TB2: Tópicos en Ciencias de la Computación

Constraint Programming in real life problems



Integrantes:

Natalia Maury  
Joaquín Galván  
Carlos Iparraguirre

Profesor

Willy Ugarte

Ciclo: 2022 - 01

**Introducción**

Constraint Programming es encontrar la solución al problema dado cumpliendo todas sus restricciones. Para ello, se utiliza un modelo que contiene las variables, restricciones, dominio de las variables, etc. Esta es una fase muy importante ya que define el problema con sus características que permitirán encontrar una solución. Una vez obtenido el modelo, se utiliza un solver para que lo resuelva, el resultado es la(s) solución(es) al problema. El solver utilizará diversas técnicas como IA, backtracking, etc. para resolver el modelo, y al existir múltiples solvers en diferentes lenguajes de programación, no se creará un software de cero sino se utilizarán los existentes.

Existen dos tipos de constraint programming, CSP y COP. Constraint Satisfaction Problem (CSP) es un problema donde se deben encontrar todas las soluciones al problema y que todas las restricciones sean cumplidas. Constraint Optimization Problem (COP) es un problema donde se busca la solución más óptima que cumpla con todas las restricciones.

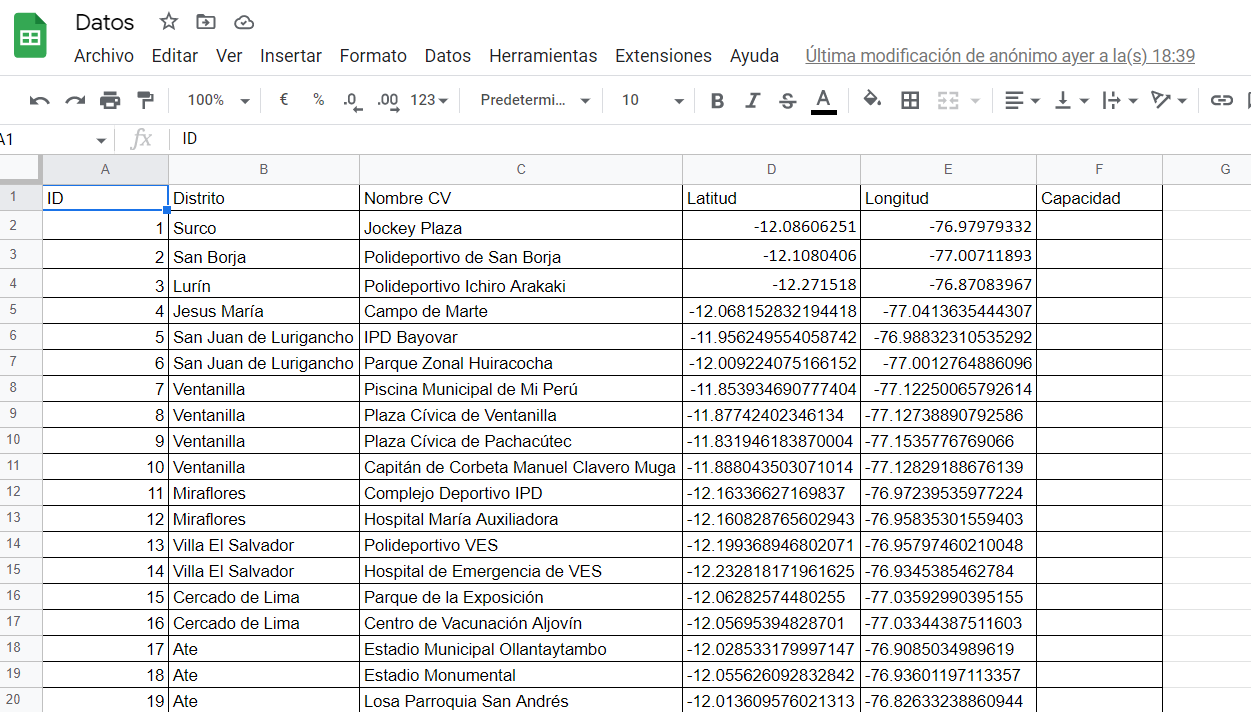
El objetivo del trabajo es aplicar constraint programming en un problema de la vida real, que en este caso es organizar a las personas en los centros de vacunación para que puedan ser vacunadas de la forma más eficiente. Para ello se debe considerar que las personas con mayor edad deben ser prioritarias al momento de ser asignadas una vacuna ya que son más vulnerables. También, al momento de asignar centros de vacunación, lo ideal es tener en cuenta los distritos y la localización de cada persona para que sea más eficiente el proceso. Por último, se debe considerar que se debe asignar como máximo una vacuna por persona para evitar la menor cantidad de personas que no tengan vacunas. Con ello, se busca resolver el problema generado en la pandemia de inmunizar a la población lo más rápido y eficientemente posible contra el covid 19.

