It is only necesary for the objective function to have a clearer meaning.

4.3.2 New objective

minimize:

$$(\sum_{i\ in\ PIVOTS} initial\_cost \cdot exist[i] + (\sum_{j in SIZES} cost[j] \cdot numSec[i,j]) + (\sum_{j in EXTRA_SIZES} cost[j] \cdot extraSize[i,j])) +$$

$$(parcel\_area - \pi \cdot \sum_{i\ in\ PIVOTS} length2[i]) \cdot coverage\_cost$$

It is clear that now the objective is to minimize the cost of irrigating the parcel. The first line computes the cost of the pivots, while the second line computes the cost of covering the rest of the parcel with coverage irrigation.

As we already stated, the value of parcel_area does not change the optimal solution (although it does change the value of the function at that point) since $parcel\_area * coverage\_cost$ is a constant.

4.4 Generalizing the parcel

At this moment we had to generalize the parcel to a **non-convex polygon** (union of convex polygons), so the model can be used for all types of polygons.

At first we thought it would be quite easy since we only had to define a set of polygons and for each polygon a set of the lines that form it. Finally, we would make an OR sentence for the constraints for each polygon (forcing the constraints to hold for some polygon).

But while this worked for the constraint 1 it was not the case for the constraint 2 since a pivot can intersect both polygons.

This forced us to significantly change that constraint. Instead of working with distances from points to lines we needed to work with **distances from points to line segments**. These line segments are the border of the polygon. And if that distance is greater than the length of the pivot for all segments, the pivot will be fully placed inside the parcel (provided that the center of the pivot is already inside the parcel).

We now show the additional sets, variables and constraints necessary for this model.

4.4.1 New sets

1. **POLY** : set of polygons that form the parcel.

2. **POLY_LINES{POLY}** : lines that form the polygon i, they are introduced as we explained in 1. Parameters (x_1, y_1, x_2, y_2) are defined for each one.

3. **SEGMENTS** : Segments that make up the border of the polygon. Parameters (x_1, y_1, x_2, y_2) are defined for each one to define its borderpoints.

4.4.2 New variables:

The variables will be discused more deeply while discussing the constraints.

1. **insidePolygon{PIVOTS, POLY} binary** If it takes the value 1, it is because the center of pivot i is in the polygon j. At least 1 has to be in it.
2. **t{PIVOTS, SEGMENTS}** represents the scalar t that appears in the calculation of the projection of a point (x,y), in this case the center of the pivot, to a line given by the points $(x_1, y_1), (x_2, y_2)$.
3. **distSeg{PIVOTS, SEGMENTS} >= 0**: distance of the center of the pivot to the segment.
4. **scenDistSeg{PIVOTS, SEGMENS, 1..3} binary**: Decide which set of constraints to use based on where the variable t is (this part is further explained below).

4.4.3 New constraints

1. The constraint 1 was simply generalized by an OR statement, which in AMPL translates to the following constraints:

$$insidePolygon[i,j] \cdot (x_2 - x_1) \cdot (y_1 - y) - (x_1 - x) \cdot (y_2 - y_1) \geq 0$$

   where i is the pivot with center in coordinates (x, y) and j is a polygon such that the line formed by (x_1, y_1), (x_2, y_2) is a border of the polygon (the line is in the set of lines that form the polygon). If $insidePolygon[i, j] = 1$, the constraint will hold only if the point is inside the polygon.

   We can force insidePolygon to be 1 for some polygon (and therefore, force the point to be inside that polygon through the previous constraint):

$$\sum_{\text{j in POLY}} insidePolygon[i,j] \geq 1$$

   where i is a pivot.

2. In order to generalize the constraint 2 we need to measure the distance between the center of a pivot and the line segments that make up the border. Remember that now we can have non-convex surfaces.

   Let (x, y) be the center of a pivot and (x_1, y_1), (x_2, y_2) the end points of the segment. The distance from the point to the segment will vary depending on 3 scenarios:

   Let's consider the points $B1 = (x_1, y_1)$ and $B2 = (x_2, y_2)$ and the segment that joins them. Note that the regions have been obtain by drawing the lines perpendicular to the segment that go through $B1$ and $B2$.

   - If the point (x, y) is **in region A1**, the distance from the point to the segment will be dist((x, y),(x_1, y_1)).

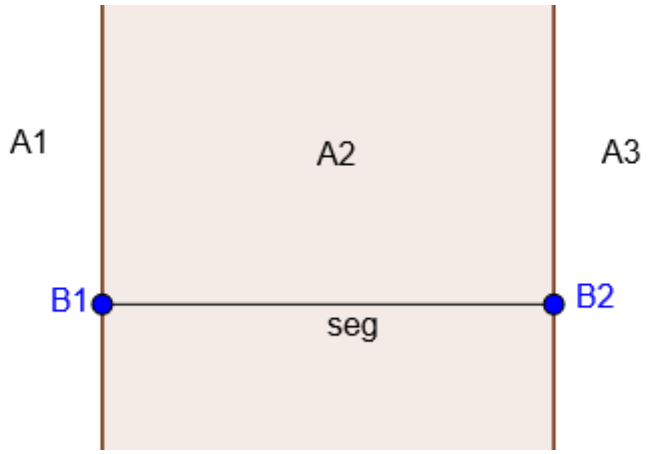

Figure 13: An ilustration of the possible scenarios when calculating the distance from a point to a segment

- If the point (x, y) is **in region A3**, the distance from the point to the segment will be dist$((x, y),(x_2, y_2))$.
- If the point (x, y) is **in region A2**, the distance from the point to the segment will be the distance from the point to the line that extends the segment (we already know how to calculate it from the previous models).

