MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MÁSTER

Sistema de optimización en coste de rutas para redes de retorno de fibra óptica basado en datos geográficos

Autor: José María Rodríguez Cano de Santayana

Director: Ignacio Ríos Calvo

Madrid

febrero 2021

Declaro, bajo mi responsabilidad, que el Proyecto presentado con el título

Sistema de optimización en coste de rutas para redes de retorno de fibra óptica basado en datos geográficos

en la ETS de Ingeniería - ICAI de la Universidad Pontificia Comillas en el

curso académico 2020/21 es de mi autoría, original e inédito y

no ha sido presentado con anterioridad a otros efectos.

El Proyecto no es plagio de otro, ni total ni parcialmente y la información que ha sido

tomada de otros documentos está debidamente referenciada.

Fdo.: José María Rodríguez Cano de Santayana Fecha: 9 / 2 / 21

Autorizada la entrega del proyecto

EL DIRECTOR DEL PROYECTO

Fdo.: Ignacio Ríos Calvo Fecha: 9 / 2 / 2021

COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIONES

TRABAJO FIN DE MÁSTER

Sistema de optimización en coste de rutas para redes de retorno de fibra óptica basado en datos geográficos

Autor: José María Rodríguez Cano de Santayana

Director: Ignacio Ríos Calvo

Madrid

febrero 2021

Agradecimientos

A todo el equipo de Rivergo, especialmente a Ignacio Ríos, Gilberto Sánchez y Alfonso Hernández por el apoyo continuo que me habéis brindado durante el desarrollo de este proyecto y el gran aporte que habéis hecho a mi formación en muchos aspectos.

Sistema de optimización en coste de rutas para redes de retorno de fibra óptica basado en datos geográficos

Autor: Rodríguez Cano de Santayana, José María**.**
Director: Ríos Calvo, Ignacio.
Entidad Colaboradora: Rivergo Advisors S.L.

RESUMEN DEL PROYECTO

En este proyecto se desarrolla e implementa una herramienta capaz de optimizar en coste las rutas en proyectos de despliegue de fibra óptica. Este problema puede reducirse a la versión de optimización del Árbol de Steiner en grafos. Para la creación de un grafo que represente todos los medios de despliegue posibles se realizará un procesado geoespacial de datos de diferentes fuentes. La herramienta constará de una interfaz en forma de *dashboard* web que permitirá tanto ejecutarla como analizar de forma dinámica los resultados obtenidos.

Palabras clave: Teoría de grafos, datos geoespaciales, optimización, redes de fibra óptica.

1. Introducción

Cuando se pretende conectar con fibra óptica (o cualquier otro medio) una serie de localizaciones a una red troncal existente, se presenta el problema de cómo trazar estas conexiones de manera que el coste sea mínimo.

Para decidir por donde se despliega la fibra se tendrán en cuenta diferentes alternativas. Algunos ejemplos serían: canalizaciones, gaseoductos o tendidos eléctricos, así como el viario completo de España, es decir, todas las vías de transporte: autopistas, autovías, calles, caminos y sendas.

En estos proyectos de conexión es común establecer un límite de inversión por ramal. De esta manera se tendrá que determinar qué puntos son viables, optimizando de esta manera el beneficio estimado teniendo en cuenta los costes e ingresos que reportará conectar cada punto.

En este proyecto se plantea una solución capaz de optimizar estas rutas de manera que el coste sea mínimo a través de una interfaz web que permita tanto la ejecución de la herramienta como la visualización de los resultados de forma dinámica, pudiendo modificar los parámetros económicos del proyecto. De esta manera se pretende mejorar el proceso de evaluación de proyectos de despliegue de fibra óptica, así como reducir notablemente su coste.

2. Definición del proyecto

A continuación, se presentan los objetivos establecidos para este proyecto.

- Crear un grafo sobre el que se ejecuten los algoritmos de optimización. Mediante el preprocesado de datos geoespaciales se creará una estructura en forma de grafo sobre la que se podrán realizar análisis y ejecutar el algoritmo de optimización de rutas.
- Desarrollar un algoritmo que permita optimizar los despliegues de fibra. Se deberá desarrollar un algoritmo que solucione el problema del Árbol mínimo de Steiner de manera que se exploten las características particulares del problema y los datos. De este modo se pretende alcanzar un equilibrio entre el grado de optimización y un tiempo de ejecución que permita que la herramienta sea interactiva.
- Desarrollar una herramienta de visualización interactiva. Se pretende desarrollar una herramienta interactiva en forma de *dashboard* web que permita ejecutar la herramienta y visualizar los resultados de forma dinámica a través de mapas y gráficos de manera que se agilice el proceso de evaluación de proyectos de despliegue de fibra óptica.

3. Arquitectura del sistema

El sistema desarrollado de puede dividir en dos secciones principales: el preprocesado y la ejecución. La parte de preprocesado se encarga de ingestar datos geoespaciales de diferentes fuentes y transformarlos en un grafo sobre el que poder ejecutar el algoritmo de optimización. Esta parte solo será necesaria cada vez que haya modificaciones en estos datos de entrada (se espera una frecuencia semanal). La parte de ejecución se encarga de realizar la optimización y mostrar los resultados a través del *dashboard* web.

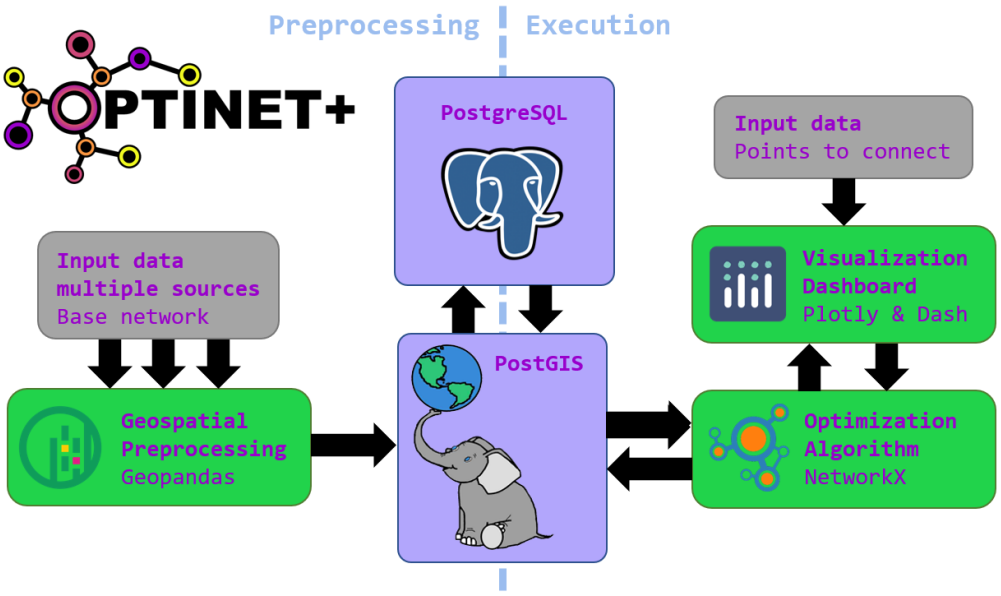

Figura 1 Arquitectura del sistema

4. Resultados

Se ha logrado desarrollar e implementar una herramienta capaz de procesar grandes cantidades de puntos a conectar. Se han desarrollado algoritmos tanto en la parte de preprocesado como en la ejecución que permiten realizar estas evaluaciones de forma interactiva en tiempos muy reducidos. Adicionalmente el algoritmo de optimización implementado reduce considerablemente los costes de las conexiones de grandes proyectos de despliegue de fibra. A continuación, se muestra una representación esquemática de una muestra de puntos reales evaluados por la herramienta.

5. Conclusiones

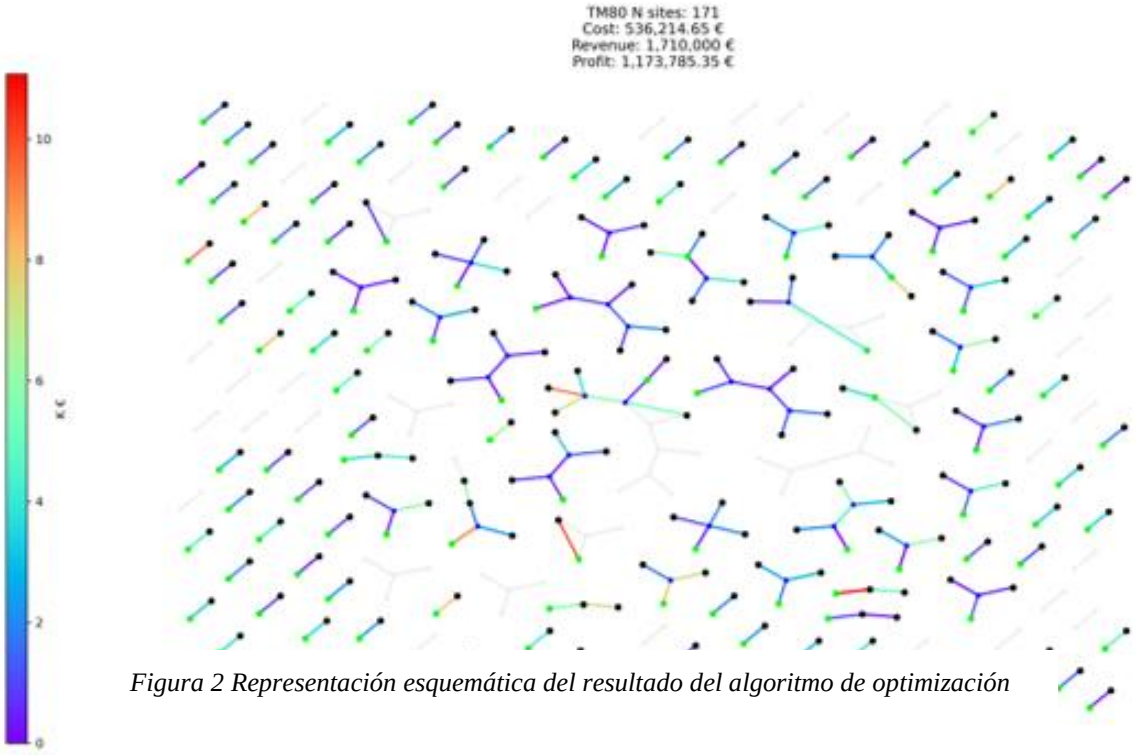

Figura 2 Representación esquemática del resultado del algoritmo de optimización

Se ha desarrollado e implementado una plataforma viable para la valoración de proyectos de despliegue de fibra óptica. Se ha demostrado tanto la viabilidad en las condiciones detalladas en este documento, así como la gran escalabilidad que ofrece la herramienta para futuras ampliaciones que se deseen hacer. Sin duda alguna, la utilización de esta herramienta supondrá una gran ventaja a la hora de valorar proyectos, ya que permite evaluar grandes cantidades de puntos a conectar, incrementando así su capacidad de reducir costes.

6. Referencias

[1] F. K. Hwang and D. S. Richards. *Steiner tree problems*. Networks, 22(1):55-89, 1992.

[2] A. B. Kahng and G. Robins, *On Optimal Interconnections for VSLI*, Kluwer publishers 1995.

[3] Zhang, Lijing and Jing Yi. *Management methods of spatial data based on PostGIS,* Second Pacific-Asia Conference on Circuits, Communications and System 1, 2010. 410-413.

Cost optimization system for fiber-optic backhaul routes based on geographic data

Author: Rodríguez Cano de Santayana, José María**.**
Director: Ríos Calvo, Ignacio.
Colaborating entity: Rivergo Advisors S.L.

ABSTRACT

The purpose of this project is to develop and implement a tool capable of optimizing the cost of connections in fiber-optic deployment projects. This problem can be reduced to the Steiner Tree problem in graphs in its optimization version. To generate a graph that represents all available deployment means geospatial data will be ingested and processed from different sources. The system will have an interactive interface implemented as a web dashboard, allowing both running the optimization algorithm and analyzing results dynamically.

Keywords: Graph theory, geospatial data, optimization, fiber-optic networks.

1. Introduction

When trying to connect with fiber optics (or any other means) a series of locations to an existing backbone, the problem arises of how to trace these connections so that the cost is minimal.

To decide where the fiber is deployed, different alternatives will be considered. Some examples would be pipelines, gas pipelines or power lines, as well as the complete roadmap in Spain, that is, all available transport routes: highways, streets, roads and paths.

In these deployment projects establishing an investment limit per branch is a common practice. Therefore, it will be necessary to determine which points are viable, thus optimizing the estimated benefit, considering the costs and revenue that connecting each point is expected to report.

2. Objectives

- Create a graph structure to later execute the optimization algorithms. Through a geospatial preprocessing pipeline, a graph structure will be generated to later perform an analysis and run the route optimization algorithm

- Develop and implement a route optimization algorithm to compute the best routes for the deployment. This algorithm will have to solve the minimum Steiner Tree

problem in a way that exploits the characteristics of the requirements and data employed in this project.

- Develop an interactive visualization tool. Develop an interactive tool in the form of a web dashboard that allows to run the algorithms and displays the results dynamically through maps and graphs to speed up the process of evaluating fiber optic deployment projects.

3. System Architecture

The developed system can be divided into two main sections: preprocessing and execution. The preprocessing part will be responsible of ingesting geospatial data from different sources and transforming it into a graph on which to run the optimization algorithm. This part will only be necessary each time there are modifications in any input data (a weekly frequency is expected). The execution part will optimize and display the results through the web dashboard.

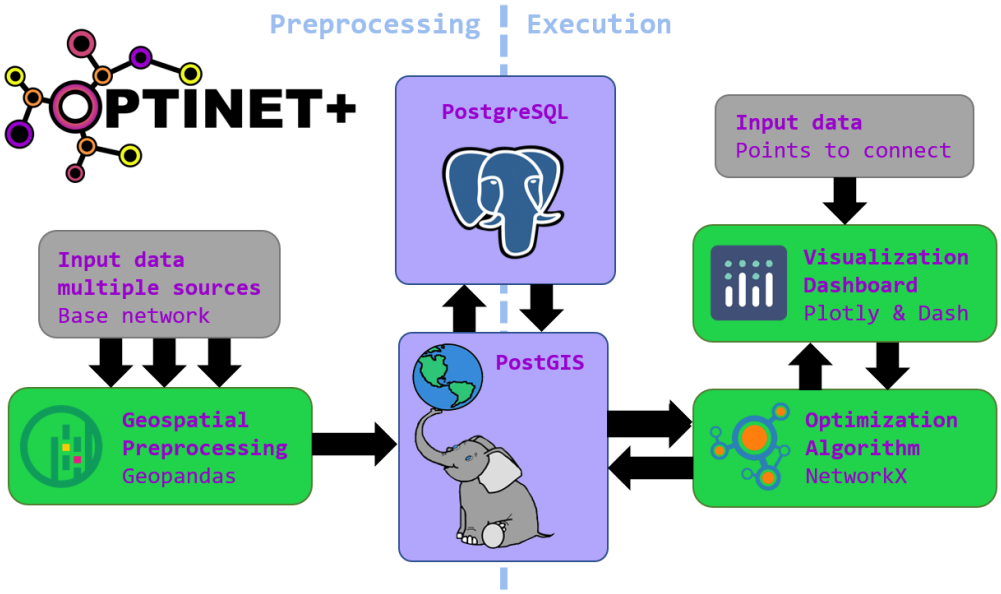

Figura 1 System Architecture

4. Results

A tool capable of processing large amounts of points to be connected has been developed and implemented. Algorithms have been developed both in the preprocessing part and in the execution that allow these evaluations to be carried out interactively in very short times. Additionally, the optimization algorithm implemented considerably reduces the costs of the connections on large fiber deployment projects. Below is a schematic representation of a sample of real points evaluated by the tool.

