Proyecto de curso:

Problema de la Ubicación de Instalaciones de Capacidad Acotada (PUICA)

Grupo 9:
Juan Sebastian Cifuentes Vallejo - 2179800
Hernan David Cisneros Vargas - 2178192
Jhon Alexander Valencia Hilamo - 2042426
Kevin Alejandro Velez Agudelo - 2123281

Profesor. Juan Francisco Díaz Frias
Profesor. Jesús Alexander Aranda
Monitor. Mauricio Muñoz

Análisis de Algoritmos II
Escuela de Ingeniería de Sistemas y Computación

Mayo de 2024

Tabla de contenido

1. introducción

En el contexto de la gestión eficiente de la producción y distribución de bienes, la correcta ubicación de instalaciones de producción es un problema crítico que afecta directamente los costos y la capacidad de satisfacer la demanda de los clientes.

En este informe, se han recopilado los aspectos más relevantes del proceso de diseño, desarrollo, implementación y prueba que realizamos para dar solución al problema

propuesto para el proyecto en cuestión, siendo este el Problema de la Ubicación de Instalaciones de Capacidad Acotada (*PUICA*), el cual nos reta a buscar la manera de determinar la ubicación óptima de instalaciones industriales para satisfacer la demanda de clientes distribuidos geográficamente, minimizando costos y maximizando beneficios.

Cumpliendo con el objetivo del presente proyecto y como equipo de trabajo, hemos buscado demostrar los conceptos y habilidades adquiridas durante esta segunda parte del curso, mostrando la capacidad para modelar y resolver problemas de optimización, explorando soluciones prácticas, mediante el uso de herramientas computacionales de modelamiento y soluciones existentes.

Se ha desarrollado un modelo genérico que resuelve el problema *PUICA,* utilizando el lenguaje de modelado *Minizinc,* el cual fue puesto a prueba con una serie de instancias diferentes del problema, que buscan probar su correctitud y analizar el comportamiento del mismo haciendo uso de los resultados de prueba obtenidos, en términos de optimización. Asimismo, se ha desarrollado una interfaz gráfica en *Python,* que facilita la configuración y ejecución del modelo construido, permitiendo visualizar los resultados de manera interactiva.

A continuación, se presentarán la formulación matemática del problema, la implementación del modelo y un análisis de los resultados obtenidos mediante pruebas realizadas sobre diversas instancias del problema.

2. El modelo

A continuación se presenta la descripción y justificación de adecuación del modelo genérico que hemos propuesto para resolver el Problema de *PUICA*. El modelo fue diseñado para optimizar la ubicación de instalaciones de producción con el objetivo de maximizar la utilidad total, considerando las restricciones de capacidad de producción y demanda de los clientes. Además, está formulado utilizando notación formal e implementado en el lenguaje de modelamiento MiniZinc, permitiendo su aplicación a diversas instancias del problema.

El modelo se formula en términos de parámetros, variables, restricciones y función objetivo, los cuales se describirán en detalle a continuación, explicando su función y su relación con el problema original.

$Sea\ C = \{1, \ldots, n\}\ el\ conjunto\ de\ clientes\ tal\ que\ c\ es\ un\ cliente\ si\ y\ sólo\ si\ c \in C.$
$Sea\ S = \{1, \ldots, m\}\ el\ conjunto\ de\ sitios\ tal\ que\ s\ es\ un\ sitio\ si\ y\ sólo\ si\ s \in S.$

Los anteriores son los conjuntos de clientes y sitios que se han establecido y que son fundamentales para la formulación del problema y la interpretación de las restricciones y la función objetivo.

El conjunto C representa a todos los clientes que necesitan ser atendidos por las instalaciones de producción. Cada cliente $c \in C$ tiene una demanda específica de productos que debe ser satisfecha por las instalaciones. Por lo tanto, $C = \{1, \ldots, n\}$ se refiere a los índices de los clientes, donde n es el número total de clientes en el problema.

El conjunto S representa todos los sitios potenciales donde se pueden ubicar las instalaciones de producción. Cada sitio $s \in S$ tiene una capacidad máxima de producción y un costo fijo asociado a su apertura. Por lo tanto, $S = \{1, \ldots, m\}$ se refiere a los índices de los sitios, donde m es el número total de sitios en el problema.

2.1 Parámetros

- $n \in N$: $el\ número\ de\ clientes.$
- $m \in N$: $el\ número\ de\ sitios.$
- $CostoFijo_s \in R$: $el\ costo\ fijo\ de\ construcción\ y\ puesta\ a\ punto\ de\ la\ instalación\ en\ el\ sitio\ s.$
- $CapacidadMaxima_s \in N$: $la\ capacidad\ máxima\ de\ producción\ de\ la\ instalación\ en\ el\ sitio\ s.$
- $Demanda_c \in R$: $la\ demanda\ del\ cliente\ c.$
- $BeneficioPorUnidad_{c,s} \in R$: $el\ beneficio\ por\ unidad\ vendida\ al\ cliente\ c\ desde\ la\ instalación\ en\ el\ sitio\ s.$

2.2 Variables de Decisión

- $Productos_{c,s} \in R$: $los\ productos\ que\ se\ enviarán\ al\ cliente\ c\ desde\ la\ instalación\ en\ el\ sitio\ s.$
- $Instalacion_s \in \{0, 1\},\ s \in S$: $el\ indicador\ de\ si\ se\ abre\ una\ instalación\ en\ el\ sito\ s\ (si\ Instalacion_s = 1\ se\ abre\ la\ instalación\ en\ el\ sitio\ s,\ si\ Instalacion_s = 0,\ no\ se\ abre\ la\ instalación\ en\ el\ sitio\ s).$

2.3 Restricciones

- $No\ se\ puede\ enviar\ una\ cantidad\ negativa\ de\ productos\ a\ un\ cliente$:

$$\forall c \in C, \forall s \in S, \left| Productos_{c,s} \geq 0 \right.$$

- $La\ demanda\ de\ los\ clientes\ se\ satisface\ (la\ demanda\ de\ cada\ cliente\ c\ se\ satisface)$:

$$\forall c \in C \left| \sum_{s \in S} Productos_{c,s} = Demanda_c \right.$$

$-$ *No se sobrepasar la cantidad de producción de ninguna de las instalaciones* (*no se sobrepasa la cantidad de producción de ninguna instalación ubicada en cualquiera de los s sitios, si no se realizó la instalación en sitio s entonces su capacidad de producción es* 0).

$$\forall s \in S \left| \sum_{c \in C} Productos_{c,s} \leq CapacidadMaxima_s * Instalacion_s \right.$$

2.4 Función Objetivo

$-$ *Maximizar la utilidad total*:

$$\sum_{s \in S} \left(\left(\sum_{c \in C} BeneficioPorUnidad_{c,s} * Productos_{c,s} \right) - CostoFijo_s * Instalacion_s \right)$$

La utilidad total se define como el beneficio total menos el costo fijo total.

