Proyecto del curso: Complejidad y Optimización de Algoritmos
Encontrar la posición más alejada posible de la ciudad más cercana.

Jhon Alejandro Córdoba Figueroa

Febrero 2022

Contents

1 Introducción

La región EcoReg tiene un problema serio de depósito de basuras, y ha decidido construir un nuevo relleno dentro de sus fronteras, con el fin de encontrar la ubicación óptima para el relleno sanitario. Se considera ubicación óptima, a la ubicación que corresponde a la más alejada de la ciudad más cercana. A continuación se describe el modelo y sus características como parámetros, variables, restricciones, función objetivo, análisis y conclusiones.

Para entender con mayor facilidad el problema y el modelo aquí planteado, se recomienda primero ver el vídeo que se encuentra en el siguiente enlace: https://youtu.be/ZSy-bH2svNs.

2 Modelo

2.1 Parámetros

- Sea n (sizeOfSquareRegion) una variable de decisión tipo entera que representa el tamaño de la región (al ser una región cuadrada, basta con esta única variable para representar el tamaño).
- Sea m (amountOfCities) una variable de decisión tipo entera que representa la cantidad de ciudades en la región.
- Sea $locations$ una matriz de tamaño mx2, tal que $i \in locations$.
- $location_i$: representa la ubicación de la ciudad i. $location_i$ es un vector tipo entero de tamaño 2, donde el primer elemento, es decir, $location_{i0}$ representa la ubicación en el Oeste y el segundo, $location_{i1}$, la ubicación en el Sur.

2.2 Variables

- $landfillWest$: Es una variable tipo flotante la cual representa la posición Oeste del relleno sanitario.
- $landfillSouth$: Es una variable tipo flotante que representa la posición Sur del relleno sanitario.
- $nearestCity$: Es una variable tipo enteram, la cual representa el index de la ubicación de la ciudad más cercana en la matriz $locations$. Aunque no es requerida para dar solución al problema, ya que, en realidad solo se necesita la ubicación del relleno sanitario, es interesante saber cuál será la ciudad más cercana y por ende la más afectada. Como esta variable representa un punto en la región, vamos a representar la coordenada Oeste como $nearestCity_0$ y a la coordenada Sur como $nearestCity_1$

2.3 Restricciones

1. $landfillWest, landfillSouth \geq 0$
2. $landfillWest, landfillSouth \leq n(sizeOfSquareRegion)$
3. $\forall i \in locations, locations_{i0} \neq landfillWest \vee locations_{i1} \neq landfillSouth$
4. $\forall i \in locations, |(landfillWest - nearestCity_0)| + |landfillSouth - nearestCity_1| \leq |(landfillWest - locations_{i0})| + |(landfillSouth - locations_{i1})|$

2.4 Función Objetivo

- Maximizar la distancia medida con el **método de Manhattan** entre el punto del relleno sanitario y la ciudad que esté más cercana a dicho punto, es decir la ciudad que encontramos gracias a la restricción #4. Las restricciones garantizan que no hay otra ciudad que pueda estar más cerca y a la vez tener la mayor distancia:

  Maximice: $|(landfillWest - nearestCity_0)| + |(landfillSouth - nearestCity_1)|$

2.5 Explicación del modelo, detalles importantes

El modelo representa la problemática (conjunto de problemas) de la región EcoReg representando los parámetros y variables de decisión necesarios, descritos anteriormente.

El aspecto más relevante, es la restricción que primero garantiza la selección de una ciudad más cercana (restricción #4) y que a su vez debe cumplir la condición de tener la distancia más alejada posible, esta última condición, de tener la mayor distancia posible, está implícita en la maximización de la función objetivo. La restricción #4 utiliza un forall para calcular la distancia con el método de Manhattan sobre la ciudad representada por *nearestCity* y el *landFill* indicando que dicha distancia debe ser menor o igual entre el resto de ciudades y el landfill.