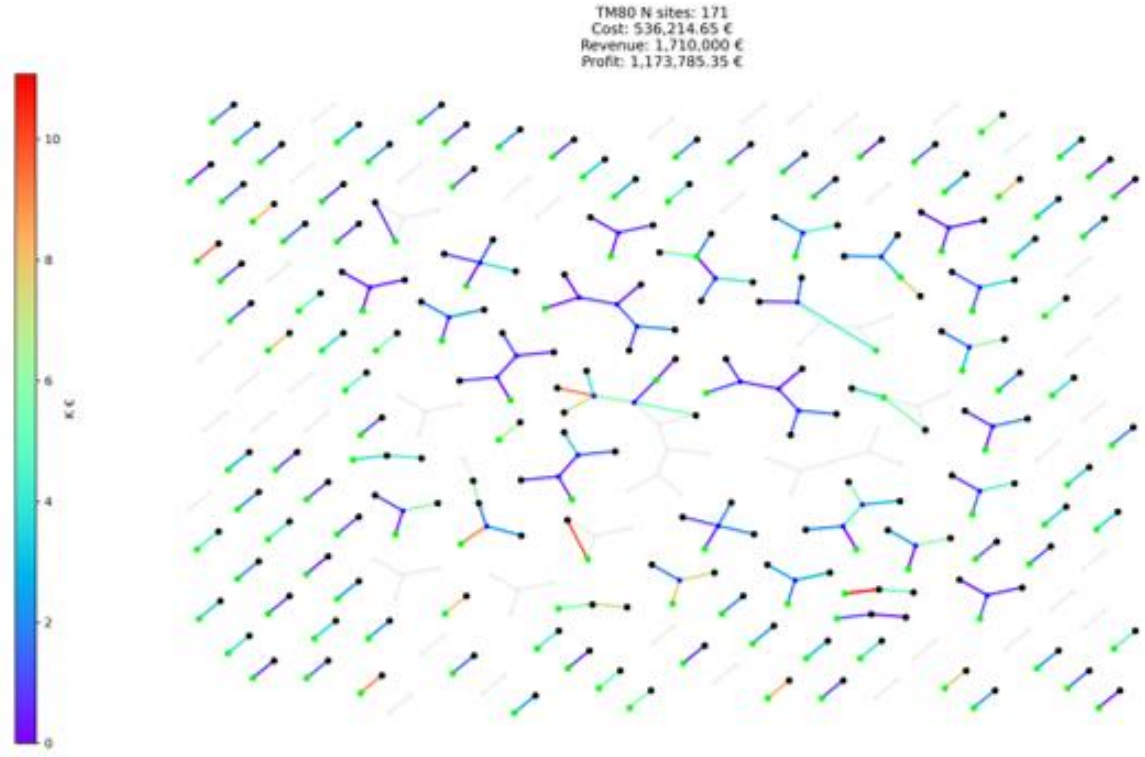

Figura 2 Schematic representation of a sample of real points evaluated by the system

5. Conclusions

A viable platform for the evaluation of fiber optic deployment projects has been developed and implemented. The viability under the conditions detailed in this document has been demonstrated, as well as the great scalability that the tool offers for future development. Undoubtedly, the use of this tool will be a great advantage evaluating projects, since it allows evaluating large numbers of points to connect, thus increasing its ability to reduce costs.

6. References

[1] F. K. Hwang and D. S. Richards. *Steiner tree problems*. Networks, 22(1):55-89, 1992.

[2] A. B. Kahng and G. Robins, *On Optimal Interconnections for VSLI*, Kluwer publishers 1995.

[3] Zhang, Lijing and Jing Yi. *Management methods of spatial data based on PostGIS,* Second Pacific-Asia Conference on Circuits, Communications and System 1, 2010. 410-413.

Índice de la memoria

Índice de figuras

Índice de tablas

Capítulo 1. INTRODUCCIÓN

Cuando se pretende conectar con fibra óptica (o cualquier otro medio) una serie de localizaciones a una red troncal existente, se presenta el problema de cómo trazar estas conexiones de manera que el coste sea mínimo. El enfoque más trivial sería simplemente conectar cada localización a través de la ruta más barata a la red troncal, sin embargo, esta solución está lejos de minimizar el coste. Esto se debe a que, al estar tratando de conectar varios puntos a la vez, podemos ir conectando los puntos entre sí antes de llegar a la red y de esta forma reducir notablemente el coste total de conectar los mismos puntos.

Para decidir por donde se despliega la fibra se tendrán en cuenta diferentes medios de despliegue alternativos. Algunos ejemplos de estos medios serían: canalizaciones, gaseoductos o tendidos eléctricos. Del mismo modo, ya que no siempre es posible o rentable utilizar alguno de estos medios se tendrá en cuenta también el viario de España. Esto incluye todas las vías de transporte: autopistas, autovías, calles, caminos y sendas. Para cada uno de estos medios se tendrá en cuenta el precio por metro lineal de desplegar fibra a través de él. En el caso del viario este precio representará el coste la obra pública necesaria.

En estos proyectos de conexión es común establecer un límite de inversión por ramal. De esta manera se tendrá que determinar qué puntos son viables, optimizando así el beneficio estimado, teniendo en cuenta los costes e ingresos (de forma aproximada) que reportará conectar cada punto.

De forma general (con algunas particularidades), este problema se puede reducir al problema del Árbol de Steiner [1] en su versión de optimización. Este es un conocido problema *NP-hard* por lo que en este proyecto se estudiarán diferentes enfoques evaluando el compromiso entre el tiempo de ejecución y la ratio de aproximación de la solución.

Capítulo 2. DESCRIPCIÓN DE LAS TECNOLOGÍAS

En este capítulo se procede a describir y explicar las principales tecnologías empleadas en el proyecto. Para ello se divide en tres secciones, correspondientes a los entornos tecnológicos más relevantes del sistema: el preprocesado de datos geoespaciales, el algoritmo de optimización y el *dashboard* de visualización.

2.1 PREPROCESADO DE DATOS GEOESPACIALES

La tecnología utilizada en esta etapa para unificar y preparar los datos de entrada es la base de datos geoespacial PostGIS. Esto no es más que una extensión de la base de datos relacional PostgreSQL, añadiendo soporte para objetos geográficos o geométricos, permitiendo realizar operaciones geométricas en las *queries*.

A la hora de elegir una herramienta para esta etapa del proyecto, se presentaban principalmente tres opciones: utilizar las librerías de Python Geopandas y Shapely, la herramienta con GUI QGIS o la base de datos PostGIS. Finalmente, tras realizar pruebas con las tres herramientas se determinó que la más apropiada para el proyecto es PostGIS por las siguientes razones:

- **Las operaciones se realizan sobre disco.** Como cualquier base de datos relacional las consultas se realizan sobre disco por lo que la RAM no supone una limitación si el tamaño de los datos procesados es mayor que la capacidad de esta. Aunque la fuente de datos no supere el límite de RAM en una máquina al hacer muchas operaciones espaciales si se puede llegar a superar este límite. Tanto QGIS como las librerías de Python necesitan cargar los datos en la RAM para poder procesarlos.

- **Índices espaciales** [5]. Esta opción existe tanto en QGIS como en PostGIS. Esto consiste en calcular previamente una “caja delimitadora” (*bounding box)* para cada objeto geométrico (se puede definir fácilmente con solo dos puntos) Figura 1.

Funciona relativamente parecido a los índices *B-Tree* de una base de datos relacional clásica. Permite realizar consultas u operaciones espaciales órdenes de magnitud más rápido que si se hicieran sobre las geometrías completas directamente. Si no existiera un índice espacial, cualquier búsqueda de un objeto geométrico requeriría un escaneo secuencial de todos los objetos, sin embargo, con un índice se organizan los datos de manera que las búsquedas se hacen a través de un árbol de búsqueda.

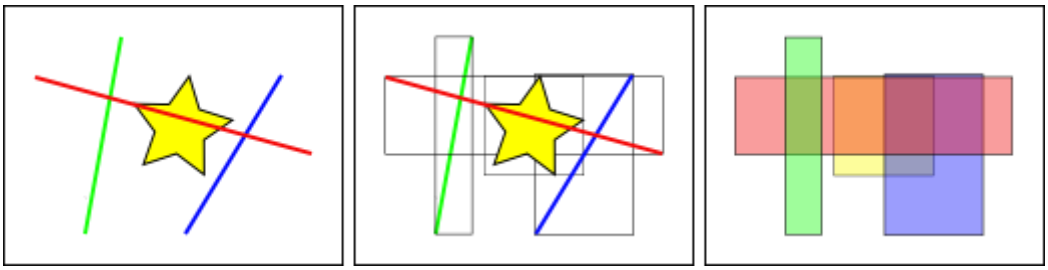

Figura 1 Funcionamiento de los índices espaciales en PostGIS

Un caso muy frecuente al trabajar con datos geoespaciales, y en este proyecto en particular, es la unión (*join)* de dos tablas con algún criterio geoespacial, por ejemplo: para cada entrada de la tabla A obtener los objetos de la tabla B que están a menos de 100 m de distancia. En este ejemplo si las tablas A y B tienen 10.000 filas, cruzarlas sin índices requeriría del orden de 10^8 comparaciones, sin embargo, utilizando índices espaciales se podría hacer con sólo 20.000.

- **Versatilidad y Simplicidad**. PostGIS ofrece una amplia gama de funciones que permite realizar prácticamente cualquier operación espacial imaginable utilizando una sintaxis familiar para muchas personas, sencilla y fácil de modificar y depurar: SQL. De esta manera una operación como la que se expresa en el punto anterior sería tan fácil como:

```
SELECT * FROM A
LEFT JOIN B
ON ST_DWithin(A.geom, B.geom, 100)
```

En cualquier herramienta GIS es necesario codificar los datos geoespaciales con un CRS (*coordinate reference system).* Esto permite proyectar la superficie geodésica de la tierra (o una parte de ella normalmente) a un plano (en el que las coordenadas tendrán dos dimensiones). En este proyecto se utilizará el sistema EPSG:25830, ya cubre la Península Ibérica y sus unidades son metros, lo cual simplifica notablemente los cálculos [6].

2.2 ALGORITMO DE OPTIMIZACIÓN.

Para el algoritmo de optimización se ha utilizado finalmente Python con ayuda de la librería NetworkX. Inicialmente se probó también la librería pgRouting [7], una extensión de PostGIS que provee funciones de enrutamiento geoespacial. Sin embargo, aunque pgRouting permite explotar la indexación espacial que aporta PostGIS, no es suficientemente flexible para cumplir con los requisitos de este proyecto. De este modo, al no poderse adaptar fácilmente a las particularidades de este problema, resulta finalmente demasiado lento para poder integrarse en una herramienta interactiva.

Finalmente, se decidió que la mejor opción era utilizar NetworkX. La ventaja principal que aporta esta librería es una estructura de datos para almacenar grafos flexible y potente. La estructura de datos básica que utiliza NetworkX es una serie de diccionarios anidados. Esto permite que el acceso a elementos del grafo (nodos y ejes) sea muy rápido e independiente del tamaño total del grafo (la complejidad temporal media es $O(1)$) [8]. Aunque esta librería aporta una gran variedad de algoritmos para grafos (incluidos algoritmos de enrutamiento) en este proyecto se han implementado de forma independiente para adaptarse a las necesidades y particularidades de este problema. Otra gran ventaja de esta librería es que proporciona una serie de funciones de visualización que permiten agilizar considerablemente el desarrollo y la depuración.

2.3 DASHBOARD DE VISUALIZACIÓN.

Para la herramienta de visualización se ha decidido utilizar Dash. Dash es una librería de Python para crear aplicaciones web de analítica de forma sencilla. Funciona sobre Flask para el entorno web y utiliza la librearía Plotly para crear los gráficos interactivos que mostrarán en el dashboard.

Ofrece una gran variedad de gráficos con distintos niveles de complejidad e "interactividad". En particular permite crear mapas interactivos, lo cual se ajusta perfectamente a las necesidades de este proyecto.

Capítulo 3. ESTADO DE LA CUESTIÓN

El problema principal que se aborda en este proyecto aplica a muchos tipos de infraestructuras: transporte, distribución, comunicaciones, etc. Por esto la investigación enfocada en solucionar el problema del Árbol mínimo de Steiner ha resultado en una gran variedad de algoritmos exactos y de aproximación, cada uno con sus ventajas y desventajas.

Dado que se conoce que el problema del Árbol mínimo de Steiner es un problema *NP-hard*, asumiendo $P \neq NP$, solo se pueden encontrar soluciones aproximadas en tiempo polinomial. La mejor solución encontrada hasta la fecha es la propuesta por Robins y Zelykovsky [9], demostrando una ratio de aproximación de 1.55, es decir que las soluciones que encuentra son, en el peor de los casos un 55% peores que el óptimo real. Sin embargo, muchas de las soluciones que consiguen una ratio de aproximación bajo también suponen una complejidad temporal elevada. De este modo, cada algoritmo ofrece una relación distinta entre la ratio de aproximación y el tiempo de ejecución, desde los algoritmos no polinomiales que encuentran una solución óptima como [10] hasta la reducción del problema al problema del MST (*Minimum Spanning Tree*, se explicará con más detalle más adelante en este documento).

Es importante destacar que en la gran mayoría de aplicaciones prácticas resulta inviable utilizar un algoritmo óptimo [11], ya que resultan muy poco escalables debido a su complejidad temporal no polinomial.

Dentro de los algoritmos de aproximación utilizados en la práctica existen principalmente tres categorías:

- **Heurísticos** (camino más corto). Estos son los más sencillos de implementar y los que antes se empezaron a descubrir. Estos basan su operativa en cálculos de la ruta más corta entre dos nodos de un grafo siguiendo algún criterio heurístico para mejorar la solución hallada por la solución más sencilla posible (MST). Algunos ejemplos de este tipo de algoritmos se desarrollan en más detalle en [12][13].

- **Contracción.** Existe una gran variedad de algoritmos de este tipo ya que para muchas aplicaciones resultan el punto de equilibrio entre complejidad, tiempos de ejecución, ratio de aproximación y escalabilidad. Algunos ejemplos de este tipo de algoritmos serían [9][14].
- **PL (Programación lineal).** Estos son los más complejos y los que mejores ratios de aproximación consiguen, sin embargo, no encuentran demasiado uso práctico ya que no son muy escalables y sus tiempos de ejecución son mayores que los de las otras dos categorías. De este modo se encuentran en un punto medio entre el resto de los algoritmos de aproximación y los algoritmos exactos en el que es menos frecuente encontrar aplicaciones prácticas. Algunos ejemplos de este tipo de algoritmos serían [15][16].

Capítulo 4. DEFINICIÓN DEL TRABAJO

4.1 JUSTIFICACIÓN

Como ya se ha mencionado el problema que se trata en este proyecto es muy complejo computacionalmente. Por esto, la resolución de este problema de forma visual, aparte de ser impreciso e inconsistente, no resulta suficientemente escalable. Se desea desarrollar una herramienta que permita analizar un gran número puntos en un tiempo muy reducido. De esta manera, la herramienta servirá para realizar análisis más profundos y dinámicos de posibles oportunidades de negocio a gran escala. Del mismo modo se pretende que esta herramienta permita aproximar razonablemente las dimensiones económicas de un proyecto de despliegue de fibra.

4.2 OBJETIVOS

1. Crear un grafo sobre el que se ejecuten los algoritmos de optimización.

Este grafo debe contener la información de todos los medios por los que es posible desplegar fibra (vías, canalizaciones, gaseoductos, etc.) e incluir el coste que tendría tender fibra en cada tramo. Esta estructura de datos se obtendrá a partir de datos geográficos mediante un preprocesado con la herramienta PostGIS. En este paso se abstraerá toda la información geométrica de manera que los datos resultantes, aparte de ser considerablemente menos pesados sean más fáciles de procesar en las siguientes partes del proyecto.

2. Desarrollar un algoritmo que permita optimizar los despliegues de fibra.

Este algoritmo tomará como datos de entrada el grafo descrito en el punto 1, una serie de puntos geográficos a conectar y un límite de inversión medio por punto. El resultado tendrá que contener las conexiones calculadas y la viabilidad de cada punto. La viabilidad de cada punto se determinará mediante la maximización de la siguiente expresión:

$$profit = CAPEX\ por\ punto \cdot n^{o}\ puntos\ conectados - coste\ de\ conectar\ dichos\ puntos$$

Aunque se denomina *profit,* está expresión no pretende representar de forma precisa el beneficio que se obtendrá en un proyecto de despliegue (ya que hay muchos otros factores económicos que no se tienen en cuenta), pero si resulta útil para determinar la viabilidad de los distintos puntos a conectar.

3. Desarrollar una herramienta interactiva

Se desarrollará una herramienta que permita alterar los parámetros de entrada (los puntos geográficos a conectar y el límite de CAPEX) y visualizar los cambios en las conexiones, la viabilidad de los distintos puntos y los parámetros económicos relevantes: coste del despliegue (por tramo y total) y el resultado de la expresión *profit* definida en el punto 2.

4.3 METODOLOGÍA

Se ha seguido una metodología de desarrollo incremental basada en *sprints* similar a Scrum. Se han realizado reuniones dos veces a la semana para revisar el estado del desarrollo, y establecer los objetivos a corto plazo del proyecto, ampliando o reduciendo el alcance del proyecto cuando se ha considerado apropiado.

