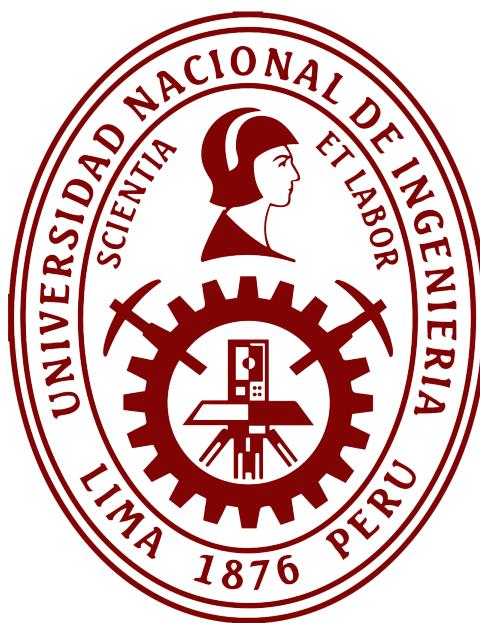


UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE CIENCIAS

**ESCUELA PROFESIONAL DE CIENCIAS DE LA
COMPUTACIÓN**



**EL BIOBJECTIVE SHORTEST PATH PROBLEM APLICADO AL
CÁLCULO DE RUTAS SEGURAS**

PROYECTO DE TESIS III

Autor: Víctor Alberto Ponce Pinedo

Asesor: Gipsy Miguel Arrunátegui Angulo

Diciembre, 2021

RESUMEN

PALABRAS CLAVE: SEGURIDAD, OPTIMIZACIÓN, ESTIMACIÓN, DENSIDAD,
GEOESPACIAL

El presente trabajo abarca un análisis y comparación de algoritmos orientados al cálculo de rutas seguras. Para ello se implementa un modelo de riesgo que nos permita obtener un grafo cuyas aristas poseen pesos de longitud y de riesgo y se implementaran un algoritmo de enrutamiento que reciba como entrada dicho grafo y traten de minimizar ambos pesos en la medida que sea posible. También se implementará una pequeña modificación a dicho algoritmo que permitan obtener resultados mucho más prácticos en términos de tiempo de ejecución y cantidad de soluciones obtenidas.

ÍNDICE

RESUMEN	1
I. Introducción	1
1.1. Motivación	1
1.2. Objetivos	2
1.2.1. Objetivo general	2
1.2.2. Objetivos específicos	2
1.3. Estructura del seminario	3
II. Estado del arte	5
2.1. Trabajos orientados directamente al cálculo de rutas seguras	5
2.1.1. Kernel Density Estimation vs. Hot-Spot Analysis - Detecting Criminal Hot Spots in the City of San Francisco	5
2.1.2. Beyond the Shortest Route: A Survey on Quality-Aware Route Naviga- tion for Pedestrians	6
2.1.3. CROWDSAFE: crowd sourcing of crime incidents and safe routing on mobile devices	6
2.1.4. Urban navigation beyond shortest route: The case of safe paths	7
2.1.5. A Mobile Information System Based on Crowd-Sensed and Official Crime Data for Finding Safe Routes: A Case Study of Mexico City	7
2.1.6. Route-The Safe: A Robust Model for Safest Route Prediction Using Crime and Accidental Data	8

2.1.7. SafeRoute: Learning to Navigate Streets Safely in an Urban Environment	9
2.1.8. Safe and Sound: Driver Safety-Aware Vehicle Re-Routing Based on Spatiotemporal Information	9
2.1.9. Better safe than sorry: a vehicular traffic re-routing based on traffic conditions and public safety issues	10
2.2. Trabajos orientados a la optimización de rutas cortas con aristas con peso bivariante	10
2.2.1. Aceleración de algoritmos de etiquetado orientados al Biobjective Shortest Path Problem	11
2.2.2. Un método exacto para el Biobjective Shortest Path Problem aplicable a grafos de gran envergadura	12
2.2.3. Algoritmo basado en el algoritmo de Dijkstra para el Biobjective Shortest Path Problem	12
2.2.4. Un algoritmo simple y rápido para el problema de la búsqueda biobjetivo	12
2.2.5. Búsqueda de doble criterio aproximada usando una representación eficiente de subconjuntos de la frontera de Pareto [21]	13
2.3. Conclusiones	13
III. Marco teórico	15
3.1. Estimación de densidad	15
3.1.1. Estimación paramétrica	15
3.1.2. Estimación no paramétrica	16
3.2. Kernel Density Estimation	16
3.3. Problemas de optimización combinatoria de criterio múltiple	18
3.4. Biobjective Shortest Path Problem	21
3.5. Mapeo de zonas de riesgo	22
3.6. Hot Spot	22

3.6.1. Definición	22
3.6.2. Tipos de Hot Spots	22
IV. Modelo de riesgo	24
V. Algoritmo para la resolución del BSPP	26
VI. Resultados Experimentales	30
6.1. Descripción general del trabajo	30
6.2. Descripción de los datos utilizados	30
6.3. Preparación de los grafos para los experimentos	33
6.4. Generación de los experimentos para la evaluación de los algoritmos	34
6.5. Resultados obtenidos	36
6.5.1. Experimentos de las categorías 1 a 7	36
6.5.2. Experimentos de las categorías 10 y 15	40
VII Conclusiones y trabajo futuro	45
7.1. Conclusiones	45
7.2. Trabajo futuro	46
REFERENCIAS	49

Índice de figuras

2.1. Arquitectura usada en el proyecto Safe and Sound [7]	10
2.2. Arquitectura usada en el proyecto Better safe than sorry [6]	11
3.1. Ilustración de la frontera de Pareto en un problema de optimización de doble criterio [10]	20
3.2. Ilustración de los efectos de la dominancia epsilon [19]	21
4.1. Geometría de una de las calles del grafo	24
5.1. Grafo de ejemplo donde se visualiza la función heurística empleada [28]	27
6.1. Resumen general del presente proyecto	31
6.2. Crímenes de San Francisco.	31
6.3. Crímenes de Chicago.	32
6.4. Modelo de riesgo de San Francisco.	34
6.5. Modelo de riesgo de Chicago.	35
6.6. Tiempos de ejecución promedio en segundos afectados por el epsilon en San Francisco y Chicago	37
6.7. Tiempos de ejecución máximos en segundos afectados por el epsilon en San Francisco y Chicago	37
6.8. Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago	38

6.9. Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$	39
6.10. Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago	39
6.11. Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$	40
6.12. Nodos expandidos en promedio afectados por el epsilon en San Francisco y Chicago	40
6.13. Tiempos de ejecución promedio en segundos afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos	41
6.14. Tiempos de ejecución máximos en segundos afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos	42
6.15. Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos	42
6.16. Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$ en la segunda ronda de experimentos	43
6.17. Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos	43
6.18. Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$ en la segunda ronda de experimentos	44
6.19. Nodos expandidos en promedio afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos	44

Lista de Algoritmos

1.	Bi-Objective A* (BOA*)	28
----	----------------------------------	----

Lista de acrónimos

BSPP Biobjective Shortest Path Problem.

KDE Kernel Density Estimation.

AGRADECIMIENTOS

Agradezco infinitamente a mis padres y a mi hermana que me han apoyado constantemente en estos tiempos en los que reina la incertidumbre. De igual manera le agradezco al profesor Miguel Arrunátegui que estuvo apoyándome y asesorándome constantemente durante todo el desarrollo de este trabajo.

Capítulo I

Introducción

El presente seminario tiene como objetivo la implementación y análisis de un algoritmo moderno orientados al Biobjective Shortest Path Problem (BSPP) aplicado al contexto del cálculo de rutas seguras, es decir, cálculo de rutas con un bajo índice de riesgo.

Para lograr esto se implementará un modelo de riesgo de una ciudad que nos permita construir un grafo que representará a dicha ciudad para posteriormente asignarles un puntaje de riesgo a cada una de las aristas del grafo que representen las peligrosidades de cada una de las calles. Luego de eso se implementará un algoritmo moderno que calcule rutas que traten de balancear tanto su peligrosidad como su longitud. Finalmente se evaluará el impacto de una pequeña modificación a dicho algoritmo analizando los tiempos de ejecución, la cantidad de soluciones y la cantidad de nodos expandidos mediante sendos experimentos.

1.1. Motivación

Hoy en día existen una gran variedad de aplicaciones que le permiten a un usuario obtener la ruta más corta desde un lugar a otro. Sin embargo, durante viajes que he realizado en vehículos de taxistas que usan dichos aplicativos los he escuchado quejarse constantemente de que estos siempre les recomiendan rutas que si bien optimizan la distancia de viaje no optimizan el riesgo

de estas rutas por lo que los conducen a lugares en los que es bastante probable que sean víctimas de algún delito. Un ejemplo particular de esto se puede ver en [5] en el cual se comenta sobre un incidente en el que el aplicativo Waze calculó una ruta que provocó que sus usuarios sean conducidos a una zona bastante peligrosa y en la que se estaba sucediendo un tiroteo. Este tipo de eventos son los que motivan a un investigador a encontrar técnicas que permitan establecer un modelo de riesgo de una ciudad y que dicho modelo de riesgo pueda ser consumido por algoritmos de enrutamiento que traten de calcular rutas que sean lo más seguras y cortas posibles, problema que se estudiará en el presente seminario.

1.2. Objetivos

1.2.1. Objetivo general

El objetivo general de este seminario es la implementación y evaluación empírica de un algoritmo moderno que resuelve el Biobjective Shortest Path Problem aplicado en el contexto del cálculo de rutas seguras y los efectos de una pequeña modificación a dicho algoritmo.

1.2.2. Objetivos específicos

- Obtención del mapa de dos ciudades en un formato que sea fácilmente manipulable mediante un lenguaje de programación.
- Identificación de las zonas de alto riesgo de las ciudades estudiadas mediante una técnica de estimación de densidad.
- Generación del modelo de riesgo de las ciudades estudiadas.
- Implementación del algoritmo que permitan calcular rutas en las ciudades que balanceen lo mejor posible su longitud y peligrosidad.
- Evaluar el impacto de una modificación al algoritmo con respecto a los tiempos de ejecución, la cantidad de soluciones obtenidas y la cantidad de nodos expandidos.

1.3. Estructura del seminario

Para que el lector tenga más clara la estructura de este documento explicaremos brevemente cada uno de sus capítulos:

- **Introducción:**

Se presenta la motivación de este trabajo así como el objetivo general y los objetivos específicos.

- **Estado del arte:**

Se presentan algunos trabajos relacionados al tema del presente seminario para que el lector conozca los trabajos realizados entorno al cálculo de rutas seguras así como el desarrollo de técnicas que resuelven el Biobjective Shortest Path Problem.

- **Marco teórico:**

Se presentan algunos conceptos importantes para que el lector comprenda el trabajo del presente seminario de la mejor manera.

- **Modelo de riesgo:** En este capítulo se abordará el uso de la técnica Kernel Density Estimation mencionada en el marco teórico para la elaboración de ecuaciones que nos permitan la implementación del modelo de riesgo de un grafo que representa a las ciudades en análisis.

- **Algoritmo para la resolución del BSPP.** En este capítulo se mostrará y explicará en líneas generales el algoritmo que trata de calcular rutas que optimicen la seguridad y la distancia y se mencionará una pequeña modificación del algoritmo para obtener resultados más prácticos en términos de tiempos de ejecución y cantidad de soluciones.