Para la resolución del problema descrito anteriormente, el enunciado describe que se tienen hard constraints y soft constraints. Las hard constraints son restricciones indispensables para resolver el problema, mientras que las soft constraints son restricciones que son ideales que se cumplan pero no indispensables para la resolución del problema. En este problema, las hard constraints son: cada vacuna debe ser aplicada como máximo a una persona, debe haber como máximo un centro asignado a cada persona, las personas vacunadas no deben ser asignadas a ningún centro y las personas infectadas no deben ser vacunadas hasta 3 meses después. Las soft constraints del problema son: cada persona sin vacunar debería ser asignada a una vacuna, cada persona debería ser asignada al centro más cercano y las personas mayores de edad deberían ser asignadas primero.

**Problema**

Para la resolución del problema, primero se tuvo que obtener los datos para poder utilizar en el modelo. Para ello, se buscaron los 30 centros de vacunación y su ubicación en los diferentes distritos de Lima. A partir de esta información, se armó un excel donde se pusieron todos los datos: nombre del centro de vacunación, distrito, coordenadas (latitud y longitud) y su capacidad. Este archivo llamado ‘Datos’ se descargó en el formato .csv y se cargó en el código. En el notebook, se lee el archivo y se almacena en un dataframe llamado ‘Info\_CV’. A partir de este dataframe, se obtendrán los datos de localización de los centros de vacunación y se utilizarán para poder dibujar los distritos en el mapa.

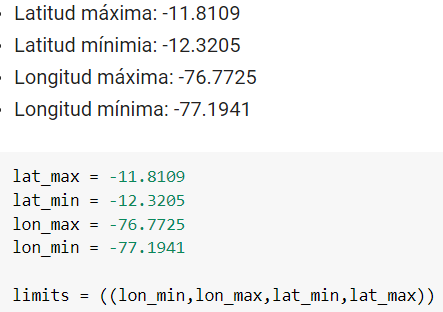
Archivo ‘Datos’ que contiene la información



Archivo cargado en el código



Para los datos de localización de las personas, se generan aleatoriamente en todos los distritos teniendo en cuenta la latitud y longitud de estos. Para ello, se deben establecer los límites máximos y mínimos de ambas cantidades: latitud máxima, latitud mínima, longitud máxima y longitud mínima.

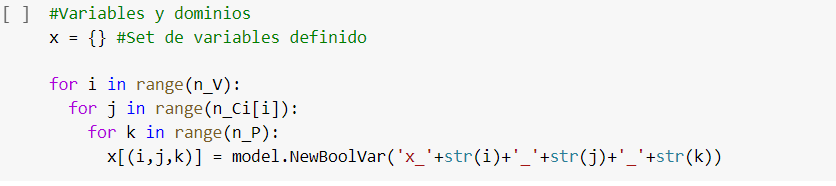


Una vez obtenidos estos valores, se generan aleatoriamente la localización de las personas.



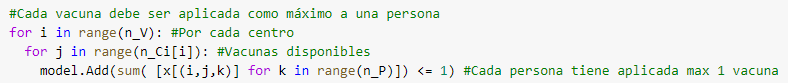
Para el modelo, se requieren múltiples variables para poder resolver el problema que son: n\_P que es la cantidad de personas, n\_CV que es la cantidad de centros de vacunación, Vac que es un vector booleano que indica si las personas están vacunadas o no, n\_Ci que es un vector que contiene la cantidad de vacunas (Capacidad) de cada centro de vacunación.