Debido a la naturaleza de la herramienta se ha seguido a grandes rasgos un orden incremental al principio del proyecto. Primero se ha desarrollado la parte de preprocesado geoespacial y posteriormente el algoritmo de optimización y la herramienta de visualización. Debido a los ajustes hechos en los objetivos y alcance del proyecto ha sido necesario desarrollar en paralelo las tres partes en las etapas finales del proyecto.

4.4 PLANIFICACIÓN Y ESTIMACIÓN ECONÓMICA

A continuación, se muestra en forma de diagrama de Gantt la planificación que se ha seguido en el proyecto con relación a los objetivos descritos en el apartado anterior. El proyecto seguirá en fase de desarrollo a partir de la entrega de este documento, sin embargo, se considera en el alcance del proyecto obtener un PMV que demuestre la utilidad de la herramienta, así como su escalabilidad de cara al futuro.

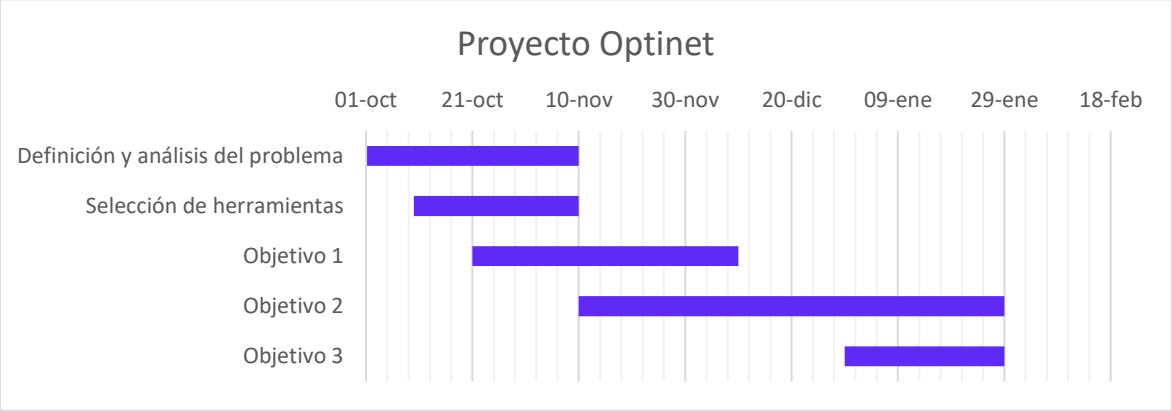

Ilustración 1 Planificación del proyecto

El proyecto no ha supuesto ningún coste fijo ya que el desarrollo se ha hecho completamente *on premises*, utilizando herramientas y dispositivos de los que ya se disponía anteriormente y sin consumir recursos que limitasen otros proyectos en producción.

Capítulo 5. SISTEMA/MODELO DESARROLLADO

5.1 PREPROCESADO DE DATOS GEOESPACIALES

En este proyecto se utilizan datos geoespaciales de distintas fuentes y con distintas características, por lo que es crítico definir e implementar una manera eficiente y consistente de preprocesar estos datos. Dado que el volumen de datos es bastante elevado, el factor eficiencia supone un requisito crítico ya que el sistema tiene que poder soportar que las fuentes de datos se actualicen con cierta frecuencia.

5.1.1 FORMATO Y CARACTERÍSTICAS DE LOS DATOS DE ENTRADA

Los datos de entrada, aunque se obtienen de múltiples fuentes siempre están en uno de esto dos formatos: tabla en base de datos PostGIS o ESRI Shapefile. Este último es un formato de archivo con extensión .shp que actualmente constituye el estándar de facto para almacenar en forma de archivo información geoespacial.

Como ya se ha mencionado, en este proyecto se utilizará la proyección espacial EPSG:25830, por lo que el primer paso será reproyectar los datos de entrada de cada fuente.

5.1.2 LIMPIEZA DE LOS DATOS

Existen diferentes razones por las que los datos geoespaciales pueden contener imprecisiones. La principal son los errores de redondeo fruto de pasar de ESRI Shapefile a PostGIS o de cambiar de una proyección a otra. Esto puede provocar que, por ejemplo, un punto y una línea coincidentes ya no coincidan (por distancias ínfimas). Para solventar este problema se requiere realizar una operación *snap* en las capas que hayan podido sufrir alguna distorsión. Esta operación permite mover objetos ligeramente (según una tolerancia establecida) para volver a hacer coincidir los puntos necesarios. Este error se suele dar en las intersecciones entre varias líneas, lo que puede provocar posteriormente que el algoritmo obvie la ruta óptima por considerar que una vía no es accesible. También supone un problema con arquetas y cajas de conexión que tendrían que coincidir con alguna capa de

canalizaciones o el propio viario y no lo hacen. En la Figura 2 se muestra un ejemplo de este proceso.

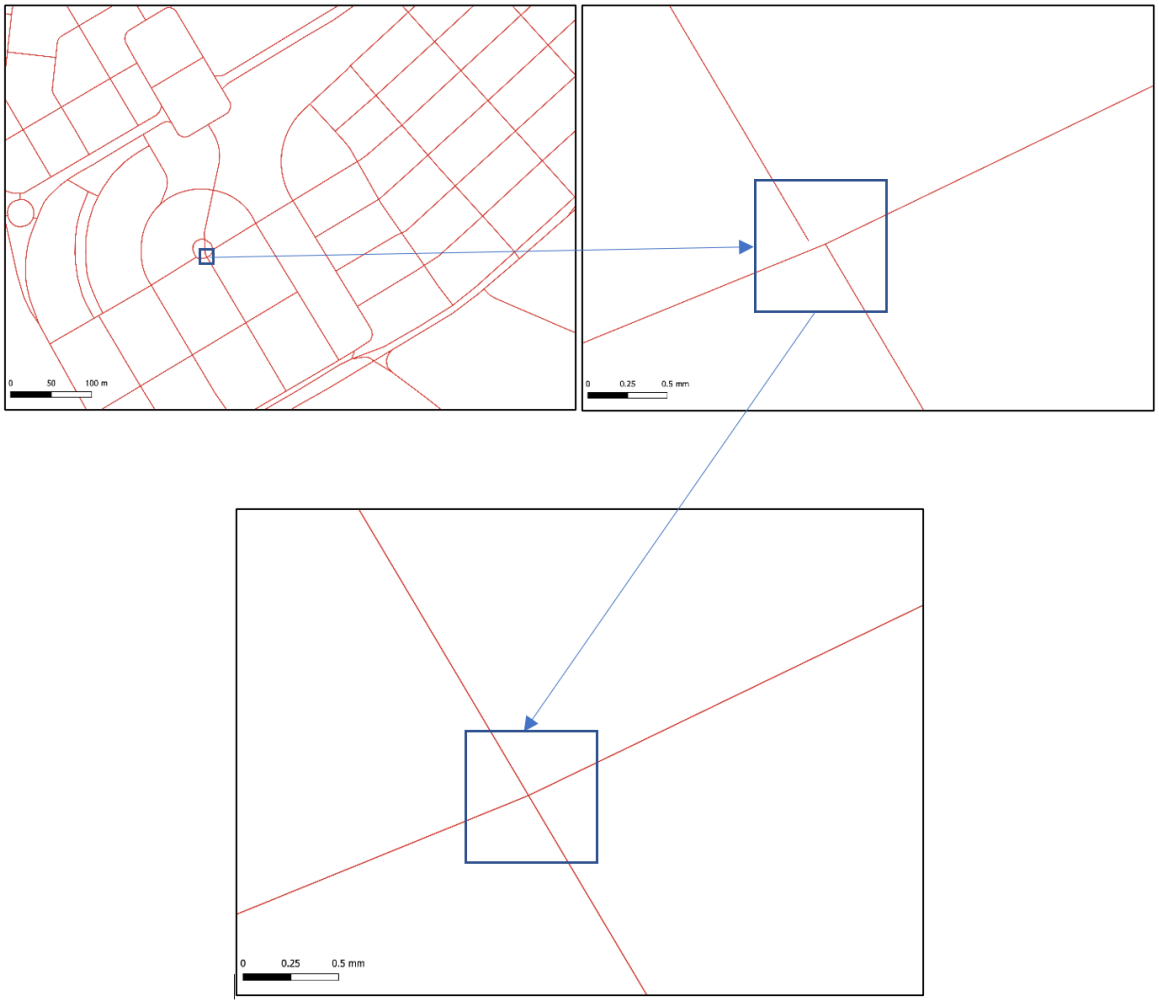

Figura 2 Limpieza de tramos no coincidentes en vías urbanas

5.1.3 REGLAS DE CONEXIÓN

Una etapa fundamental en el preprocesado de datos geográficos es establecer e implementar diferentes reglas para definir la forma de conectar las distintas capas entre sí. Estas reglas responden a las restricciones y posibilidades reales que se encuentran a la hora de desplegar la fibra.

Se utilizan dos métodos de interconexión: "*any to any*" y "puntos clave".

- *Any to any*: este método de conexión consiste en establecer que se puede "saltar" de cualquier punto de una capa a cualquier punto de otra con una distancia máxima. El enfoque más inmediato para resolver este problema sería subdividir los segmentos en intervalos de unos pocos metros y realizar interconexiones entre los nodos de las dos capas que estén a menos de la distancia máxima. Sin embargo, aunque sencillo de implementar, este sistema supone un incremento muy importante en el número de nodos y ejes que tendrá el grafo en el futuro. Ver Figura 3.

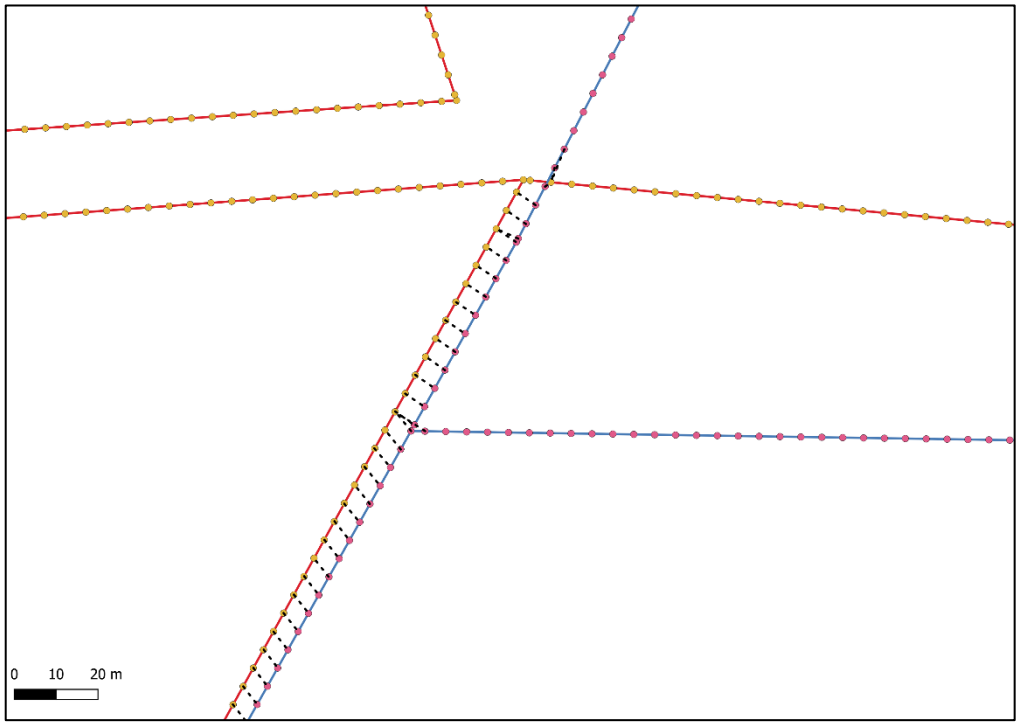

Figura 3 Conexiones entre dos capas según el criterio "any to any"

Por esto, se ha desarrollado una forma alternativa de implementar este método. Consiste en encontrar únicamente los puntos en los que puede llegar a darse un "salto". Estos puntos son los que estén exactamente a la distancia establecida como máximo. Para hallar estos puntos se realiza una operación *buffer* sobre una de las capas y se calcula la intersección entre la frontera de este *buffer* y la capa restante. De esta forma la ruta podrá cambiar de capa en cualquier momento, pero no se

aumenta tanto la complejidad del grafo. Suponiendo que se establece una distancia máxima de 10 m este método se basa en la premisa de que si la ruta va por la capa A y saltar a la capa B resulta más barato lo hará en cuanto sea posible. Del mismo modo si la ruta, va por la capa A y está a menos de 10 m de la capa B significa necesariamente que el camino a través de la capa B es más caro, ya que si no lo fuera habría "saltado" antes.

- Puntos clave. Esta regla de conexión se da cuando el acceso a una capa solamente se puede dar en algunos puntos preestablecido. Algunos ejemplos en los que se tiene que implementar esta regla son: canalizaciones a las que solamente se puede acceder a través de arquetas o tendidos aéreos a los que solo se puede acceder a través de los postes que lo sostienen. Esta regla es más sencilla de implementar y supone añadir menos computación al preprocesado que la regla anterior. Ver Figura 4.

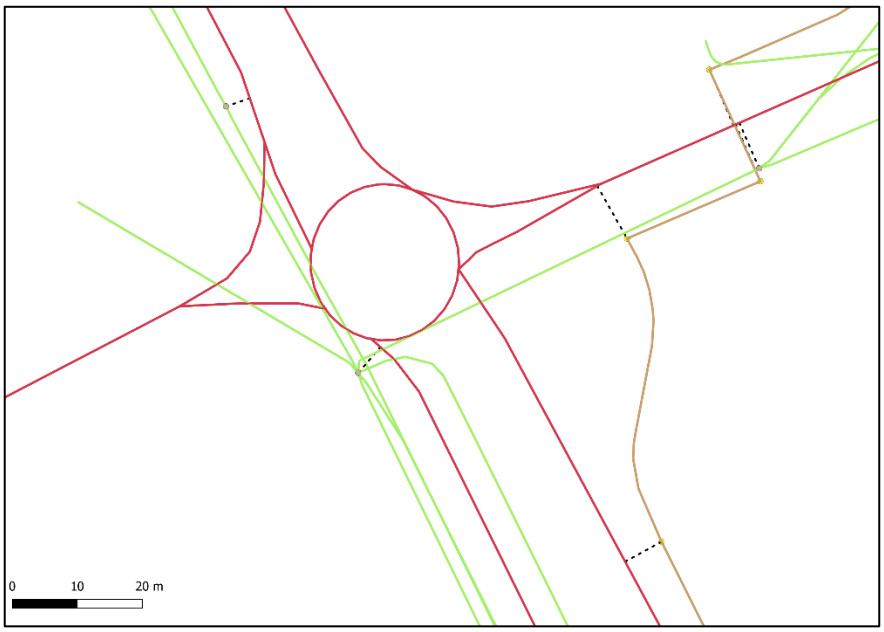

Figura 4 Conexiones entre dos capas según el criterio "puntos clave"

5.1.4 DE DATOS GEOESPACIALES A GRAFO

Una etapa fundamental en el funcionamiento de la herramienta desarrollada en este proyecto es la transformación de dato geoespacial a un grafo sobre el que poder realizar el análisis que se describirá más adelante.

Dado que las geometrías de las infraestructuras de transporte y distribución que se contemplan en este proyecto están representadas como conjuntos de líneas en dos dimensiones sobre una proyección, es crítico que en la etapa de limpieza de datos se hayan hecho coincidir exactamente las coordenadas de las geometrías que deban ser coincidentes. De esta manera para transformar el dato geoespacial a grafo basta con identificar estos puntos de coincidencia, asignarlos a nodos en un grafo y unir dichos nodos según si forman parte de la misma línea. Estas líneas que unen distintos nodos de la red total se denominarán tramos de la red total.

Para poder recuperar la información geográfica perdida en este proceso se asignará un identificador único a cada nodo y cada línea, de tal manera que una vez identificadas las conexiones óptimas se pueda recuperar exactamente a que objetos geoespaciales corresponden. Como se va a perder la información geométrica en este paso es necesario realizar las medidas correspondientes antes de finalizar la transformación. De esta manera se asignarán una serie de atributos a los ejes del grafo para utilizarlos posteriormente en el algoritmo de optimización:

- **Origen/tipo**: para cada tramo se asignará un atributo al eje correspondiente del grafo que especifique que tipo de infraestructura representa, es decir: red de transporte (vía), canalización, gaseoducto, tendido aéreo, etc.
- **Subtipo**: para cada tramo se almacenará también como atributo las características concretas que puedan afectar al coste del despliegue por ese medio. En el caso de la vía pública se especificará el tipo (vía urbana, autopista, camino, senda, etc.); si es una canalización el tipo: (PVC, corrugada, etc.). Esto será útil posteriormente por varios motivos. En primer lugar, es necesario para calcular el coste de despliegue en este tramo. Adicionalmente esta información será utilizada para analizar las

conexiones calculadas y para aplicar filtros si se considera necesario, por ejemplo: filtrar las autopistas y autovías para que el algoritmo no las considere ya que suelen suponen un gran retraso en el proyecto debido al tiempo que tardan en tramitarse las licencias necesarias.