Note that the proyection of (x, y) on the line that extends the segment will always fall in the same region as the point. Also note that if we are able to determine the value t that satisfies

$$proy((x, y), ((x_1, y_1), (x_2, y_2))) = t \cdot (x_1, y_1) + (1 - t) \cdot (x_2, y_2) = (x_2, y_2) + t \cdot ((x_1, y_1) - (x_2, y_2))$$

where proy defines the proyection of the point on a given line, we can easily know in which region is the projection and therefore the point.

In summary, depending on the value of t we will find ourselves in one of the scenarios.

- If $t < 0$ the point is in region A3.
- If $t > 1$ the point is in region A1.
- If $0 < t < 1$ the point is in region A2.

Luckily for us, we can easily figure out the value of t through the constraint:

$$\frac{(x[i] - x_2[j]) \cdot (x_1[j] - x_2[j]) + (y[i] - y_2[j]) \cdot (y_1[j] - y_2[j])}{(x_2[j] - x_1[j])^2 + (y_2[j] - y_1[j])^2} = t[i, j]$$

Now we can compute the value of the distance of the center of the pivot to the segment that will be stored in the variable distSeg, taking into account in each case the way to calculate the distance based on the case in which we are, as we have already commented before. Therefore, we need to code the following implications:

$$t \leq 0 \implies distSeg = dist((x,y),(x_2,y_2))$$

$$t \leq 1 \implies distSeg = dist((x,y),(x_1,y_1))$$

$$0 < t < 1 \implies distSeg = dist((x,y), LINE((x_1,y_1),(x_2,y_2)))$$

In order to implement these constraints we need binary variables *scenDistSeg* defined in PIVOTS cross SEGMENTS cross 1..3. This variable will help us choose which group of constraints must be met based on the case in which we find ourselves. Only one $scenDistSeg[i,j,k]$ will be one for each pivot and line (k in 1,2,3), and it will mark the scenario we find ourselves in

$t < 0 \implies scenDistSeg[i,j,1] = 1$

$0 < t < 1 \implies scenDistSeg[i,j,2] = 1$

$1 < t \implies scenDistSeg[i,j,3] = 1$

With the constraints (applied in PIVOTS cross SEGMENTS) :

$$scenDistSeg[i,j,1] \cdot T \leq 0$$

$$scenDistSeg[i,j,2] \cdot T \geq 0$$

$$scenDistSeg[i,j,2] \cdot (T-1) \leq 0$$

$$scenDistSeg[i,j,3] \cdot (T-1) \geq 0$$

$$scenDistSeg[i,j,1] + scenDistSeg[i,j,2] + scenDistSeg[i,j,3] = 1$$

we make sure that scenDistSeg will only be 1 if t is in the correct interval.

Finally, we calculate the value of distSeg (distance from pivot center to segment) through the constraints :

$$scenDistSeg[i,j,1] \cdot ((x-x_2)^2 + (y-y_2)^2) = scenDistSeg[i,j,1] \cdot distSeg$$

$$scenDistSeg[i,j,2] \cdot \frac{(x_2-x_1)\cdot(y_1-y) - (x_1-x)\cdot(y_2-y_1))}{((x_2-x_1)^2+(y_2-y_1)^2)} = scenDistSeg[i,j,2] \cdot distSeg$$

$$scenDistSeg[i,j,3] \cdot ((x-x_1)^2 + (y-y_1)^2) = scenDistSeg[i,j,3] \cdot distSeg$$

note that these constraints will hold for any value of distSeg in case that $scenDistSeg = 0$, this means that the value of distSeg will be the left-hand side of the constraint where $scenDistSeg = 1$.

5 AMPL model implementation

For executing and solving an optimization model in AMPL, we need three files: a file.mod, a file.dat, a file.run. In this section we are going to briefly describe how we can implement this mathematical model in AMPL, an optimization language where we can choose different solvers to obtain an optimal solution to any problem. Since we have already commented the code in the section before, we are now specially interested in explaining how the .dat file has to be defined, and how to actually run the model.

5.1 Data

```
param num_pivots := 2;
param coverage_cost := 0.5;
param initial_cost := 6000;
param parcel_area := 1458888.8282936495;
param max_length := 800;

param : SIZES : cost :=
  0 0
  37 8036
  39 8667
  47 9705
  49 13500
  52 10737.5
  54 11750;
set EXTRA_SIZES := 0 37 39;
set POLY := P1 P2 P3 P4;
set LINES_POLY[P1] := L11 L12 L13 L14 L15 L16 L17 L18;
set LINES_POLY[P2] := L21 L22 L23;
set LINES_POLY[P3] := L31 L32 L33 L34;
set LINES_POLY[P4] := L41 L42 L43;

param: x1_lines y1_lines x2_lines y2_lines :=
    P1 L11 603.87 1818.61 1133.98 1559.28
  P1 L12 1133.98 1559.28 2716.24 335.98
  P1 L13 2716.24 335.98 2309.82 24.19
  P1 L14 2309.82 24.19 2192.08 45.88
  P1 L15 1998.38 120.64 1568.74 285.01
  P1 L16 1568.74 285.01 1153.94 693.77
  P1 L17 1153.94 693.77 815.28 921.84
    P1 L18 815.28 921.84 603.87 1818.61
    P2 L21 1998.38 120.64 1568.74 285.01
    P2 L22 1568.74 285.01 1541.63 354.78
    P2 L23 1541.63 354.78 1998.38 120.64
    P3 L31 1998.38 120.64 1541.63 354.78
    P3 L32 1541.63 354.78 1345.1 485.78
    P3 L33 1345.1 485.78 1153.94 693.77
    P3 L34 1153.94 693.77 1998.38 120.64
    P4 L41 1345.1 485.78 1217.91 498.5
```

```
    P4 L42 1217.91 498.5 1153.94 693.77
    P4 L43 1153.94 693.77 1345.1 485.78
    ;


param: SEGMENTS: x1 y1 x2 y2 :=
    S1 603.87 1818.61 1133.98 1559.28
    S2 1133.98 1559.28 2716.24 335.98
    S3 2716.24 335.98 2309.82 24.19
    S4 2309.82 24.19 2192.08 45.88
    S5 2192.08 45.88 1998.38 120.64
    S6 1998.38 120.64 1568.74 285.01
    S7 1568.74 285.01 1541.63 354.78
    S8 1541.63 354.78 1345.1 485.78
    S9 1345.1 485.78 1217.91 498.5
    S10 1217.91 498.5 1153.94 693.77
    S11 1153.94 693.77 815.28 921.84
    S12 815.28 921.84 603.87 1818.61;
```