El beneficio total es la suma de todos los beneficios individuales. Un beneficio individual se obtiene al vender una cierta cantidad de productos a un cliente c desde un sitio s. Esto se calcula como el beneficio por unidad multiplicado por el número de unidades vendidas a ese cliente desde ese sitio.

El costo fijo total es la suma de todos los costos fijos individuales. Un costo fijo individual es el costo de instalar una planta en un sitio s.

3. Detalles Importantes de la Implementación

La implementación del modelo de optimización se llevó a cabo utilizando Python y MiniZinc. Se desarrolló una interfaz gráfica en Python que permite al usuario seleccionar un archivo .txt que contiene la información sobre los parámetros del problema. Esta interfaz procesa el archivo .txt y genera un archivo .dzn con el mismo nombre en una carpeta llamada DatosPUICA. Adicionalmente, la información del .txt se utiliza para sobrescribir un archivo llamado input.dzn.

Posteriormente, por medio de un botón en la interfaz gráfica, se ejecuta un modelo en MiniZinc (.mzn) utilizando el solver ***HIGHS 1.6.0***, que importa los datos de input.dzn y realiza la optimización de acuerdo con la función objetivo definida en el modelo.

El programa permite al usuario seleccionar el archivo .txt desde el equipo, y luego genera el archivo .dzn correspondiente. Las restricciones del modelo en MiniZinc se definieron utilizando la palabra reservada `constraint`. Para aplicar restricciones a conjuntos, se utilizó el cuantificador `forall`, y para las restricciones que contienen sumatorias, se empleó la función `sum`.

En el modelo, la función objetivo se representa directamente mediante los parámetros $BeneficioPorUnidad_{c,s}$ y $CostoFijo_s$, junto con las variables de decisión $Productos_{c,s}$ e $Instalacion_s$. Sin embargo, en la implementación se establece a partir de una restricción que expresa la diferencia entre las variables `beneficio_total` y `costo_total_fijo_sitios`. `beneficio_total` representa la suma de todos los beneficios individuales, mientras que `costo_total_fijo_sitios` denota la suma de todos los costos fijos individuales. Ambas variables se construyen mediante restricciones que contienen la implementación de las variables de decisión que sí están definidas en el modelo.

4. Análisis de Árboles Generados por el Modelo y explicación del mecanismo de Branch and Bound

Los árboles fueron generados mediante el solver ***Geocode Gist 6.3.0*** y contienen diversos tipos de nodos, cada uno identificado por su forma y color. Estos nodos incluyen:

Círculos Azules: Representan las instancias consideradas durante la búsqueda. En general, indican la posibilidad de encontrar varias soluciones.

Cuadrados Rojos: Son instancias incorrectas que no cumplen con las restricciones del problema.

Rombos Verdes: Representan soluciones factibles, es decir, configuraciones que cumplen con todas las restricciones.

Triángulos Rojos: Indican ramas podadas durante la búsqueda, es decir, instancias que se descartan al no llevar a una solución factible.

Rombo Amarillo: Es la solución óptima del problema.

La salida en la consola de Gist muestra dos arrays y un valor numérico. "X_INTRODUCED_0_" es un array que representa la variable de decisión $Productos_{c,s}$ del modelo. Por otro lado, "instalaciones" es otro array que representa la variable de decisión $Instalacion_s$ del modelo. Finalmente, "utilidad_total" es un valor numérico que representa el objetivo a maximizar.

"X_INTRODUCED_0_" es un array que representa la matriz de productos enviados al cliente c desde el sitio s. Cada m elementos del array conforman una fila, y la cantidad de productos enviados al cliente c desde el sitio s se encuentra en la posición $(c * m) + s - m$ del array.

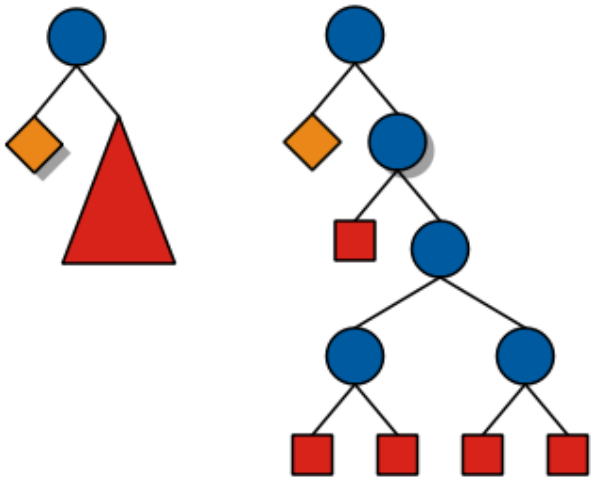
Para esta sección se referirá como “nodo A del nivel B” al nodo presente en el nivel B que visualmente tiene A-1 nodos del nivel B a su izquierda y se utilizarán los términos “característica” y “características” para referirse a nuevas restricciones que acotan los valores que las variables pueden tomar, estas “características” cumplen con las restricciones definidas inicialmente en el modelo.