- **Dueño**: para cada tramo se especificará el dueño de la infraestructura, ya sea público (en el caso de la vía) de la empresa cliente o de otra empresa. Esto podrá afectar al precio de despliegue, se podrá utilizar como filtro, como el subtipo, y permitirá analizar los resultados calculados.
- **Distancia**: se medirá la distancia de cada tramo. Aunque el algoritmo no considerará la distancia si no el coste a la hora de optimizar, esta medida es útil para analizar posteriormente las características de las conexiones calculadas.
- **Coste**: este es el parámetro más importante. Se añadirá como atributo a cada eje del grafo. Se calculará el coste de desplegar fibra óptica en cada tramo teniendo en cuenta la distancia del tramo en cuestión y el coste (por metro lineal) en función de las características del tramo: tipo, subtipo y dueño.

De esta manera se obtendrá un grafo que representa la red total con potencial de despliegue y los atributos necesarios para ejecutar el algoritmo de optimización y analizar los resultados.

En el grafo sobre el que se ejecutará en algoritmo también se especificaran una serie de características en los nodos que lo componen. Cada nodo podrá ser de uno de los siguientes tipos:

- **Puntos a conectar:** estos puntos representarán las oficinas, torres o cualquier otra localización que se desea conectar con fibra óptica. Aunque la información geográfica no coincida inicialmente de forma exacta con ningún objeto de la red total de despliegue se preprocesarán para hacerlos coincidir con el viario. En la ejecución del algoritmo estos nodos se pueden considerar los puntos de origen.
- ***Splice*:** estos nodos representan las cajas de conexión existentes en la red de fibra óptica de la empresa cliente. El objetivo final del algoritmo será calcular las rutas entre los puntos a conectar y cualquier *splice* (el que considere óptimo en cada caso). En la ejecución del algoritmo estos nodos se pueden considerar los puntos de destino.

- **Nodos normales:** todos los nodos que no estén en ninguna de las categorías anteriores se considerará un nodo normal. Como ya se ha mencionado estos nodos representan intersecciones y puntos de coincidencia entre distintos tramos de la red total.

Adicionalmente cada nodo tendrá un identificador único que permitirá cruzarlo con la base de datos geoespacial para poder recuperar las geometrías de las rutas calculadas.

Para simplificar el cálculo de rutas se creará un único nodo virtual que representa de forma abstracta la red existente de la empresa cliente. Este nodo y los ejes que lo conectan no tendrán equivalente geográfico ni coste alguno. Este nodo virtual (al que se asignará el identificador -1) estará conectado a todos los *splices* existentes en el grafo con coste 0.

De este modo para calcular la ruta más barata entre cualquier nodo y la red de fibra existente de la empresa cliente bastará con calcular el SP entre el nodo en cuestión y el nodo -1. Este camino se representaría como una lista de nodos que componen el SP, siendo el primer elemento el nodo a conectar, el último el nodo virtual -1 y el penúltimo el *splice* al que es óptimo conectarse.

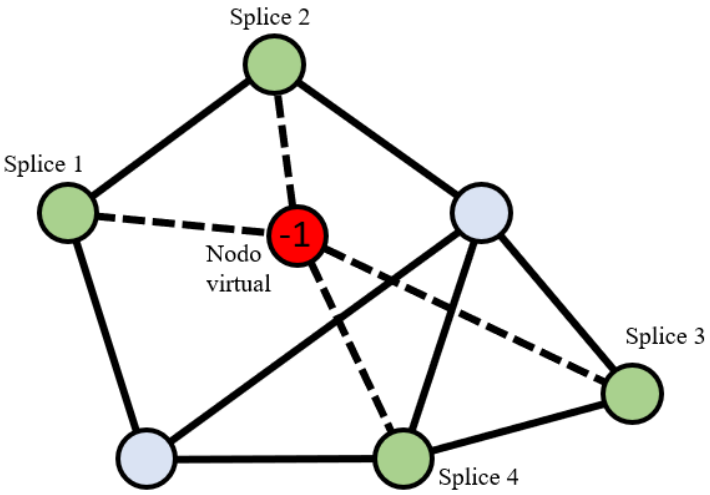

Figura 5 Ejemplo esquemático mostrando un grafo con nodo virtual

Como cabría esperar, entendiendo que se quieren representar redes de transporte y distribución, el grafo resultante es poco denso (*sparce)*, es decir que el orden de magnitud del número de ejes es más parecido al número de nodos que al número de nodos al cuadrado. De esta manera la estructura más adecuada para almacenar el grafo es una lista de ejes en contraposición a una matriz de conexiones. Adicionalmente el grafo sobre el que trabajará en este proyecto es a-direccional, es decir que es indiferente considerar que un eje representa la conexión A→B que B→A. A continuación, se muestra un ejemplo simplificado de estas dos estructuras posibles en las que almacenar un grafo:

Lista de ejes

Node A	Node B	Cost
1	2	158,23
1	3	1455,34
2	3	578,66

Matriz de conexiones

Nodes	1	2	3
1	-	158,23	1455,34
2	158,23	-	578,66
3	1455,34	578,66	-

Figura 6 Tipos de estructuras de datos para almacenar un grafo

5.2 ALGORITMO DE OPTIMIZACIÓN

5.2.1 DEFINICIÓN DEL PROBLEMA MATEMÁTICO SUBYACENTE

El problema que se pretende resolver en este proyecto se puede reducir a la versión de optimización del problema del Árbol de Steiner. A continuación, se describe de forma precisa el enunciado de dicho problema:

Dado un grafo $G = (V, E)$ formado por un conjunto de vértices V y ejes E de orden $n = |V|$, costes en los ejes $c \in \mathbb{R}_+^E$ y un subconjunto de vértices terminales $S \subseteq V$ de orden $k = |S|$ se pretende encontrar un subárbol $T = (V', E')$ de G que contenga todos los nodos terminales y un coste total mínimo. Los nodos no terminales que forman parte del Árbol de Steiner se denominan nodos de Steiner. En el problema abordado en este proyecto el grafo G lo componen todas las capas a través de las cuales se puede desplegar fibra, y el subconjunto

de vértices V son los puntos que se pretende conectar con fibra y la red ya existente a la que se conectarán.

A continuación, en la Figura 7, se muestra una representación geográfica del grafo resultante del preprocesado. En azul se representan una serie de puntos que se desea conectar y en verde los *splices* a los que estos puntos se pueden conectar.

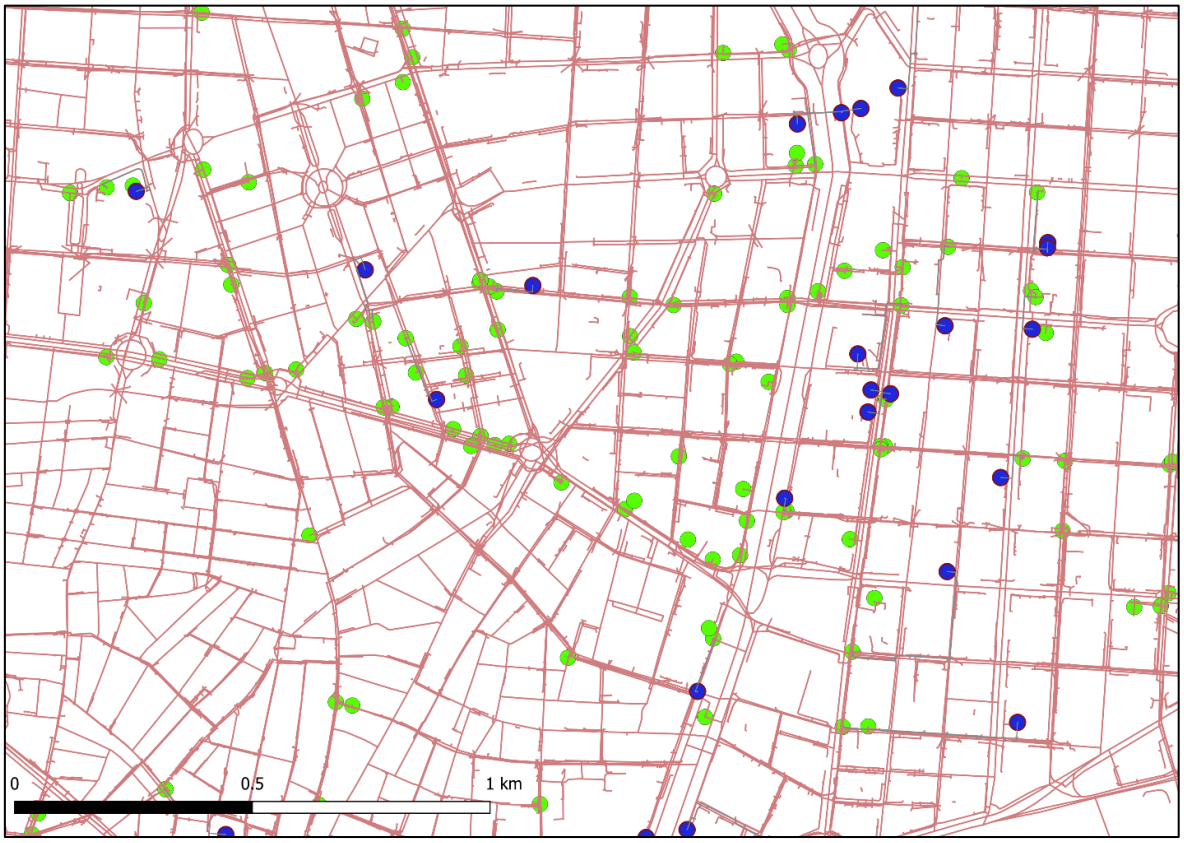

Figura 7 Captura del grafo en una zona urbana junto con splices y puntos a conectar

5.2.2 ENFOQUE INGENUO

Como ya se ha mencionado el problema del Árbol Mínimo de Steiner es *NP-hard* por lo que gran parte del esfuerzo de investigación se ha enfocado a encontrar algoritmos de aproximación con una complejidad temporal logarítmica. Estos algoritmos de aproximación se valoran según dos parámetros principales: el tiempo (tanto en el peor caso posible como la media) y el factor de aproximación. Este factor determina, en el peor caso posible, cuanto se puede alejar la aproximación hallada del óptimo real.

Uno de los primeros enfoques que históricamente se desarrollaron para un algoritmo de aproximación es el árbol de expansión mínimo (MST o *minimum spanning tree)*. Este método consiste en computar el MST sobre el cierre métrico (MC o *metric closure)* del grafo y los nodos terminales. El MC de un grafo G y un subconjunto de vértices $S \subseteq V$ es el grafo completo que contiene únicamente los vértices S y que conecta cada par de vértices u, v con ejes cuyo peso es el del camino de coste mínimo entre estos en G. Una vez calculado el MST del MC, los ejes contenidos en este se sustituyen por los caminos más cortos del grafo G de los que se extrajeron. Este método tiene un factor de aproximación $f = 2 - \frac{2}{k}$, es decir $f \simeq 2$ cuando k es considerablemente grande. Este factor lo comparten muchos algoritmos de aproximación, pero es importante notar que se trata de el peor caso posible. La mayoría de los algoritmos de aproximación más sofisticados, aunque puedan compartir el mismo factor, en la práctica encuentran soluciones con menor coste en la mayoría de los casos.

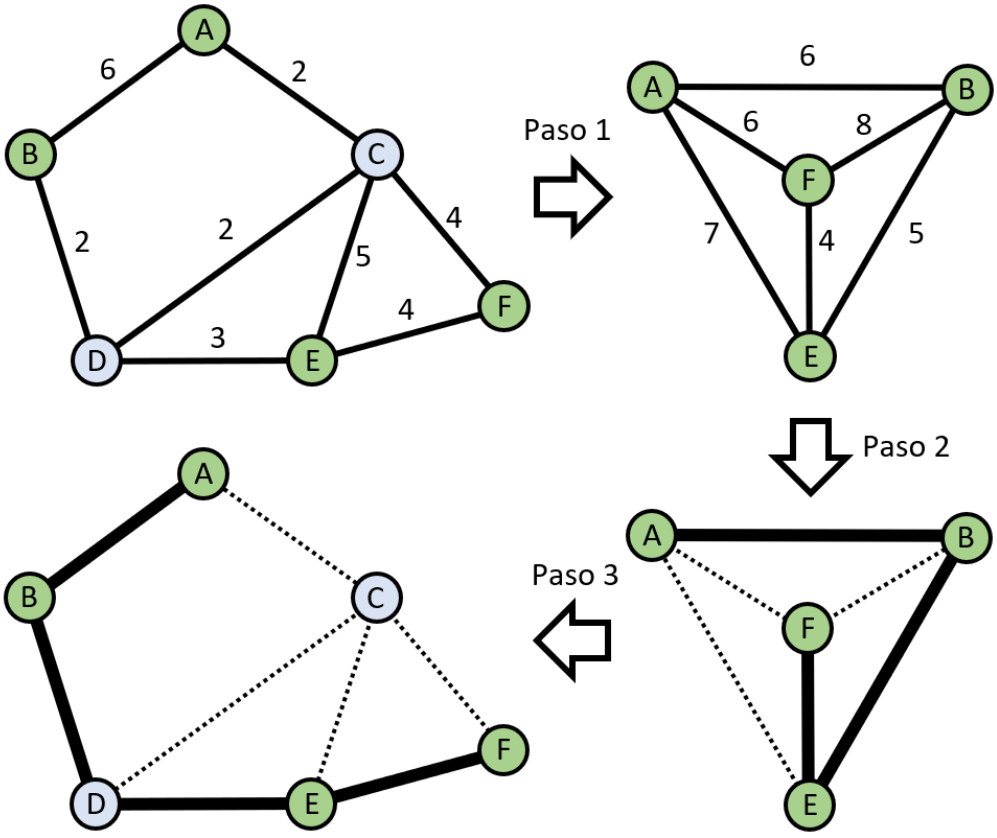

Figura 8 Cálculo del Árbol mínimo de Steiner mediante MST

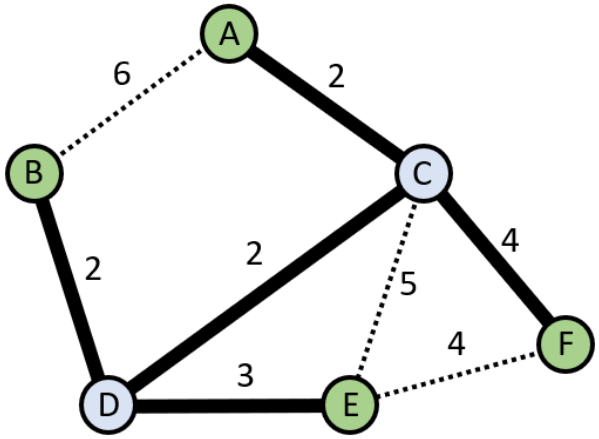

Figura 9 Resultado final con el método del MST

5.2.3 PRECÓMPUTO DE RUTAS

Para generar un grafo sobre el que ejecutar un algoritmo como el descrito anteriormente es necesario calcular las rutas y los costes entre los distintos puntos a conectar. Este proceso es el descrito como "Paso 1" en la Figura 8. De esta manera se puede reducir considerablemente el tamaño del grafo. Como se explicará más adelante este precómputo es útil también para otros algoritmos de aproximación.

Analizando el funcionamiento de cualquier algoritmo para calcular el MST, ya sea el de Prim o el de Kruskal se puede asumir una premisa que simplificará y reducirá notablemente el número de cálculos necesarios. Es importante añadir que en el caso del algoritmo de aproximación por MST el número de ejes del árbol de Steiner resultante será igual al número de vértices menos uno: $|E'| = |V'| - 1$

Premisa: Ningún eje del árbol de Steiner resultante tendrá un coste mayor que cualquiera de los SP entre los nodos que conecta y el nodo virtual -1.