While most parameters and sets have already been explained we will explain how to introduce the sets POLY, POLY_LINES and SEGMENTS.

First you need to separate your parcel into convex polygons, give each polygon a name and create the set POLY by introducing the names. This division can always be made by selecting the edges of the polygon and combining them into convex polygons. The next step is to introduce the lines that bound the polygons into the set POLY_LINES for each polygon. This will be done as it was explained in the first restriction of the first model 1.

In the images below it can be seen the parcel and its division. The point A_i denotes the first point of the segment S_i in the code, as we can see, all the points that appear in the segments are the same ones that appear in the lines of the polygons and are also the edges of our polygon.

In order to introduce the set SEGMENTS we simply introduce segments defined by the edges of the polygon.

While executing the model we found out that it was too big for the solver to handle. It didn't return feasible solutions even after hours of execution. Therefore, since we do not expect that the optimal solutions will have pivots centers in the 3 small polygons we did not introduce them in the .dat file we actually used. Said file is shown below.

```
param num_pivots := 2;
param coverage_cost := 0.5;
param initial_cost := 12000;
param parcel_area := 1458888.8282936495;
param max_length := 800;

param : SIZES : cost :=
  0 0
  37 8036
  39 8667
  47 9705
```

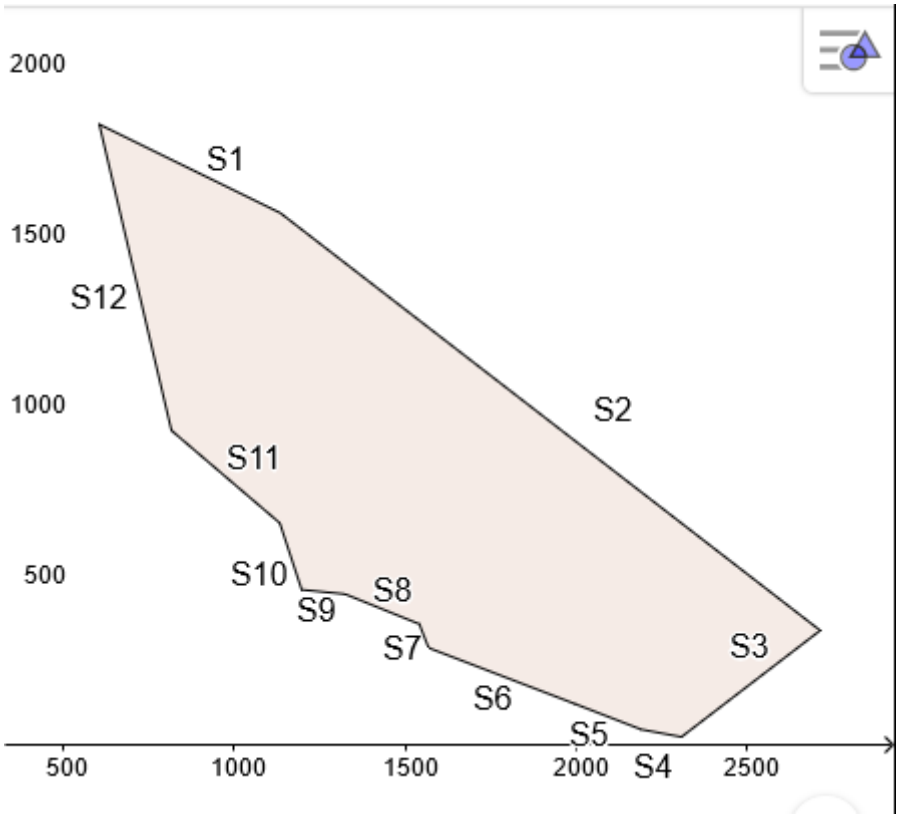

Figure 14: Parcel we are working with.