4.1 Árbol 1

4.1.1 Parámetros

```
int: n_clientes = 2;
int: n_sitios = 2;
array[1..n_sitios] of float: costo_fijo_sitios = [2,2];
array[1..n_sitios] of int: capacidad_maxima_sitios = [6,6];
array[1..n_clientes] of float: demanda_clientes = [1.5,2.0];
array[1..n_clientes, 1..n_sitios] of float: beneficio_por_unidad =
[|4.0,4.0
|4.5,4.5|];
```

4.1.2 Árbol generado



4.1.3 Explicación

Nodo raíz:

Este nodo considera las instancias con las siguientes características.

X_INTRODUCED_0_ = array1d(1..4, [[[0..1.5]], [[0..1.5]], [[0..2]], [[0..2]]]);
instalaciones = array1d(1..2, [false..true, false..true]);
utilidad_total = [[-4..30]];

Los elementos de la primera y segunda fila de “X_INTRODUCED_0_” están acotados en los intervalos [0, 1.5] y [0, 2] respectivamente, ya que de acuerdo a las restricciones, se debe cumplir que no se puede enviar una cantidad negativa de productos a ningún cliente y se debe satisfacer la demanda de todos los clientes.

Los elementos de “instalaciones” pueden tomar los valores de verdadero o falso, según cómo se definió e implementó el modelo.

La utilidad está acotada en el intervalo [-4, 30] considerando el peor y mejor caso posible teniendo en cuenta las restricciones de la implementación pero no su coherencia.

Nodo 1 del nivel 1:

Este nodo encuentra una solución óptima con las siguientes características.

X_INTRODUCED_0_ = array1d(1..4, [1.5, 0.0, 2.0, 0.0]);
instalaciones = array1d(1..2, [true, false]);
utilidad_total = 13.0;

Esta solución instala únicamente la planta del sitio 1 y envía desde ella 1.5 productos al cliente 1 y 2.0 productos al cliente 2.

El nodo 2 del nivel 1:

Este nodo considera las instancias con las siguientes características.

X_INTRODUCED_0_ = array1d(1..4, [[[0..1.5]], [[0..1.5]], [[0..2]], [[0..2]]]);
instalaciones = array1d(1..2, [false..true, true]);
utilidad_total = [[13..28]];

Los elementos de la primera y segunda fila de “X_INTRODUCED_0_” están acotados en los intervalos [0, 1.5] y [0, 2] por las mismas razones que se explican para el nodo raíz.

El segundo elemento de “instalaciones” siempre es verdadero mientras que el primero puede ser tanto verdadero como falso, indicando que se considerarán las instancias en las que siempre se realiza la instalación en el sitio 2.

La utilidad está acotada en el intervalo [13, 28], considerando que el peor caso ahora debe ser mayor o igual al resultado óptimo actual y el mejor caso debe ajustarse a la nueva característica, "siempre instalar en el sitio 2", lo cual reduce la utilidad en 2 unidades, ya que instalar al menos un sitio cuesta al menos 2 unidades.

Nodo 1 del nivel 2:

Este nodo encuentra una solución óptima con las siguientes características, pero se presenta con un cuadrado rojo debido a que ya se había encontrado una solución óptima con el mismo valor de utilidad_total y no con uno menos (peor).

X_INTRODUCED_0_ = array1d(1..4, [0.0, 1.5, 0.0, 2.0]);
instalaciones = array1d(1..2, [false, true]);
utilidad_total = 13.0;

Nodo 2 del nivel 2:

Este nodo considera las instancias con las siguientes características.

X_INTRODUCED_0_ = array1d(1..4, [[[0..1.5]], [[0..1.5]], [[0..2]], [[0..2]]]);
instalaciones = array1d(1..2, [true, true]);
utilidad_total = [[13..26]];

Los elementos de la primera y segunda fila de “X_INTRODUCED_0_” están acotados en los intervalos [0, 1.5] y [0, 2] por las mismas razones que se explican para el nodo raíz.

Tanto el primer como el segundo elemento de “instalaciones” es verdadero, indicando que se considerarán las instancias en las que se realizan instalaciones en ambos sitios.

La utilidad está acotada en el intervalo [13, 26] considerando que el peor caso debe ser mayor o igual al resultado óptimo actual y el mejor caso debe ajustarse a la nueva característica, "siempre instalar en los sitios 1 y 2", lo cual reduce la utilidad en 4 unidades, ya que instalar en 2 sitios cuesta 4 unidades.

El nodo 1 del nivel 3:

Este nodo considera las instancias con las siguientes características.

X_INTRODUCED_0_ = array1d(1..4, [[[0.75..1.5]], [[0..0.75]], [[0..2]], [[0..2]]]);
instalaciones = array1d(1..2, [true, true]);
utilidad_total = [[13..23]];

Los elementos de la segunda fila de “X_INTRODUCED_0_” están acotados en el intervalo [0, 2] por las mismas razones que se explican para el nodo raíz pero ahora el primer y segundo elemento de la primera fila están acotados en los intervalos [0.75, 1.5] y [0, 0.75] respectivamente, con el fin de considerar las instancias con estas características, que por cierto también cumplen con las restricciones establecida.

Esta nueva característica descarta algunas soluciones que infringen la restricción “Se debe satisfacer la demanda de los clientes” que en el modelo se definió mediante una igualdad. Sin embargo esta nueva característica NO descarta todas las soluciones que infringen dicha restricción, por ejemplo, descarta la solución con “X_INTRODUCED_0_=[1.5, 1.5, 2.0, 0.0]” pero no la solución “X_INTRODUCED_0_=[1.5, 0.75, 2.0, 0.0]”.

Tanto el primer como el segundo elemento de “instalaciones” es verdadero, indicando que se considerarán las instancias en las que se realizan instalaciones en ambos sitios.

La utilidad está acotada en el intervalo [13, 23] considerando que el peor caso debe ser mayor o igual al resultado óptimo actual y el mejor caso debe ajustarse a la nueva característica relacionada a la primera fila de “X_INTRODUCED_0_”.

El nodo 2 del nivel 3:

Este nodo considera las instancias con las siguientes características.