Todas estas variables serán utilizadas por la variable booleana del modelo que es en 3 dimensiones llamada ‘x’. La variable ‘x’ está compuesta por el set de variables de n\_P, n\_CV y n\_Ci, y al ser una variable booleana su dominio es de 0-1: xijk ∈ {0, 1} donde (i, j, k) ∈ V × Ci × P. Si en el centro i, la vacuna j es tomada por la persona k, entonces xijk será 1. Si en el centro i, la vacuna j no es tomada por la persona k, entonces xijk será 0.

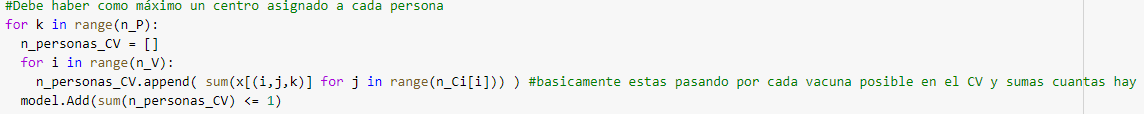


Una vez agregadas todas las variables al modelo, se deben implementar las restricciones. Primero se empezará por agregar las hard constraints y luego se implementarán las soft constraints en el modelo.

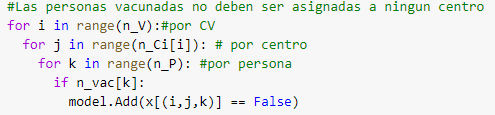
Primero se debe verificar que cada vacuna sea asignada a máximo una persona.



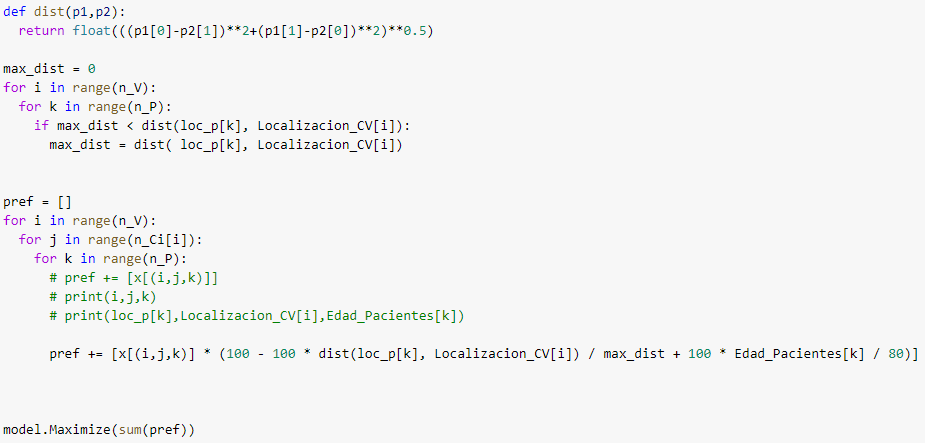
Luego se debe asignar a cada persona como máximo un centro de vacunación



Finalmente, se verifica que las personas vacunadas no sean asignadas a ningún centro de vacunación

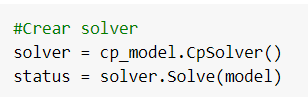


Una vez implementadas las hard constraints, se procede a implementar las soft constraints. Para ello se debe añadir primero la función de distancia para poder calcular la distancia entre las personas y los centros de vacunación. Luego se debe encontrar la distancia máxima para que al agregar la variable al modelo se pueda realizar la maximización. El .maximize() buscará maximizar la edad del paciente para que los mayores tengan preferencia y al mismo tiempo minimizar la distancia entre la persona y el centro de vacunación.