- **Resultados Experimentales:**

En este capítulo se explica como se ha construido los experimentos que nos permitirán la evaluación de nuestro algoritmo de enrutamiento, se mostrarán los resultados obtenidos y se tratará de extraer conclusiones de dichos resultados.

■ Conclusiones y trabajo futuro:

En este capítulo se exponen las conclusiones obtenidas de este trabajo. También se proponen trabajos futuros para mejorar los resultados obtenidos así como consolidar los detalles que quedaron pendientes.

Capítulo II

Estado del arte

En este capítulo se abarcarán trabajos realizados durante los últimos años en el área del cálculo de rutas. Los trabajos abarcan no solamente el cálculo de rutas seguras sino también rutas que cumplan con otros criterios tales como evitar el ingreso a zonas excesivamente congestionadas de autos, zonas de alta probabilidad de accidentes automovilísticos, entre otros criterios. Tras la explicación de los trabajos en la sección final se expondrán las conclusiones extraídas como resultado de su estudio.

2.1. Trabajos orientados directamente al cálculo de rutas seguras

2.1.1. Kernel Density Estimation vs. Hot-Spot Analysis - Detecting Criminal Hot Spots in the City of San Francisco

En este trabajo se analizan y comparan dos métodos para determinar zonas de alta concentración de crimen o hot spots: KDE y Getid Ord Gi*. Para evaluar como se desempeñan estas técnicas los autores usaron un conjunto de datos geoespaciales de crímenes que corresponden a la ciudad de San Francisco y estudiaron que tan capaces son estas técnicas de realizar la detección de los hot spots. Los autores concluyen que KDE te permite ubicar donde se encuentran

los hot spots pero que estos no poseen una gran importancia estadística, Getis Ord Gi* permite localizar mejor zonas de gran concentración de crimen más no distingue zonas de baja concentración de crimen y finalmente que para la obtención de mejores resultados lo más conveniente es el uso de estas dos técnicas de manera conjunta [14].

2.1.2. Beyond the Shortest Route: A Survey on Quality-Aware Route Navigation for Pedestrians

En este trabajo, tal y como dice su nombre, los autores realizan una revisión de la literatura para recopilar investigaciones acerca de sistemas de navegación que permitna calcular rutas que cumplan no solo con ser lo más cortas posibles o cuyo tiempo de viaje sea el menor posible sino que también otros tipos de criterios tales como rutas de riesgo mínimo en lo que respecta a crímenes y/o accidentes automovilísticos. Tambien detallan para dichos criterios que tipos de datos se usaron para modelar adecuadamente el sistema de navegación y como procedieron a evaluar sus implementaciones [26]. Este trabajo resultó de mucha utilidad para el presente proyecto dado que menciona la existencia de OpenStreetmap [18] y servicios de datos abiertos como los de la web de Chicago [4] para la obtención de mapas y datos de eventos de crímenes respectivamente.

2.1.3. CROWDSAFE: crowd sourcing of crime incidents and safe routing on mobile devices

En este trabajo se presenta uno de los primeros esfuerzos realizados para el cálculo de rutas seguras . Aquí los autores presentan un sistema completo compuesto por una base de datos geoespacial, un servirdor para la lógica de negocio y un aplicativo móvil que permite a sus usuarios la visualización de las zonas más peligrosas de su ciudad, el reporte de crimen y el cálculo de rutas seguras. Para el desarrollo de esta última funcionalidad, el modelo de riesgo utilizado por los autores fue de realizar el conteo de crímenes en un radio determinado de cada una de las calles de la ciudad de modo que las cantidades calculadas servirán como pesos en un grafo cuyas

aristas contendrán como atributos tanto la longitud de la calle como la cantidad de crímenes calculadas. Sin embargo, para poder lograr el objetivo de obtener rutas lo más cortas y seguras posibles se propone un nuevo peso en el grafo que trate de combinar los dos pesos mencionados anteriormente y con este nuevo peso se aplica el algoritmo de Dijkstra para el cálculo de la ruta final [25]. Cabe recalcar que dicho algoritmo de Dijkstra no fue implementado por los autores sino que se encuentra implementado en el gestor de datos geoespaciales PostGIS, ya que toda la información de la ciudad se encuentra almacenada en una base de datos PostgreSQL.

2.1.4. Urban navigation beyond shortest route: The case of safe paths

En este trabajo los autores desarrollaron un modelo de riesgo de una ciudad y utilizaron un algoritmo orientado al Biobjective Shortest Path Problem para calcular rutas que sean lo más cortas y seguras posibles. En lo que respecta al modelo de riesgo de la ciudad los autores descargaron datos de OpenStreetMap y mediante una herramienta llamada osm4routing obtuvieron las ciudades en forma de grafos. Después de obtener dichos grafos utilizaron una técnica estadística no paramétrica llamada Kernel Density Estimation que les permite asignar un puntaje de riesgo a cada una de las calles de la ciudad que vendría a representar la probabilidad de que en dicha calle ocurra un determinado crimen. Una vez que obtuvieron los grafos con las aristas etiquetadas tanto con su longitud como con su peligrosidad diseñan un algoritmo al cual llaman **TotalPaths** que les permitiría ir calculando soluciones pareto-óptimas que balanceen la peligrosidad de las rutas así como su longitud. [1]

2.1.5. A Mobile Information System Based on Crowd-Sensed and Official Crime Data for Finding Safe Routes: A Case Study of Mexico City

En este trabajo los autores desarrollan un aplicativo móvil que posee la funcionalidad del cálculo de rutas seguras en Mexici City. Lo novedoso en este trabajo es que para el modelo de riesgo de la ciudad los investigadores emplean no solamente datos geoespaciales de crímenes, sino también tweets que son analizados mediante el uso de técnicas de procesamiento de len-

guaje natural para la extracción de información de los mismos. Una vez que se haya extraído toda la información necesaria de los tweets y de los datos geoespaciales y que se tenga finalmente una base de datos compuesta por diversos crímenes y la ubicación geográfica que se ha realizado se implementa el modelo de riesgo de la ciudad, el cual consiste básicamente en asignarle a cada calle de la ciudad un puntaje de riesgo igual a la cantidad de crímenes ocurridos en un radio determinado. Con estos scores que funcionarán como pesos en un grafo se realiza la búsqueda de la ruta con menos riesgo mediante el algoritmo de Dijkstra y la ruta devuelta por dicho algoritmo será la que visualice el aplicativo móvil [16].

2.1.6. Route-The Safe: A Robust Model for Safest Route Prediction Using Crime and Accidental Data

En este trabajo los autores presentan un aplicativo móvil que permita calcular rutas seguras para sus usuarios. Sin embargo, a diferencia de los trabajos enteriores que calculaban las rutas prácticamente desde cero, en este trabajo lo que se hace es utilizar el servicio DirectionService de Google [13] el cual te proporciona muchas rutas distintas desde un lugar a otro y determinar cual de estas rutas es la más segura. Para poder determinar la ruta más segura lo que se hace en primera instancia es determinar los hot spots de una ciudad (en este caso se trabaja con información de crímenes de la ciudad de Nueva York) mediante el uso de algoritmos de clusterización y una vez que se han determinado los hot spots se usan algoritmos de clasificación que permiten determinar la ruta más segura que nos haya proveído el servicio DirectionService [27]. La propuesta luce interesante de implementar, sin embargo los resultados no resultan demasiado atractivos ni fáciles de interpretar por lo que no fue muy tomado muy en cuenta en el presente seminario. Aún así constituye un esfuerzo más en el área de algoritmos para determinar rutas seguras.

2.1.7. SafeRoute: Learning to Navigate Streets Safely in an Urban Environment

En este trabajo los autores proponen un enfoque novedoso para ese entonces con el objetivo de la resolución del problema del cálculo de rutas con criterios múltiples. En este caso los criterios son solo dos: la optimización tanto de la distancia de la ruta calculada como del riesgo de la misma. Para ello los autores proponen un enfoque basado en Deep Learning, más específicamente Deep Reinforcement Learning. Este tipo de aprendizaje profundo está basado en darle incentivos a una red neuronal durante su proceso de aprendizaje de modo que con el paso de los entrenamientos vaya calculando rutas entre dos puntos que optimicen la longitud y la seguridad de dichas rutas de la mejor manera posible [15].

2.1.8. Safe and Sound: Driver Safety-Aware Vehicle Re-Routing Based on Spatiotemporal Information

En este trabajo los autores llevan el concepto de enrutamiento mucho más allá ya que a diferencia de la mayoría de trabajos hasta ahora explicados cuyo propósito general es brindarte una ruta que cumpla con un determinado criterio, este trabajo también abarca otras tecnologías tales como Cloud Computing o incluso el internet de las cosas de modo que se desarrolle un sistema de enrutamiento dinámico capaz de ser “consciente” de las condiciones del tráfico de la ciudad y que los vehículos partícipes de dicho sistema se comuniquen entre ellos y en base a los mensajes que se reciben se ejecutan algoritmos que te redirijan de modo que la ruta cumpla con un criterio determinado. La arquitectura general del sistema completo puede verse en la figura 2.1. Algo interesante que se propone en este trabajo es el uso de redes neuronales recurrentes para la predicción tanto de crímenes como de tráfico, criterios usados por su sistema para el cálculo de sus rutas [7].

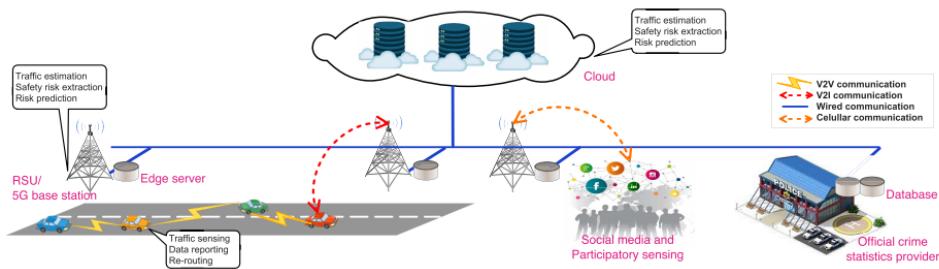


Figura 2.1: Arquitectura usada en el proyecto Safe and Sound [7]

2.1.9. Better safe than sorry: a vehicular traffic re-routing based on traffic conditions and public safety issues

Este es otro trabajo en el que los autores proponen un sistema de enrutamiento dinámico con entidades conscientes del contexto, es decir, vehículos que van a estar comunicándose entre sí constantemente de modo que se conozca con el mayor detalle posible el estado del lugar en el que se están movilizando. La arquitectura general del sistema puede verse en la figura 2.2. Con respecto al enrutamiento el desafío es el mismo que entodos los trabajos hasta ahora abarcados, es decir, el cálculo de rutas que tratan de optimizar múltiples criterios (optimización multiobjetivo). Para este trabajo los autores tomarán como criterios las condiciones de tráfico y de seguridad ciudadana. Con respecto al tráfico su monitoreo se realiza gracias a la comunicación constante de los vehículos que conforman el sistema y para el aspecto de la seguridad se manipulará información de redes sociales así como bases de datos de crímenes oficiales de las fuerzas del orden [6].