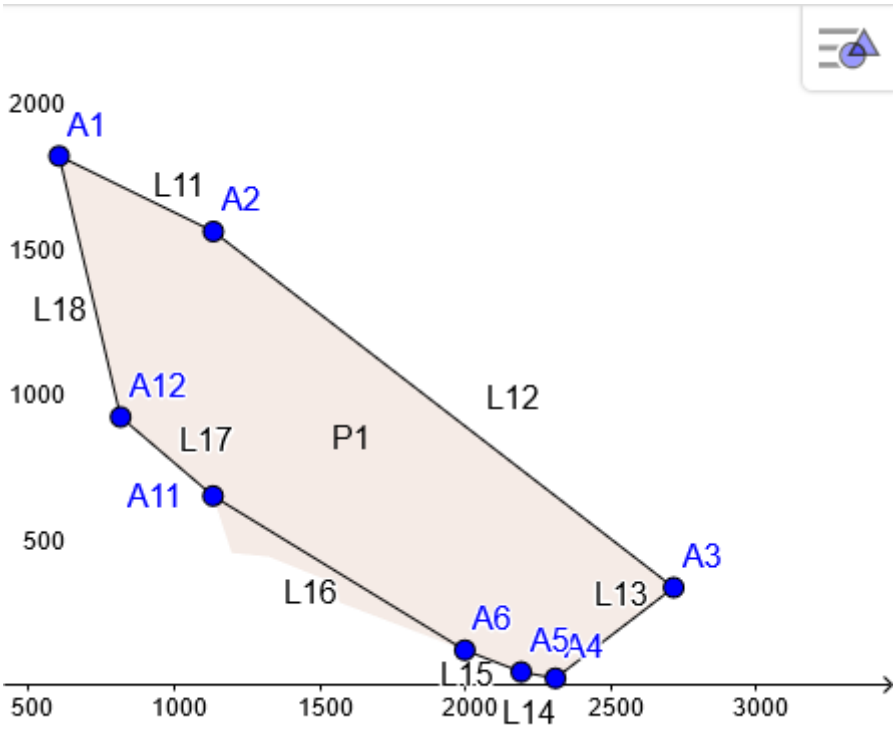

Figure 15: First polygon in our division.

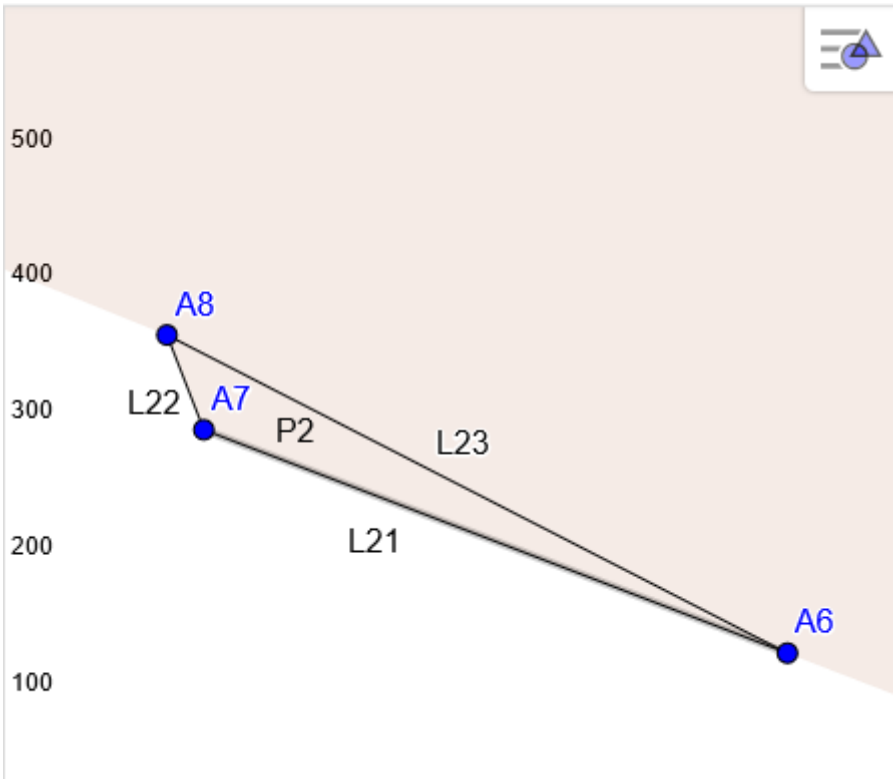

Figure 16: Second polygon in our division.

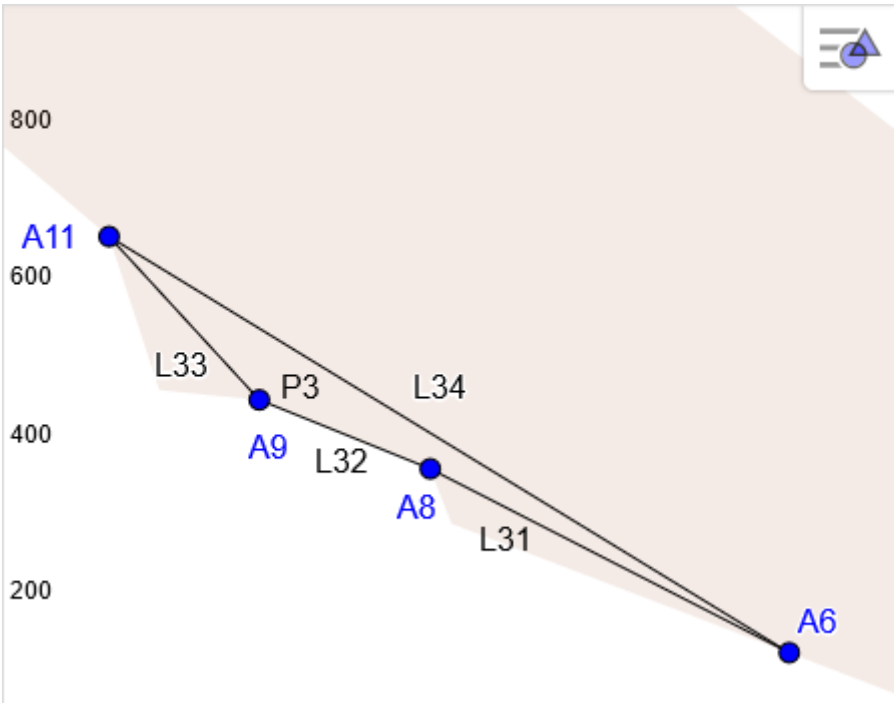

Figure 17: Third polygon in our division.