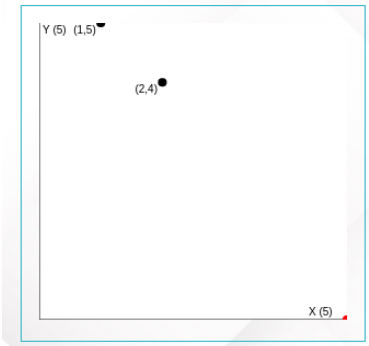
En otras palabras, la restricción #4, es la que garantiza que el solver encuentre el punto del *landFill*, tal que, halla la distancia más grande *posible* entre la ciudad *posible* más cercana y la *posible* ubicación del *landFill*, calculando la distancia de Manhattan y a su vez maximizando la distancia entre dicha ciudad y el punto del relleno sanitario.

3 Pruebas

A continuación se describen algunas de las pruebas realizadas a la implementación del modelo planteado.

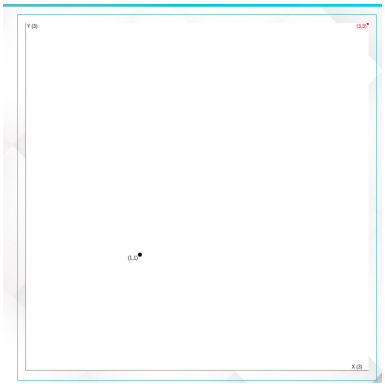
3.1 Prueba 1

sizeOfSquareRegion = 5; amountOfCities = 2; locations = [1,5; 2,4];



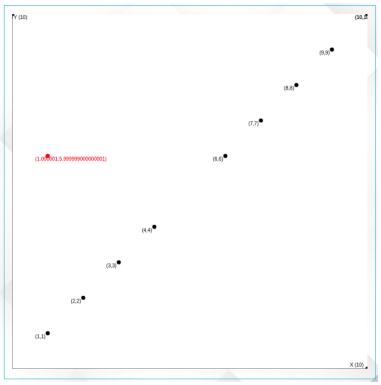
3.2 Prueba 2

sizeOfSquareRegion = 3; amountOfCities = 1; locations = [1, 1];



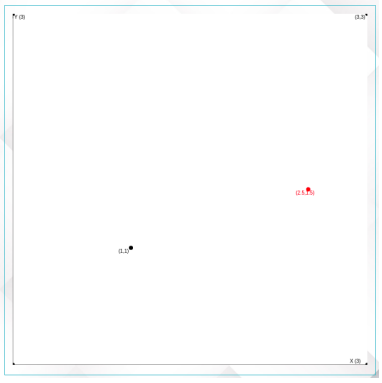
3.3 Prueba 3

sizeOfSquareRegion = 10; amountOfCities = 12; locations = [1, 1 ;2, 2 ;3, 3 ;4, 4 ;6, 6 ;7, 7 ;8, 8 ;9, 9 ;10, 10 ;0, 10 ;10, 0 ;10, 10];



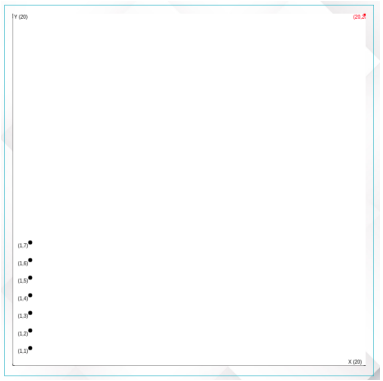
3.4 Prueba 4

sizeOfSquareRegion = 10; amountOfCities = 12; locations = [1, 1 ;2,2 ;3,3 ;4,4 ;6,6 ;7,7 ;8,8 ;9,9 ;10,10 ;0,10 ;10,0 ;10,10];



3.5 Prueba 5

sizeOfSquareRegion = 20; amountOfCities = 7; locations = [;1,1 ;1,2 ;1,3 ;1,4 ;1,5 ;1,6 ;1,7];

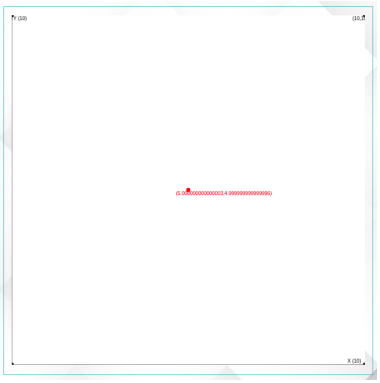


3.6 Prueba 6

sizeOfSquareRegion = 10;
amountOfCities = 5; locations = [5,
5 ;0, 0 ;10, 0 ;10, 10 ;0, 10 ;];