X_INTRODUCED_0_ = array1d(1..4, [[[0..0.75]], [[0.75..1.5]], [[0..2]], [[0..2]]]);
instalaciones = array1d(1..2, [true, true]);

utilidad_total = [[13..23]];

Los elementos de la segunda fila de “X_INTRODUCED_0_” están acotados en el intervalo [0, 2] por las mismas razones que se explican para el nodo raíz pero ahora el primer y segundo elemento de la primera fila están acotados en los intervalos [0, 0.75] y [0.75, 1.5] respectivamente, con el fin de considerar las instancias con estas características, que por cierto también cumplen con las restricciones establecida.

Esta nueva característica descarta algunas soluciones que infringen la restricción “Se debe satisfacer la demanda de los clientes” que en el modelo se definió mediante una igualdad. Sin embargo esta nueva característica NO descarta todas las soluciones que infringen dicha restricción, por ejemplo, descarta la solución con “X_INTRODUCED_0_=[1.5, 1.5, 0.0, 2.0]” pero no la solución “X_INTRODUCED_0_=[0.75, 1.5, 0.0, 2.0]”.

Tanto el primer como el segundo elemento de “instalaciones” es verdadero, indicando que se considerarán las instancias en las que se realizan instalaciones en ambos sitios.

La utilidad está acotada en el intervalo [13, 23] considerando que el peor caso debe ser mayor o igual al resultado óptimo actual y el mejor caso debe ajustarse a la nueva característica relacionada a la primera fila de “X_INTRODUCED_0_”.

Todos los nodos del nivel 4:

Estos nodos consideran las instancias con las siguientes características, pero se presentan con cuadrados rojos debido a que en todos los nodos la cota de utilidad_total solo permite valores muy cercanos al valor óptimo ya obtenido (13) por lo tanto se considera que la brecha de optimización es aceptable y no se sigue expandiendo nodos.

X_INTRODUCED_0_ = array1d(1..4, [[[1.375..1.5]], [[0.25..0.375]], [[1..1.11111]], [[1..1.11111]]]);
instalaciones = array1d(1..2, [true, true]);
utilidad_total = [[13..13.5]];

X_INTRODUCED_0_ = array1d(1..4, [[[1..1.125]], [[0.625..0.75]], [[1..1.11111]], [[1..1.11111]]]);
instalaciones = array1d(1..2, [true, true]);
utilidad_total = [[13..13.5]];

X_INTRODUCED_0_ = array1d(1..4, [[[0.625..0.75]], [[1..1.125]], [[1..1.11111]], [[1..1.11111]]]);
instalaciones = array1d(1..2, [true, true]);
utilidad_total = [[13..13.5]];

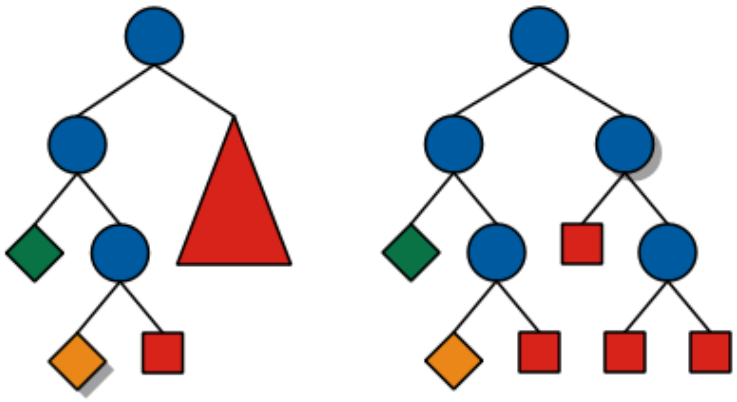
X_INTRODUCED_0_ = array1d(1..4, [[[0.25..0.375]], [[1.375..1.5]], [[1..1.11111]], [[1..1.11111]]]);
instalaciones = array1d(1..2, [true, true]);
utilidad_total = [[13..13.5]];

4.2 Árbol 2

4.2.1 Parámetros

```
int: n_clientes = 3;
int: n_sitios = 3;
array[1..n_sitios] of float: costo_fijo_sitios = [2,2,2];
array[1..n_sitios] of int: capacidad_maxima_sitios = [6,6,6];
array[1..n_clientes] of float: demanda_clientes = [1.5,2.0,1.0];
array[1..n_clientes, 1..n_sitios] of float: beneficio_por_unidad =
[|1.0,4.5,1.0
|1.5,4.5,1.5
|1.5,4.5,1.5|];
```

4.2.2 Árbol generado



4.2.3 Explicación

Nodo raíz:

Este nodo considera las instancias con las siguientes características.

X_INTRODUCED_0_ = array1d(1..9, [[[0..1.5]], [[0..1.5]], [[0..1.5]], [[0..2]], [[0..2]], [[0..2]], [[0..1]], [[0..1]], [[0..1]]]);
instalaciones = array1d(1..3, [false..true, false..true, false..true]);
utilidad_total = [[-6..32.25]];

Los elementos de la primera, segunda y tercera fila de “X_INTRODUCED_0_” están acotados en los intervalos [0, 1.5], [0, 2] y [0, 1] respectivamente, ya que de acuerdo a las restricciones, se debe cumplir que no se puede enviar una cantidad negativa de productos a ningún cliente y se debe satisfacer la demanda de todos los clientes.

Los elementos de “instalaciones” pueden tomar los valores de verdadero o falso, según cómo se definió e implementó el modelo.

La utilidad está acotada en el intervalo [-6, 32.25] considerando el peor y mejor caso posible teniendo en cuenta las restricciones de la implementación pero no su coherencia.

Nodo 1 del nivel 2:

Este nodo encuentra una solución (que al final resulta no ser la óptima) con las siguientes características.

X_INTRODUCED_0_ = array1d(1..9, [1.5, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 0.0, 0.0]);
instalaciones = array1d(1..3, [true, false, false]);
utilidad_total = 4.0;

Esta solución instala únicamente la planta del sitio 1 y envía desde ella 1.5 productos al cliente 1, 2.0 productos al cliente 2 y 1.0 productos al cliente 3.

Nodo 2 del nivel 2:

Este nodo considera las instancias con las siguientes características.

