Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Herramienta para resolver el Problema de la Separación de Vértices Capacitado

Tool for solving the Capacitated Vertex Separation Problem

Sergio Tabares Hernández

La Laguna, 13 de septiembre de 2022

D. **Juan José Salazar González**, con N.I.F. 43.356.435-D profesor Titular de Universidad adscrito al Departamento de Matemáticas, Estadística e Investigación Operativa de la Universidad de La Laguna, como tutor

C E R T I F I C A

Que la presente memoria titulada:

"Herramienta para resolver el Problema de la Separación de Vértices Capacitado"

ha sido realizada bajo su dirección por D. **Sergio Tabares Hernández**, con N.I.F. 78.729.840-N.

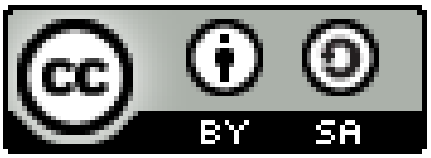
Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firma la presente en La Laguna a 13 de septiembre de 2022

Agradecimientos

A Juan José Salazar González por aportar la idea para la realización de este trabajo y por todo el asesoramiento ofrecido durante el transcurso del mismo.

Y a todos mis familiares y amigos por estar siempre presentes tanto en los buenos como en los malos momentos.

Licencia



Resumen

La optimización de recursos es una tarea de vital importancia ya que nos ayuda a saber cual es la mejor forma de llevar a cabo otras actividades para así poder minimizar los costes y maximizar sus beneficios.

El propósito de este proyecto ha sido el desarrollo de una herramienta que sea capaz de buscar la solución a un problema de optimización, en este caso el Problema de la Separación de Vértices Capacitado, y que cualquier persona pueda utilizarlo.

Además, como objetivo adicional, también se ha llevado a cabo un análisis comparativo de diferentes formulaciones implementadas para resolver dicho problema.

Palabras clave: Herramienta, Optimización, Problema de la Separación de Vértices Capacitado

Abstract

Resource optimization is a task of vital importance because it helps us to know which is the best way to carry out other activities in order to minimize costs and maximize their benefits.

The purpose of this project has been the development of a tool that is capable of finding the solution to an optimization problem, in this case the Capacitated Vertex Separation Problem, and that anyone can use it.

Furthermore, as an additional goal, a comparative analysis of different implemented formulations to solve this problem has also been carried out.

Keywords: Tool, Optimization, Capacitated Vertex Separation Problem

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

1.1. Motivación

La **optimización de recursos** ha estado a nuestro alrededor desde que vivíamos en cavernas, es un instinto básico de supervivencia ya que nos ayuda a saber cual es la mejor forma de hacer algo para así obtener los mejores resultados, la mejor eficiencia y/o la mejor eficacia [1].

Esta sección de las **matemáticas**, englobada dentro de la **ingeniería logística**, puede ser aplicada a multitud de escenarios de nuestra vida cotidiana. Desde la cantidad de comida que preparar, para que no sobre en exceso pero tampoco falte; hasta como meter nuestro equipaje en la maleta cuando nos vamos de viaje, para que entre toda la ropa posible que nos pueda hacer falta; o incluso la elección de la mejor ruta de recogida de tus amigos para ir a la playa, para evitar malgastar la tan preciada gasolina dando vueltas innecesarias por la ciudad; aunque su aplicación suele darse principalmente en empresas de gran tamaño para **minimizar costes y maximizar beneficios**.

Es de estos problemas de donde surge la necesidad de buscar un método que sea capaz de dar con la mejor forma de afrontarlos, de optimizar los recursos disponibles sujetos a una serie de restricciones. Este método se conoce como **Optimización Matemática** (o **Programación Matemática**) [2] ya que el problema y sus restricciones se modelan con notación matemática para poder resolverlos como si de una ecuación de segundo grado se tratase, eso si, de forma algo más compleja; dando lugar a una formulación matemática del problema.

1.2. Objetivos

El objetivo principal de este trabajo ha sido **desarrollar una herramienta** que pueda ser utilizada para la resolución de un problema de optimización y que además permita realizar un **análisis comparativo** entre las implementaciones de diversas formulaciones.

Para ello se ha seleccionado el **Problema de la Separación de Vértices Capacitado** (en adelante **CVSP**, de su nombre en inglés: **Capacitated Vertex Separator Problem**) [3]. Este problema de optimización busca eliminar el menor número de nodos de un grafo para que este quede separado en grupos de nodos con un tamaño limitado y que los grupos resultantes queden totalmente separados unos de otros.

Un claro ejemplo de aplicación del problema elegido sería la optimización del orden de vacunación de la población ante una nueva pandemia como la del SARS-CoV-2, que hemos sufrido recientemente, ya que si se prioriza la vacunación en las personas específicas puede minimizarse el contagio de la enfermedad al separar a la población no vacunada en una serie de grupos aislados; aunque para poder llevarlo a cabo habría que tener disponible un grafo con las relaciones interpersonales de toda la población mundial, tarea que complica ligeramente el proceso para llevarlo a cabo.

Capítulo 2

Tecnologías utilizadas

2.1. Hardware

Para el desarrollo del proyecto se utilizó un ordenador de sobremesa equipado con un procesador AMD Ryzen 5 3500X (6 hilos) @ 3.600 GHz y 16 GB de memoria RAM aunque para ejecutar el programa sobre grafos de gran tamaño y obtener los resultados en menor tiempo se solicitó una máquina virtual en los servidores de la Universidad de La Laguna a la cual le fué asignado un procesador Intel Xeon E312xx (Sandy Bridge) (20 hilos) @ 2.095 GHz y 20 GB de memoria RAM.

2.2. Software

El objetivo del trabajo se ha alcanzado haciendo uso de **Python** [4] como lenguaje de programación junto con las siguientes librerías externas:

- **PyQt5** [5] para la implementación de la interfaz gráfica de usuario.
- **NetworkX** [6] y **Matplotlib** [7] para manipular y representar los grafos.
- **Google OR-Tools** [8] y **Gurobi Optimization** [9] para transcribir las formulaciones matemáticas con las que se expresan los diferentes métodos para resolver el problema de optimización matemática.

El código completo del programa junto con los grafos utilizados, sus soluciones, algunas imágenes y demás han sido alojados en un repositorio publico en **GitHub** [10] [11].

Además, para la redacción de esta memoria se utilizó **Overleaf** [12], herramienta online de elaboración de documentos en formato LaTeX [13].

Capítulo 3

Fases del desarrollo

En primer lugar se realizó un **estudio del estado actual del problema**, aunque este no fué muy extenso ya la mayor parte de la información sobre este se encuentra reunida en el artículo "Casting Light on the Hidden Bilevel Combinatorial Structure of the Capacitated Vertex Separator Problem", por Fabio Furini, Ivana Ljubić, Enrico Malaguti y Paolo Paronuzzi (en adelante "el artículo de referencia") [3]. En este artículo introducen el problema y aportan varias formulaciones clásicas desde diversas aproximaciones para posteriormente expandir la visión del problema desde un acercamiento binivel basado en el modelo de liderazgo de Stackelberg [14].

Luego se realizó una primera versión del programa con el cual poder **cargar y visualizar los grafos** que se dan como entrada del problema, sobre los que se ejecutarán los algoritmos de resolución del CVSP, y también se le asignó un formato a la solución para poder realizar pruebas de cómo se visualizará la solución sobre el grafo una vez calculada.