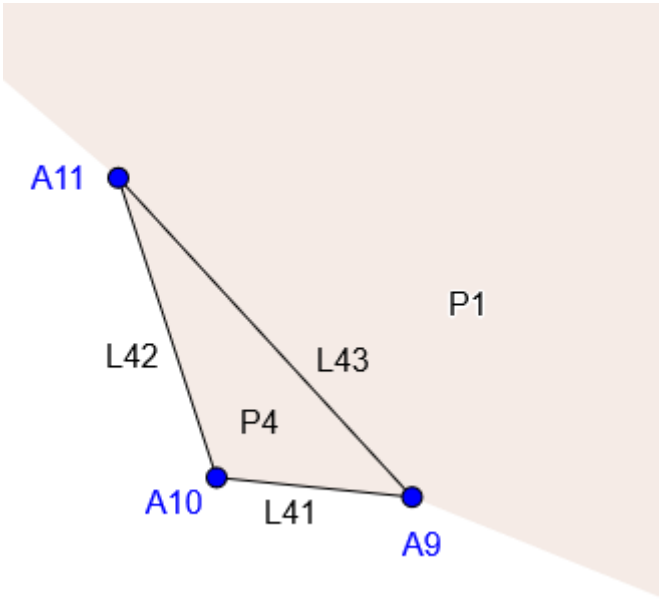

Figure 18: Fourth polygon in our division.

```
  49 13500
  52 10737.5
  54 11750;
set EXTRA_SIZES := 0 37 39;
set POLY := P1;
set LINES_POLY[P1] := L11 L12 L13 L14 L15 L16 L17 L18;

param: x1_lines y1_lines x2_lines y2_lines :=
    P1 L11 603.87 1818.61 1133.98 1559.28
  P1 L12 1133.98 1559.28 2716.24 335.98
  P1 L13 2716.24 335.98 2309.82 24.19
  P1 L14 2309.82 24.19 2192.08 45.88
  P1 L15 1998.38 120.64 1568.74 285.01
  P1 L16 1568.74 285.01 1153.94 693.77
  P1 L17 1153.94 693.77 815.28 921.84
    P1 L18 815.28 921.84 603.87 1818.61
    ;


param: SEGMENTS: x1 y1 x2 y2 :=
    S1 603.87 1818.61 1133.98 1559.28
    S2 1133.98 1559.28 2716.24 335.98
    S3 2716.24 335.98 2309.82 24.19
    S4 2309.82 24.19 2192.08 45.88
    S5 2192.08 45.88 1998.38 120.64
    S6 1998.38 120.64 1568.74 285.01
    S7 1568.74 285.01 1541.63 354.78
    S8 1541.63 354.78 1345.1 485.78
    S9 1345.1 485.78 1217.91 498.5
    S10 1217.91 498.5 1153.94 693.77
    S11 1153.94 693.77 815.28 921.84
    S12 815.28 921.84 603.87 1818.61;
```