De este modo, si se utiliza algún algoritmo similar a Dijkstra solo con calcular el SP de cada nodo terminal al nodo virtual -1 ya habremos calculado todos los posibles caminos que pueden contenerse en el árbol de Steiner. Con este método se reduce notablemente el número de cálculos necesarios, así como el espacio de grafo explorado. Esto es especialmente importante cuando se analizan una gran cantidad de puntos con distancias considerables entre ellos. En la Figura 10 se muestra de forma esquemática este concepto. Las regiones definidas en azul serían las regiones del grafo que se explorar desde cada nodo terminal (A, B, D) hasta el nodo virtual, en la práctica el *splice* más cercano (C). Como se puede observar A y D no se "ven" mutuamente, por lo que se descartará la conexión entre ellos ya que, aunque se añadiera no podría aparecer el árbol de Steiner resultante. En resumen, si se calculan todas las combinaciones posibles se obtiene un grafo con del orden de $|E| = \frac{(k-1)k}{2}$ ejes y aplicando este método se estudian del orden de $|E| = k$ ejes, siendo más pronunciada esta diferencia cuantos más nodos haya originalmente. De esta forma, si aplicásemos el método del MST el grafo simplificado quedaría como se representa en la Figura 11.

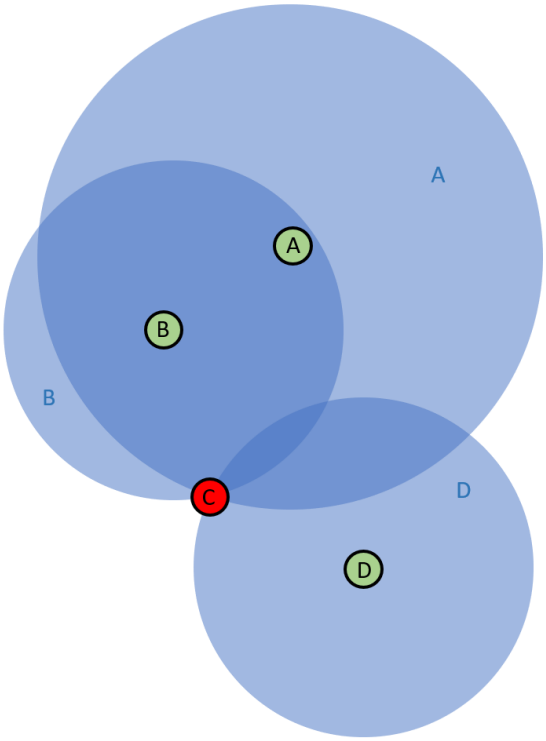

Figura 10 Ejemplo en un espacio euclídeo de la premisa utilizada en el cálculo de rutas

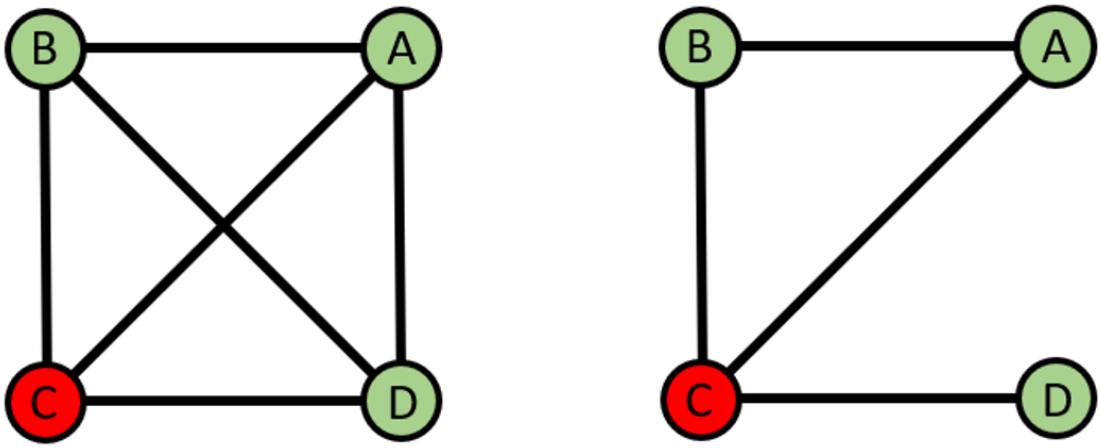

Figura 11 Grafo completo (izda.) frente a conexiones resultantes con premisa (dcha.)

En este proyecto se ha implementado el algoritmo *Uniform Cost Search* (UCS) para el cálculo de rutas (s*hortest path* o SP). Este algoritmo es muy similar a Dijkstra, teniendo una complejidad temporal del peor caso posible (*worst case time complexity*) igual Dijkstra, es decir: $O(E \log V)$ en caso de utilizar montículos binarios de mínimos (*min binary heap*) para la cola de prioridad. Sin embargo, la diferencia fundamental entre UCS y Dijkstra es que la cola de prioridad no se inicializa con todos los nodos del grafo, por lo que el tiempo de ejecución en la práctica es mucho menor. Esto se debe a que la complejidad temporal de ambos algoritmos se deriva de las complejidades de las operaciones que se realizan sobre la cola. Ver Tabla 1.

Operación	**Complejidad**
find-min	$O(1)$
delete-min	$O(\log n)$
insert	$O(\log n)$
decrease-key	$O(\log n)$
meld	$O(n)$

Tabla 1 Complejidades temporales de las distintas operaciones de un montículo binario

Debido a que las complejidades de *delete-min* e *insert* son logarítmicas en función de *n*, reducir todo lo posible el tamaño de la cola de prioridad a lo largo de la ejecución del algoritmo tiene un gran impacto en el tiempo de ejecución. UCS básicamente inserta nodos a la cola de prioridades según son explorados, en vez de insertarlos todos inicialmente como Dijkstra. Esto no solo supone una gran mejora en la velocidad del algoritmo si no que aporta una gran escalabilidad, se puede ejecutar este algoritmo sobre grafos arbitrariamente extensos y el tiempo de ejecución solo dependerá de la distancia entre los nodos que se están estudiando. Esto resulta de gran utilizad en este proyecto ya que se trabajará con un grafo que representa redes de transporte y distribución a nivel nacional, pero en la práctica las distancias de cualquier punto al *splice* más cercano no suelen ser superiores a unos pocos kilómetros. Ver Figura 7.

Normalmente los algoritmos para resolver el problema SP devuelven el coste total de la ruta y la secuencia de nodos que la componen. Sin embargo, en este proyecto, para ahorrar cómputos redundantes se retorna una estructura con todos los predecesores de los nodos explorados. A continuación, se explica en más detalle en que consiste esto.

Tanto en UCS como en Dijkstra u otros algoritmos similares (como A*) el proceso funciona de la siguiente manera (expresado en pseudocódigo):

```
function ShortestPath(Graph G, Source node A, Target node B)

      create priority queue Q
      create distance map dist
create predecessor map pred

      while Q not empty:
            u ← delete-min(Q)

            if (u == B)
                  return dist, pred

            for each neighbor v of u
                  aux ← distance from A to v through u
                  if (aux < dist[v])
                        dist[v] ← aux
                        pred[v] ← u
                        decrease-key(Q, v, new priority)
```

Estos mapas *dist* y *pred* son estructuras clave-valor. En *dist* se almacena para cada nodo explorado *n* la distancia del nodo A a *n* y en *pred* se almacena el nodo anterior (predecesor) en la ruta A → *n*. De esta manera se puede calcular fácilmente (con complejidad $\Theta(\delta(A,n) \leq \Theta(E \log V))$ la ruta A → *n* de la siguiente manera:

```
function GetPath(predecessor map pred, Source node A, node n)

        create path list path
        last_node = n

        while pred[last_node] exists
                append pred[last_node] to path
```

De esta manera solo tenemos que ejecutar UCS para cada nodo que se quiere conectar una vez, obteniendo varios mapas *pred* y *dist*. Posteriormente se pueden consultar estas estructuras mediante la función *GetPath* para calcular cualquier ruta necesaria de manera mucho más rápida.

Para ejemplificar el funcionamiento del algoritmo se representa en la Figura 12 como funcionaría este en un espacio euclídeo, esto se puede generalizar fácilmente a un grafo sustituyendo las distancias por caminos entre nodos de un grafo. Se representa así porque se considera que representa de forma clara el funcionamiento y posteriormente las diferencias entre los distintos algoritmos.

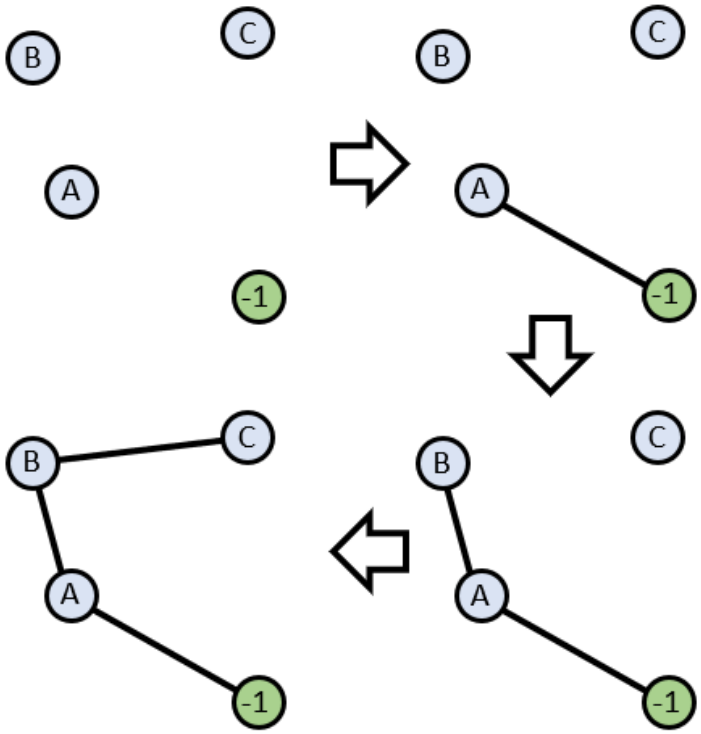

Figura 12 Proceso que sigue el algoritmo MST para calcular las conexiones

5.2.4 ALGORITMOS DE APROXIMACIÓN

Con un enfoque similar al algoritmo descrito anteriormente existen multitud de algoritmos de aproximación para el problema del Árbol mínimo de Steiner que mejoran el factor de aproximación, es decir que aun siendo algoritmos que no garantizan una solución óptima son capaces de acercarse más a ella.

En ese proyecto se han estudiado los algoritmos de aproximación más comunes, estudiando su idoneidad en función de dos factores principalmente: el tiempo de ejecución medio y el coste total de la solución encontrada. El tiempo se evaluará en cuanto que permita cierta interactividad. Para ello se han utilizado tanto simulaciones como ejemplos con datos reales.

5.2.4.1 Loss-Contracting Algorithm (k-LCA)

El primer algoritmo desarrollado en este proyecto que mejora la aproximación del MST es el sugerido en [9]. Este algoritmo ofrece una ratio de aproximación de $1 + \frac{\ln 3}{2} \approx 1.55$. Es decir que no depende del número de nodos terminales o del número de nodos de Steiner, de modo que, a priori, resulta una solución más escalable. Se ha elegido este algoritmo debido a que se considera el mejor [11] dentro de la familia de algoritmos de contracción para solucionar el problema del Árbol Mínimo de Steiner. Estos algoritmos de contracción basan su enfoque en encontrar subconjuntos de nodos terminales para los que es fácil encontrar soluciones locales óptimas. Por esto los algoritmos de contracción suelen describirse mediante un parámetro k que indica el tamaño máximo de los mencionados subconjuntos de nodos terminales. Esta familia de algoritmos como norma general comienzan con un grafo completo $G = (V, E)$ que satisface la desigualdad de Minkowski [17] (ver Figura 13), un conjunto de nodos terminales $S \subset V$ y un parámetro $k \leq |S|$. Podemos asumir que el grafo es completo ya que podemos sustituir cualquier eje $e \in E$ con el SP entre los nodos de e.

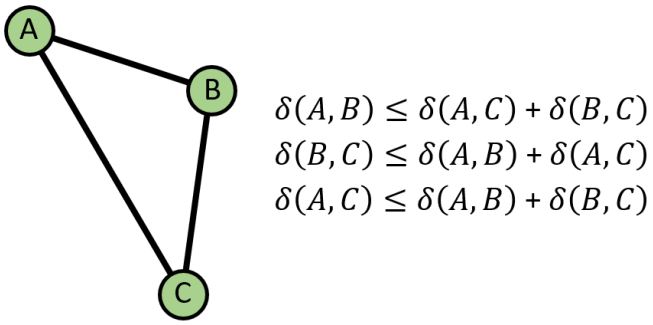

Figura 13 Desigualdad de Minkowski o desigualdad triangular

Es preciso para comprender el algoritmo definir lo que se considera un componente completo (*full component*). Un componente completo es un Árbol de Steiner sobre un subconjunto de nodos terminales $S' \subset S$ en el que los nodos terminales S' son hojas, es decir, nodos pertenecientes a un árbol y con un solo eje coincidente.

Una característica fundamental de este algoritmo es que permite un parámetro k arbitrariamente grande (únicamente limitado en la práctica por el número de nodos terminales), sin embargo, su tamaño repercutirá negativamente en el tiempo de ejecución.

A continuación, se describe el algoritmo desarrollado, denominado por los autores *Loss-Contracting Algorithm (k-LCA)*. Las funciones definidas en el pseudocódigo son las siguientes:

- **MST**: Árbol mínimo de expansión de un grafo (*Minimum Spanning Tree)*
- **cost**: coste total de un grafo, definido como la suma de los costes de todos sus ejes.
- **loss**: coste total del grafo denominado Loss(K). El grafo Loss(K) es un bosque de coste mínimo que alcanza los nodos de Steiner en un *full component* K de manera que cada componente conectado contenga al menos un nodo terminal.

```
function kLCA(Graph G, Terminal nodes S, max contraction k)

      T = MST(G)
      H = G
      while True

            create empty map win

for each full component K of size k in G
win[K] = (cost(T)-cost(K))/loss(K)

            max_K, max_r = pair with max value in win

if (max_r ≤ 0)
      break loop

H = H ∪ max_K
T = MST(T ∪ C[max_K])

      return MST(H)
```

Siguiendo el mismo ejemplo conceptual utilizado en el caso del MST el funcionamiento del algoritmo (con un parámetro $k = 3$) podría describirse como se muestra en la Figura 14. En

este caso dado que el ejemplo es extremadamente sencillo la solución hallada por el algoritmo coincide con el óptimo real que podría encontrarse con un algoritmo de tiempo exponencial como [10].

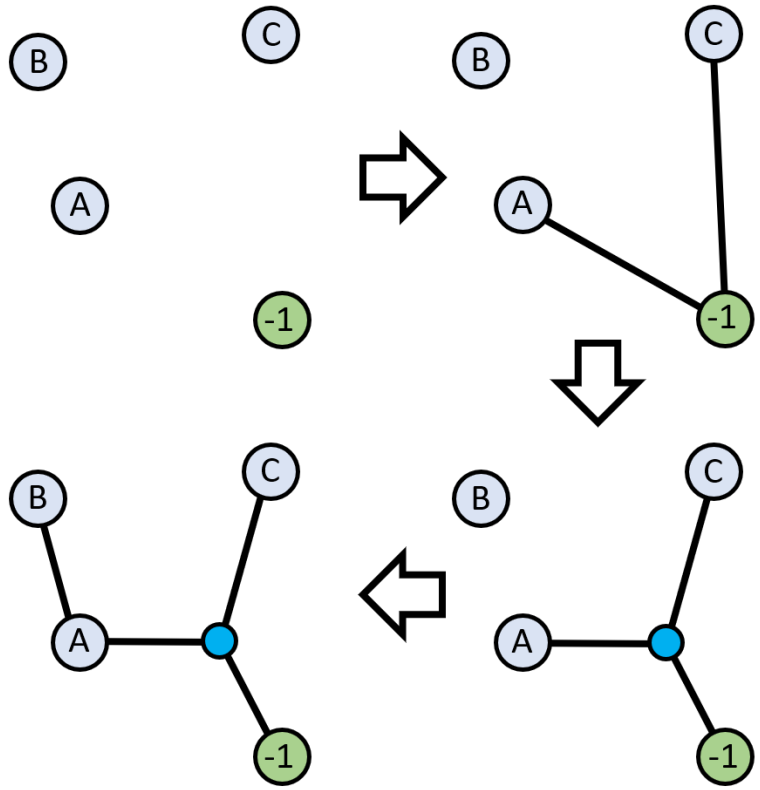

Figura 14 Cálculo del Árbol mínimo de Steiner mediante el algoritmo k-LCA

5.2.4.2 Algoritmo de H. Takahashi & A. Matsuyama

El tercer y último algoritmo desarrollado en este proyecto es el propuesto por H. Takahashi y A. Matsuyama en [12]. Dado que no le asignan ningún nombre en concreto en adelante será referido como T&M.