A continuación se implementó una **primera formulación** usando la librería OR-Tools para resolver el problema de optimización y también el método para visualizar la solución obtenida por el algoritmo sobre el grafo que se le ha dado como entrada.

Tras ello se implementaron **varias formulaciones alternativas** con OR-Tools pero, como algunas de ellas requerían de tiempos de ejecución demasiado largos, se implementaron también usando la librería Gurobi. Esta última permite implementar las formulaciones de forma que las restricciones sean generadas dinámicamente, lo cual permite que se generen únicamente las restricciones necesarias para poder llegar a la solución dando así el resultado en un tiempo de cómputo considerablemente inferior y haciendo uso de una menor cantidad de memoria.

Cabe destacar que en un primer momento se planteó la implementación de las formulaciones binivel nombradas en el artículo de referencia como un

posible objetivo a alcanzar para este trabajo pero tras implementar varias de las formulaciones clásicas y ver que daban resultados bastante prometedores además de la gran complejidad de las formulaciones binivel, la implementación de estas últimas fue descartada.

De forma simultanea se añadió la posibilidad de ejecutar el programa con distintos argumentos para poder configurar el resultado de la ejecución del programa, se creó una interfaz gráfica para facilitar su uso y se implementaron soluciones para que sea igual de útil en ordenadores con o sin entorno gráfico.

Una vez completada la herramienta se generó un conjunto de grafos con diferentes formas, tamaños y densidades con los que se llevó a cabo un **análisis de la herramienta** desarrollada utilizando cada formulación implementada y aportándole distintos valores a los parámetros de entrada del problema. De estas ejecuciones se compararon los tiempos de cómputo y la calidad de las soluciones obtenidas para realizar dicho análisis y obtener los resultados y conclusiones que se exponen en este documento.

Por último se llevó a cabo la **redacción de esta memoria** en la que se plasmó el trabajo realizado y se le dio forma a los resultados obtenidos del análisis junto con las conclusiones generadas en consecuencia.

Capítulo 4

Características de la herramienta

4.1. Formulaciones implementadas

Para poder probar la herramienta desarrollada se han implementado diversas formulaciones, algunas de las cuales fueron extraídas del artículo [3] y otras son derivaciones de las anteriores buscando mejorar la velocidad de obtención de la solución o la calidad de esta sin comprometer la velocidad de cómputo.

Para que sirva como guía para saber el índice de cada una y facilitar su uso posterior, comentaremos cada una en el mismo orden en el cual se encuentran en la herramienta.

En estas formulaciones la variable V hace referencia al conjunto de vértices del grafo y E al conjunto de aristas. Además, el CVSP toma como entrada, además del grafo, dos valores: k, que indica el número mínimo de grupos que deben quedar tras separar el grafo y que da lugar a K como el rango de índices entre 1 y k; y b, que indica el número máximo de nodos que pueden quedar en cada grupo resultante.

4.1.1. Librería Google OR-Tools

- Formulación 1

$$\text{máx} \sum_{i \in K} \sum_{v \in V} \xi_v^i \tag{1a}$$

$$\sum_{i \in K} \xi_v^i \leq 1 \qquad v \in V \tag{1b}$$

$$\xi_w^i + \sum_{j \in K \ i} \xi_v^j \leq 1 \qquad i \in K, wv \in E \tag{1c}$$

$$\sum_{v \in V} \xi_v^i \leq b \qquad i \in K \tag{1d}$$

$$\xi_v^i \in \{0, 1\} \qquad i \in K, v \in V \tag{1e}$$

Esta formulación tiene como función objetivo maximizar el número de nodos que se asignan a cada uno de los grupos resultantes (1a) teniendo como restricciones que un nodo no puede estar en más de un grupo a la vez (1b), que los nodos que están unidos por una arista no pueden estar en grupos diferentes (1c), que un grupo no puede estar formado por más de b nodos (1d) y que a cada combinación de nodo y grupo solo se le pueden asignar valores binarios: 1 si el nodo v pertenece al grupo i y 0 en caso contrario (1e).

- Formulación 2

$$\begin{aligned}
\text{máx} \sum_{i \in K} \sum_{v \in V} \xi_v^i & & & (1a) \\
\sum_{i \in K} \psi_Q^i \leq 1 & & Q \in \mathcal{Q} & (2a) \\
\xi_v^i + \psi_Q^i \leq 0 & & i \in K, Q \in \mathcal{Q}, v \in Q & (2b) \\
\sum_{v \in V} \xi_v^i \leq b & & i \in K & (1d) \\
\xi_v^i \in \{0,1\} & & i \in K, v \in V & (1e) \\
\psi_Q^i \in \{0,1\} & & i \in K, Q \in \mathcal{Q} & (2c)
\end{aligned}$$

Esta formulación es similar a la anterior, la diferencia es que se ha reformulado definiendo un conjunto $\mathcal{Q}$ con los cliques del grafo, lo cual da lugar a la sustitución de las restricciones (1b) y (1c). Las nuevas restricciones hacen que cada clique solo pueda pertenecer a un mismo grupo de nodos resultante de la separación (2a), que un nodo solo puede ser asignado a un grupo resultante si y solo si pertenece al clique seleccionado para dicho grupo (2b) y que a cada combinación de clique y grupo solo se le pueden asignar valores binarios: 1 si algún nodo del clique Q pertenece al grupo i y 0 en caso contrario (2c).

- Formulación 3

$$\begin{aligned}
\text{mín} \sum_{v \in V} x_v & & & (3a) \\
\sum_{v \in W} x_v \geq 1 & & W \subseteq V : \sigma(W) > k & (3b) \\
x_v \in \{0,1\} & & v \in V & (3c)
\end{aligned}$$

Para esta formulación se ha tenido en cuenta una aproximación diferente, aunque fundamentalmente tiene el mismo objetivo. Esta vez se tiene como función objetivo minimizar el número de nodos que se asignan al conjunto de la solución (3a) teniendo como restricciones que cualquier conjunto de nodos que separe al grafo en grupos con componentes conexas con más de b nodos debe ser descartado (3b) y que a cada nodo solo se le pueden asignar valores binarios: 1 si el nodo v pertenece al grupo de la solución y 0 en caso contrario (3c).

En esta formulación aparece la variable $\sigma(W)$, esto hace referencia al resultado del *bin packing problem* [15] que nos da el número de contenedores de tamaño b necesarios para empaquetar las componentes conexas del subgrafo $G[W]$. En el caso de que una componente conexa esté compuesta por un mayor número de nodos que b, el empaquetamiento no es factible por lo que $\sigma(W) = \infty$.

- Formulación 4

$$\text{mín} \sum_{v \in V} x_v \tag{3a}$$

$$\sum_{v \in C} x_v \geq 1 \qquad C \subseteq W : C \textit{ connected and } |V(C)| = b + 1 \tag{4}$$

$$x_v \in \{0, 1\} \qquad v \in V \tag{3c}$$

Esta vez, la formulación 4 es un caso especial de la número 3. La diferencia reside en que la restricción (3b) es sustituida por la (4) que hace que para una componente conexa C perteneciente a un subgrafo arbitrario $G[W]$, si esta componente tiene un mayor número de nodos que b, al menos uno de esos nodos debe pertenecer al conjunto de la solución.