5.2 Model

Here we can see the model we will send to the solver to obtain a result. We have put some comments to facilitate the understanding of the code, although not too many so as not to hinder the reading of it.

```
#--PARAMETERS AND SETS--------------------------

param num_pivots;               #maximum number of pivots you should try to put


set PIVOTS := {1..num_pivots};

set SIZES;
set EXTRA_SIZES within SIZES;         #possible sizes of the last section
```

```
set POLY;                    #set of polygons that form the surface
set LINES_POLY{POLY};              #edges that form each plygon
set SEGMENTS;

param x1_lines{i in POLY, j in LINES_POLY[i]};  #(x1,y1),(x2,y2) are the points that form a
                                                 polygon edge
param y1_lines{i in POLY, j in LINES_POLY[i]};
param x2_lines{i in POLY, j in LINES_POLY[i]};
param y2_lines{i in POLY, j in LINES_POLY[i]};

param x1{SEGMENTS};
param y1{SEGMENTS};
param x2{SEGMENTS};
param y2{SEGMENTS};

param UB := 100;                #upper bound to the number of section a pivot can have
param max_length;

param coverage_cost;
param initial_cost;
param cost{SIZES};                #cost of each section size
param parcel_area;

#--DECISION VARIABLES----------------------------
var x{PIVOTS};                   #(x,y) = center pivot
var y{PIVOTS};
var numSec{PIVOTS,SIZES} >= 0 integer;      #it counts the number of sections of each size

#--AUXILIAR VARIABLES----------------------------
var hasSize{PIVOTS,SIZES} binary;       #mark whether a certain size is used in a pivot
var extraSize{PIVOTS, EXTRA_SIZES} binary;    #marks whether a pivot has extra size
var length{PIVOTS} >= 0;             #total size of the pivot
var length2{PIVOTS} >= 0;            #square of total size of the pivot
var exist{PIVOTS} binary;            #marks whether a pivot exists

var insidePolygon{PIVOTS, POLY} binary;     #mark whether a pivot is inside the polygon
var t{PIVOTS, SEGMENTS};             #scalar t that appears in the calculation of the
                                              projection of the center (x,y), to a line given
                                               by the points (x1, y1), (x2, y2)
var distSeg{PIVOTS, SEGMENTS} >= 0;
var scenDistSeg{PIVOTS, SEGMENTS, 1..3} binary;


#--FUNCION OBJETIVO-------------------------------

# We want to minimize the cost of installing the pivots (first line) and the cost of covering
                                              the area not covered by the pivots by coverage
                                              irrigation (second line)
```

```
minimize objective: (sum{i in PIVOTS} (initial_cost*exist[i] + (sum{j in SIZES} cost[j]*numSec
                                                [i,j]) + (sum{j in EXTRA_SIZES} cost[j]*
                                                extraSize[i,j])))
    + (parcel_area - 3.14*(sum{i in PIVOTS} length2[i]))* coverage_cost;


#what we want to obtain is how many pivots, where to place them and with how many sections and
                                                what sizes


#--CONSTRAINTS----------------------------------------

# the pivot must be placed in one of the polygons (second constraint) and must be inside it (
                                                first constraint)

s.t. Inside_Polygon{i in PIVOTS, j in POLY, k in LINES_POLY[j]}:
  insidePolygon[i,j]*((x2_lines[j,k]-x1_lines[j,k])*(y1_lines[j,k]-y[i]) - (x1_lines[j,k]-x[i]
                                                )*(y2_lines[j,k]-y1_lines[j,k])) >= 0;



s.t. Inside_Some_Polygon{i in PIVOTS}: sum{j in POLY} insidePolygon[i,j] >= 1;


# must not concern the edges (first constraint) and cannot collide with another pivot when
                                                turning (second)

s.t. Far_Borders{i in PIVOTS, j in SEGMENTS}:
  distSeg[i,j] >= length2[i];

s.t. Far_Between{(i,j) in PIVOTS cross PIVOTS: i < j}:
  (x[i] - x[j])^2 + (y[i] - y[j])^2 >= (length[i] + length[j])^2;



# makes sure that the variable hasSize has the correct value (first), no more than 1 section
                                                length (second),at least only 1 extraSize to be
                                                 used (third)

s.t. Has_Size{i in PIVOTS, j in SIZES}:
  numSec[i,j] <= hasSize[i,j]*UB;

s.t. Single_Pivot_Section_Size{i in PIVOTS}:
  sum{j in SIZES} hasSize[i,j] <= 1;

s.t. Single_Extra_Size{i in PIVOTS}: sum{j in EXTRA_SIZES} extraSize[i,j] <= sum{k in SIZES
                                                diff EXTRA_SIZES} hasSize[i,k];

s.t. Calculating_Length{i in PIVOTS}:
```

```
  length[i] = ((sum{k in EXTRA_SIZES} k*extraSize[i,k]) + (sum{j in SIZES} j*numSec[i,j]));

s.t. Calculating_Length2{i in PIVOTS}:
  length2[i] = length[i]^2;


# force a maximum length for eachpivot

s.t. Pivot_Exists{i in PIVOTS}:
  length[i] <= max_length*exist[i];



# we calculate the projection of the center of the pivot to the edges of the polygon and
                                                obtain the scalar t that appears, it will help
                                                us to calculate the distance from the center to
                                                 the edge

s.t. Calculating_T{i in PIVOTS,j in SEGMENTS}:
  ((x[i]-x2[j])*(x1[j]-x2[j])+(y[i]-y2[j])*(y1[j]-y2[j]))/((x2[j]-x1[j])^2 + (y2[j]-y1[j])^2)
                                                = t[i,j];


# calculate the distance according to the case in which we find ourselves
    # Objetivo T < 0 => distSeg = dist((x,y),(x2,y2))

s.t. T_Less_0{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,1]*t[i,j] <= 0;

s.t. Calculating_Distance_Seg_1{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,1]*((x[i]-x2[j])^2 + (y[i]-y2[j])^2 ) = scenDistSeg[i,j,1]*distSeg[i,j];


    # Objetivo 0 < T < 1 => distSeg = distLine

s.t. T_Greater_0{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,2]*t[i,j] >= 0;

s.t. T_Less_1{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,2]*(t[i,j]-1) <= 0;

s.t. Calculating_Distance_Seg_2{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,2]*((x2[j]-x1[j])*(y1[j]-y[i]) - (x1[j]-x[i])*(y2[j]-y1[j]))^2 /
  ((x2[j]-x1[j])^2 + (y2[j]-y1[j])^2) = scenDistSeg[i,j,2]*distSeg[i,j];

    # Objetivo T > 1 => distSeg = dist((x,y),(x1,y1))

s.t. T_Grater_1{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,3]*(t[i,j]-1) >= 0;
```

```
s.t. Calculating_Distance_Seg_3{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,3]*((x[i]-x1[j])^2 + (y[i]-y1[j])^2 ) = scenDistSeg[i,j,3]*distSeg[i,j];

s.t. In_Some_Scenario{i in PIVOTS, j in SEGMENTS}:
  scenDistSeg[i,j,1]+scenDistSeg[i,j,2]+scenDistSeg[i,j,3] = 1;
```

5.3 Executable .run file

We are going to explain what we have to code in the .run file to solve our problem in AMPL. First of all, we need to load the contents of the .mod file using the command sequence *model codigo.mod*. After this, we load the contents of the .dat file using *data codigo.dat*. Now, we tell the solver that we wish to make use of the gurobi solver, through the line *option solver gurobi* and then, we also need to specify options for this solver: *option gurobi_ options ' NonConvex = 2 '*. This is because our problem is non-convex and gurobi assumes the problem is convex. Finally, we ask AMPL to solve the problem with all the indications given by writing *solve* and ask them to show us the solution through the function display.

```
reset;

model codigo.mod;
data codigo.dat;
option solver gurobi;
option gurobi_options 'NonConvex=2';

solve;

for{i in PIVOTS} {
  if exist[i]=1 then {
    print "There is a pivot in the position (",x[i],",",y[i],") of length ",length[i],".";
    for{j in SIZES} {
      if hasSize[i,j]=1 then
        print "The pivot has ",numSec[i,j]," sections of length ",j,".";
    };
    for{k in EXTRA_SIZES} {
      if extraSize[i,k]=1 then
        print "In addition, it has an extra section of length ",k,".";};

  };
}
print "";
print "";
display _ampl_time;
display _total_solve_time;
```

6 Execution and output

Before moving on with the results we will discuss weird behaviour of the solver during execution.

When executing the model we found ourselves with a problem. Sometimes, if there were many pivots, the solver would not find any factible solutions or the solutions given were clearly suboptimal even after running for a long time. What is even more weird is that when multiplying the objective function by a constant (which does not change the optimal solution) it found better (or at least just feasible) solutions. We do not know why this happens, but in order to get the result for 4 pivots we multiplied the objective funtion by 10.