Este algoritmo ofrece un factor de aproximación de $2 \cdot (1 - \frac{1}{k})$, siendo k el número de nodos terminales. De este modo, el factor de aproximación convergerá a 2, del mismo modo que el

algoritmo básico con MST. Sin embargo, en la práctica, como se detallará más adelante suele encontrar soluciones mejores que MST. De hecho, como se verá cuando se detalle el funcionamiento del algoritmo, las soluciones son siempre estrictamente mejores que MST.

Este algoritmo, a diferencia del desarrollado en el punto anterior, no basa su funcionamiento en iteraciones de contracción. Sigue un proceso aditivo añadiendo rutas de forma definitiva a lo que finalmente será un Árbol de Steiner completo. Una de las grandes ventajas de este algoritmo es su gran rapidez, presentando una complejidad temporal de $O(kn^2)$, siendo k el número de nodos terminales y n el número total de nodos en el grafo. Sin embargo, como se verá posteriormente, aprovechando las particularidades del problema que se aborda en este proyecto se puede mejorar considerablemente el rendimiento del algoritmo.

Al igual que k-LCA, T&M permite obtener Árboles de Steiner con nodos de Steiner de grado superior a 2. Esto quiere decir que puede reducir considerablemente la distancia y coste de fibra desplegada creando nodos intermedios en los que convergen varias rutas.

A continuación, se detalla, en forma de pseudocódigo, el funcionamiento del algoritmo.

```
function T&M(Graph G, Terminal nodes S)

      create empty set of spanned terminal nodes st
      create empty graph T
      add one arbitrary terminal node to st

      while st ≠ S
            create empty array best_edges
            min_cost ← −∞

            for each node v in (S - st)
                  cost, edges ← shortest path in G between
                  v and any node in T

                  if cost < min_cost
                        min_cost ← cost
                        best_edges ← edges

            T = T ∪ best_edges

      return T
```

El algoritmo original propone realizar un cálculo de SP para cada nodo terminal. Adicionalmente, cada ejecución tendrá como nodos de destino todos los nodos en T, por lo que a medida que se construye el Árbol de Steiner, estas ejecuciones serán más lentas ya que tendrá que construirse y consultarse un *set* de nodos cada vez más grande. Sin embargo, si en vez un nodo arbitrario como se propone en el *paper* original inicializamos el algoritmo con el nodo virtual -1, garantizamos que se cumpla la premisa enunciada en el apartado 5.2.3. De esta manera que podemos simplemente precalcular todas las rutas posibles como se explicó para el caso del MST, y de esta manera, reducir considerablemente el tiempo de ejecución. Esto se debe a que, en vez de realizar cálculos de SP en cada iteración, se puede consultar una estructura de datos tipo mapa o diccionario en tiempo $O(1)$. Ver Figura 15.

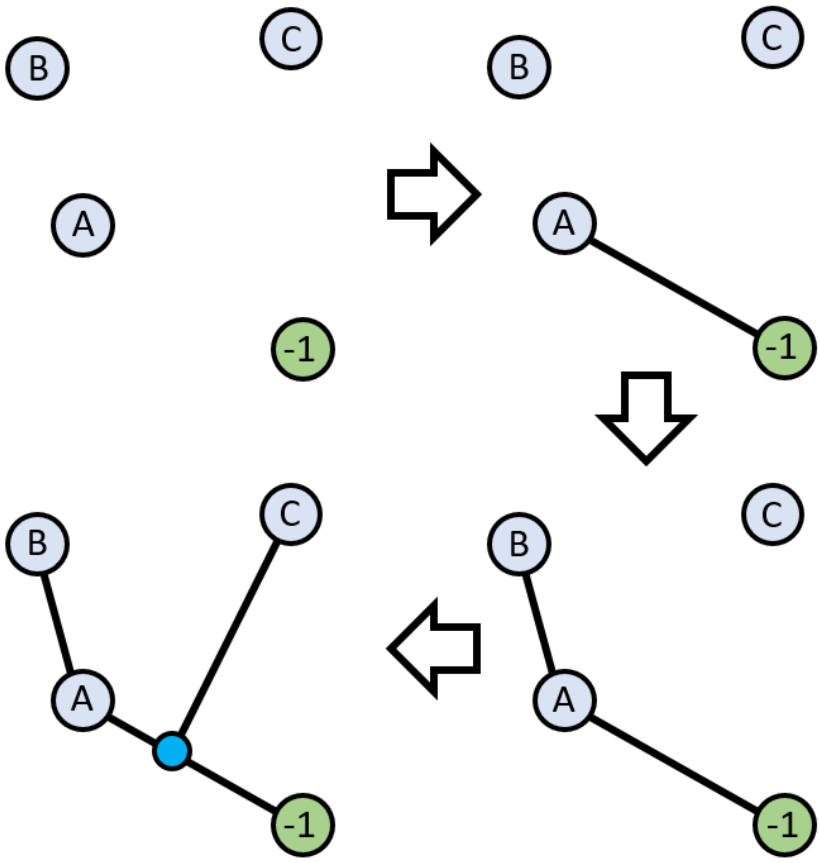

Figura 15 Cálculo del Árbol mínimo de Steiner mediante el algoritmo propuesto por de H. Takahashi y A. Matsuyama

5.2.4.3 Diferencias y características de los diferentes algoritmos sobre datos reales.

Hasta ahora se ha ejemplificado el funcionamiento de los distintos algoritmos en un espacio euclídeo para acentuar las diferencias en un ejemplo lo más sencillo posible. Sin embargo, en un grafo real como el que se aborda en este proyecto las diferencias en los resultados pueden ser menores o incluso no existir. Esto se debe principalmente a que el grafo es prácticamente plano. Esto quiere decir que se podría dibujar en un plano sin que ninguna arista se cruce. No es absolutamente verdad, ya que en ocasiones ejes que representan canalizaciones o tenidos pueden cruzar una vía sin crear una intersección, sin embargo, de forma general podemos afirmar que el grafo es suficientemente parecido a un grafo planar en la mayoría de las regiones como para limitar considerablemente las diferencias en los resultados de los diferentes algoritmos.

Si analizamos un caso más parecido a la realidad se puede observar que en muchas ocasiones los distintos algoritmos encontrarán exactamente la misma solución, frecuentemente la óptima, debido a las restricciones del propio grafo. Ver figura 16.

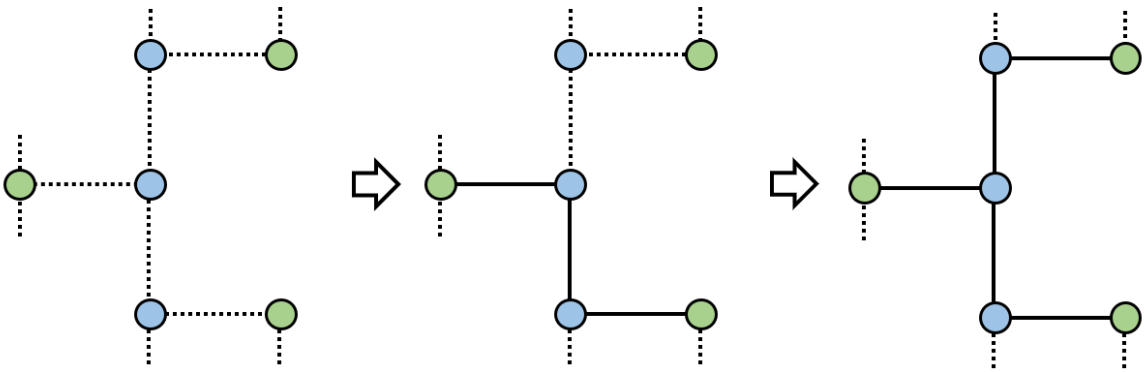

Figura 16 Ejemplo de grafo en el que distintos algoritmos pueden hallar la misma solución

A continuación, se muestra mediante una representación esquemática en la figura 17 los ramales resultantes tras ejecutar el algoritmo sobre datos reales, tanto los puntos que se desea conectar como el grafo subyacente que representa los medios de despliegue.

- **Splices** (●): representados como puntos verdes. En el grafo todos estos puntos estarían conectados a un nodo virtual -1 que representa de forma abstracta la red existente de fibra de la empresa cliente, sin embargo, por facilitar la comprensión de la figura se ha eliminado. De este modo cada componente conectado representa un ramal distinto del despliegue.
- **Nodos en ramales viables** (●): representados como puntos negros. Estos puntos representan los sitios que se desea conectar con fibra y que pertenecen a un ramal viable. Más adelante se detalla como determinar esta viabilidad y como eliminar determinados puntos de estos ramales para viabilizar el resto del ramal.
- **Nodos en ramales viables** (x): representados como "x" moradas. Estos puntos representan los sitios que se desea conectar con fibra y que pertenecen a un ramal no viable. Más adelante se detalla como determinar esta viabilidad y como eliminar determinados puntos de estos ramales para viabilizar el resto del ramal.
- **Nodos de Steiner** (●): Estos puntos representan nodos en el Árbol mínimo de Steiner que no son nodos terminales (es decir en el grafo original eran parte de un medio de despliegue: una intersección entre vías, canalizaciones, etc.) de grado superior a 2. Esto quiere decir que a la hora de desplegar fibra en estos puntos será necesario conectar dos o más tramos de fibra con una caja de empalme. La aparición de estos nodos es señal de que el algoritmo está encontrando correctamente formas de optimizar y reducir la distancia y coste total de fibra desplegada.
- **Ejes del grafo.** Los ejes del grafo representar tramos de fibra entre puntos a conectar, nodos de Steiner y *splices*¸ es importante destacar que en esta representación están simplificados. Cualquiera de estos ejes puede estar compuesto de varios nodos de grado 2. Estos nodos de grado 2 han sido contraídos para obtener un grafo esquemático fácil de representar y que ofrezca una visión general del resultado obtenido del algoritmo. El color de cada eje representa el coste de este de acuerdo con la escala mostrada a la izquierda.
- **Parámetros económicos y del proyecto:** En la parte superior de la figura a modo de título se muestran distintos parámetros relevantes para la evaluación del resultado:

- Número de nodos: es el número de nodos que se consideran viables en cada iteración (más adelante se explica el proceso iterativo de podado de nodos)
- Coste: suma del coste de cada tramo de despliegue contenido en el Árbol de Steiner calculado, posteriormente en la representación más avanzada en el *dashboard* será posible desgranar esta información para evaluar de forma más precisa el resultado.
- Ingresos: esto pretende ser únicamente una aproximación a los ingresos que reporta la conexión de los puntos en una determinada iteración. A medida que se desarrolle más el proyecto esta estimación será cada vez más precisa, sin embargo, en el estado actual del proyecto resulta útil únicamente para determinar de forma general la viabilidad de los distintos puntos y ramales.
- Beneficio: representa sencillamente la resta entre ingresos y coste. El objetivo del algoritmo de podado que se detallará más adelante es maximizar esta cifra.

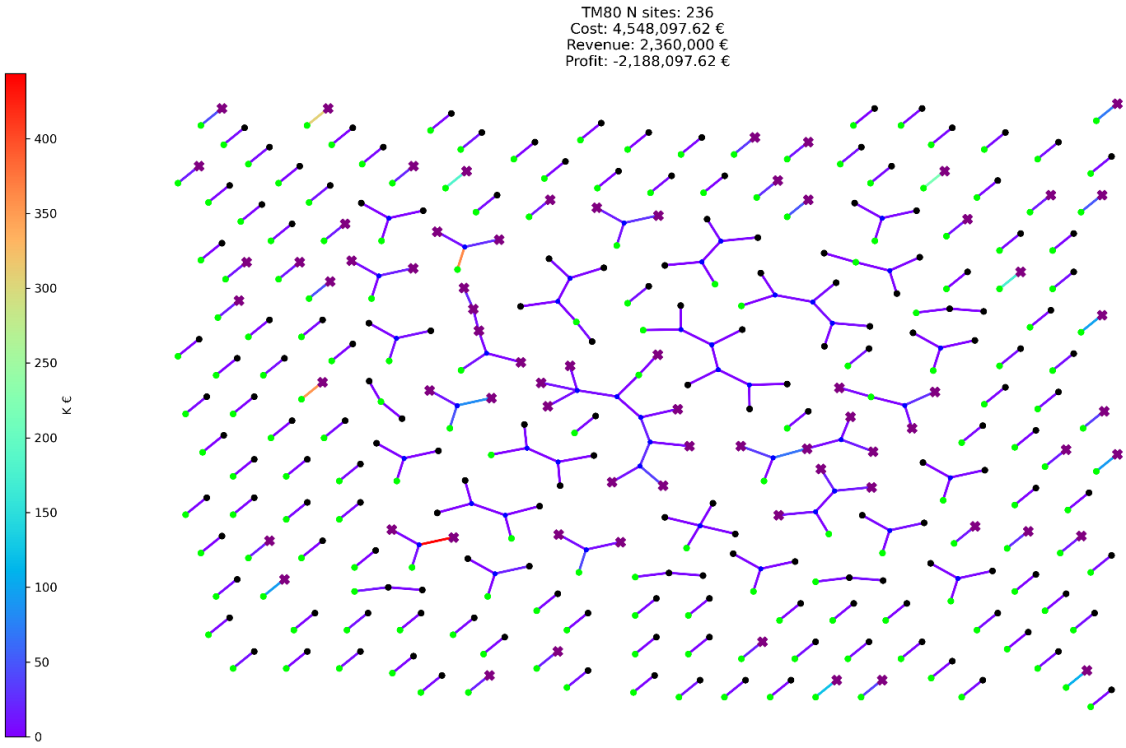

Figura 17 Representación esquemática de todos los ramales calculados y la viabilidad inicial de estos

5.2.5 PODA DEL ÁRBOL RESULTANTE PARA ADAPTARLO AL LÍMITE DE CAPEX

Una de las funcionalidades adicionales que ofrece la herramienta desarrollada en este proyecto es la capacidad de calcular y determinar que nodos terminales y ramales resultan viables según un límite de inversión determinado. De este modo se podrán optimizar los despliegues de manera que cuantos más puntos se evalúen a la vez mayor retorno será posible obtener de media por cada punto conectado. Debido a las limitaciones de tiempo en el proyecto se estable un único límite de CAPEX medio por punto, sin embargo, como se desarrollará más adelante, la herramienta permitirá tener en cuenta una información más granular de los parámetros económicos del proyecto, aumentando aún más la capacidad de la herramienta de maximizar los beneficios de un proyecto de despliegue de fibra óptica.

Es importante definir como se considera si un punto es viable o no. Esto está definido puramente como una regla de negocio.

Viabilidad de un punto a conectar: Se define un ramal cualquiera como viable si el coste total del ramal (subárbol a partir del splice más cercano) es menor que el límite de CAPEX por el número total de puntos viables conectados. Si un ramal se determina no viable se procederá a eliminar el nodo hoja con coste de conexión máximo. De esta manera se procederá a recalcular el Árbol mínimo de Steiner sin el nodo eliminado. Esto se debe a que eliminando la necesidad de conectar un nodo que ya se ha determinado no viable las conexiones para el resto de los nodos calculadas por el algoritmo pueden ser distintas y en caso de serlo será con un coste estrictamente menor al anterior.

A continuación, se muestra este proceso en un ramal, en el que tras eliminar los dos nodos más caros es posible viabilizar el resto del ramal (ver Figura 18). Nótese que los colores cambian en relación también al resto de ramales siendo podados, no es una escala absoluta.

Figura 18 Proceso de poda de un ramal

De este modo el pseudocódigo que representa el algoritmo de podado de los ramales sería:

```
function PruneTree(Graph T, capex limit cl)

      remove virtual node -1 from T
      create empty graph T'

      for each connected component cc in T
            if cost(cc) > nodes in cc * cl
                  add cc to T'

if T' is empty
            return T

      max_cost_leaf ← null
      max_cost ← −∞

      for each leaf l in T'
            if l is not a splice and cost(edge(l)) > max_cost
                  max_cost ← cost(edge(l))
                  max_cost_leaf ← l

      remove l in T
      return PruneTree(T, cl)
```

A continuación, se muestra una serie de pasos en el proceso completo de podado con datos reales (ver Figura 19). En este caso concreto se reduce el número de puntos conectados de 226 a 171, llevando la estimación de beneficio de -1.120.472,49 € hasta +1.173.785,35 €.