4.1.2. Librería Gurobi Optimization

- Formulación 1

$$\text{máx} \sum_{i \in K} \sum_{v \in V} \xi_v^i \tag{1a}$$

$$\sum_{i \in K} \xi_v^i \leq 1 \qquad v \in V \tag{1b}$$

$$\xi_w^i + \sum_{j \in K \setminus \{i\}} \xi_v^j \leq 1 \qquad i \in K, wv \in E \tag{1c}$$

$$\sum_{v \in V} \xi_v^i \leq b \qquad i \in K \tag{1d}$$

$$\xi_v^i \in \{0, 1\} \qquad i \in K, v \in V \tag{1e}$$

Esta formulación es idéntica a la número 1 implementada con OR-Tools.

- Formulación 1 - Alternativa b

$$\text{máx} \sum_{i \in K} \sum_{v \in V} \xi_v^i \tag{1a}$$

$$\sum_{i \in K} \xi_v^i \leq 1 \qquad v \in V \tag{1b}$$

$$\xi_w^i + \xi_v^j \leq 1 \qquad i \in K, j \in K \setminus \{i\}, wv \in E \tag{1c alt b}$$

$$\sum_{v \in V} \xi_v^i \leq b \qquad i \in K \tag{1d}$$

$$\xi_v^i \in \{0, 1\} \qquad i \in K, v \in V \tag{1e}$$

Para esta primera alternativa se ha modificado la restricción (1c) extrayendo el sumatorio para así tener las combinaciones en restricciones separadas y a su vez seguir cumpliendo con el mismo objetivo.

- Formulación 1 - Alternativa c

$$\text{máx} \sum_{i \in K} \sum_{v \in V} \xi_v^i \tag{1a}$$

$$\sum_{i \in K} \xi_v^i \leq 1 \qquad v \in V \tag{1b}$$

$$\sum_{i \in L} \xi_w^i + \sum_{j \notin L} \xi_v^j \leq 1 \qquad L \subseteq K, wv \in E \tag{1c alt c}$$

$$\sum_{v \in V} \xi_v^i \leq b \qquad i \in K \tag{1d}$$

$$\xi_v^i \in \{0, 1\} \qquad i \in K, v \in V \tag{1e}$$

Para esta segunda alternativa se ha modificado nuevamente la restricción (1c) pero esta vez se ha introducido un nuevo sumatorio para unificar aún más combinaciones en una misma restricción pero sin dejar de cumplir el mismo objetivo.

- Formulación 2

$$\begin{aligned}
\text{máx} \sum_{i \in K} \sum_{v \in V} \xi_v^i & & & (1a) \\
\sum_{i \in K} \psi_Q^i \leq 1 & & Q \in \mathcal{Q} & (2a) \\
\xi_v^i + \psi_Q^i \leq 1 & & i \in K, Q \in \mathcal{Q}, v \in Q & (2b) \\
\sum_{v \in V} \xi_v^i \leq b & & i \in K & (1d) \\
\xi_v^i \in \{0,1\} & & i \in K, v \in V & (1e) \\
\psi_Q^i \in \{0,1\} & & i \in K, Q \in \mathcal{Q} & (1f)
\end{aligned}$$

Esta formulación es idéntica a la número 2 implementada con OR-Tools.

- Formulación 3

$$\begin{aligned}
\text{mín} \sum_{v \in V} x_v & & & (3a) \\
\sum_{v \in W} x_v \geq 1 & & W \subseteq V : \sigma(W) > k & (3b) \\
x_v \in \{0,1\} & & v \in V & (3c)
\end{aligned}$$

Esta formulación es idéntica a la número 3 implementada con OR-Tools.

- Formulación 3 - Método *lazy*

  En este caso, se ha implementado la formulación del punto anterior pero de forma que la familia de restricciones 3b sean generadas siguiendo el método de la Generación Dinámica de Restricciones para así poder obtener la solución al problema en un menor tiempo de cómputo y utilizando menor cantidad de memoria. Esto es debido a que, al emplear dicho método, solo se van generando restricciones que acoten la solución del problema cuando realmente son necesarias.

- Formulación 4

$$\text{mín} \sum_{v \in V} x_v \tag{3a}$$

$$\sum_{v \in C} x_v \geq 1 \qquad C \subseteq W : C \textit{ connected and } |V(C)| = b + 1 \tag{4}$$

$$x_v \in \{0, 1\} \qquad v \in V \tag{3c}$$

Esta formulación es idéntica a la número 4 implementada con OR-Tools.

- Formulación 4 - Método *lazy*

  Para esta formulación se ha realizado lo mismo que para la formulación 3 *lazy*, se implementó la formulación 4 pero haciendo uso del método de la Generación Dinámica de Restricciones para agilizar el proceso de búsqueda de la solución.

4.2. Uso de la herramienta

Para utilizar la herramienta desarrollada primeramente se debe clonar el repositorio de la siguiente manera:

```
$ git clone https://github.com/alu0101124896/TFG-Capacitated-
  Vertex-Separator-Problem.git cvsp_package
```

Nota: En el comando anterior, al clonar el repositorio también se le cambia el nombre a "cvsp_package", esto se hace para facilitar el uso de la herramienta en los pasos futuros.

Luego se deben instalar las dependencias del proyecto para su correcto funcionamiento:

```
$ cd ./cvsp_package
$ pip install -r ./requirements.txt
```

Una vez realizados los pasos anteriores se dispone de tres métodos diferentes para su utilización:

4.2.1. Interfaz Gráfica de Usuario

La primera opción disponible, y la más sencilla de utilizar, se trata de una Interfaz Gráfica de Usuario (en adelante GUI, de su nombre en inglés: Graphical User Interface).

Para ello, en primer lugar se debe ejecutar el programa "gui_main.py", tras lo que debería aparecer una ventana como la mostrada en la figura 4.1.

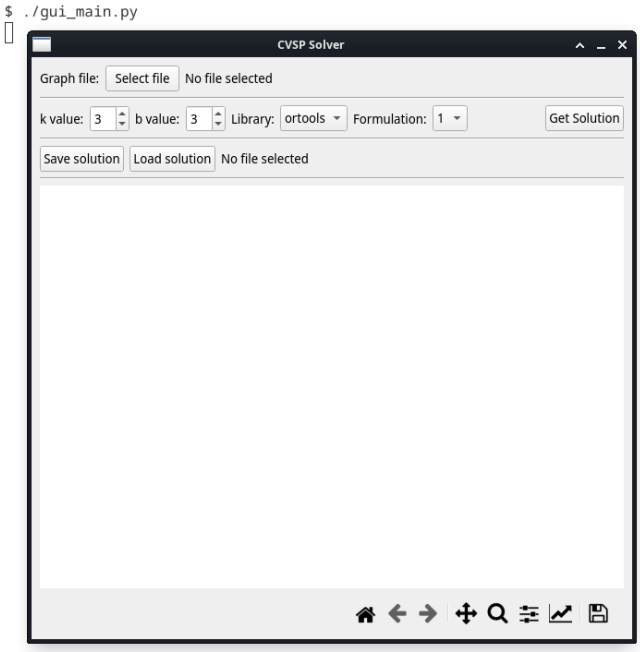


Figura 4.1: Ventana principal de la GUI

Una vez dentro de la aplicación se deberá cargar el grafo sobre el cual se desea trabajar. Para ello, se abrirá una nueva ventana del sistema de archivos del ordenador en el cual se podrá elegir el grafo deseado, tal y como se muestra en la figura 4.2.