X_INTRODUCED_0_ = array1d(1..9, [[[0..1.5]], [[0..1.5]], 0.0, [[0..2]], [[0..2]], 0.0, [[0..1]], [[0..1]], 0.0]);
instalaciones = array1d(1..3, [false..true, true, false]);
utilidad_total = [[4..24.25]];

Los elementos de la primera, segunda y tercera fila de “X_INTRODUCED_0_” están acotados en los intervalos [0, 1.5], [0, 2] y [0, 1] por las mismas razones que se explican para el nodo raíz.

El segundo y tercer elemento de “instalaciones” son siempre verdadero y falso respectivamente mientras que el primero puede ser tanto verdadero como falso, indicando que se considerarán las instancias en las que siempre se realiza la instalación en el sitio 2 y nunca en el sitio 3.

La utilidad está acotada en el intervalo [4, 24.25], considerando que el peor caso ahora debe ser mayor o igual al resultado óptimo actual y el mejor caso debe ajustarse a la nueva característica, "siempre instalar en el sitio 2 y nunca en el sitio 3".

El nodo 1 del nivel 3:

Este nodo encuentra una solución óptima con las siguientes características.

X_INTRODUCED_0_ = array1d(1..9, [0.0, 1.5, 0.0, 0.0, 2.0, 0.0, 0.0, 1.0, 0.0]);
instalaciones = array1d(1..3, [false, true, false]);
utilidad_total = 18.25;

Esta solución instala únicamente la planta del sitio 2 y envía desde ella 1.5 productos al cliente 1, 2.0 productos al cliente 2 y 1.0 productos al cliente 3.

Nodos 2, 3 y 4 del nivel 3:

Estos nodos consideran las instancias con las siguientes características, pero se presentan con cuadrados rojos debido a que en todos los nodos la cota de utilidad_total solo permite valores muy cercanos al valor óptimo ya obtenido (18.25) por lo tanto se considera que la brecha de optimización es aceptable y no se sigue expandiendo nodos.

X_INTRODUCED_0_ = array1d(1..9, [[[0..0.345679]], [[1.15432..1.5]], 0.0, [[0..0.345679]], [[1.65432..2]], 0.0, [[0..0.345679]], [[0.654321..1]], 0.0]);
instalaciones = array1d(1..3, [true, true, false]);
utilidad_total = [[18.25..19.8056]];

X_INTRODUCED_0_ = array1d(1..9, [0.0, [[1.15432..1.5]], [[0..0.345679]], 0.0, [[1.65432..2]], [[0..0.345679]], 0.0, [[0.654321..1]], [[0..0.345679]]]);
instalaciones = array1d(1..3, [false, true, true]);
utilidad_total = [[18.25..19.8056]];

X_INTRODUCED_0_ = array1d(1..9, [[[0..0.302782]], [[1.19722..1.5]], [[0..0.302782]], [[0..0.302782]], [[1.69722..2]], [[0..0.302782]], [[0..0.302782]], [[0.697218..1]], [[0..0.302782]]]);
instalaciones = array1d(1..3, [true, true, true]);
utilidad_total = [[18.25..19.6125]];

5. Pruebas

En esta sección, se presentan y analizan los resultados obtenidos de la ejecución de la batería de pruebas entregada y de las pruebas construidas como grupo, utilizando el modelo y programa desarrollado para resolver el problema de optimización planteado.

El modelo ha sido implementado en el lenguaje de programación *MiniZinc*, y las pruebas se ejecutaron utilizando la interfaz gráfica de usuario (GUI) que hemos diseñado en *Python*, la cual facilita la carga de los archivos .txt de entrada, correspondientes a las pruebas propuestas por el profesor y monitor para el proyecto, la ejecución del modelo, y la visualización de las soluciones resultantes junto al tiempo de ejecución correspondientes a cada prueba.

Es importante indicar que para la ejecución de las pruebas, se utilizaron dos *solvers* diferentes, buscando encontrar los mejores resultados en cuestión de solución y tiempos de ejecución. de esta manera, utilizando la GUI diseñada, se utilizó el solver por defecto ***COIN-BC 2.10.10/1.17.8***; para la segunda opción de solver, se utilizó el solver ***HIGHS 1.6.0*** desde el IDE de *minizinc*, el cual nos entregó mejores resultados en cuestión de tiempo de ejecución.

Con estas pruebas se busca evaluar la eficacia y eficiencia del modelo en términos de su capacidad para proporcionar soluciones óptimas en un tiempo razonable. Para ello, se emplearon múltiples conjuntos de datos de entrada que varían en tamaño y complejidad, con el fin de simular diferentes escenarios posibles y medir el rendimiento del modelo bajo diversas condiciones.

5.1 Descripción y análisis de pruebas

Las pruebas fueron ejecutadas en dos equipos de cómputo con características técnicas diferentes, uno por solver utilizado, así:

5.1.1 Resultados Solver *COIN-BC*

Las pruebas a continuación fueron ejecutadas con el solver ***COIN-BC 2.10.10/1.17.8*** en la versión de ***MiniZinc 2.8.3.***
Información técnica del equipo de cómputo encargado de su procesamiento:

- Procesador: Intel(R) Core(TM) i5-10210U CPU @ 1.60GHz 2.11 GHz
- Memoria física instalada: 16,0 GB
- Sistema operativo: Windows 10 Pro

	Solver COIN-BC					
N° Prueba	Tiempo 1 (s)	Tiempo 2 (s)	Tiempo 3 (s)	Tiempo promedio (s)	Solucion	Es Optima?
1	0,301	0,272	0,278	0,2836666667	44.45	SI
2	0,205	0,222	0,210	0,212	421,7853	SI
3	UNSATISFIABLE					
4	0,222	0,206	0,216	0,215	154,6069	SI
5	0,214	0,194	0,187	0,198	116,2332	SI
6	0,216	0,207	0,211	0,211	191,9874	SI
7	0,227	0,232	0,213	0,224	220,2822	SI
8	0,196	0,193	0,215	0,201	389,0531	SI
9	0,533	0,357	0,333	0,408	842,2872	SI
10	0,333	0,318	0,332	0,328	529,5838	SI
11	0,458	0,451	0,454	0,454	3156,398	SI
12	0,523	0,575	0,564	0,554	3186,5358	SI
13	0,250	0,240	0,241	0,244	2963,9357	SI
14	0,278	0,279	0,257	0,271	2555,7563	SI
15	UNSATISFIABLE					
16	0,585	0,563	0,581	0,576	5689,3322	SI
17	0,654	0,628	0,651	0,644	4507,0857	SI
18	0,581	0,562	0,565	0,569	4183,792	SI
19	0,299	0,217	0,352	0,289	8143,03129	SI
20	2,123	2,002	1,926	2,017	14955,8316	SI
21	1,854	1,457	1,632	1,648	3292,07579	SI
22	UNSATISFIABLE					
23	21,903	16,982	19,994	19,626	6080,20149	SI
24	14,040	14,213	15,528	14,594	-601,00849	SI
25	UNSATISFIABLE					
26	131	98	103	110,667	12220,4745	SI
27	133	141	133	135,667	27073,1983	SI
28	179	183	177	179,667	23427,6752	SI
29	63	6	6	25,000	31281,554	SI
30	198	178	187	187,667	89765,2695	NO SE SABE