2.2. Trabajos orientados a la optimización de rutas cortas con aristas con peso bivariable

La optimización de rutas cortas con aristas con peso bivariable, tambien conocido como Biobjective Shortest Path Problem, ha gozado de bastante interés por parte de los investigadores en la última década. Existen adaptaciones de algoritmos tradicionales como Dijkstra y DFS así

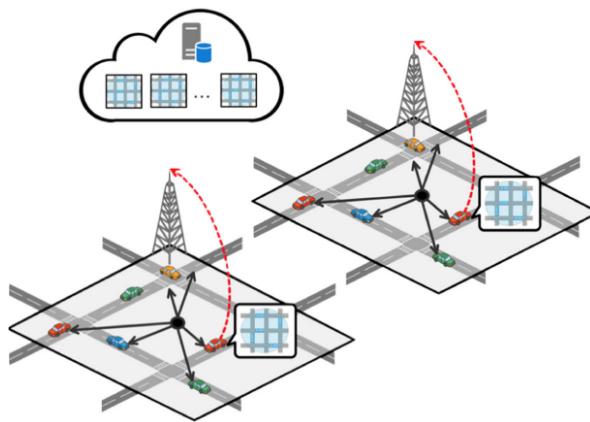


Figura 2.2: Arquitectura usada en el proyecto Better safe than sorry [6]

como adaptaciones de algoritmos basados en heurísticas tales como el famosísimo A estrella. En esta sección exploraremos en orden cronológico los trabajos que más han influenciado y destacado en el estudio y resolución del Biobjective Shortest Path Problem.

2.2.1. Aceleración de algoritmos de etiquetado orientados al Biobjective Shortest Path Problem

En este trabajo los autores proponen un método para la aceleración de algoritmos que resuelven el Biobjective Shortest Path Problem. Dicha técnica consiste en un podado que se hace etiquetando a los vértices de un grafo con pares de la forma (c_1, c_2) que vendría a representar el costo de cada una de las rutas que llega hasta cada uno de los vértices y posteriormente esas etiquetas se usarán para podar rutas que hallan superado lexicográficamente el costo de alguno de los labels. Los autores también establecen una comparación de diferentes estructuras de datos tales como los Fibonacci heaps, el Binary Heap y los Buckets y su incidencia en los tiempos de ejecución [20].

2.2.2. Un método exacto para el Biobjective Shortest Path Problem aplicable a grafos de gran envergadura

En este trabajo que para el momento de su publicación fue el estado del arte en lo que respecta al cálculo de todas las soluciones de la frontera de Pareto los autores desarrollan un algoritmo basado en el algoritmo DFS y le aplicaron varias técnicas de podado para detener el cálculo de rutas que no pertenezcan a la frontera de Pareto y comparan sus resultados con los obtenidos en [20].

2.2.3. Algoritmo basado en el algoritmo de Dijkstra para el Biobjective Shortest Path Problem

En este trabajo los autores adaptan el algoritmo de Dijkstra para el Biobjective Shortest Path Problem y lo comparan con los resultados obtenidos en [8] superándolos y convirtiéndose en el estado del arte para la fecha de su publicación.

2.2.4. Un algoritmo simple y rápido para el problema de la búsqueda biobjetivo

[28] Este es uno de los algoritmos más modernos que están orientados al Biobjective Shortest Path Problem y lo novedoso de este algoritmo es que a diferencia de los anteriores trabajos este está basado en el algoritmo A estrella y se apoya en una heurística consistente bastante genérica que propone el autor con la cual obtiene resultados bastante competitivos superando al Dijkstra biobjetivo presentado en [24] y me atrevería a decir que de momento este es el nuevo algoritmo estado del arte para el Biobjective Shortest Path Problem. De momento hay algunos trabajos que se están apoyando en este, por ejemplo hay un trabajo que lo convierte en bidireccional para tratar de acelerar la búsqueda y existe otro trabajo que se explicará en la siguiente subsección el cual compute un subconjunto de la frontera de Pareto.

2.2.5. Búsqueda de doble criterio aproximada usando una representación eficiente de subconjuntos de la frontera de Pareto [21]

Este trabajo es de los más recientes en torno al Biobjective Shortest Path Problem, está basado fuertemente en el trabajo presentado en [28] y lo que proponene es un nuevo algoritmo y una nueva estructura de datos para calcular un subconjunto de soluciones de la frontera de Pareto y es precisamente eso lo que lo distingue del algoritmo **BOA-star** presentado en [28] ya que estos últimos calculan la frontera de Pareto completa. Para hacer una comparación justa los autores establecen una pequeña variante al algoritmo **BOA-star** por medio de únicamente dos líneas de código modificadas con respecto al código original y lo comparan con el algoritmo que ellos mismos desarrollaron aparte y utilizando diversos factores de aproximación que sirven para determinar que tanto queremos restringir a nuestro conjunto solución resultante.

2.3. Conclusiones

En lo que respecta estrictamente al cálculo de rutas seguras no es mucho lo que se ha encontrado y por lo general solo los trabajos más antiguos son los que se enfocan a un modelo de riesgo estático de una ciudad mientras que trabajos más modernos toman en cuenta otros modelos de riesgo, toman en cuenta el tráfico de una ciudad y son diseñador para sistemas IoT en los cuales los vehículos son conscientes del contexto en el que se desenvuelven por medio de sensores y satélites. De los trabajos más confiables y entendibles que he podido estudiar con respecto a modelos de riesgo tenemos al desarrollado por la doctora Galbrun y sus colegas en [1] y por ello en este seminario implementaremos dicho modelo de riesgo en dos ciudades para obtener los grafo que utilizaremos con el fin de evaluar a nuestros algoritmos de enruteamiento. Ahora, dado que el modelo que hemos escogido nos permite enfocar el problema del cálculo de rutas seguras como un Biobjective Shortest Path Problem, esto nos da la posibilidad de estudiar y evaluar a fondo los mejores algoritmos propuestos en torno a este problema, más específicamente en el presente seminario evaluaremos los algoritmo BOA* propuesto en [28]

y la modificación a BOA* propuesta en [21] y sacaremos conclusiones del impacto de dicha modificación con una gran cantidad de experimentos.

Capítulo III

Marco teórico

3.1. Estimación de densidad

Consideremos una variable aleatoria X que tiene como función de densidad a una función f . Dicha función nos permite conocer las propiedades asociadas a ciertos intervalos en los que reside nuestra variable aleatoria X y esto se rige mediante la siguiente ecuación:

$$P(a \leq X \leq b) = \int_a^b f(x)dx$$

Ahora, supongamos que tenemos en nuestro poder un conjunto de puntos o datos y que dichos datos poseen una función de densidad desconocida que nos permita describir su comportamiento. La estimación de densidad consiste en calcular una estimación de dicha función de densidad a partir de los datos con los que se cuentan. En cuanto a las técnicas existentes para la estimación de dicha función de densidad, estas se pueden dividir en dos categorías que se detallan a continuación.

3.1.1. Estimación paramétrica

En este tipo de técnicas de estimación lo que se hace es asumir una función de densidad de probabilidad que describa nuestra data. Una vez que se escoge dicha función se procede a la

búsqueda de los parámetros inherentes de dicha función para describir el comportamiento de los datos de la mejor manera posible. Por ejemplo, si escogieramos la función de distribución gaussiana los parámetros que tendríamos que definir son ma media y la varianza.

3.1.2. Estimación no paramétrica

En esta tipo de técnicas no es necesario asumir una fucnión de probabilidad que describa nuestro conjunto de datos, sino que a partir de ellos se tratará de construir nuestra propia función de probabilidad. Una de estas técnicas y dicho sea de paso es ampliamente usade es el Kernel Density Estimation (KDE) y que se explicará en la siguiente sección.

3.2. Kernel Density Estimation

También conocida por sus siglas KDE, esta es una técnica de estimación de densidad no paramétrica que nos ayuda a construir la función de densidad de probabilidad de un conjunto de datos mediante lo que se denomina una función Kernel que será escogida por el experimentor y una variable denominada bandiwth (ancho de banda). Por ejemplo, la fórmula para esta técnica en el caso de que se posea una única variable aleatoria es la siguiente:

$$f(x) = \frac{1}{nh} \sum_{i=1}^n K\left(\frac{x - x_i}{h}\right) \quad (3.1)$$

En la ecuación 3.1 n es la cantidad de datos que se tiene, h es el ancho de banda, x_i es uno de los datos con los que se cuenta y K es cualquier función de distribución conocida. Entre los kernels más usados tenemos:

- Distribución uniforme: $f(x) = U(-1, 1)$
- Distribución triangular: $f(x) = (1 + |x|)_+$
- Distribución de Epanechnikov: $f(x) = \frac{3}{4}(1 - x^2)_+$
- Distribución de peso doble: $f(x) = \frac{15}{16}(1 - x^2)_+^2$

- Distribución de peso triple: $f(x) = \frac{35}{32}(1 - x^2)_+^3$
- Distribución normal: $f(x) = N(0, 1)$

Cabe recalcar que los kerneles mencionados trabajan (al menos en el caso de una sola variable aleatoria) en el intervalo $[-1, 1]$.

Ahora, si se desea trabajar con más de una variable aleatoria se tiene que modificar la ecuación mostrada en 3.1, de modo que la ecuación correspondiente al Kernel Density Estimation multivariable [11] vendría dada por:

$$f(x) = \frac{1}{n|H|^{\frac{1}{2}}} \sum_{i=1}^n K(H^{\frac{-1}{2}}(x - X_i)) \quad (3.2)$$

En esta última ecuación n vendrían a ser la cantidad de datos usados para la construcción de nuestra función de densidad, H vendría a ser la matriz de ancho de banda. Esta matriz tiene que ser simétrica, definida positiva y con la misma dimensión que la cantidad de variables aleatorias que manejemos. Por último K es una función que vendría a describir el kernel utilizado para construir nuestra función de densidad.

Como se puede apreciar en las ecuaciones 3.1 y 3.2, además del Kernel tambien tiene que escogerse el valor del o de los anchos de banda y para ellos existen criterios a los que se les suele denominar **rules of thumb** [23]. Entre estos criterios tenemos:

- Regla de Scott: en este método la matriz de ancho de banda estaría determinada por una multiplicación entre un factor denominado el **factor de Scott** y la matriz de covarianza de nuestras variables aleatorias. El factor de Scott viene dado por la ecuación $n^{\frac{-1}{d+4}}$, donde n es la cantidad de datos y d es la dimensionalidad de los datos.
- Regla de Silverman: esta regla es similar a la regla de Scott y la única diferencia es que el factor de Scott (ahora llamado **factor de Silverman**) vendría dado por la ecuación $\frac{n(d+2)^{\frac{-1}{d+4}}}{4}$

En particular para el desarrollo de este proyecto se usará la regla de Scott que ya viene implementada en la librería Scipy de Python. Esta librería y su implementación del Kernel Density

Estimation fueron utilizados en [1] para el desarrollo de su modelo de riesgo.