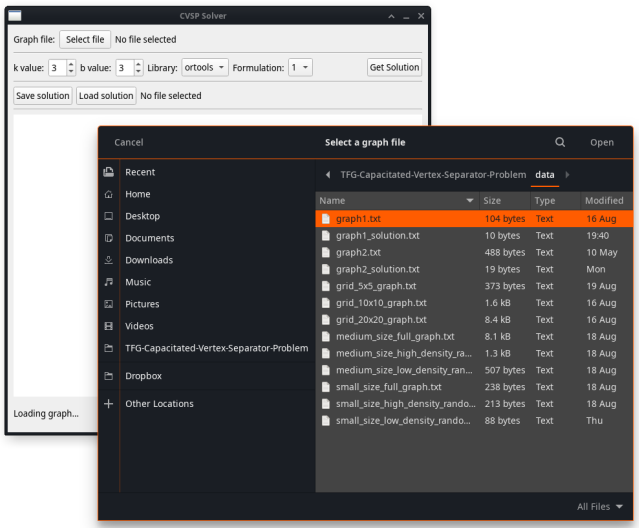


Figura 4.2: Ejemplo de selección de un grafo

A continuación se podrán introducir los parámetros deseados para obtener una nueva solución, como puede apreciarse en la figura 4.3.

Una vez calculada la solución, esta podrá ser almacenada en un archivo para su posterior consulta o utilización. Para ello también se abrirá una nueva ventana

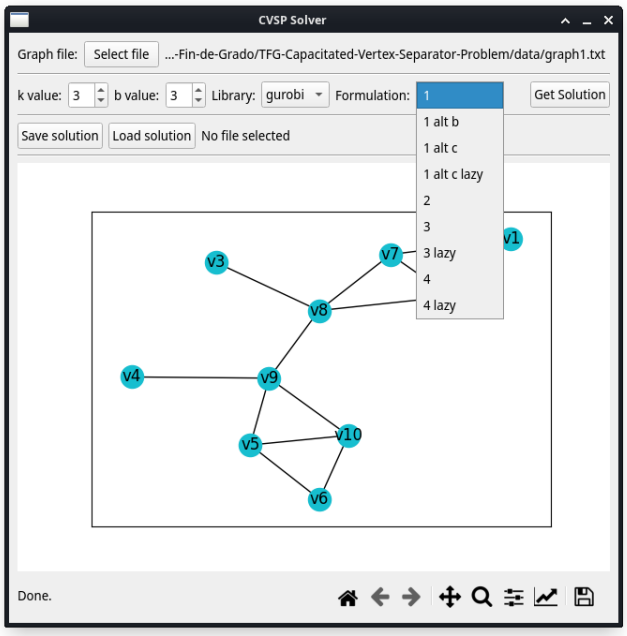


Figura 4.3: Ejemplo de configuración de los parámetros

del sistema de archivos pero esta vez para elegir la ubicación y el nombre del archivo con la solución.

Además es posible cargar una solución previamente computada y que haya sido almacenada en un archivo, aunque primero debe estar cargado el grafo del cual se ha obtenido la solución; en caso contrario puede dar error.

En la figura 4.4 se puede ver un ejemplo del resultado de obtener una solución.

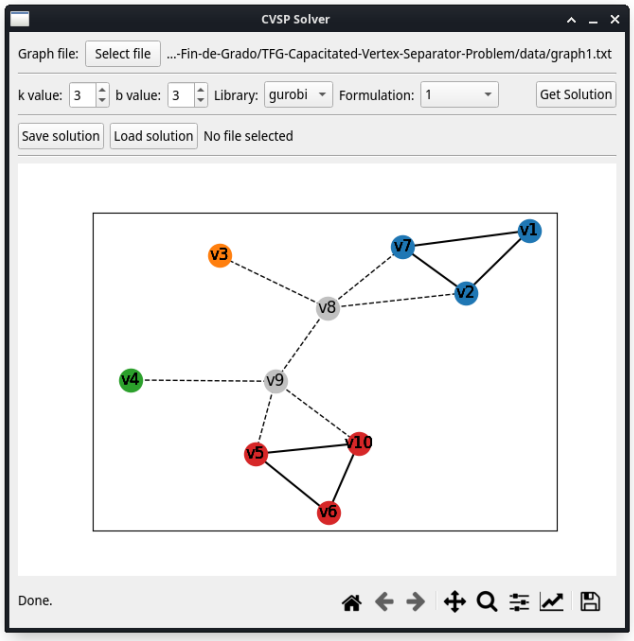


Figura 4.4: Ejemplo del estado del programa tras obtener una solución

4.2.2. Interfaz de Línea de Comandos

Otra posibilidad se trata del uso mediante la Interfaz de Línea de Comandos (en adelante CLI, de su nombre en inglés: Command Line Interface).

Esta opción está disponible ejecutando el programa "cli_main.py" y además se le puede aportar cualquiera de los siguientes parámetros para modificar su comportamiento:

- `[--input-file | -i] INPUT_FILE`:
  Importar la definición del grafo desde `INPUT_FILE`.

- `[--output-file | -o] OUTPUT_FILE`:
  Exportar la solución obtenida a `OUTPUT_FILE`.

- `[--library-name | -l] LIBRARY_NAME`:
  Seleccionar una librería de optimización para utilizar:
  - Para la librería Google OR-Tools: 'ortools'
  - Para la librería Gurobi Optimization: 'gurobi'

- `[--formulation-index | -f] FORMULATION_INDEX`:
  Seleccionar una formulación del problema a utilizar:
  - Para la librería Google OR-Tools: [1-4]
  - Para la librería Gurobi Optimization: [1-8]

- `[--k-value | -k] K_VALUE`:
  Número mínimo de grupos restantes.

- `[--b-value | -b] B_VALUE`:
  Número máximo de nodos en los grupos restantes.

- `[--no-gui | -g]`:
  No mostrar la solución gráficamente.

- `[--quiet | -q]`:

  Eliminar toda interacción normal a través de la terminal (Exceptuando un mensaje de la librería Gurobi el cual indica que se está utilizando una licencia académica para uso no comercial y su fecha de vencimiento).

En el caso de que el programa sea ejecutado sin especificar ninguno de los parámetros anteriores, se le pedirán de forma interactiva los datos que sean imprescindibles para la ejecución, excepto si se activa la opción "quiet" que forzará a que el programa utilice valores predeterminados para aquellos parámetros a los que no se les haya asignado uno.

En las figuras 4.5 y 4.6 se muestran ejemplos de uso de este método.