Enlace: https://docs.google.com/spreadsheets/d/10YXJPUPLFBcOO6IAmExkgv1Ol6wSHcNeU7BChKkGoc0/edit?usp=sharing

Los datos mostrados en la tabla anterior, corresponden a los resultados de la ejecución de las 30 pruebas de la batería entregada con el solver indicado.
Se tomaron los tiempos de ejecución, en escalas de segundos, arrojados por el programa a través de la etiqueta configurada en la GUI para tal objetivo. Se realizaron 3 ejecuciones para cada prueba, para obtener de esta manera un tiempo de ejecución promedio.

En cada una de las pruebas, el valor de la solución resultante corresponde adecuadamente a los valores de las soluciones entregadas junto con la batería de pruebas, es decir que el modelo está funcionando adecuadamente.

5.1.1 Resultados Solver *HIGHS*

Las pruebas a continuación fueron ejecutadas con el solver ***HIGHS 1.6.0.*** en la versión de ***MiniZinc 2.8.3.***

Información técnica del equipo de cómputo encargado de su procesamiento:

- Procesador: AMD Ryzen 7 5800U with Radeon Graphics @ 1.90 GHz
- Memoria física instalada: 16,0 GB
- Sistema operativo: Windows 11 Pro

	Solver HiGHS					
N° Prueba	Tiempo 1 (s)	Tiempo 2 (s)	Tiempo 3 (s)	Tiempo promedio (s)	Solucion	Es Optima?
1	0.38	0,33	0,32	0,325	44.45	SI
2	0,31	0,31	0,34	0,320	421,7853	SI
3	UNSATISFIABLE					
4	0,31	0,32	0,35	0,327	154,6069	SI
5	0,33	0,32	0,32	0,323	116,2332	SI
6	0,31	0,3	0,31	0,307	191,9874	SI
7	0,33	0,31	0,36	0,333	220,2822	SI
8	0,36	0,3	0,31	0,323	389,0531	SI
9	0,48	0,5	0,49	0,490	842,2872	SI
10	0,42	0,41	0,42	0,417	529,5838	SI
11	0,64	0,6	0,68	0,640	3156,3981	SI
12	0,77	0,75	0,74	0,753	3186,5358	SI
13	0,38	0,37	0,42	0,390	2963,9357	SI
14	0,37	0,36	0,4	0,377	2555,7563	SI
15	UNSATISFIABLE					
16	0,55	0,54	0,,54	0,545	5689,3322	SI
17	1,63	1,56	1,57	1,587	4507,0857	SI
18	2,64	2,71	2,61	2,653	4183,792	SI
19	2,7	2,99	2,76	2,817	8143,0313	SI
20	5,84	5,78	5,67	5,763	14955,8316	SI
21	1,45	1,39	1,43	1,423	3292,07579	SI
22	UNSATISFIABLE					
23	19,46	19,56	19,46	19,493	6080,2015	SI
24	8,15	8,24	8,53	8,307	-601,0084	SI
25	UNSATISFIABLE					
26	155,86	95.81	95,63	125,745	12220,4745	SI
27	23,55	23,47	23,28	23,433	27073,19839	SI
28	7,94	7,82	7,88	7,880	23427,6753	SI
29	6,55	6,57	6,63	6,583	31281,554	SI
30	12,88	12,67	12,81	12,787	89765,2696	NO SE SABE

Enlace: https://docs.google.com/spreadsheets/d/10YXJPUPLFBcOO6IAmExkgv1Ol6wSHcNeU7BChKkGoc0/edit?usp=sharing

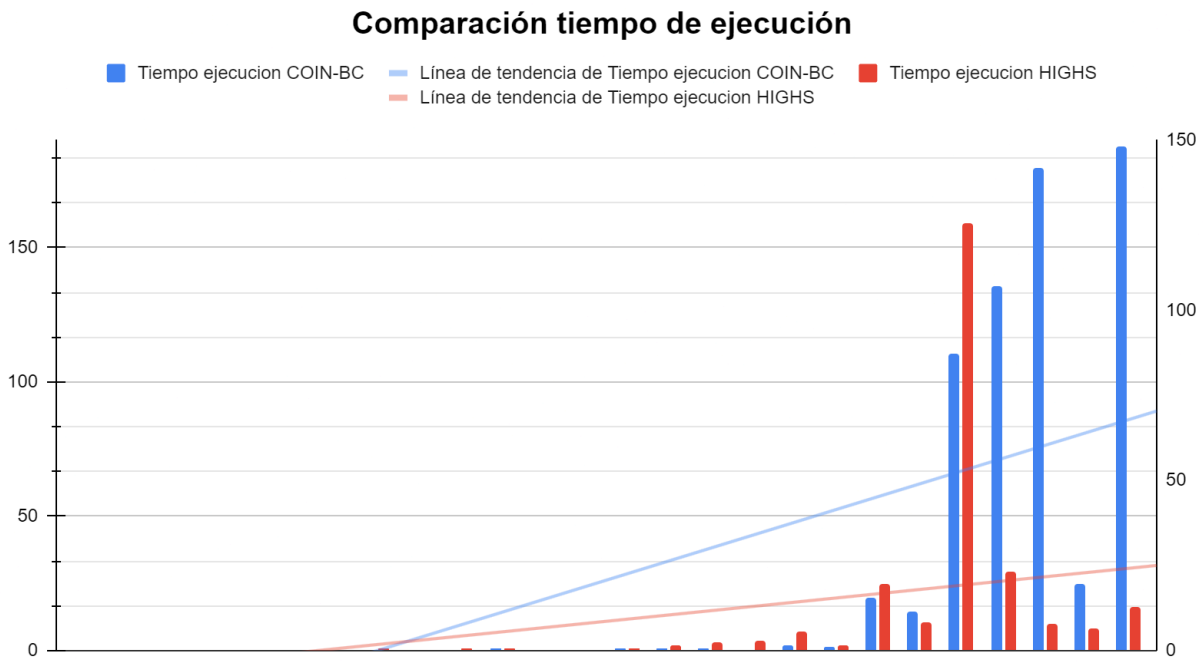
Los datos mostrados en la tabla anterior, corresponden a los resultados de la ejecución de las 30 pruebas de la batería entregada con el solver indicado.
Se tomaron los tiempos de ejecución, en escalas de segundos, arrojados por el programa a través de la etiqueta configurada en la GUI para tal objetivo. Se realizaron 3 ejecuciones para cada prueba, para obtener de esta manera un tiempo de ejecución promedio.