3.3. Problemas de optimización combinatoria de criterio múltiple

Los problemas de optimización combinatoria se pueden definir de manera general por la siguiente ecuación:

$$\min_{x \in \chi} \{f(x)\} \quad (3.3)$$

En dicha ecuación $x = (x_1, \dots, x_n)$ donde x vendría a ser un vector de decisión, $\chi \subseteq \mathbf{R}^m$ vendría a ser el espacio de todos los vectores de decisión admisibles y $f : \chi \rightarrow \mathbf{R}$ es una función de criterios que mapea una solución en el espacio de vectores de decisión admisible a un valor escalar. Ahora, los **problemas de optimización combinatoria con criterio múltiple** difieren en que nuestra función de criterios ahora es de la forma $f : \chi \rightarrow \mathbf{R}^k$ donde $k \geq 2$ haciendo que nuestro espacio de soluciones admisibles sea multidimensional. Ahora, algo importante que recalcar es que a diferencia de los problemas con espacio de soluciones admisibles de dimensión en el cual podemos hablar de soluciones óptimas máximas o mínimas, si tenemos un espacio de soluciones multidimensional se hace necesario definir otras nociones de optimalidad. A continuación se presentaran unas definiciones relacionadas al concepto de **dominancia** y finalmente hablaremos de un concepto relacionado a la optimalidad llamado **eficiencia de Pareto** [17].

Definición 3.3.1. Sean $x_1, x_2 \in \chi$ dos soluciones en el espacio de soluciones admisibles, entonces x_2 se dice que está **dominado** por x_1 si

$$f(x_1) \leq f(x_2) \text{ y } f(x_1) \neq f(x_2)$$

Esto quiere decir que x_1 debe ser distinto y menor que x_2 en al menos uno de todos los criterios

u objetivos evaluados. Esta relación de dominancia se escribe de la siguiente manera:

$$x_1 \prec x_2$$

Ahora, en base a esta definición podemos pasar a definir la optimalidad de Pareto y la frontera de Pareto:

Definición 3.3.2. Una solución $x \in \chi$ se dice que es **Pareto óptima** si no existe ninguna solución $\bar{x} \in \chi$ tal que $\bar{x} \prec x$.

Finalmente, todas las soluciones que no sean dominadas por ninguna otra solución conforman la denominada **frontera de Pareto**. Una ilustración de este concepto puede verse en la figura 3.1 en la cual se ven dos puntos negros resaltados y solo el punto que está más a la izquierda es el que no es dominado por ningún otro. En este caso la frontera de pareto consiste en las curvas pequeñas que están resaltadas.

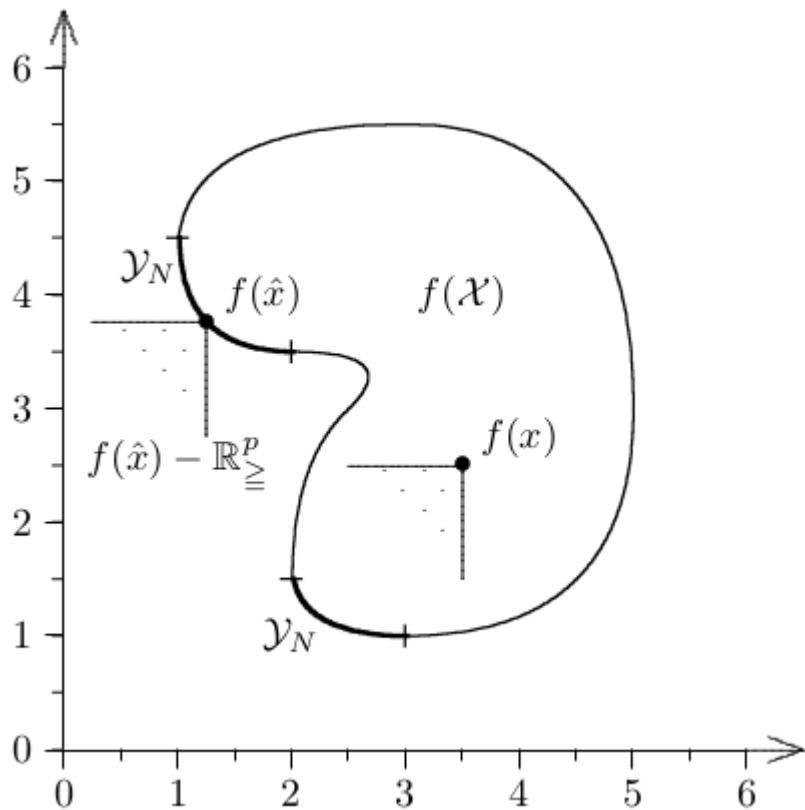


Figura 3.1: Ilustración de la frontera de Pareto en un problema de optimización de doble criterio [10]

Estas son las definiciones clásicas respecto a la dominancia, sin embargo, exista una pequeña variante de esta definición que fue usada en [21] para calcular solo una parte de la frontera de Pareto de su problema. Esta definición se presenta a continuación:

Definición 3.3.3 (La dominancia ϵ [19]). La relación ϵ – dominance en vectores de coste de Z_+^m está definido por :

$$x \preceq_\epsilon y \iff x_i \leq y_i, 1 \leq i \leq m \quad (3.4)$$

En la figura 3.2 se puede apreciar los efectos de la dominancia epsilon en un problema de optimización combinatoria de doble criterio y como influyen dos valores distintos del valor epsilon (el izquierdo es un valor ϵ mas grande que el derecho). Notar como las líneas punteadas han sido movilizadas de modo que su origen no coincide con los puntos oscuros en cuestión y por ende abarcan o dominan a más soluciones.

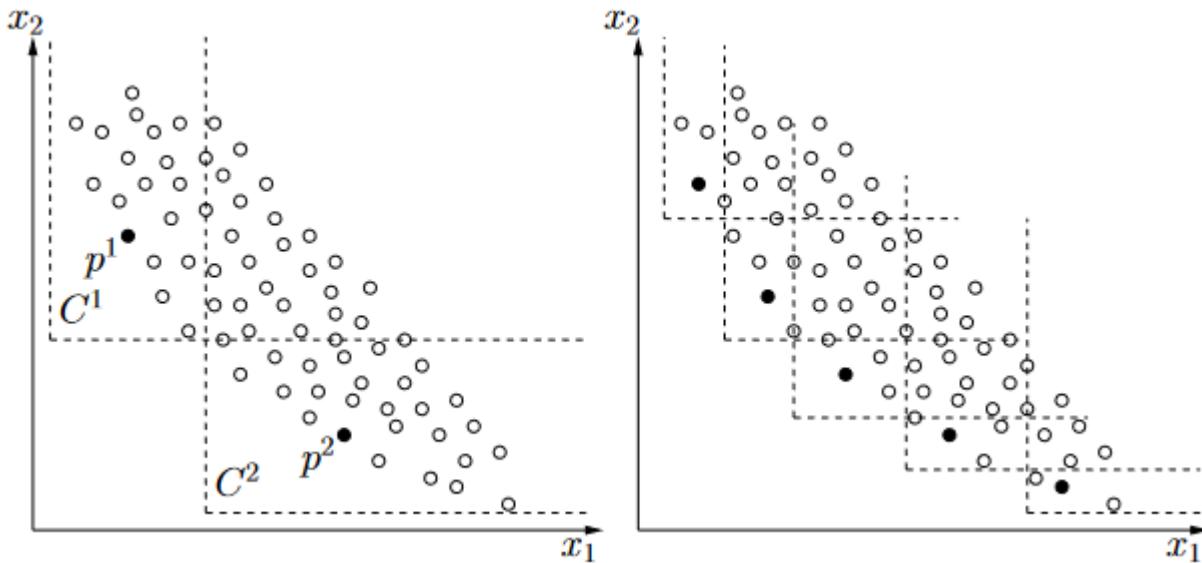


Figura 3.2: Ilustración de los efectos de la dominancia epsilon [19]

3.4. Biobjective Shortest Path Problem

Sea un grafo $G = (N, A)$ donde $N = \{1, \dots, n\}$ son un conjunto de nodos o vértices y $A \subseteq N \times N$ representa un conjunto de aristas o arcos. Sean también un nodo inicial u origen $s \in N$ y un nodo final u objetivo $t \in N$. Sea finalmente una función de coste $c : A \rightarrow \mathbb{R}^2$ la cual asigna dos costes a cada una de las aristas de nuestro grafo de modo que $c_{ij} = (c_{ij}^1, c_{ij}^2)$ vendría a representar el coste de la arista ij .

El Biobjective Shortest Path Problem consiste en lo siguiente [17]:

$$\min_{p \in P} \left\{ c(p) = \left(\sum_{(i,j) \in p} c_{ij}^1, \sum_{(i,j) \in p} c_{ij}^2 \right) \right\} \quad (3.5)$$

En la ecuación 3.5 P representa todo el conjunto de rutas o caminos simples desde el origen hacia el objetivo, de modo que el problema vendría a ser un problema de optimización combinatoria de criterio múltiple ya que estamos tratando de minimizar dos funciones que son las sumatorias de los primeros y segundos costes de las aristas que conforman una ruta.

3.5. Mapeo de zonas de riesgo

El mapeo de crímenes o zonas de riesgo hace referencia al estudio y análisis de datos espaciales que están relacionados a crímenes. El mapeo de crímenes cumple 3 funciones importantes en el análisis del crimen [22]:

1. Facilita la visualización y el análisis de eventos espaciales relacionados al crimen.
2. Permite relacionar información en base a variables geográficas.
3. Proveen mapas que ayudan a entender mejor los resultados obtenidos de los análisis.

3.6. Hot Spot

3.6.1. Definición

En lo que respecta al mapeo de zonas de riesgo, un HOT SPOT es una zona es un mapa que está catalogada como altamente peligrosa. La manera de catalogar una zona determinada puede estar basada en el número de crímenes ocurridos en esa zona o también en la gravedad de dichos crímenes. El hecho de hablar de HOT SPOTS nos lleva también a hablar de COLD SPOTS, que vendrían a ser áreas de bajo peligro, así como SPOTS en los que la peligrosidad no sea ni alta ni baja [3].

3.6.2. Tipos de Hot Spots

Existen diversas maneras de visualizar y analizar las zonas de alto riesgo, entre ellas tenemos [9]:

1. Repeat places hot spots: en estos hot spots se analiza lugares específicos tales como restaurantes, bares o mercados en los que hayan ocurrido una cantidad significativa de crímenes. Este enfoque resulta útil ya que si bien hay regiones irregulares en las que se identifica una alta concentración de crímenes, estos suelen estar concentrados en lugares específicos.

2. Repeat victimization hot spot: a diferencia de los enfoques usuales en los que se analizan regiones geográficas, este enfoque se centra en personas que hayan sido víctimas de algún delito en repetidas ocasiones y estas personas pueden estar distribuidas en un gran número de lugares.
3. Repeat Street hot spots: este enfoque se concentra en analizar las calles en los que existe un gran riesgo de sufrir un delito o en las que se han cometido una gran cantidad de crímenes o eventos peligrosos. Este enfoque es específico a resultado difícil de analizar e implementar, sin embargo, en un trabajo reciente se ha logrado identificar HOT STREETS en ciudades usando algoritmos de machine learning y visualizarlos apropiadamente en un mapa.
4. Neighborhoods hot spots: este enfoque se centra en analizar los vecindarios (en nuestro caso podrían ser las manzanas) en las que ocurren una gran cantidad de crímenes.

Capítulo IV

Modelo de riesgo

En este capítulo se explican las ecuaciones que permiten puntuar las aristas de cada grafo con su peligrosidad apoyándonos en la técnica estadística Kernel Density Estimation y una serie de ecuaciones propuestas en [1]. Nos referiremos a la ecuación 3.2 presentada en el marco teórico como $\lambda(p)$ donde p representa un punto en el espacio geográfico (ya sea proyectado o no).