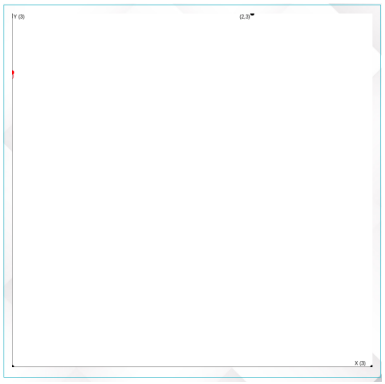
3.7 Prueba 7

sizeOfSquareRegion = 10;
amountOfCities = 4; locations = [0,
0 ;10, 0 ;10, 10 ;0, 10 ;];



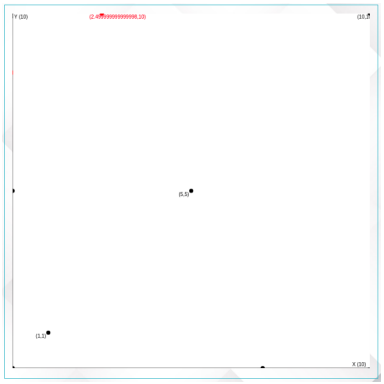
3.8 Prueba 8

sizeOfSquareRegion = 3;
amountOfCities = 3; locations = [0,
0 ;2, 3 ;3, 0 ;];



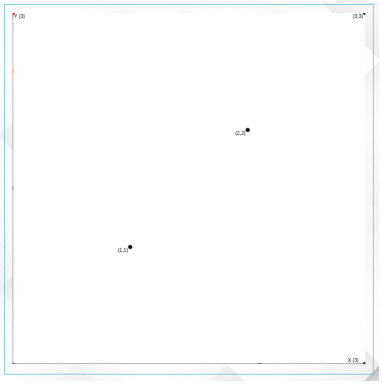
3.9 Prueba 9

sizeOfSquareRegion = 10;
amountOfCities = 6; locations = [0,
0 ;0, 5 ;5, 5 ;7, 0 ;1, 1 ;10, 10 ;];



3.10 Prueba 10

sizeOfSquareRegion = 3;
amountOfCities = 4; locations = [1, 1 ;2, 2 ;3, 3 ;3, 0 ;];



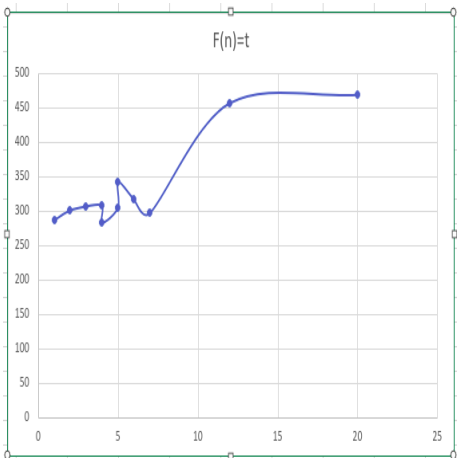
4 Análisis y Conclusión

Para analizar las pruebas realizadas, se ejecutó 5 veces cada prueba y se calculó el tiempo promedio de ejecución de cada una de ellas, ya que cada vez que se ejecutaba daba un tiempo diferente, por lo tanto utilizando el promedio se intentó trabajar con valores más aproximados a la realidad o más correctos. Luego se creó una tabla teniendo en cuenta el número (#) de prueba, tamaño de la entrada y el promedio de ejecución mencionado, y luego se graficó la función tiempo de ejecución:

$$F(n) = t$$

Donde n es el tamaño de la entrada y t el tiempo de ejecución en milisegundos

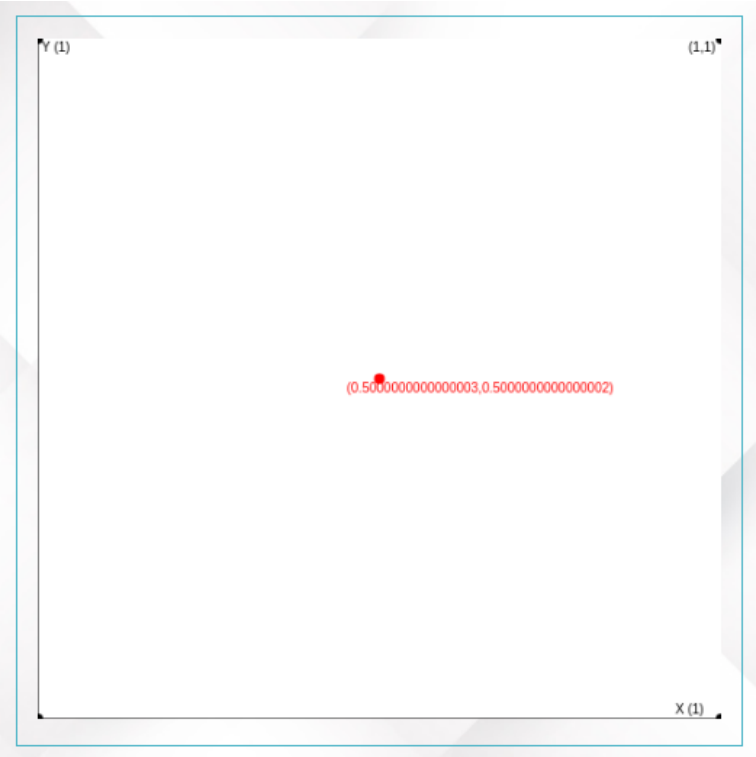
# de prueba	tamaño de entrada	promedio tiempo de ejecución (ms)
# 2	1	287
# 1	2	301
# 8	3	307
# 10	4	308
# 7	4	283
# 4	5	305
# 6	5	343
# 9	6	317
# 5	7	297
# 3	12	457
$dat20_1$	20	469



Podemos observar que en este modelo, el tiempo de ejecución no está determinado solo por el tamaño de la entrada, porque aunque es evidente que hay una relación entre estas 2 variables, hay casos como el de la **Prueba 5** que con 7 datos de entrada logró obtener un mejor tiempo promedio de ejecución que el de la **Prueba 1** , en este problema el tiempo de ejecución está más relacionado con la complejidad o dificultad de hallar la ubicación óptima para el *landfill* que con la cantidad de datos de entrada; lo cual, no es lo mismo, porque podemos tener una entrada **n** grande; pero con una solución "obvia" porque todos los puntos están concentrados, por ejemplo en la parte superior de la región, el solver rápidamente encuentra que el punto óptimo está en la parte más superior de la región.

Es importante resaltar la implicación que conlleva, el tipo de las variables $landfillWest$ y $landfillSouth$, al ser tipo flotante (continuas) y que no estén restringidas a quedar sobre intersecciones, implica que siempre encontraremos una solución a cualquier instancia del problema. Por ejemplo:

la región más pequeña que podemos tener, es una región de tamaño 1x1, supongamos que hay 4 ciudades en los 4 posibles puntos de dicha región, el sistema da como solución el punto (0.5, 0.5). Ésta es la prueba más clara, de que no hay una instancia del modelo insatisfactible, ya que hay infinitos puntos posibles en dicha región 1x1, en donde podemos colocar el relleno sanitario sin que éste esté sobre una ciudad:



5 Sobre la GUI

Para construir la interfaz gráfica de usuario, se utilizaron varias tecnologías:

- NestJS: Framework de NodeJS, para crear una pequeña API-REST, la cual permite guardar información en archivos, ejecutar el solver en minizinc, etc.
- Vue JS: Aproveché la reactividad que brinda este framework para ahorrar trabajo al momento de lograr una GUI más interactiva, realizar validaciones y aumentar la experiencia de usuario.
- Canvas: Esta API que brindan los navegadores, me fue útil para representar el modelo en un plano cartesiano, no se utilizaron librerías adicionales para lograr esto, se creo el plano cartesiano solo usando el poder de Canvas. Algo muy interesante fue el hecho de que se necesito escalar las dimensiones de la región especificada por el usuario al tamaño del Canvas que se mide en píxeles, y lo programé para que tomará el tamaño de acuerdo a la pantalla, esto se logró, usando una regla de 3.