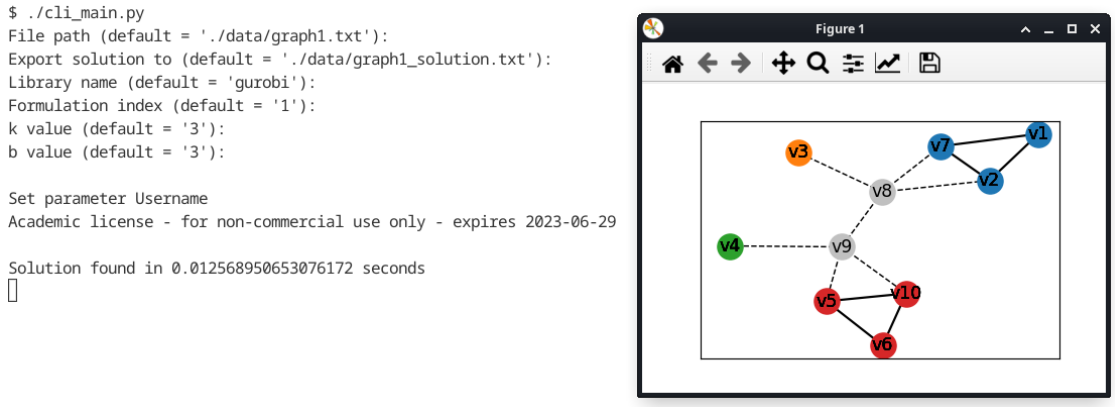


Figura 4.5: Ejemplo de uso de forma interactiva

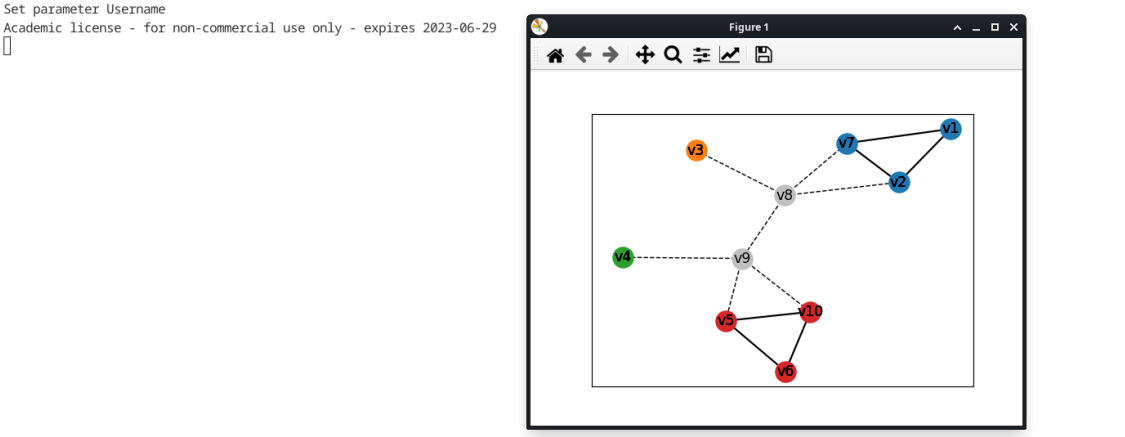


Figura 4.6: Ejemplo de uso de forma silenciosa

4.2.3. Librería de código

Por último, también está disponible su uso como librería para añadir la herramienta al código de otro programa.

Lo único que habría que hacer en este caso es importar las funciones deseadas y hacer uso de las mismas.

A continuación se muestra un ejemplo de cómo podría utilizarse:

```
from cvsp_package import solve_cvsp

solve_cvsp(
    input_file = "./cvsp_package/data/graph1.txt",
    output_file = "./graph1-solution.txt",
    library_name = "gurobi",
    formulation_index = 1,
    k_value = 3,
    b_value = 3,
    no_gui = True,
    quiet = True,
)
```

4.3. Formato de los archivos

Para poder utilizar un mismo grafo múltiples veces y para facilitar el posterior uso de una solución obtenida, estos pueden ser almacenados en archivos de texto siguiendo un formato especifico que el programa es capaz de interpretar.

La estructura de estos archivos se detalla a continuación para que cualquier usuario también pueda interpretar su contenido con un simple vistazo y para que también pueda introducir nuevos grafos al programa o cargar una solución diferente para visualizarla gráficamente.

4.3.1. Definición del grafo

Para el formato de la definición del grafo se ha optado por seguir la siguiente estructura. En la primera línea del archivo se definen las características generales del grafo, osea ser el número de nodos, el número de aristas y si el grafo es dirigido o no, este último expresado con un “0” o un “1”; y a partir de la segunda línea se definen las aristas que componen al grafo, una por línea, e indicando los dos nodos que la conforman. Todos los datos se encuentran separados por comas (,).

En la figura 4.7 se puede observar un ejemplo con un grafo y su codificación en el formato especificado

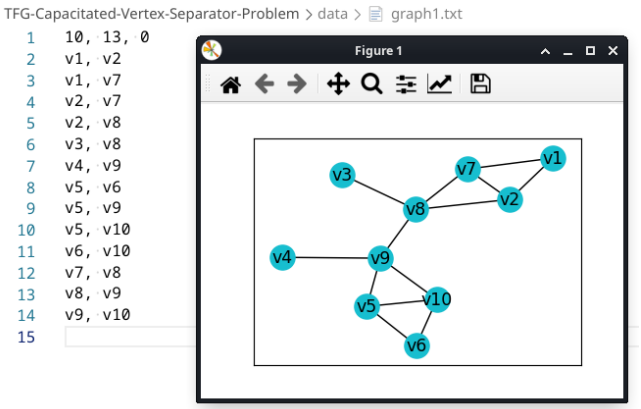


Figura 4.7: Ejemplo del formato de un archivo de grafo

4.3.2. Solución obtenida

Para el formato de la solución obtenida se ha optado por una estructura mucho más sencilla; como el objetivo del problema es separar un grafo eliminando una serie de nodos y la definición del grafo ya la tenemos en un archivo específico para ello, la solución puede ser especificada simplemente con el conjunto de elementos que se deben eliminar. De esa forma, la estructura del archivo se compone del identificador de cada nodo a eliminar separados por comas (,).

Siguiendo con el grafo del ejemplo anterior, su solución sería codificada tal y como se muestra en la figura 4.8.

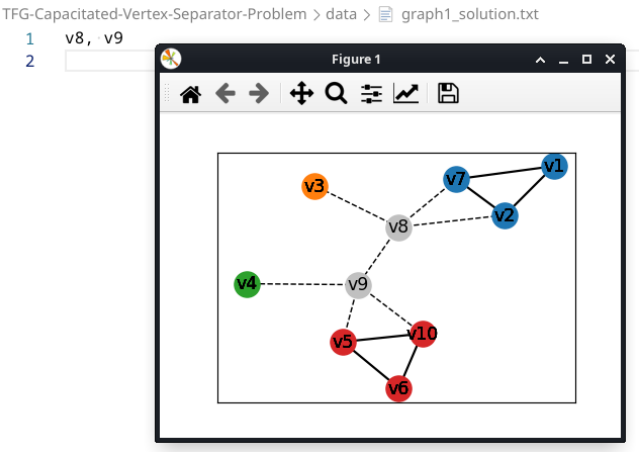


Figura 4.8: Ejemplo del formato de un archivo de solución

Capítulo 5

Análisis de la herramienta

5.1. Preparación

5.1.1. Grafos

Para realizar el análisis se ejecutó el programa utilizando cada una de las implementaciones sobre una serie de grafos con diversos números de nodos y de aristas. Entre ellos se encuentran los grafos de las figuras 5.1 y 5.2, grafos que se usan de ejemplo en el artículo [3].