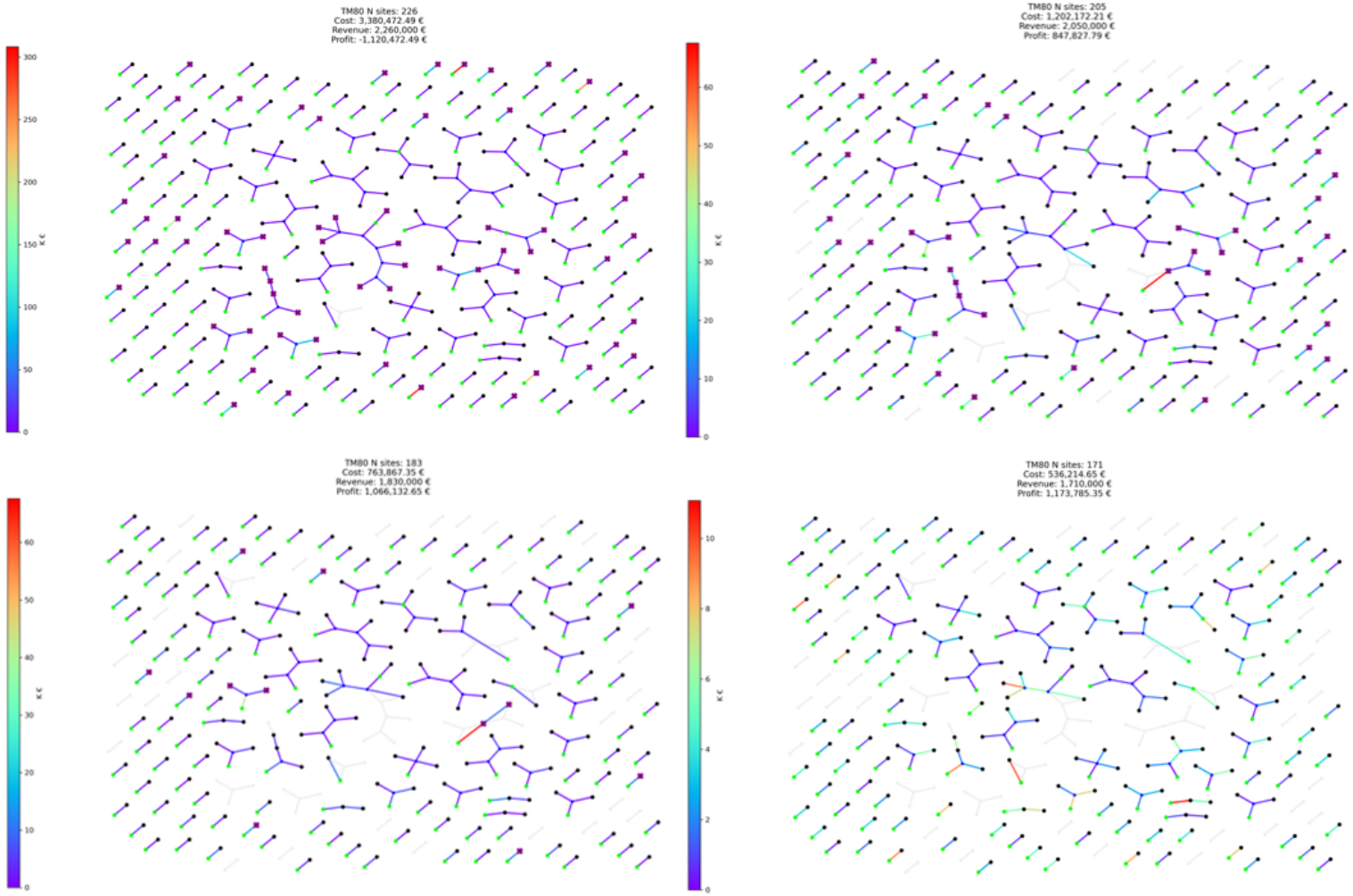

Figura 19 Muestra de 4 estados de los ramales en el proceso de podado

5.3 *DASHBOARD DE VISUALIZACIÓN*

La parte final en este proyecto es el desarrollo e implementación de un dashboard interactivo de visualización desde el cual se podrán analizar los puntos que se quieran considerar en tiempo real, variando los distintos parámetros de entrada para poder evaluar un proyecto.

Debido a la limitación temporal del proyecto no ha sido posible implementar todos los objetivos deseados para el *dashboard* de visualización. Se ha implementado un PMV que permite leer un archivo .shp situado en el mismo directorio que el *script* de Python y muestra a través de una interfaz web lo mostrado en la Figura 20.

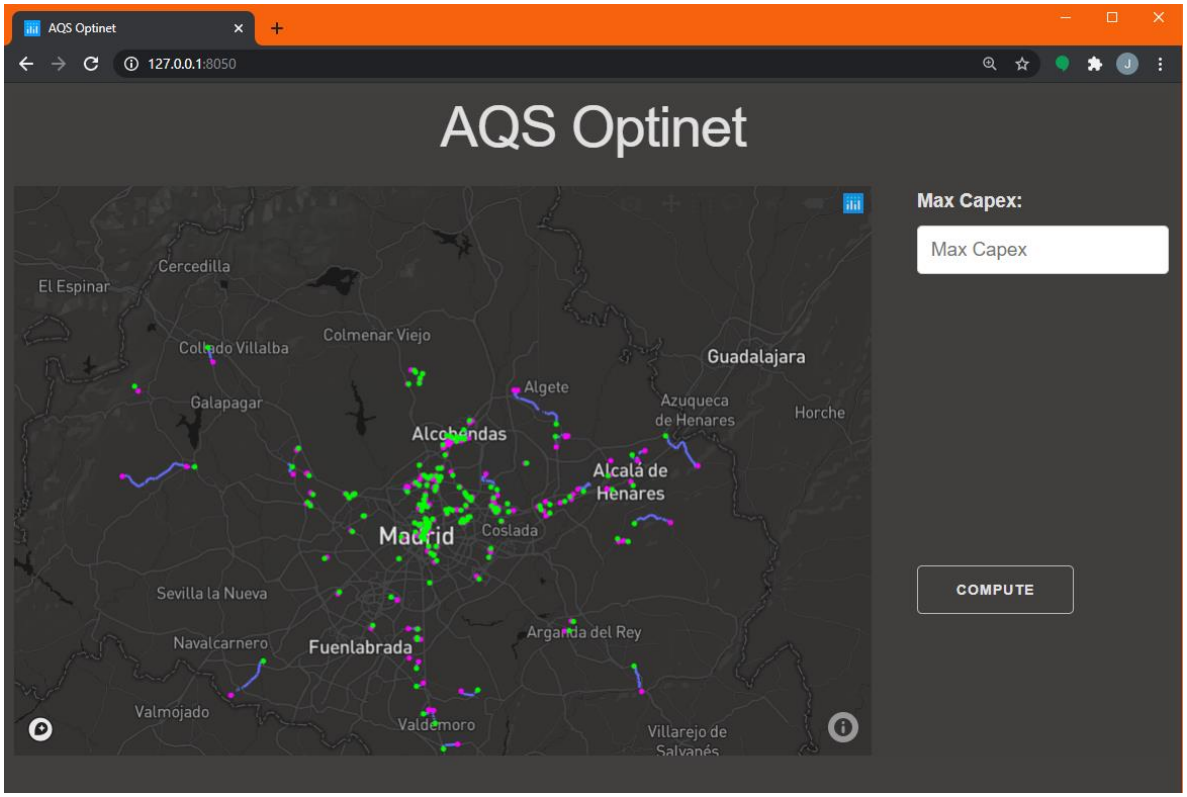

Figura 20 Dashboad de visualización

En el estado actual del *dashboard* es posible establecer un límite de CAPEX y calcular las conexiones. Los resultados se mostrarán en forma de mapa y se almacenarán en un archivo .shp en el mismo directorio. De esta manera se puede almacenar en PostGIS si se desea o analizarlo en más detalle en alguna herramienta GIS como QGIS.

A pesar de las limitaciones del *dashboard* web implementado si existen otras funcionalidades listas para implementar en cuento sea posible. Estas están actualmente operativas en *jupyter notebooks*, de manera que esta transición será rápida en cuanto sea posible. A continuación, se muestran los algunos de los gráficos listos para implementar en el *dashboard*. Para la realización de esta memoria se han ofuscado los tipos y subtipos de medio de despliegue, así como los dueños de estos.

- **Mapa interactivo**. Figura 21. En este mapa se pueden apreciar exactamente las rutas calculadas para el despliegue de fibra, la localización de los *splices* utilizados (verde) y los puntos a conectar (magenta). Adicionalmente se diferencia el tipo de medio de despliegue según el color de las conexiones.

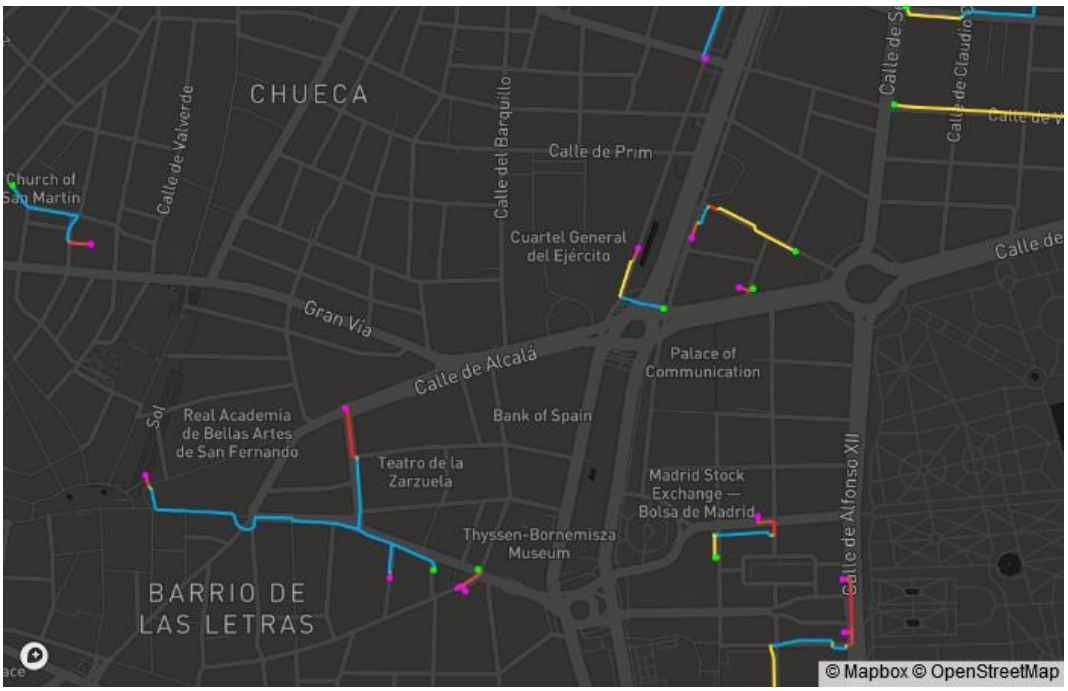

Figura 21 Mapa interactivo en el que se muestran las rutas calculadas y los tipos de medios de despliegue

- **Histograma de distancias por tipo**. Figura 22. Esta visualización permite comprender rápidamente en que medios se ha de desplegar la fibra. De este modo se puede comprender que medios de despliegue resultan más rentables en la zona de despliegue, aportando información valiosa de cara a nuevos proyectos o acuerdos.

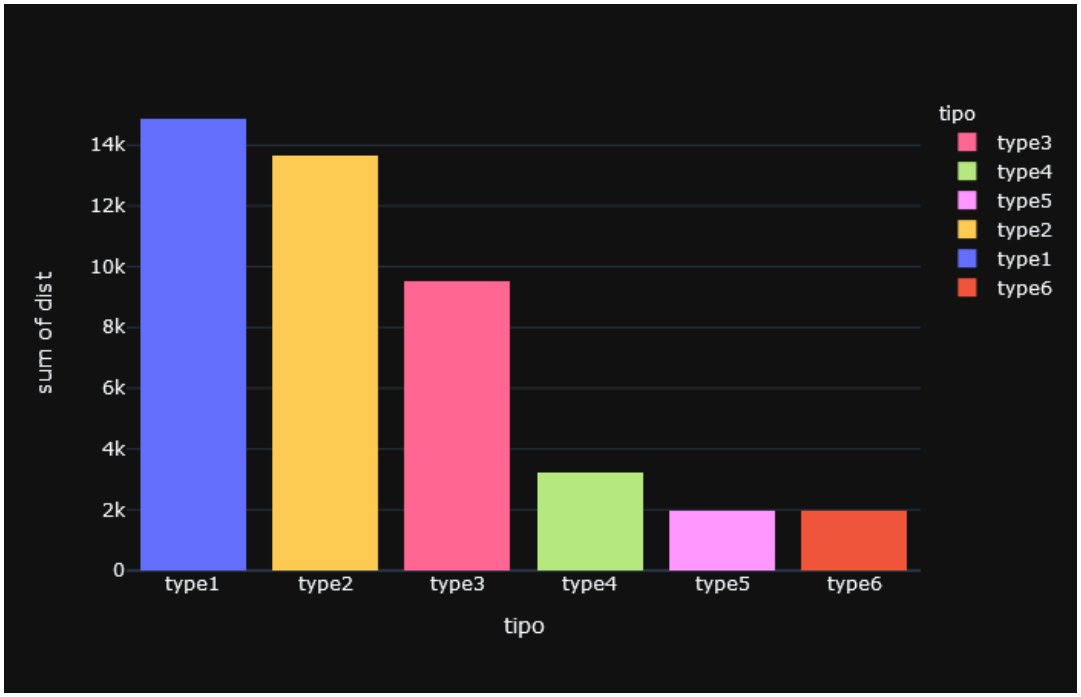

Figura 22 Histograma de distancias por tipo

- **Histograma de distancias por coste**. Figura 23. Esta visualización permite comprender rápidamente en que medios se está enfocando el capital invertido. En conjunción con el gráfico anterior, permite comprender que medios de despliegue resultan más rentables en la zona de despliegue, aportando información valiosa de cara a nuevos proyectos o acuerdos.

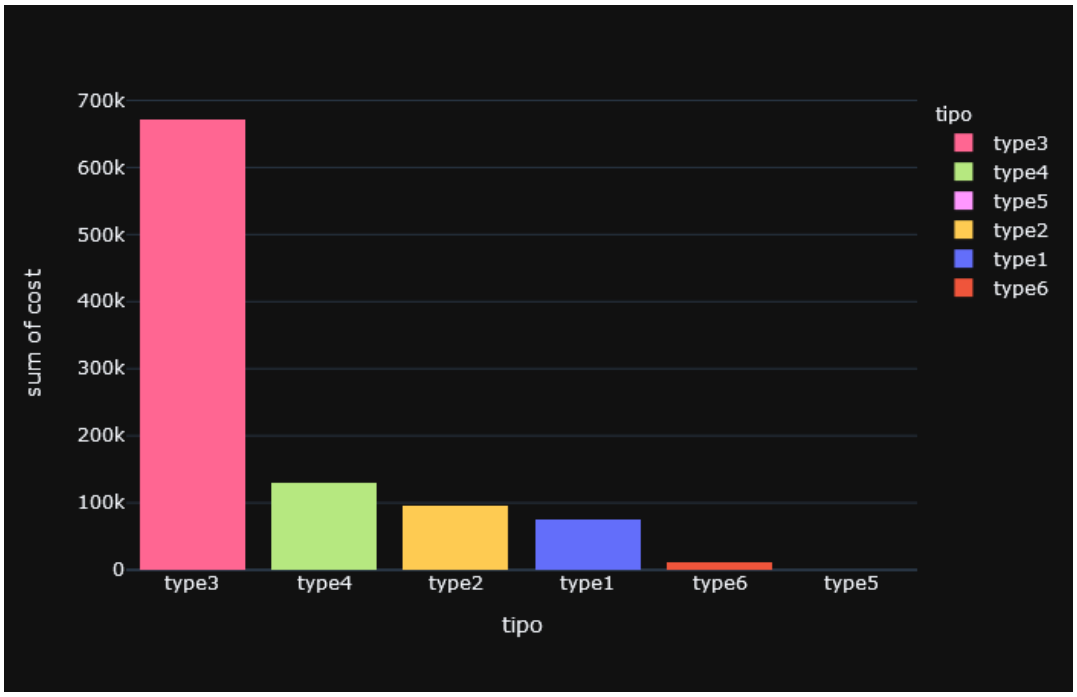

Figura 23 Histograma de distancias por coste

- **Gráfico *Sunburst* por coste y distancia**. Figuras 24 y 25. Este gráfico contiene de forma concentra la distribución del capital a invertir en el proyecto de despliegue. Permite ver facilmente esta distribucion según el tipo, subtipo y dueño del medio de despliegue. Los tamaños relativos de las regiones expresan la cantidad de fibra desplegada (m) y el color expresa el coste del despliegue (k€). Como algunas regiones pueden resultar muy pequeñas para ser leidas correctamente este gráfico ofrece una opción interactiva para filtrar según cualquier categoría. Por ejemplo, si se hace click sobre la region “type2” el gráfico se transforma a través de una animación permitiendo visualizar únicamente los subtipos y dueños dentro de esa categoría.