Una vez que tenemos nuestro modelo generado a partir de los eventos de crímenes procedemos a asignar los puntajes de riesgo a las aristas de nuestro grafo. Con esto en mente y en función de la estructura que posee nuestra data proveniente de OSMNX denotamos por $\Gamma(e)$ la geometría de cada una de las aristas del grafo ya que OSMNX nos proporciona líneas compuestas por varios puntos denotados por $\xi_i \in \Gamma(e)$ para representar a cada calle tal y como se visualiza en la figura 4.1

```
In [14]: list(edges_proj.iloc[0].geometry.coords)
Out[14]: [(2774199.33074235, 8436519.179924924),
           (2774187.764647256, 8436467.60467333),
           (2774187.0855983626, 8436463.084863747),
           (2774187.041070566, 8436453.620122986),
           (2774187.5420082747, 8436438.248335937)]
```

Figura 4.1: Geometría de una de las calles del grafo

Luego definimos la siguiente ecuación que nos permite obtener la densidad de crimen en la

arista e :

$$\Lambda(e) = \sum_{\xi_i \in \Gamma(e)} \lambda(\xi_i)$$

Para obtener el riesgo final de una arista e se utilizará la siguiente ecuación:

$$r(e) = \frac{\Lambda(e)}{\sum_{e' \in G} \Lambda(e')}$$

Finalmente, para calcular el nivel de riesgo total de una ruta P los autores de [1] proponen la siguiente ecuación:

$$r_{total}(P) = 1 - \prod_{e \in P} (1 - r(e))$$

Sin embargo, para poder usar este modelo de riesgo y resolverlo mediante un algoritmo orientado al Biobjective Shortest Path Problem es necesario obtener el riesgo total de una ruta P en función de una suma de valores asociados a las aristas del grafo, de modo que los autores proponen utilizar **la suma de los logaritmos negativos de uno menos el riesgo de cada arista**. Cabe recalcar que los autores no proporcionan la fórmula de la ecuación asociada a dicha sugerencia de modo que la interpretación que le dimos y la cual finalmente implementamos es la siguiente:

$$r_{total}(P) = \sum_{e \in P} (-\log_{10}(1 - r(e)))$$

Capítulo V

Algoritmo para la resolución del BSPP

El algoritmo encargado de calcular toda la frontera de Pareto se muestra en 1. Un componente bastante del algoritmo que forma parte de los datos de entrada es la función heurística consistente y en el presente seminario se utilizó una función heurística bastante genérica que fue propuesta en [28] y que consiste en lo siguiente: dado que las aristas poseen dos costos la función heurística también mapea dos costos a cada nodo de modo los costos sean las distancias obtenidas con el algoritmos de Dijkstra desde el nodo objetivo hacia cada nodo en específico para cada uno de los costos en cuestión. Esto se ilustra en la figura 5.1 donde cada nodo tiene etiquetado la distancia mínima de cada coste desde el nodo en cuestión hacia el nodo objetivo y para realizar el cálculo de dichos valores lo que tiene que hacerse es **ejecutar un algoritmo one-to-all shortest path desde el nodo objetivo hacia el resto de nodos en el grafo con las aristas invertidas** lo cual puede hacerse con el algoritmo de Dijkstra igual que en [28].

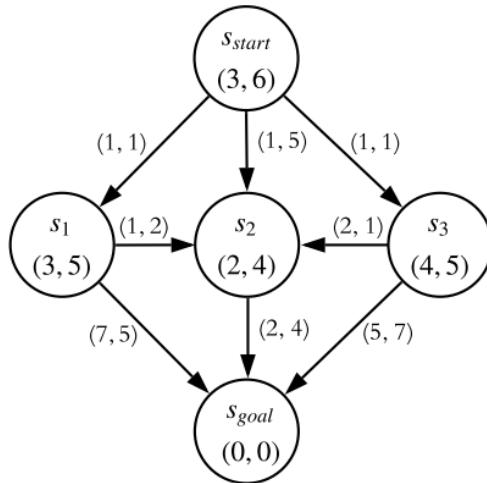


Figura 5.1: Grafo de ejemplo donde se visualiza la función heurística empleada [28]

Otro componente bastante importante del algoritmo es la lista **Open** y los elementos o **nodos** que dicha lista almacena. Cada uno de estos nodos tiene asociados un estado o vértice del grafo que se está manipulando, un *g*-value que indica el costo hasta dicho nodo del camino que se está calculando, un *f*-value que indica el costo estimado de la ruta que atraviesa el nodo en cuestión hasta llegar al objetivo y finalmente un *parent* que representa literalmente el padre o antecesor del nodo en el grafo. Los nodos están ordenados lexicográficamente en función de su *f*-value. Se sugiere consultar el capítulo 5 de [10] para una explicación más detallada de este tipo de ordenamiento.

Por lo general los algoritmos orientados al Biobjective Shortest Path Problem tratan de computar todas las rutas posibles entre un origen y un destino, sin embargo, una parte crucial de dichos algoritmos es la de verificar si al momento de ir calculando una de estas rutas esta es dominada o no por alguna ruta de nuestro conjunto solución final. Es esta verificación la que hace resaltar a este algoritmo ya que a diferencia de trabajos anteriores en donde esta verificación se hace en tiempo lineal en este algoritmo esto se realiza en tiempo constante. Estas verificaciones de si una ruta que está siendo calculada es o no es dominada, se realiza por ejemplo en la línea 11 en la que verificamos si el segundo valor del coste, es decir g_2 de la ruta que llega hasta el nodo x ha excedido el valor g_2^{min} asociado al estado de dicho nodo

Algorithm 1: Bi-Objective A* (BOA*)

Data: Un problema de búsqueda ($S, E, \mathbf{c}, s_{start}, s_{goal}$) y una función heurística consistente h

Result: El conjunto de soluciones Pareto-óptimas de costo único

```

1  $sols \leftarrow \emptyset$ 
2 foreach  $s \in S$  do
3    $g_2^{min}(s) \leftarrow \infty$ 
4    $x \leftarrow$  new node with  $s(x) = s_{start}$ 
5    $g(x) \leftarrow (0, 0)$ 
6    $parent(x) \leftarrow \text{null}$ 
7    $f(x) \leftarrow (h_1(s_{start}), h_2(s_{start}))$ 
8   Inicializar Open y agregar  $x$  al mismo
9   while Open  $\neq \emptyset$  do
10    | Remover nodo  $x$  de Open cuyo f-value sea el valor más pequeño
        | lexicográficamente
11    | if  $g_2(x) \geq g_2^{min}(s(x))$  or  $f_2(x) \geq g_2^{min}(s_{goal})$  then
12    |   | continue
13    |   |  $g_2^{min}(s(x)) \leftarrow g_2(x)$ 
14    |   | if  $s(x) = s_{goal}$  then
15    |   |   | Add  $x$  to  $sols$ 
16    |   |   | continue
17    |   | foreach  $t \in Succ(s(x))$  do
18    |   |   |  $y \leftarrow$  new node with  $s(y) = t$ 
19    |   |   |  $g(y) \leftarrow g(x) + \mathbf{c}(s(x), t)$ 
20    |   |   |  $parent(y) \leftarrow x$ 
21    |   |   |  $f(y) \leftarrow g(y) + h(t)$ 
22    |   |   | if  $g_2(y) \geq g_2^{min}(t)$  or  $f_2(y) \geq g_2^{min}(s_{goal})$  then
23    |   |   |   | continue
24    |   |   |   | Add  $y$  to Open
25   | return  $sols$ 

```

o si el costo final estimado de la ruta asociada a dicho nodo sobrepasa el valor g_2^{min} asociado al estado o vértice objetivo. Un proceso similar se puede ver en la línea 22 del algoritmo al momento de analizar los hijos del nodo expandido y . Son estas verificaciones y la manera en la que organizamos nuestros nodos lexicográficamente (consultar el capítulo 5 de [10] para una mejor explicación) en la lista **Open** los que permiten las verificaciones de dominancia en tiempo constante y el cálculo de la frontera de Pareto completa. Los detalles de la correctitud del algoritmo están detallados en [28].

Finalmente, un inconveniente de calcular la frontera de Pareto completa es que el tamaño de esta puede llegar a ser muy grande resultando ser poco práctico si es que se desea usar este algoritmo como el motor de un aplicativo de cálculo de rutas. Con esto en mente, una sugerencia que se presenta en [21] es una pequeña modificación al algoritmo en las líneas 11 y 22 apoyadas en la definición de la ϵ -dominancia para calcular un subconjunto de la frontera de pareto. La modificación realizada es la siguiente:

$$g_2(x) \geq g_2^{\min}(s(x)) \text{ or } (1 + \epsilon)f_2(x) \geq g_2^{\min}(s_{goal})$$

Lo que propone básicamente es añadir el factor epsilon en la verificación de dominancia de modo que se calcule un subconjunto del conjunto solución inicial. Con todo lo expuesto hasta ahora, el objetivo de los experimentos a realizar es básicamente analizar empíricamente el desempeño del algoritmo propuesto en [28] en nuestro problema de cálculo de rutas seguras y ver los efectos del factor epsilon sugerido en [21].

Capítulo VI

Resultados Experimentales

6.1. Descripción general del trabajo

Una ilustración de todo el presente proyecto puede visualizarse en la figura 6.1. Tal y como se puede apreciar, los pasos son los siguientes:

1. Se utilizan los datos geográficos de los crímenes de la ciudad para construir el modelo de riesgo de manera conjunta con un grafo que representa a las calles de dicha ciudad.
2. Se puntuán las aristas de los grafos con las ecuaciones explicadas en el capítulo 4 y se obtiene el grafo final.
3. Se codifica el algoritmo y se usa el grafo como datos de entrada para el algoritmo.
4. Se analizan resultados tales como tiempos de ejecución, cantidad de soluciones y cantidad de nodos expandidos en promedio.

6.2. Descripción de los datos utilizados

- Para poder generar el modelo de riesgo de nuestras ciudades se utilizaron datos geoespaciales relacionados a crímenes de las ciudades de San Francisco y Chicago. En lo que

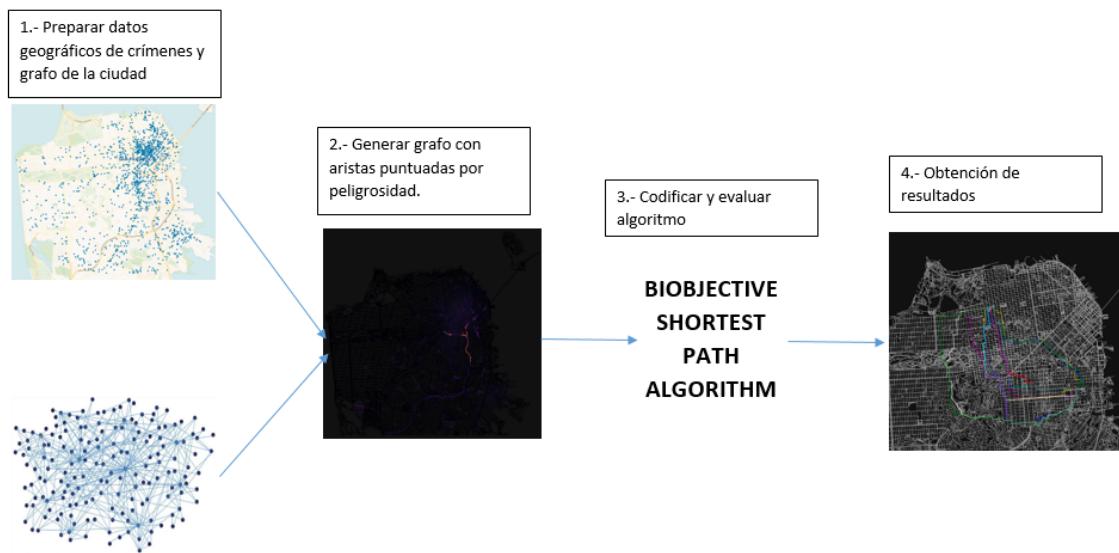


Figura 6.1: Resumen general del presente proyecto

respecta a San Francisco se utilizó un dataset que consta de 2306 eventos de Asaltos del año 2016 en los meses de Noviembre y Diciembre (utilicé estos eventos debido a que ya me encontraba familiarizado con el mismo por haberlo manipulado en el anterior seminario) y en lo que respecta a Chicago utilicé 4865 datos geoespaciales de crímenes del mes de Diciembre del año 2021.