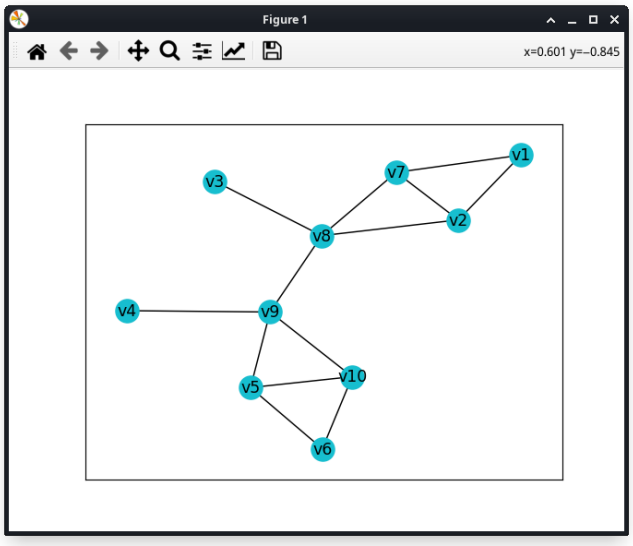


Figura 5.1: Primer grafo del artículo [3]

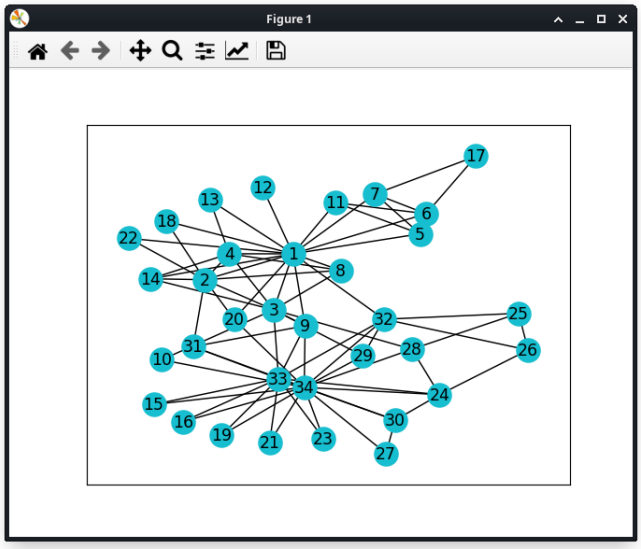


Figura 5.2: Segundo grafo del artículo [3]

El resto de los grafos utilizados se pueden dividir en dos categorías detalladas a continuación.

Por un lado tenemos una serie de grafos en forma de cuadrícula, de 5x5, 10x10 y 20x20 nodos cada uno, los cuales han sido generados haciendo uso de la función `grid_2d_graph()`, de la librería NetworkX, y dándole como parámetros las dimensiones del grafo.

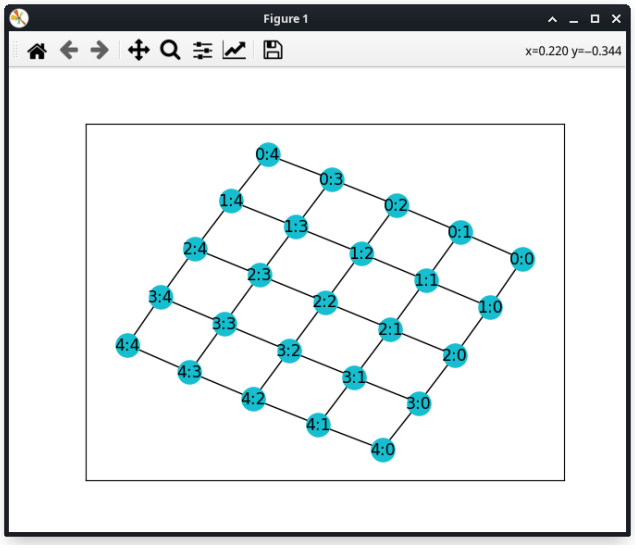


Figura 5.3: Ejemplo de un grafo en forma de cuadrícula de 5x5

Y por otro lado, disponemos de una serie de grafos generados aleatoriamente empleando la función `gnm_random_graph()`, también perteneciente a la librería NetworkX. Esta función se ha ejecutado aportandole como parámetros el número total de nodos y de aristas deseados; para ello se ha preestablecido el número

de nodos a 10 para los grafos pequeños, 50 para los medianos y 300 para los grandes además de dos densidades diferentes de aristas para cada tamaño de grafo, siendo el número de aristas calculado según la siguiente regla: número de nodos del grafo multiplicado por 1,5 para una densidad de aristas baja y por 4 para una densidad de aristas alta; lo cual nos da 15 y 40 aristas respectivamente para los grafos de 10 nodos; 75 y 200 para los de 50 nodos; y 450 y 1200 para los de 300 nodos. ¶ No se van a poder ejecutar todas ¶

En las figuras 5.3 y 5.4 se pueden observar un ejemplo de cada una de estas categorías.

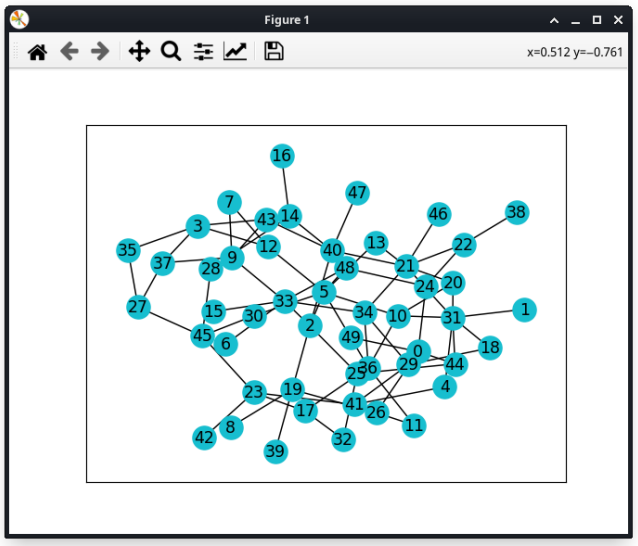


Figura 5.4: Ejemplo de un grafo generado aleatoriamente de tamaño mediano y baja densidad

5.1.2. Entorno de ejecución

Para llevar a cabo la obtención de los resultados y poder comparar las distintas formulaciones implementadas, se solicitó una máquina virtual en los servidores de la universidad con las características que fueron detalladas en el capítulo de tecnologías utilizadas.

En esta máquina virtual se clonó el repositorio de GitHub en el que está alojado el código del programa [11] y se instalaron los requisitos necesarios. A continuación se ejecutó el programa multitud de veces para recopilar los suficientes datos como para realizar el posterior análisis, configurando cada ejecución con diferentes combinaciones de librerías, formulaciones, grafos y valores para los parámetros k y b que tiene como entrada el problema implementado.

5.2. Resultados obtenidos

Para empezar, en la tabla 5.1 se pueden observar los resultados de la comparación del tiempo de cómputo de las distintas formulaciones utilizando el grafo de la figura 5.1 y aportándole a la herramienta los valores de entrada $k = 3$ y $b = 3$.

Librería	**Formulación**	**Ejecución**	**Tiempo**
OR-Tools	1	1	0.10058355599176139 s
OR-Tools	1	2	0.09391754295211285 s
OR-Tools	1	3	0.09487786598037928 s
Gurobi	1	1	0.020040108007378876 s
Gurobi	1	2	0.01726607303135097 s
Gurobi	1	3	0.020142749999649823 s
OR-Tools	2	1	0.12856424890924245 s
OR-Tools	2	2	0.09169619297608733 s
OR-Tools	2	3	0.11438244197051972 s
Gurobi	2	1	0.019990194938145578 s
Gurobi	2	2	0.017910355934873223 s
Gurobi	2	3	0.018399001099169254 s
OR-Tools	3	1	5.182858854997903 s
OR-Tools	3	2	5.148053912911564 s
OR-Tools	3	3	5.238816987955943 s
Gurobi	3	1	1.2711157971061766 s
Gurobi	3	2	1.2730468650115654 s
Gurobi	3	3	1.2381053649587557 s
OR-Tools	4	1	0.0860305760288611 s
OR-Tools	4	2	0.06933053699322045 s
OR-Tools	4	3	0.0699106240645051 s
Gurobi	4	1	0.0632564719999209 s
Gurobi	4	2	0.06178715103305876 s
Gurobi	4	3	0.06987129501067102 s

Tabla 5.1: Comparativa de librerías con el grafo 1

Aunque este es un grafo muy pequeño y que aparentemente no tiene complejidad, en esta tabla ya se pueden observar algunas diferencias entre los tiempos de cómputo de las diferentes implementaciones.