En cada una de las pruebas, el valor de la solución resultante corresponde adecuadamente a los valores de las soluciones entregadas junto con la batería de pruebas, es decir que el modelo está funcionando adecuadamente

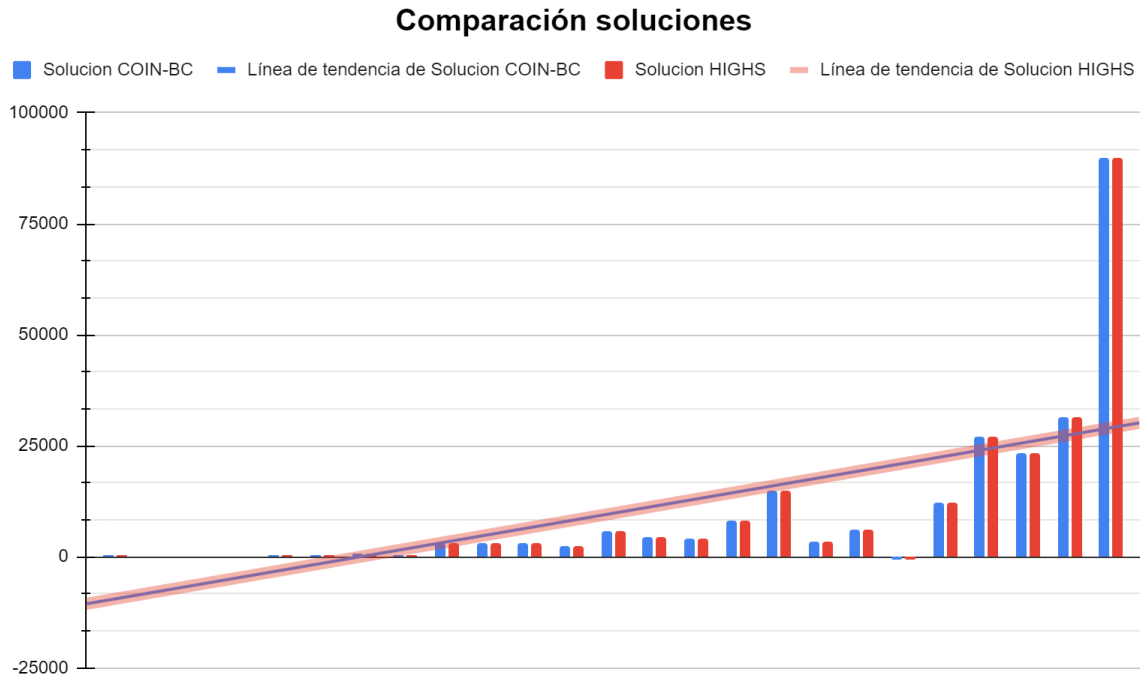
5.1.1 Análisis de resultados

A continuación se presentan dos gráficos de barras, cada uno con dos variables involucradas, para representar y comparar los resultados de tiempo de ejecución y

soluciones, respectivamente, de las pruebas, usando los dos solvers en cuestión (***COIN-BC*** y ***HIGHS)***



El anterior gráfico de barras muestra la comparación de los resultados de tiempos de ejecución de la batería de pruebas utilizando el solver ***COIN-BC*** (azul) y el solver ***HIGHS*** (rojo). En el eje X se encuentran cada una de las pruebas (excluyendo aquellas que resultaron insatisfactorias: prueba 3, prueba 15, prueba 22, prueba 25), para cada unos de los dos solvers. En el eje Y, se representan los tiempos de ejecución, la columna izquierda para ***COIN-BC*** y la columna derecha para ***HIGHS.***



El anterior gráfico de barras muestra la comparación de las soluciones de la batería de pruebas utilizando el solver ***COIN-BC*** (azul) y el solver ***HIGHS*** (rojo).
En el eje X se encuentran cada una de las pruebas (excluyendo aquellas que resultaron insatisfactorias: prueba 3, prueba 15, prueba 22, prueba 25), para cada unos de los dos solvers. En el eje Y, se representan las soluciones obtenidas, la columna izquierda para ***COIN-BC*** y la columna derecha para ***HIGHS.***

Basándonos en los resultados obtenidos para la batería de pruebas, podemos afirmar que el modelo construido es capaz de alcanzar la solución óptima esperada para cada instancia del problema que se le pase. Sin embargo, esto no necesariamente significa que el programa sea eficiente. En algunas de las pruebas, los resultados fueron insatisfactorios, esto debido a que no se encontró solución, en otras pruebas, los tiempos de ejecución se prolongaron un poco más.