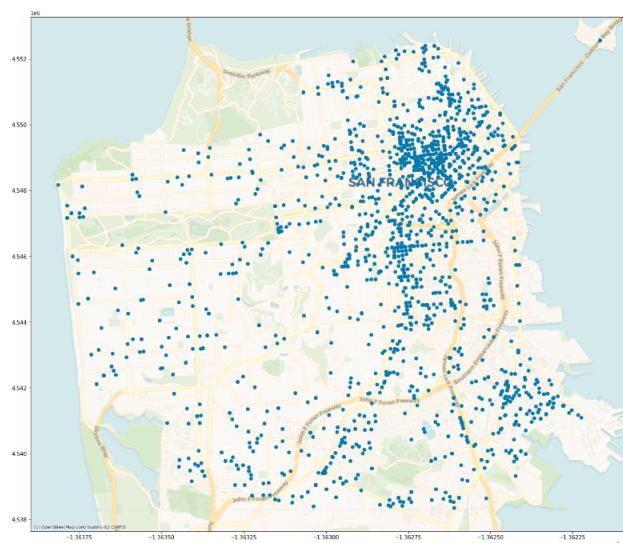


Figura 6.2: Crímenes de San Francisco.

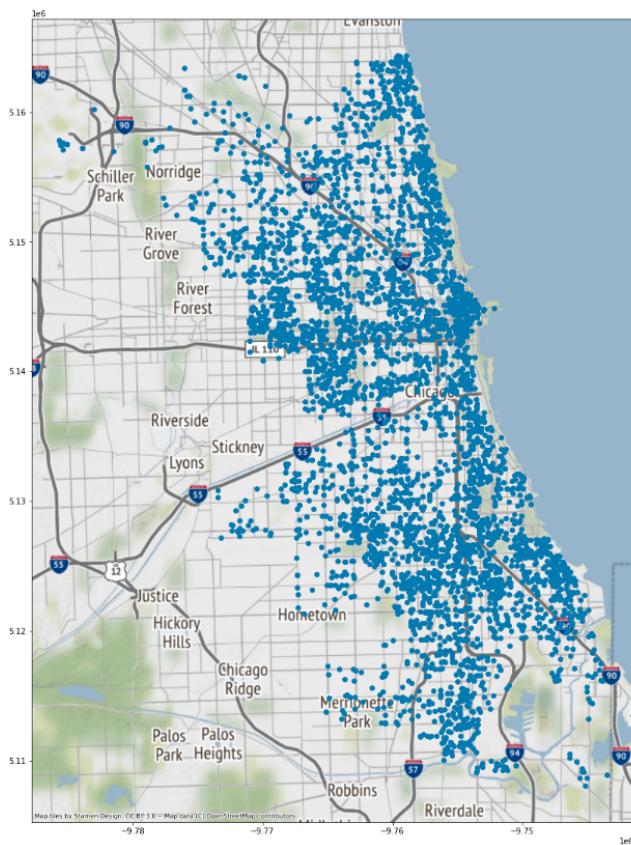


Figura 6.3: Crímenes de Chicago.

- En lo que respecta a los grafos correspondientes a San Francisco y Chicago estos se obtuvieron como objetos de la librería NetworkX mediante la funcionalidad **graph_from_place** de la librería de Python OSMNX [2] pasándole como únicos argumentos los nombres de las ciudades en cuestión (“San Francisco, USA” y “Chicago, USA”) respectivamente. Algo importante que recalcar es que si bien es factible que los algoritmos que van a evaluarse puedan ser adaptados para trabajar con grafos con aristas paralelas opté por no considerarlas para evitar posibles complicaciones. Finalmente, el grafo de San Francisco con el que se trabajó consta de 41020 vértices y 116338 aristas mientras que el grafo de la ciudad de Chicago consta de 139225 vértices y 386477 aristas.

6.3. Preparación de los grafos para los experimentos

Básicamente aquí se implementan las ecuaciones presentadas en el capítulo 4 sobre los grafos proporcionados por la librería OMSNX. Sin embargo, un paso importante antes de generar el modelo del Kernel Density Estimation y de puntuar cada una de las aristas es conveniente proyectar tanto las coordenadas geográficas de nuestros eventos de crímenes así como los grafos obtenidos de OSMNX. En lo que respecta a los crímenes la librería Geopandas nos permite hacerlo muy fácilmente gracias a su funcionalidad **to_crs**. Esta funcionalidad también la presenta OSMNX para proyectar sus grafos. Geopandas recomienda utilizar el código EPSG **epsg:3857** para graficar coordenadas (ver [12]) y utilicé ese sistema de proyección para los ploteos de las figuras 6.2 y 6.3, pero en la web www.epsg.io recomiendan que se usen sistemas de proyecciones distintos dependiendo de la región de la Tierra que se analice. Con ello en mente utilicé el sistema **epsg:6454** en San Francisco y el sistema **epsg:3651** en Chicago para tener los datos de los crímenes y la información de las ciudades proyectadas adecuadamente. Una vez que cuento con todos mis datos proyectados puedo usar la técnica Kernel Density Estimation implementada en la librería Scipy para construir la función de distribución de los crímenes de cada ciudad y finalmente implementar las ecuaciones proporcionadas en [1] y mencionadas en el capítulo 4 para la puntuación de las aristas de mi grafo. Una representación de la peligrosidad de las aristas en los grafos puede visualizarse en las figuras 6.4 y 6.5 correspondientes a San Francisco y Chicago respectivamente. En dichas imágenes se pueden ver aristas con un tono bastante intenso que indica que son más peligrosas que el resto de las aristas con una intensidad de color menor.

Finalmente, dado que los algoritmos para la resolución del Biobjective Shortest Path Problem se implementarán en Java es necesario pasar los grafos obtenidos en Python a un formato fácilmente manipulable. Para ello lo que hice fue pasar los identificadores de los vértices de mis grafos a un archivo de texto y para las aristas genere archivos CSV con los campos **from**, **to**, **length**, **log_risk_score**.

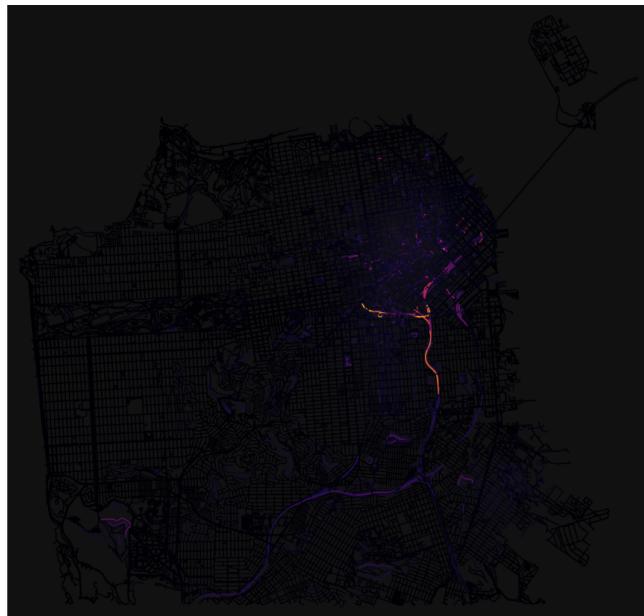


Figura 6.4: Modelo de riesgo de San Francisco.

6.4. Generación de los experimentos para la evaluación de los algoritmos

Tendremos experimentos de distintas categorías. Por ejemplo, un experimento de la categoría **X** consta de un origen y un destino que distan entre **X-1** y **X** kilómetros. Con esto en mente, realizaremos dos rondas de experimentos de distintas categorías. Estas rondas se explican a continuación:

1. En esta ronda inicial de experimentos en ambas ciudades se escogieron cien nodos al azar que nos servirán como nodos iniciales. Luego, a cada nodo le asociaremos 4 nodos de las categorías del 1 al 7 teniendo finalmente para cada nodo de origen 28 nodos de destino y teniendo finalmente 2800 experimentos a realizar (400 por cada categoría) en cada una de las ciudades.
2. Para tratar de exigir un poco más al algoritmo en ambas ciudades, en esta segunda y última ronda de experimentos para la ciudad de San Francisco se escogieron 36 nodos iniciales de los nodos seleccionados en la anterior ronda y se les asoció a cada uno 4 nodos de destino de la categoría 10. Solo se escogieron 36 nodos ya que la ciudad es

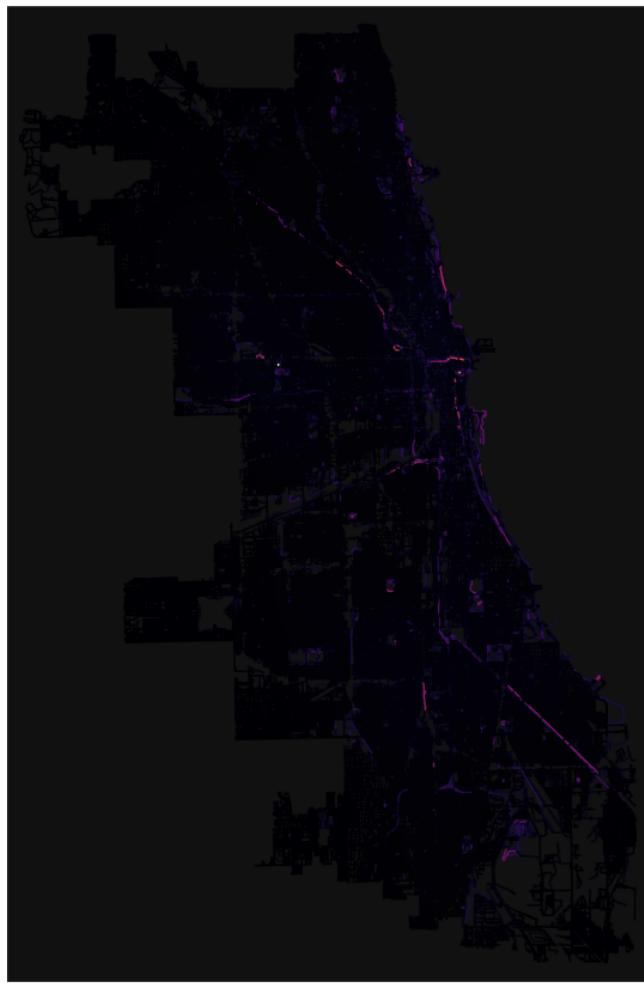


Figura 6.5: Modelo de riesgo de Chicago.

relativamente pequeña (121 km^2 de superficie) y era bastante complicado encontrar más experimentos viables. En el caso de la ciudad de Chicago se trabajo del mismo modo que en los experimentos de la primera ronda, con los mismos 100 nodos iniciales y asociándoles a cada uno 4 nodos de destino de la categoría 10 y también de la categoría 15. Se tendría, por tanto, 144 experimentos de la categoría 10 en la ciudad de San Francisco y 400 experimentos en las categorías 10 y 15 para la ciudad de Chicago.