En primer lugar, lo que más destaca es la diferencia de la formulación número 3 del resto, sobre todo al utilizar la librería OR-Tools. Esto puede ser debido a que dicha formulación tiene en su interior el *bin packing problem*, que es resuelto múltiples veces para cada ejecución del problema principal debido a que forma parte de la familia de restricciones (3b).

Por otro lado también se puede observar que por lo general las implementaciones que utilizan la librería Gurobi obtienen la solución en un menor tiempo de cómputo que sus iguales implementadas con la librería OR-Tools.

Para corroborar esas diferencias se repitió el análisis pero esta vez utilizando un grafo de mayor tamaño, específicamente con el de la figura 5.3, y aportándole a la herramienta los valores de entrada $k = 4$ y $b = 9$. Estos resultados se ven reflejados en la tabla 5.2

Librería	**Formulación**	**Ejecución**	**Tiempo**
OR-Tools	1	1	2.0304450099356472 s
OR-Tools	1	2	2.0125140419695526 s
OR-Tools	1	3	2.013226584997028 s
Gurobi	1	1	0.22595032502431422 s
Gurobi	1	2	0.21032686601392925 s
Gurobi	1	3	0.21032686601392925 s
OR-Tools	2	1	4.858895273064263 s
OR-Tools	2	2	4.304193728021346 s
OR-Tools	2	3	4.789025265024975 s
Gurobi	2	1	0.27234301099088043 s
Gurobi	2	2	0.2724232239415869 s
Gurobi	2	3	0.23106516001280397 s
OR-Tools	3	1	Process killed
OR-Tools	3	2	Process killed
OR-Tools	3	3	Process killed
Gurobi	3	1	Process killed
Gurobi	3	2	Process killed
Gurobi	3	3	Process killed
OR-Tools	4	1	Process killed
OR-Tools	4	2	Process killed
OR-Tools	4	3	Process killed
Gurobi	4	1	Process killed
Gurobi	4	2	Process killed
Gurobi	4	3	Process killed

Tabla 5.2: Comparativa de librerías con el grafo 2

Para nuestra sorpresa, parece ser que las formulaciones 3 y 4 requieren de tanto espacio en memoria que el programa no es capaz de obtener la solución sin antes ser matado por el sistema operativo por saturar la memoria RAM.

Es por ello por lo que se decidió implementar estas formulaciones nuevamente pero esta vez siguiendo un método de generación dinámica de las restricciones, o método *lazy* para abreviar, de esta forma se generan únicamente las restricciones que son realmente necesarias para llegar a la solución. El único problema es que OR-Tools no dispone de funcionalidades para implementar formulaciones *lazy* por lo que solo se pudieron implementar con la librería Gurobi.

Como puede verse en la tabla 5.3, estas formulaciones *lazy* si que son capaces de encontrar la solución aunque el tiempo de cómputo es excesivamente alto en comparación con las formulaciones 1 y 2. Para esta tabla se utilizaron los mismos parámetros que para la tabla 5.2.

Librería	**Formulación**	**Ejecución**	**Tiempo**
Gurobi	3 lazy	1	1083.4114536510315 s
Gurobi	3 lazy	2	1085.626199353952 s
Gurobi	3 lazy	3	1080.8011025439482 s
Gurobi	4 lazy	1	243.86323273205198 s
Gurobi	4 lazy	2	242.85018600302283 s
Gurobi	4 lazy	3	244.13149936008267 s

Tabla 5.3: Comparativa de las formulaciones *lazy*

En cuanto a las formulaciones alternativas b y c de la número 1, estas fueron propuestas por el tutor en búsqueda de alguna posible mejoría ya que la formulación 1 es la que mejores resultados da por lo que también fueron tomadas en cuenta en el análisis.

Para destacar sus diferencias se utilizó el grafo en forma de cuadrícula de 20x20, por el gran tamaño que facilita mostrar las diferencias en el tiempo requerido por cada formulación además de los patrones que se generan en el grafo resultante, los cuales son bastante curiosos y podrían ser de aplicación en otro tipo de proyectos.

Para las ejecuciones de la tabla 5.4 se le aportaron a la herramienta los valores de entrada $k = 5$ y $b = 70$.

Librería	Formulación	Ejecución	Tiempo
Gurobi	1	1	3.9381233590029296 s
Gurobi	1	2	3.969851205998566 s
Gurobi	1	3	3.9381599059997825 s
Gurobi	1 alt b	1	5.666013857997314 s
Gurobi	1 alt b	2	5.666013857997314 s
Gurobi	1 alt b	3	5.736349938997591 s
Gurobi	1 alt c	1	169.85730715499813 s
Gurobi	1 alt c	2	168.91667046899965 s
Gurobi	1 alt c	3	170.07103360199835 s

Tabla 5.4: Comparativa de las formulaciones alternativas a la número 1

En esta última tabla podemos observar que la formulación original sigue siendo la que mejores resultados ofrece aunque las otras dos formulaciones también son bastante buenas en grafos de menor tamaño.

Y por último nombrar que en las figuras 5.5 y 5.6 se pueden observar los patrones generados por la separación en los grafos en forma de cuadrícula. Esto se debe a que es un grafo plano, osea que todos sus nodos pueden ser representados en un mismo plano sin que ninguna de sus aristas se cruce, a diferencia de los grafos generados aleatoriamente, que pueden tomar formas muy enrevesadas.

Es por esto último por lo que no han sido tomados en cuenta a la hora de realizar el análisis ya que es muy difícil discernir si las soluciones generadas son realmente válidas o no por estar eliminando más nodos de los que debiera o dejar muchos más grupos separados de los que se desean.

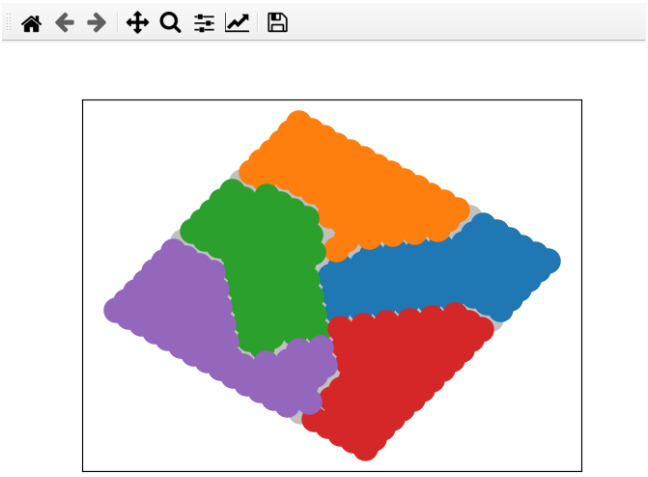


Figura 5.5: Ejemplo 1 de un patrón generado en un grafo de cuadrícula

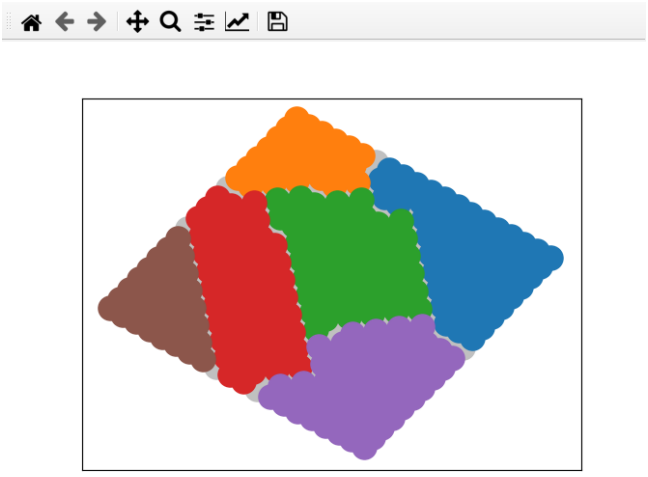


Figura 5.6: Ejemplo 2 de un patrón generado en un grafo de cuadrícula

Capítulo 6

Conclusiones y líneas futuras

La optimización de recursos es, sin duda, un campo de las matemáticas altamente versátil en el mundo moderno y que se ha visto beneficiado recientemente gracias a la increíble expansión que han sufrido las grandes empresas tecnológicas, ya que son estas las que más se benefician de inclinar la balanza hacia la maximización de los beneficios reduciendo los costes al mínimo.