As we stated in the previous section we did not allow for the center of a pivot to be in the polygons P2, P3 or P4, due to computational limitations. This should not be an issue since we do not think any optimal solution would have a pivot center there, since the pivot would be too small.

Solutions for 3 and 4 pivots are shown below:

```
# Obtained output for 4 pivots
There is a pivot in the position ( 1495.59902607124 , 819.4604823326696 ) of length  364 .
The pivot has  7  sections of length  52 .
There is a pivot in the position ( 990.8286548175666 , 1354.223757650374 ) of length  247 .
The pivot has  4  sections of length  52 .
In addition, it has an extra section of length  39 .
There is a pivot in the position ( 2102.000520902596 , 389.6448490843216 ) of length  284 .
The pivot has  5  sections of length  49 .
In addition, it has an extra section of length  39 .

_ampl_time = 0.010587

_total_solve_time = 15801.3
```

```
# Obtained output for 3 pivots
There is a pivot in the position ( 1002.351683078641 , 1319.762010064251 ) of length  270 .
The pivot has 5 sections of length 54 .
There is a pivot in the position ( 2080.17385773558 , 433.3538280204568 ) of length  312 .
The pivot has 6 sections of length 52 .
There is a pivot in the position ( 1499.018683440716 , 816.9451288139087 ) of length 364 .
The pivot has 7 sections of length 52.

_ampl_time = 0.010496

_total_solve_time = 112347
```

These solutions might be suboptimal since we had to abort the execution while running, and not let the execution run until the process was completely finished.

As we can see, the solution for 4 pivots installs three pivots but does not use the fourth one. This would mean the optimal solution does indeed only use 3 pivots, so we are not considering finding solutions for more pivots, and we will run once again our code to find the optimal allocation of 3 pivots. We cannot be sure since as it was already stated, the solutions could be suboptimal.