Ahora, para estudiar los efectos del factor epsilon explicado en los capítulos 3 y 5, en cada experimento que tengamos se utilizarán los valores de epsilon 0.0, 0.025, 0.05, 0.075, 0.1 y 0.125. Se evaluarán los tiempos de ejecución, la cantidad promedio de nodos expandidos y el tamaño del conjunto solución en cada experimento.

Una última acotación que hacer es que en los tiempos de ejecución mostrados **no se incluye el tiempo que demora el cálculo de la función heurística relacionados a los vértices del grafo**. Esto se debe a que estos no se ven afectados por el valor del epsilon y que al incluirlos en el análisis no se suele apreciar el impacto del epsilon en los experimentos más computacionalmente sencillos por lo que serán omitidos del análisis. Sin embargo es importante señalar lo máximo que ha demorado calcular estos valores heurísticos en nuestros grafos, de modo que habiendo señalado eso tenemos que el tiempo máximo que demoro calcular la función heurística en **San Francisco** es de **0.184 segundos** y en **Chicago** demoró **0.69 segundos**

6.5. Resultados obtenidos

6.5.1. Experimentos de las categorías 1 a 7

Tiempos de ejecución

En la figura 6.6 se pueden apreciar los tiempos de ejecución promedio en ambas ciudades. Tanto en San Francisco como en Chicago pueden apreciarse una disminución en los tiempos de ejecución promedio (sobre todo en los experimentos de las categorías 4 a la 7 que son los más difíciles.) Las reducciones en los tiempos promedio son un poco más difíciles de apreciar en los experimentos más sencillos (los de categorías inferiores que son de la categoría 3 para abajo) y incluso pareciera que empeoran pero las variaciones son mínimas ya que en esas dimensiones (prácticamente 0.01 segundos) las operaciones computacionales no suelen ser constantes.

En el caso de los tiempos de ejecución máximos (los cuales se muestran en la figura 6.7) el comportamiento conforme va aumentándose el epsilon luce bastante errático. En el caso de San Francisco si puede apreciarse una reducción de los tiempos de ejecución en las categorías superiores (salvo en los experimentos de la categoría 5 donde se ve uno de los comportamientos erráticos). Por otro lado, es difícil poder afirmar algo en la ciudad de Chicago en esta primera ronda de experimentos con respecto a los tiempos de ejecución máximos ya que en las primeras 4 categorías de experimentos los tiempos máximos exhiben un comportamiento errático. Esto

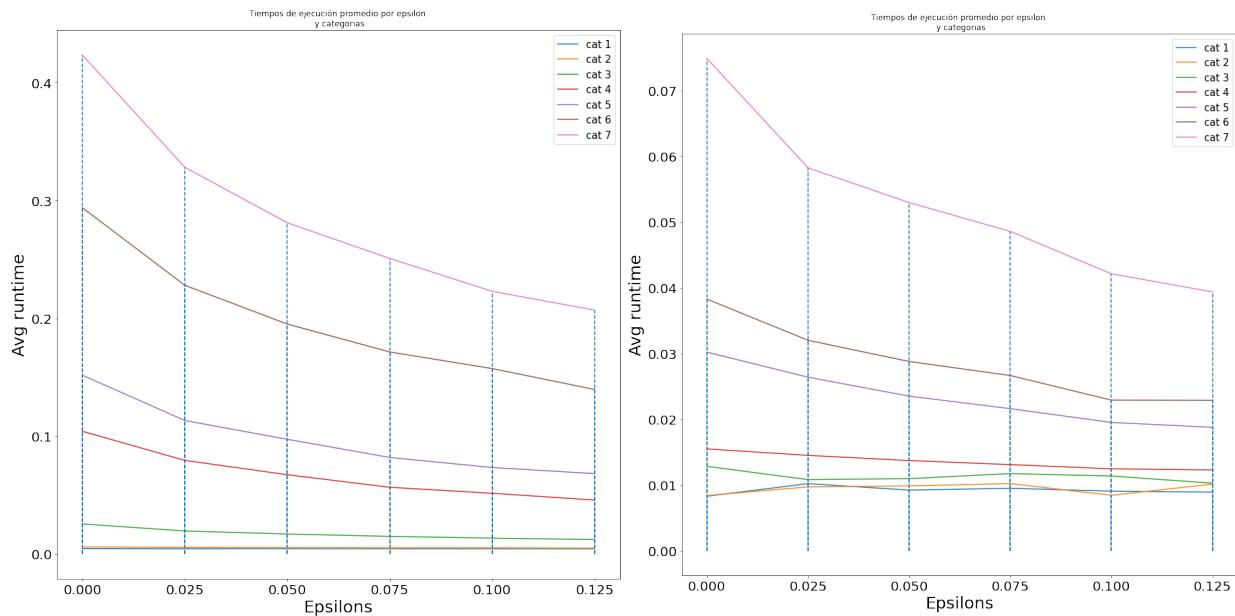


Figura 6.6: Tiempos de ejecución promedio en segundos afectados por el epsilon en San Francisco y Chicago

puede deberse a la variabilidad de los tiempos en algunas operaciones computacionales, lo cual considero era de esperarse.

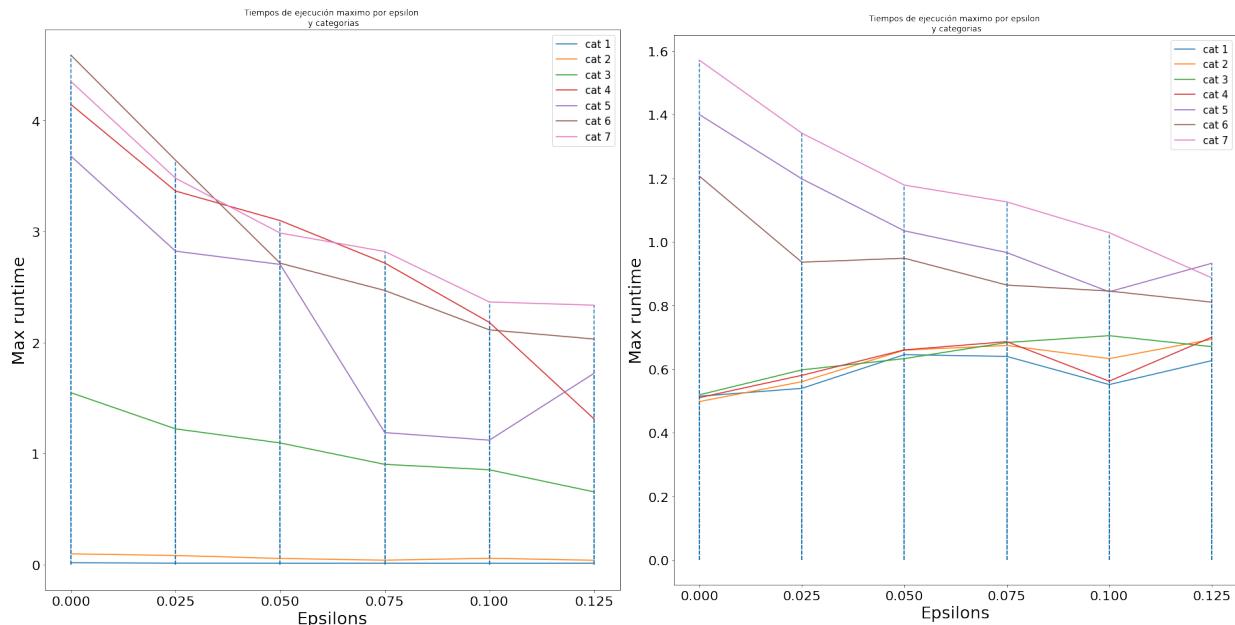


Figura 6.7: Tiempos de ejecución máximos en segundos afectados por el epsilon en San Francisco y Chicago

Cantidad de soluciones

Donde verdaderamente se ve el impacto del factor epsilon es en los conjuntos de solución resultantes y dicho impacto se puede apreciar en ambas ciudades. Por ejemplo, basta con ver la figura 6.8 para apreciar el impacto de aplicar un valor de epsilon relativamente pequeño para reducir enormemente la cantidad de soluciones en promedio. En la figura 6.9 se puede apreciar la misma gráfica sin contar $\epsilon = 0$ para apreciar mejor los resultados. En ambas gráficas se puede apreciar una reducción en la cantidad de soluciones en promedio encontradas conforme se va incrementando el valor de epsilon.

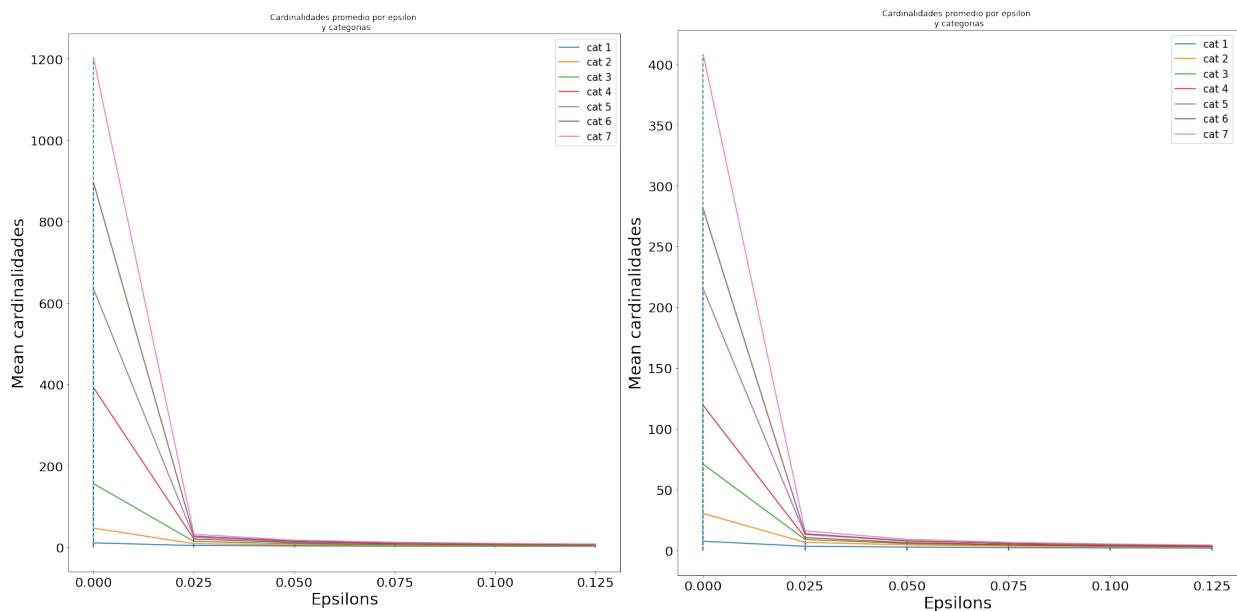


Figura 6.8: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago

Las figuras 6.10 y 6.11 muestran las cantidades máximas de soluciones halladas y aquí también puede apreciarse el impacto significativo del factor epsilon conforme este va incrementándose.