También ha ayudado enormemente la evolución de los computadores que cada vez tienen mayor poder de cómputo, lo que reduce en gran medida el tiempo de resolución de los problemas de optimización y que a su vez facilita que se puedan resolver problemas de mayor tamaño sin tener que esperar toda una vida para obtener la solución deseada.

La herramienta ha sido desarrollada con éxito y dando resultados realmente buenos para algunas de las formulaciones aunque otras resultaron ser peores de lo esperado.

Por último, cabe destacar que el análisis realizado tiene un amplio margen de mejora, yo solo he rascado ligeramente la superficie ya que para algunos de los grafos y configuraciones se requiere de grandes cantidades de potencia y tiempo de cómputo para obtener una solución que yo no he sido capaz de proveer pero que con los recursos necesarios estaría encantado de realizar.

Capítulo 7

Summary and Conclusions

Resource optimization is undoubtedly a highly versatile field of mathematics in the modern world and one that has recently been benefited thanks to the incredible expansion of large technological companies, since they are the ones that benefit the most from tipping the balance towards maximizing profits by reducing costs to a minimum.

It has also been greatly helped by the evolution of computers with ever-increasing computing power, which greatly reduces the time it takes to solve optimization problems and in turn makes it easier to solve larger problems without having to wait a lifetime in order to obtain the desired solution.

The tool has been developed successfully and giving really good results although for limited applications if it is executed in a common computer, in case of wanting to solve large size problems it will be required to resort to the use of high performance computers if it is desired to obtain the solution in a reasonable time.

Finally, it should be noted that the analysis performed has a wide room for improvement, I have only slightly scratched the surface since some of the graphs and configurations require large amounts of computational time and power to obtain a solution that I have not been able to provide but with the needed resources I would be glad to perform.

Capítulo 8

Presupuesto

En la tabla 8.1 se han detallado las tareas realizadas y el tiempo que se ha requerido para la realización de cada una de ellas de manera aproximada además de los recursos que se han utilizado para llevar a cabo el proyecto y el coste de cada elemento.

En cuanto al coste por hora trabajada, de media en España un ingeniero informático recién egresado cobra una media de 20.500 € brutos al año [16], además en España se ha acordado que como máximo se pueden trabajar, de forma legal, 40 horas a la semana y como un año se divide en 52 semanas. Si hacemos los cálculos necesarios nos da que aproximadamente un ingeniero informático cobra 10 € por cada hora trabajada.

Descripción	**Cantidad**	**Precio**	**Total**
Lecturas y planificación	15 h	10 €	150 €
Desarrollo y documentación	80 h	10 €	800 €
Realización del análisis	45 h	10 €	450 €
Interpretación de los resultados	10 h	10 €	100 €
Redacción de la memoria final	30 h	10 €	300 €
Ordenador personal	1 u	1000 €	1000 €
Máquina virtual de la ULL*	1 u	3000 €	3000 €
Total:			**5800 €**

Tabla 8.1: Presupuesto del proyecto

*Precio de un ordenador con características parecidas a lo ofrecido por la máquina virtual que se me ha concedido.

Bibliografía

[1] J. A. Guerra Sánchez, "Optimización de recursos. Concepto y tipos.", *Gestiopolis*, 24-jun-2020. [En línea]. Disponible en: `https://www.gestiopolis.com/concepto-de-optimizacion-de-recursos/`. (Accedido: 13-sep-2022).

[2] G. B. Dantzig, "The Nature of Mathematical Programming", *Mathematical Programming Glossary*. [En línea]. `https://glossary.informs.org/second.php?page=nature.html`. (Accedido: 13-sep-2022).

[3] F. Furini, I. Ljubić, E. Malaguti, P. Paronuzzi, "Casting Light on the Hidden Bilevel Combinatorial Structure of the Capacitated Vertex Separator Problem", *Informs - PubsOnLine*, 16-sep-2021. [En línea]. `https://pubsonline.informs.org/doi/10.1287/opre.2021.2110`. (Accedido: 13-sep-2022).

[4] Python Software Foundation, *Python*. [En línea]. Disponible en: `https://www.python.org/`. (Accedido: 13-sep-2022).

[5] Riverbank Computing Limited, "What is PyQt?", *Riverbank Computing*. [En línea]. Disponible en: `https://www.riverbankcomputing.com/software/pyqt/`. (Accedido: 13-sep-2022).

[6] NetworkX Developers, *NetworkX, Network Analysis in Python*. [En línea]. Disponible en: `https://networkx.org/`. (Accedido: 13-sep-2022).

[7] The Matplotlib development team, *Matplotlib: Visualization with Python*. [En línea]. Disponible en: `https://matplotlib.org/`. (Accedido: 13-sep-2022).

[8] L. Perron, V. Furnon, "OR-Tools", *Google*. [En línea]. Disponible en: `https://developers.google.com/optimization/`. (Accedido: 13-sep-2022).

[9] Gurobi Optimization LLC., *Gurobi Optimization*. [En línea]. Disponible en: `https://www.gurobi.com/`. (Accedido: 13-sep-2022).

[10] Github Inc., *GitHub*. [En línea]. Disponible en: `https://github.com/`. (Accedido: 13-sep-2022).

[11] S. Tabares Hernández, "TFG-CVSP-solver", *GitHub*, 13-sep-2022. [En línea]. Disponible en: `https://github.com/alu0101124896/TFG-CVSP-solver`. (Accedido: 13-sep-2022).

[12] "LaTeX, Evolved", *Overleaf*. [En línea]. Disponible en: https://www.overleaf.com/. (Accedido: 13-sep-2022).

[13] "LaTeX– A document preparation system", *The LaTeXProject*. [En línea]. Disponible en: https://www.latex-project.org/. (Accedido: 13-sep-2022).

[14] "Stackelberg model", *Breaking Down Finance*. [En línea]. Disponible en: https://breakingdownfinance.com/finance-topics/finance-basics/stackelberg-model/. (Accedido: 13-sep-2022).

[15] Colaboradores de Wikipedia, "Bin packing problem", *Wikipedia, The Free Encyclopedia*. [En línea]. Disponible en: https://en.wikipedia.org/wiki/Bin_packing_problem. (Accedido: 13-sep-2022).

[16] "Sueldo del Ingeniero Informático en España", *Jobted*. [En línea]. Disponible en: https://www.jobted.es/salario/ingeniero-inform%C3%A1tico. (Accedido: 13-sep-2022).