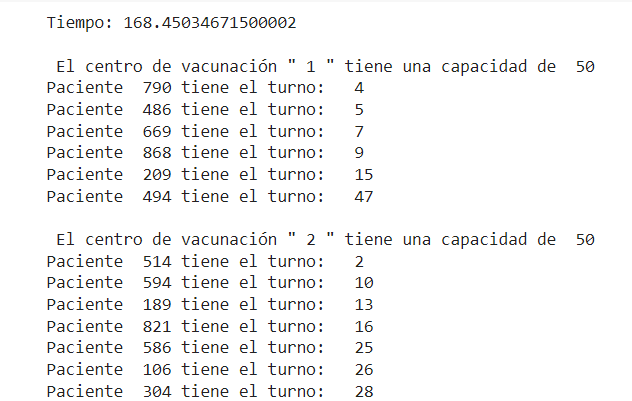


**Resultados**

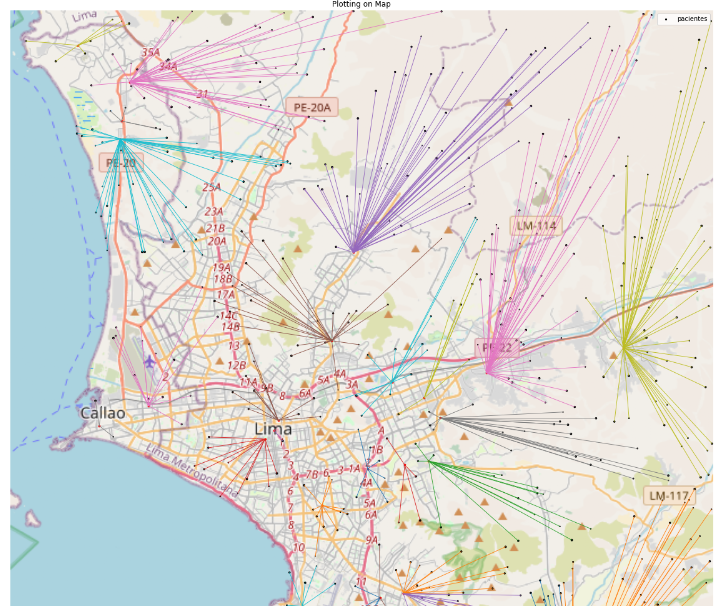
Una vez realizado el modelo, se creó en el solver para obtener la solución al problema.



Como se puede observar, el modelo funciona y tarda 169 segundos aproximadamente en resolver el problema planteado, siendo bastante eficiente. Logra asignar a la mayor cantidad de personas al centro de vacunación más cercano. Sin embargo, por problemas de memoria no se pudo probar con los 10 000 datos de personas y por falta de tiempo no se pudo implementar una interfaz.



El mapa final con las personas y sus centros de vacunación es el siguiente:



**Conclusiones**

El modelo mostró ser efectivo para resolver el problema planteado. Si bien este problema también podría haber sido resuelto con un modelo de data mining o machine learning, constraint programming era más efectivo y la mejor solución para ello. Los modelos de machine learning o data mining necesitan datos de entrenamiento para poder funcionar y tener la mejor eficacia, mientras que constraint programming no necesita ningún tipo de entrenamiento. Por ende, el problema tendría la mejor solución aplicando constraint programming ya que puede soportar múltiples parámetros, no requiere entrenamiento de datos y es bastante rápido.

**Links:**

gob.pe (2022). *Consultar centros de vacunación contra la COVID-19 e influenza en Lima Metropolitana y Callao*. Recuperado de <https://www.gob.pe/centrosdevacunacion/> [Consultado el 22 de mayo del 2022]

Collab con el código (modelo): <https://colab.research.google.com/drive/1gXbmru2a_XmMF7vl9ADSWOyBkW-BD589?usp=sharing>

Excel con los datos: <https://docs.google.com/spreadsheets/d/19lirOgkcPSptq-yFP-oHlq1UciccC2XjcRzW6-Msny8/edit?usp=sharing>

Word con los datos (investigación): <https://docs.google.com/document/d/1XeoDEp7WPeoxTKnqib9FmkfpjDGnk2o0yaW4WOwe32g/edit?usp=sharing>