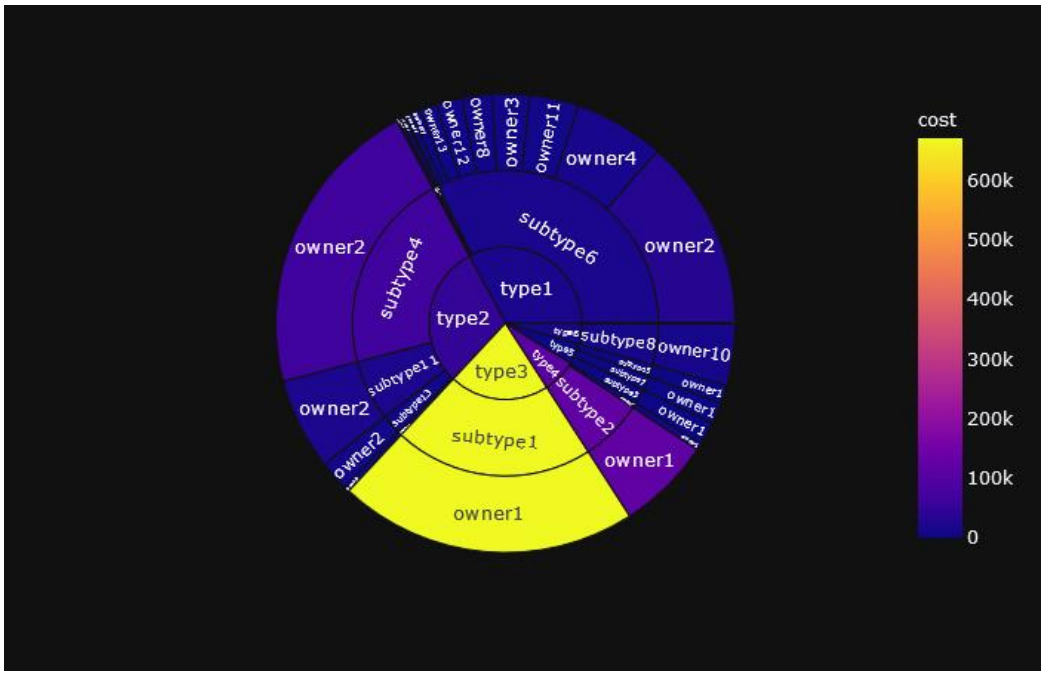

Figura 24 Gráfico Sunburst total

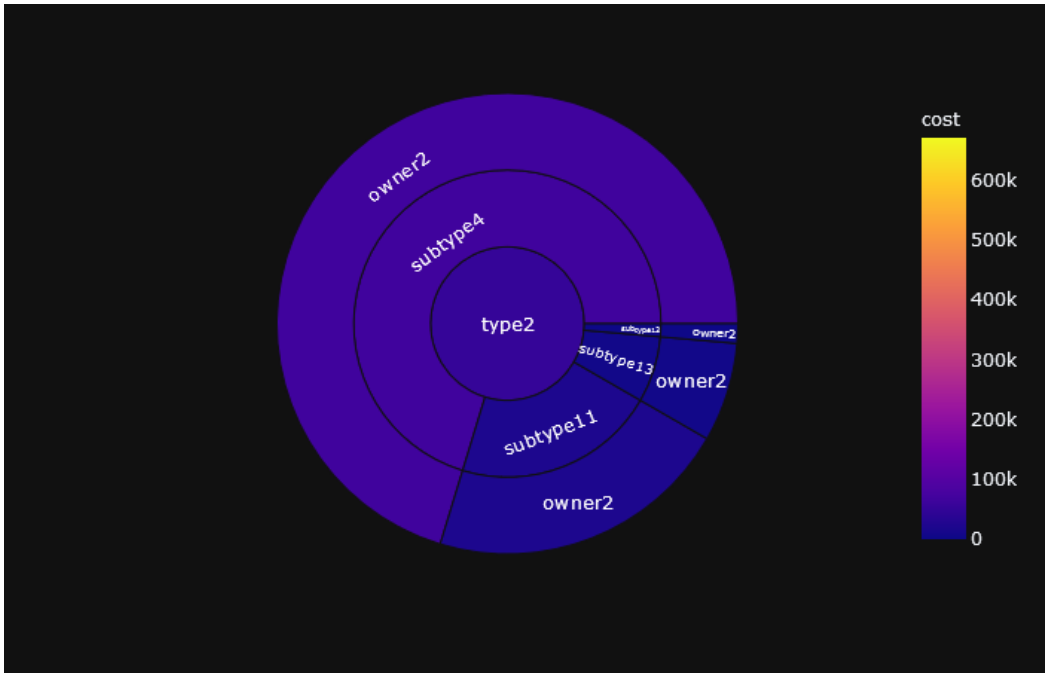

Figura 25 Gráfico Sunburst filtrado por "type2"

- **Gráfico *tree map* por coste y distancia**. Figura 26. Similar al gráfico anterior, permite visualizar las relaciones entre todas las características de un medio de despliegue, su coste y su distancia acumulada. En este gráfico el tamaño de las regiones representadas tiene una relación logarítmica con la distancia desplegada de modo que las regiones más pequeñas siguen resultando visibles. De este modo se puede visualizar de forma general las características generales del proyecto de despliegue y detectar rápidamente cualquier anomalía.

Figura 26 Gráfico Tree Map por coste y distancia

5.4 ARQUITECTURA DE LA SOLUCIÓN

Finalmente, tras detallar cada componente de la solución propuesta en este proyecto, en este apartado se procederá a describir a alto nivel la arquitectura de la plataforma. En la Figura 27 se muestra de forma esquemática la arquitectura y el flujo de datos de la herramienta desarrollada.

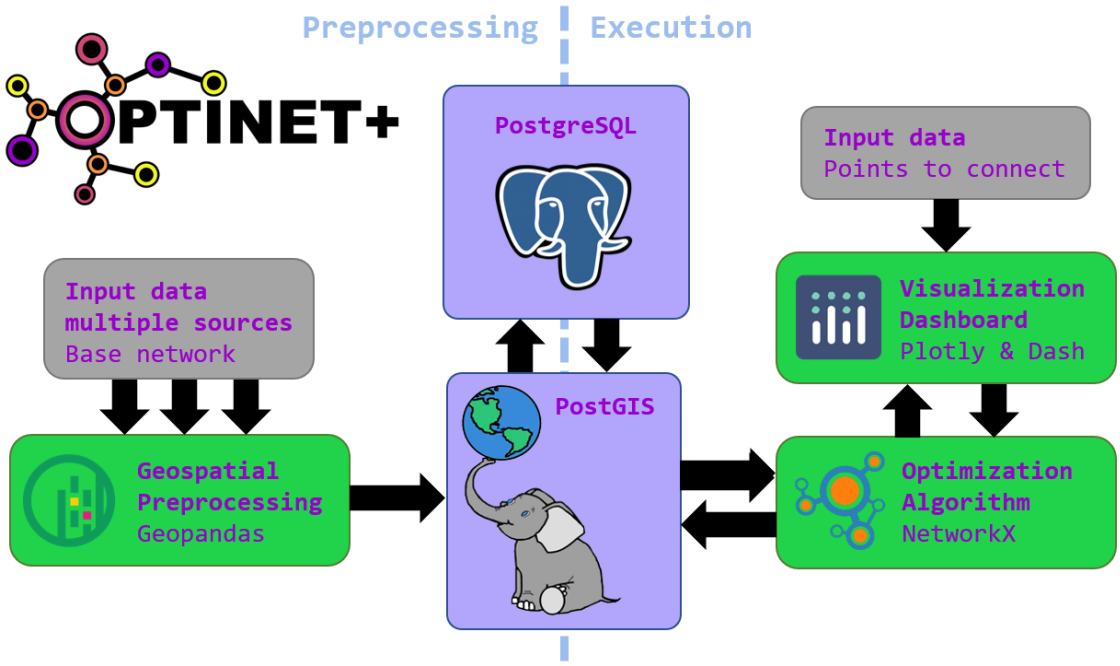

Figura 27 Esquema de la arquitectura de la herramienta

La arquitectura de la plataforma está centrada en el almacenamiento de datos geoespaciales en PostgreSQL/PostGIS. Los elementos mostrados a la izquierda en la figura anterior muestran la fase de preprocesado de las geometrías que representan los medios de despliegue de fibra. Esta parte del proceso sólo es necesaria cuando se reciben nuevos datos de este tipo, lo cual es de esperar que ocurra como mucho semanalmente. Por el otro lado queda representado el flujo de datos de la operativa normal de la plataforma. Esto ocurrirá cada vez que quiera realizar una valoración de un proyecto de despliegue de fibra.

Capítulo 6. ANÁLISIS DE RESULTADOS

En este apartado se analizará el correcto comportamiento de los distintos componentes que forman la infraestructura, siguiendo el orden en que se han desarrollado y descrito en este documento.

6.1 PREPROCESADO DE DATOS GEOESPACIALES

Durante el desarrollo de este proyecto se han utilizado datos geoespaciales correspondientes a los distintos medios de despliegue en la Comunidad de Madrid. Todo el proceso descrito en el apartado de desarrollo supone un tiempo de 87s o 1m 27s. Este es un excelente resultado ya que este sería el cuello de botella en cuanto a escalabilidad en el proyecto. Cuando se amplíen los datos de entrada a todo el territorio nacional se puede esperar un incremento importante (superior a lineal) en este tiempo de procesado. Dado que este proceso no se requiere con frecuencia el sistema podría soportar un incremento en el tiempo de ejecución de varios órdenes de magnitud sin afectar a la operabilidad de la herramienta.

En cuanto a espacio consumido tampoco resulta un problema. Los datos ya procesados, listos para ser consultados por el algoritmo de optimización suponen aproximadamente 1GB, de modo que el paso a nivel nacional no supondrá un problema.

6.2 ALGORITMO DE OPTIMIZACIÓN

Una gran parte del esfuerzo dedicado en este proyecto ha estado enfocado en desarrollar un algoritmo que encontrase soluciones suficientemente óptimas en un tiempo razonable. Este tiempo razonable está definido por el margen que deja para construir una herramienta interactiva a partir del algoritmo. A continuación, se muestra una medida de los tiempos y costes encontrados por los distintos algoritmos desarrollados (ver Tabla 2). Esta medida de coste representa el coste de la solución al ejecutar el algoritmo con datos reales, sin aplicar ninguna limitación de inversión inicial.

Algoritmo	**Tiempo (s)**	**Coste total (M €)**
MST (Minnimun Spanning Tree)	0.16	5.22
TM80 (H. Takahashi & A. Matsuyama)	0.74	4.55
k-LCA (k Loss Contraction Algorithm)	12.32	4.63

Tabla 2 Tiempos de ejecución y costes de la solución encontrada por los distintos algoritmos

De estos resultados se deduce que el algoritmo más adecuado para esta herramienta es TM80, ya que ofrece la mejor aproximación en un tiempo razonable. Es interesante apuntar que k-LCA encuentra puede encontrar soluciones locales (para ramales concretos) mejores o peores que TM80. Aunque en teoría k-LCA ofrece una ratio de aproximación mejor que TM80, dadas las características del grafo esta mejoría rara vez se materializa en una solución local ligeramente mejor que la encontrada por TM80. Adicionalmente un tiempo de 12s no resulta viable ya que en función del límite de inversión que se establezca esta operación tiene que ejecutarse decenas de veces. Por ejemplo, estableciendo un límite de CAPEX de 10.000€, algo normal en un proyecto de FTTO, esta operación se realiza unas 60 veces durante el proceso de podado.

6.3 DASHBOARD DE VISUALIZACIÓN

Esta parte del proyecto se ha visto algo limitada debido al tiempo, sin embargo, se ha llegado a desarrollar un *dashboard* operativo que permite realizar valoraciones y visualizar los resultados, así como exportar los resultados tras modificar los límites de inversión si se consideran satisfactorios.

Gran parte de los trabajos futuros tendrán que enfocarse en ampliar esta parte de la herramienta, sin embargo, se ha considerado que cumple con los requisitos para un PMV.

Capítulo 7. CONCLUSIONES Y TRABAJOS FUTUROS

Se ha desarrollado e implementado una plataforma viable para la valoración de proyectos de despliegue de fibra óptica. Se ha demostrado tanto la viabilidad en las condiciones detalladas en este documento, así como la gran escalabilidad que ofrece la herramienta para futuras ampliaciones que se deseen hacer.

Sin duda alguna, la utilización de esta herramienta supondrá una gran ventaja a la hora de valorar proyectos, ya que permite tener en cuenta grandes cantidades de puntos a conectar. De esta manera, como se ha detallado en la memoria, cuantos más puntos se evalúen simultáneamente más se podrán optimizar las conexiones.

La herramienta desarrollada ofrece una interfaz sencilla e intuitiva que permite realizar estas evaluaciones rápidamente y de una forma más dinámica respecto a los parámetros económicos del proyecto.

A continuación, se enumeran los principales objetivos a partir de ahora para mejorar y ampliar la funcionalidad de la herramienta:

- Agregación de más datos de medios de despliegue. Como ya se ha mencionado el desarrollo en este proyecto se ha hecho sobre datos que representan la Comunidad de Madrid. Un paso fundamental para empezar a utilizar esta herramienta en producción será obtener y agregar los datos de medios de despliegue del resto del territorio nacional.
- Mejoras en la herramienta de visualización. Otro factor importante que hay que tener en cuenta es la versatilidad de la herramienta de visualización. Será necesario añadir más gráficos que permitan realizan un análisis detallado de forma rápida y visual, añadiendo más agilidad al proceso de evaluación.
- Mejoras en el algoritmo de optimización. Aunque el algoritmo en si ya es satisfactorio, se puede seguir mejorando esta etapa del procesado. El principal

objetivo en este ámbito será permitir una granularidad mayor en los parámetros económicos a la hora de calcular la viabilidad de los puntos analizados.

Capítulo 8. BIBLIOGRAFÍA

[1] F. K. Hwang and D. S. Richards. *Steiner tree problems*. Networks 22, 1992. 55-89.

[2] A. B. Kahng and G. Robins, *On Optimal Interconnections for VSLI*, Kluwer publishers 1995.

[3] F. K. Hwang, D. S. Richards, and P. Winter. *The Steiner Tree Problem*, North-Holland, 1992.

[4] Zhang, Lijing and Jing Yi. *Management methods of spatial data based on PostGIS*, Second Pacific-Asia Conference on Circuits, Communications and System 1, 2010. 410-413.

[5] Spatial indexing in PostGIS. https://postgis.net/workshops/postgis-intro/indexing.html

[6] Coordinate reference system ESPG25830. https://spatialreference.org/ref/epsg/25830/

[7] PGRouting PostGIS library. https://pgrouting.org/

[8] NetworkX graph data structure implementation. https://networkx.org/documentation/stable/reference/introduction.html#data-structure

[9] G. Robins and A. Zelikovsky. *Improved Steiner tree approximation in graphs*, SODA 00, 2000. 770-779

[10] S. E. Dreyfus and R. A. Wagner. *The Steiner problem in graphs*, Networks 1, 1972. 195-207.

[11] M. Chimani and M. Woste. *Contraction-Based Steiner Tree Approximations in Practice.* Algorithms and Computation, 2011. 40-49

[12] H. Takahashi and A. Matsuyama. *An approximate solution for the Steiner problem in graphs*. Math. Japonica 24, 1980. 573-577.

[13] P. Winter and J. MacGregor Smith. *Path-distance heuristics for the Steiner problem in undirected networks*. Algorithmica 7, 1992. 309-327.

[14] A. Zelikovsky. An 11/6-aproximation algorithm for the Steiner tree problem in graphs. Information Processing Letters 46, 1993. 79-83.

[15] L. G. Khachiyan. *A polynomial algorithm for linear programming.* Soviet Math. Doklady 20, 1979. 191-194.

[16] J. Byrka, F. Grandoni, T. Rothvoß, and L. Sanità. *An Improved LP-based Approximation for Steiner Tree*. Proc. 42nd STOC, 2010.

[17] R. Gardner. *The Brunn-Minkowski inequality*. Bulletin of the American Mathematical Society 39, 2002. 355-405.