Parámetros importantes incluyen la capacidad de producción, la demanda de energía y la disponibilidad de recursos. Sin embargo, todos estos parámetros están interrelacionados, por lo que cada uno tiene su importancia. Alteraciones significativas en un par de ellos pueden producir resultados distintos.
Es este análisis que realizamos, la razón de la importancia de diseñar un modelado cuidadoso y detallado de todas las restricciones y parámetros involucrados en un problema.

5.1.3 Presentación y análisis de Instancias creadas

A continuación se muestran los resultados de la ejecución de las instancias que hemos construido, junto con su tiempo de ejecución en segundos. Estas 5 instancias se encuentran dentro del repositorio en la carpeta ***MiInstancias,*** en formato **.txt**, que luego serán transformadas, por la interfaz, en formato **.dzn.**

	Solver COIN-BC					
Entrada	Tiempo 1 (s)	Tiempo 2 (s)	Tiempo 3 (s)	Tiempo promedio (s)	Solucion	Es Optima?
instancia1	0,2	0,197	0,181	0,193	31620	SI
instancia2	0,218	0,186	0,187	0,197	16000	SI
instancia3	0,23	0,18	0,196	0,202	55065	SI
instancia4	UNSATISFIABLE					
instancia5	0,27	0,236	0,254	0,253	533150	SI

Se utilizó el solver ***HIGHS 1.6.0*** para ejecutar estas pruebas, esto debido a que en los resultados de las pruebas anteriores, este solver arrojó mejores tiempos de ejecución.

La creación de las instancias solicitadas se llevó a cabo teniendo en cuenta las características de los parámetros que componen el problema.
Se centró la atención principalmente en el número de clientes ***(n_clientes)*** y en el número de sitios ***(n_sitios)***, buscando mantener una proporcionalidad relativa entre ambos elementos. Tomamos esta decisión reconociendo que el cumplimiento de las restricciones y la satisfacción de la demanda dependen críticamente de la relación entre estos dos factores. También se consideraron aspectos clave como los **costos fijos de los sitios**, las

capacidades máximas de los sitios, **las demandas de los clientes**, y **los beneficios por unidad.** Además, se tomaron también en cuenta los límites de capacidad, las restricciones de costos, y las demandas específicas de cada cliente.

Con esto en mente, exploramos diferentes escenarios, ajustando los parámetros para generar instancias que representarán situaciones realistas y con diferentes características. Se experimentó con costos fijos y variables, Por ejemplo:

En la **Instancia 3**, diseñamos un caso donde había menos recursos disponibles que la demanda de los clientes, desafiando así la capacidad del sistema para manejar la escasez de recursos.
En contraste, en la **Instancia 5**, exploramos un escenario con una mayor disponibilidad de recursos, y una demanda creciente y variada, para poner a prueba la capacidad del sistema para adaptarse a situaciones de alta carga.
Por otro lado, **la Instancia 4** fue encontrada como insatisfacible, lo que significa que no hay una solución factible que cumpla con todas las restricciones del problema simultáneamente. Esto puede ser el resultado de una combinación de costos irrealistas, negativos o una demanda excesiva en relación con la capacidad de los sitios.

Para cada instancia,se establecieron límites proporcionales para las capacidades y las demandas indicadas. En el caso de los beneficios por unidad, se asignaron valores aleatorios que cumplían con las restricciones del problema, generando así varias soluciones posibles, identificando una de ellas como la óptima.

Nos dimos cuenta de que el diseño estratégico de instancias implicaba la manipulación consciente de parámetros para explorar diversos escenarios y comprender cómo estos afectan la solución del problema. La sensibilidad del modelo a diferentes configuraciones de entrada se destacó mediante la creación de instancias que reflejaban condiciones variadas y específicas.
Estas variaciones permitieron evaluar la robustez del modelo y su capacidad para adaptarse a diferentes contextos operativos, proporcionando así una visión integral sobre el desempeño y la eficiencia del sistema en situaciones reales y simuladas.

6. Conclusiones

Como respuesta a lo solicitado para este proyecto, hemos desarrollado satisfactoriamente un modelo y un programa que incluye una interfaz gráfica, el cual da solución al problema de Ubicación de Instalaciones con Asignación de Clientes (PUICA) de manera adecuada e interactiva.
En base a los resultados de las pruebas obtenidos y presentados previamente, podemos asegurar que nuestra representa una solución para la resolución de este problema, proporcionando resultados precisos y consistentes.}
El uso de diferentes solvers para la prueba de nuestro modelo, ha resaltado la importancia de seleccionar el solver adecuado en función de las características específicas del problema.

Desempeño de Solvers:

- **HiGHs Solver:** Durante las pruebas realizadas, HiGHs demostró ser eficiente en la resolución del problema PUICA. Este solver proporciona soluciones rápidas y consistentes, especialmente en instancias de tamaño mediano a grande. La capacidad de HiGHs para manejar grandes conjuntos de datos lo hace ideal para aplicaciones en escenarios industriales.
- **COIN-BC Solver:** Aunque también es efectivo, COIN-BC mostró un rendimiento ligeramente inferior en comparación con HiGHs en términos de tiempo de ejecución. Sin embargo, en algunas instancias, COIN-BC fue capaz de encontrar soluciones óptimas que HiGHs no pudo identificar. Esto sugiere que el uso combinado de ambos solvers puede ser beneficioso para asegurar una mayor calidad en las soluciones obtenidas.

La implementación de una GUI intuitiva facilita la interacción con el sistema, permitiendo a los usuarios cargar datos de entrada, ejecutar el solver y visualizar los resultados de manera eficiente.
Además, la solución desarrollada es escalable y puede adaptarse para manejar diferentes tamaños de instancias del PUICA. La flexibilidad del diseño permite integrar fácilmente nuevos solvers o ajustar los parámetros existentes, proporcionando una herramienta robusta para la resolución de problemas de optimización.

En conclusión, la programación lineal nos brinda herramientas extremadamente poderosas en la resolución de problemas de optimización discreta, donde premia el análisis intuitivo de las variables y parámetros de decisión.

7. Video Explicativo de la Interfaz Gráfica y Otros Elementos

Enlace al video explicativo en la plataforma YouTube: https://youtu.be/MZjFonTIkQA
Enlace al repositorio: https://github.com/AlexanderValencia21/Proyecto2-ADA