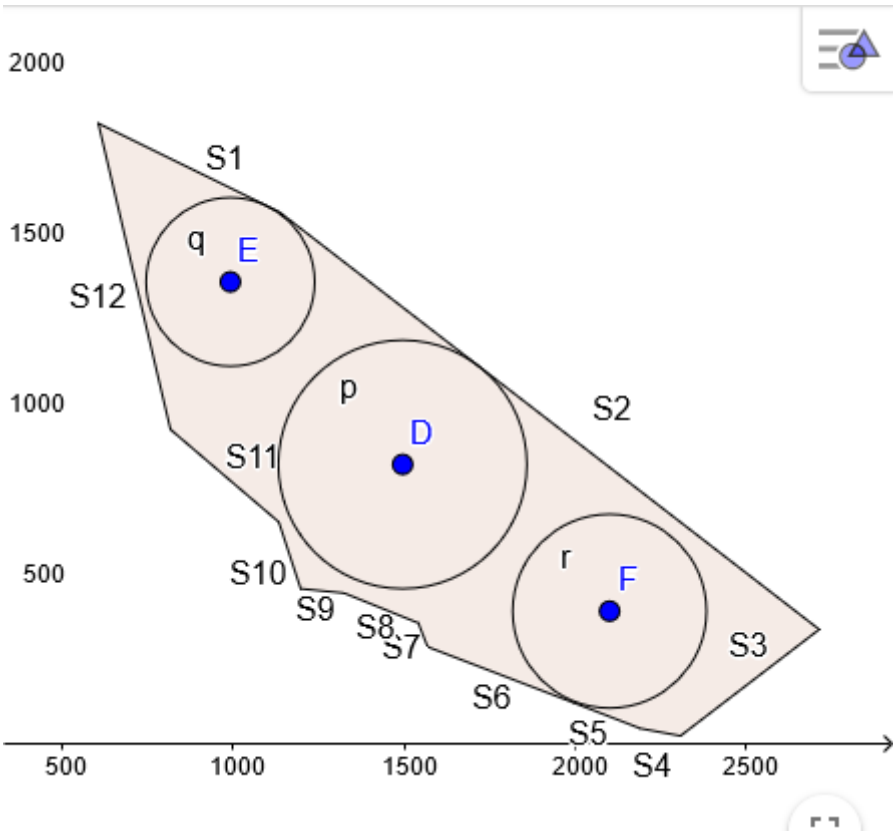

Figure 19: Solution with 4 pivots

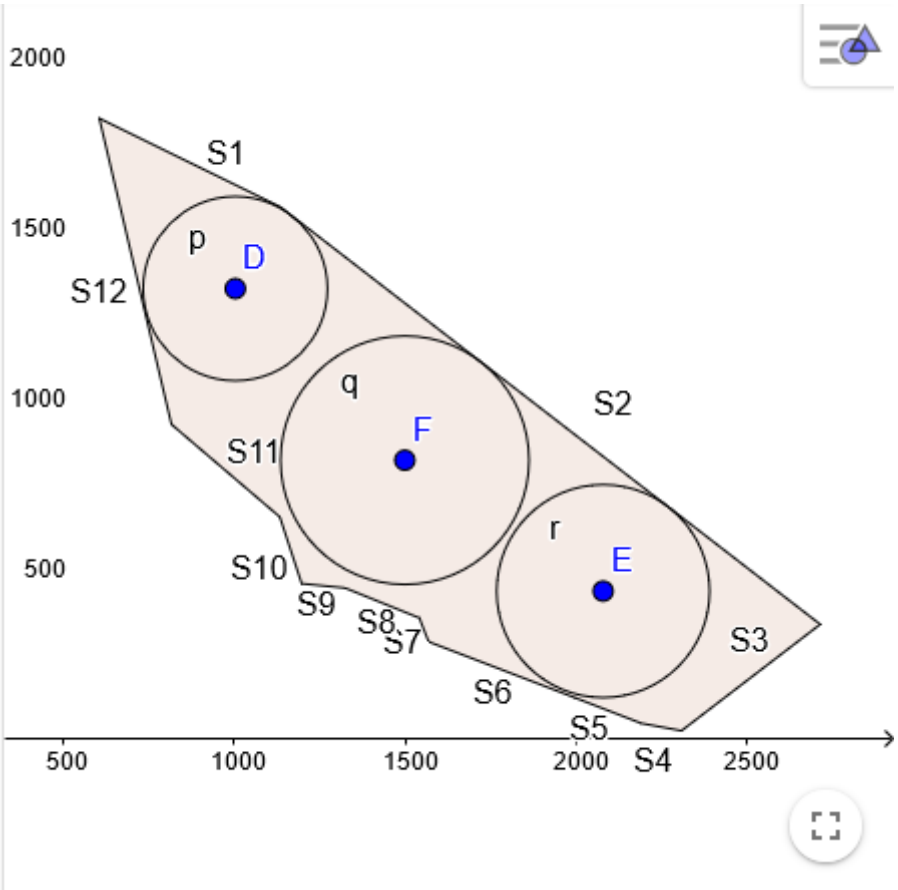

Figure 20: Solution with 3 pivots

Figure 21: Current pivot arrangement

Because the parcel is quite small the result is not that interesting, although it could be useful for determining the exact value of the radius of the pivots.

When comparing our result with the current distribution (shown below) we can see it is basically the same distribution. Therefore we have not been able to find a better distribution to the current pivot arrangement, which was installed four decades ago now.

When trying to install more than 3 pivots, the program decides that the most optimal approach, given the farm, its size, and the purchase and installation costs, is to reinstall 3 pivots in a manner almost identical to the existing setup. It's not exactly the same arrangement, because in our problem we are using machinery cost data, which, as we previously explained in Section 2, might not be very accurate estimates. This somewhat skews the solution. However, the program demonstrates that the arrangement of the pivots established 40 years ago in this farm is actually optimal, a thing we did not really expect, and cannot be improved by attempting to install more pivots. This was the information the company wanted to know and what we sought to discover.

On the one hand, these results are satisfying to us because they indicate that our program works as expected. However, it is also frustrating to see that we have not been able to find a better arrangement for the property's redesign, which was our initial proposal to the company, as a no better layout seems to exist.

7 Conclusions

In conclusion, mathematical modeling techniques and optimization can be employed in tandem to describe, simulate, and derive outcomes for a myriad of real-life situations.

In this particular study, we have successfully recreated a farm using the AMPL optimization language. To achieve this, we translated the geometric shape of the farm into an approximation using a union of convex polygons. We utilized Python to define the edges of these polygons, which provided us with the coordinates of the corners based on an image of the farm. Through another system, we scaled the points so that the distances between them accurately reflected reality, resulting in a faithful representation of the land.

By delineating the farm's boundaries in AMPL with pairs of points (one for the beginning of the segment and another for the end, denoting an edge), we addressed the primary challenge of positioning irrigation systems within the farm, as initially intended.

This study allowed us to determine if there was a better way to relocate and install new irrigation systems to cover a larger area of the farm than what the previous arrangement provided.

Through a series of constraints, we developed a model that identifies the ideal location based on maximizing coverage and minimizing cost, considering two different types of irrigation systems. However, we encountered a capacity limitation issue with the solver, which prevented us from obtaining results beyond a certain number of pivots.

In the future, this model could be used by installers and other agricultural businesses for every farm and parcel as long as their borders are straight lines, streamlining the process of locating and configuring irrigation systems. A software like this one, selecting the best irrigation system arrangement does not currently exist: all calculations by most technicians and installers are done on paper using rulers, pencils and compasses. Our model and program now makes this process easier, automatic, and it also ensures optimal irrigation system arrangement in any polygon, whether it is convex, or non-convex.

Specifically, for this farm, we found that the optimal solution involves using 3 pivots and covering the rest of its surface with coverage sprinkler irrigation; in an arrangement similar to the current one, so our advice to the company would be to keep the current disposition unchanged, and to simply change parts, since it's already optimal, and our model has shown there exists no better way to cover this surface.

Our model highlights the importance of information science and mathematics in the technical field, serving as both a substitute and supplement to the current method of determining these factors: using a compass and a ruler on a map.

We believe that the obtained results and program could be much more interesting for larger parcels, and especially, those with a greater width, as this would allow us to better observe and understand the performance of our model.

It is worth noting that implementing certain considerations due to the irregularity of the surface proved to be costly and unintuitive. Working with this problem in general was challenging, particularly due to the presence of non-convex shapes, the non-linear nature of the problem, and the combinatorial possibilities of the segments. Furthermore, this nature of the problem has resulted in long execution times of the code, and the solver having trouble to find optimal arrangements. This has given us an understanding of the limits of integer

non-linear programming software. Mathematical optimization is an intriguing field, but there are times when some of the tools used, and our models may take a considerable time to return a solution, or be developed, and times the solvers available may have trouble finding a solution for the problem posed.

References

[1] *https://gis.stackexchange.com/questions/193482/how-to-calculate-how-many-polygons-i-can-put-insid*

[2] Jose Fernando Oliveira Jeinny Peralta, Marina Andretta. Packing circles and irregular polygons using separation lines. *https://pdfs.semanticscholar.org/28d1/23ade38205987a36ece611c1ef87a83d44c7.pdf*.

[3] David Eppstein Marshall Bern. Quadrilateral meshing by circle packing. *https://www.ics.uci.edu/~eppstein/pubs/BerEpp-MRT-97.pdf*.