Cantidad promedio de nodos expandidos

Finalmente otro factor a analizar es la cantidad de nodos expandidos en promedio y estos resultados pueden apreciarse en la figura 6.12. Esto nos da una idea de cuantas operaciones se

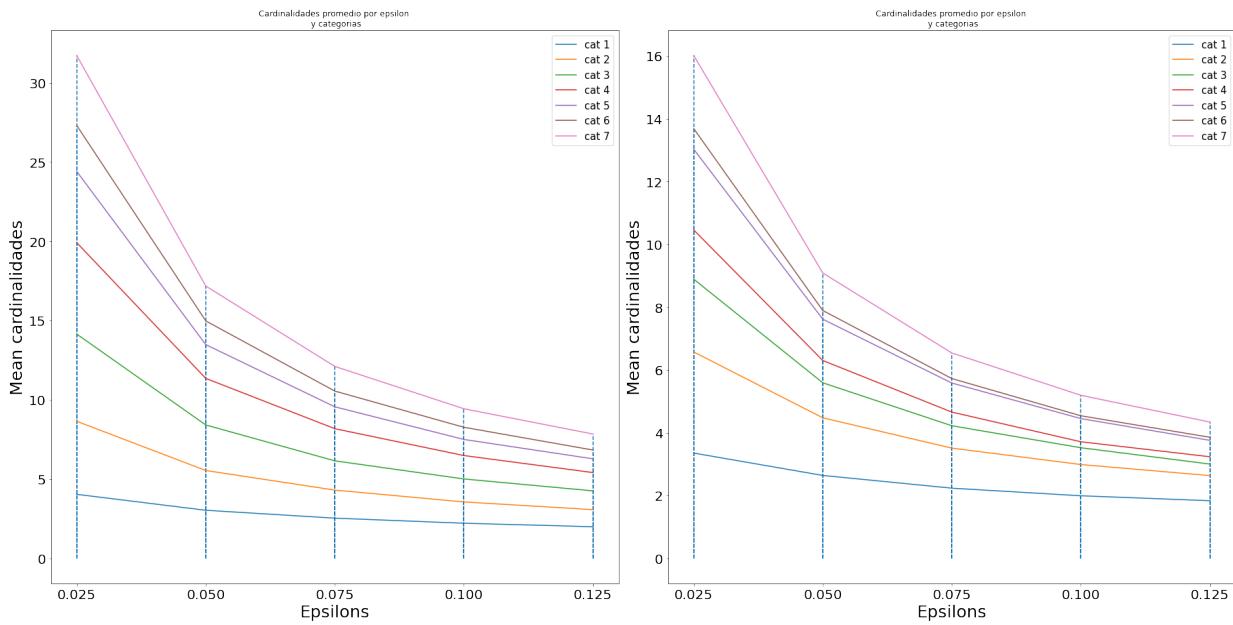


Figura 6.9: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$

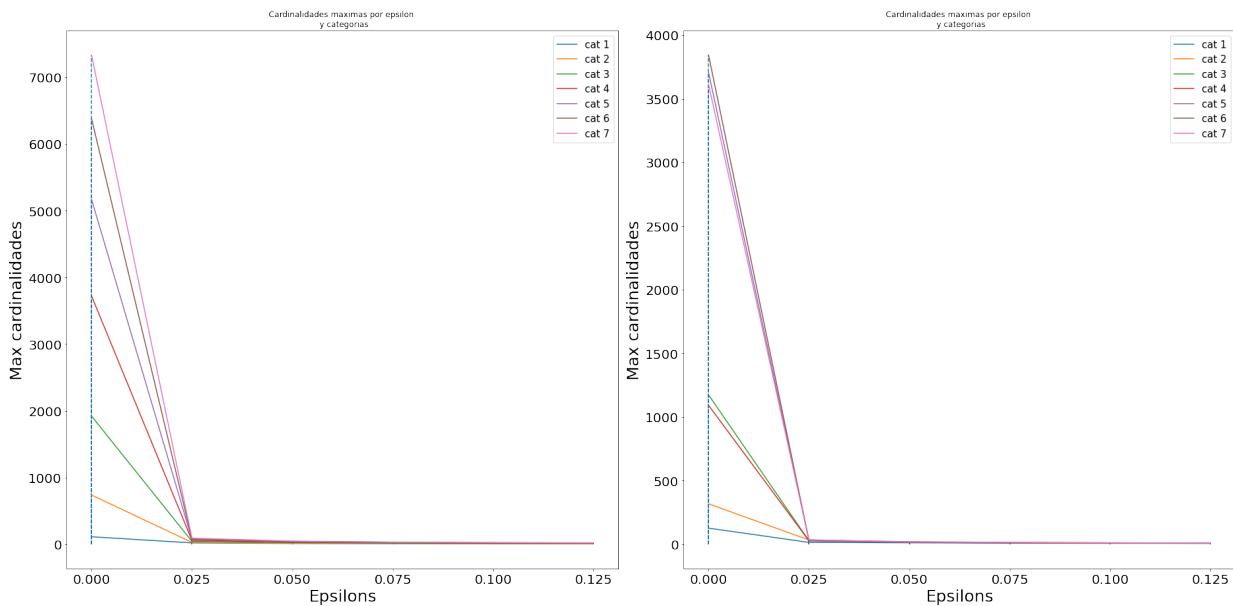


Figura 6.10: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago

están realizando internamente en el algoritmo y es de ayuda ya que en los trabajos orientados a algoritmos de búsqueda multiobjetivo no suelen hablarse de complejidades teóricas ni de tiempo ni de memoria (al menos no en trabajos como [28] ni [21]) y es por eso que se decidió incluir este factor en el presente proyecto.

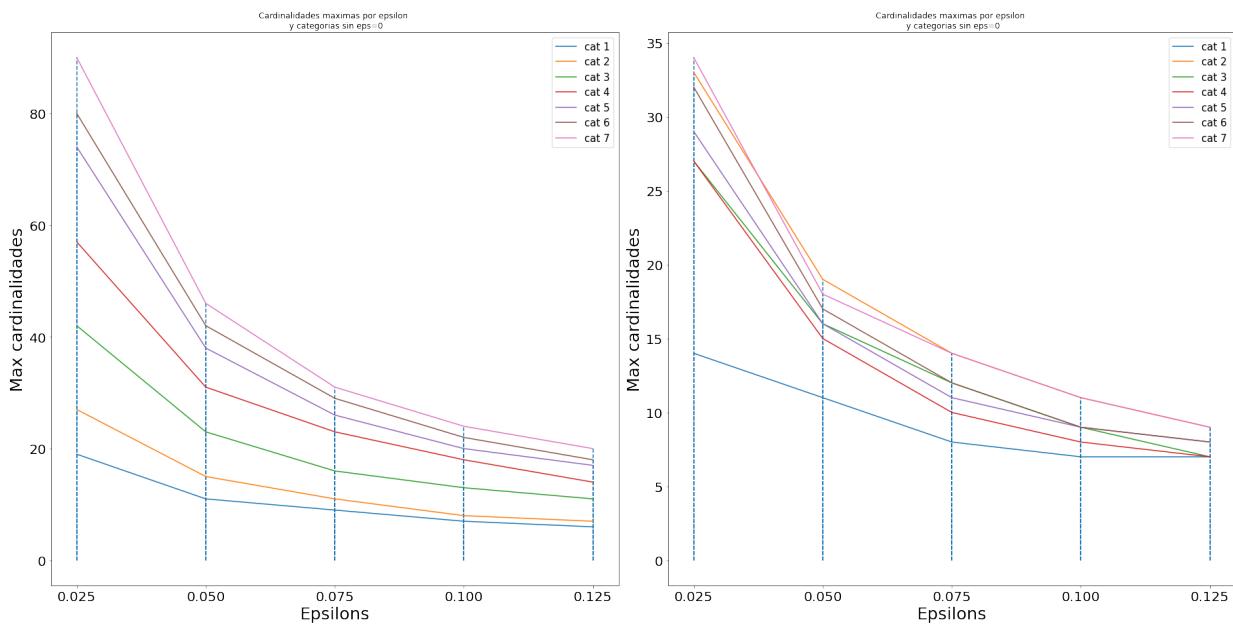


Figura 6.11: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$

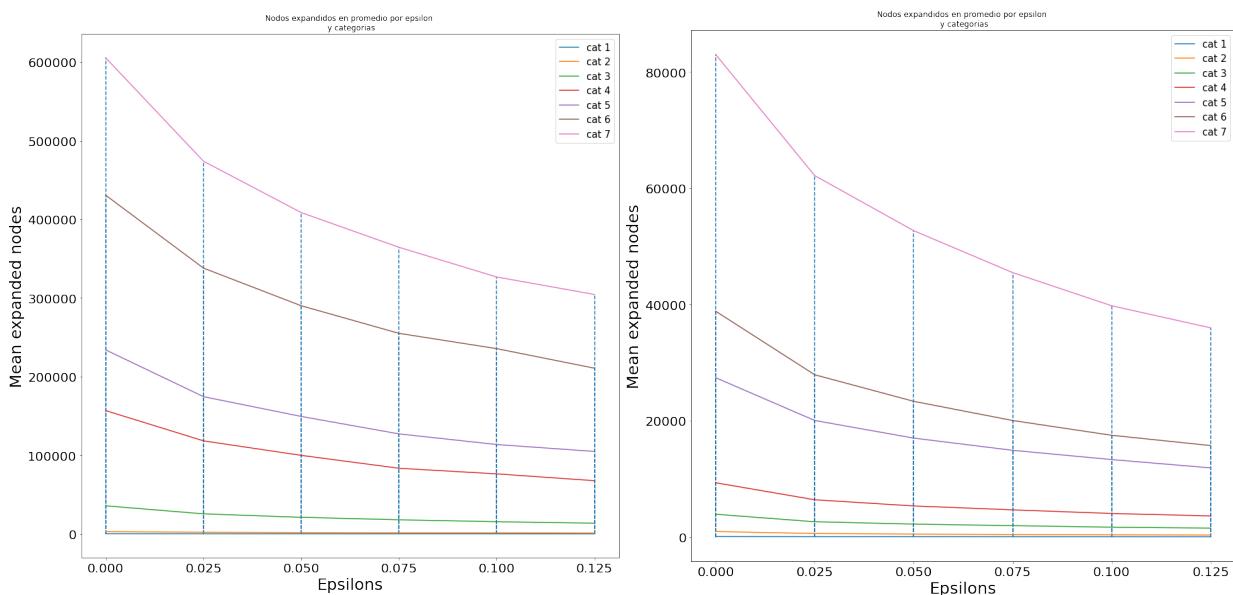


Figura 6.12: Nodos expandidos en promedio afectados por el epsilon en San Francisco y Chicago

6.5.2. Experimentos de las categorías 10 y 15

Tiempos de ejecución

En estas categorías ya se empiezan a apreciar un poco mejor las mejoras que trae el uso del factor epsilon en términos del tiempo de ejecución en ambas ciudades. En las figura 6.13 puede

verse como van disminuyendo los tiempos de ejecución promedio en ambas ciudades conforme se va aumentando el valor de epsilon. También puede apreciarse el mismo efecto del valor de epsilon en los tiempos de ejecución máximos de la figura 6.14 (salvo por un ligero incremento en la categoría 15). Aquí puede apreciarse como es que la reducción de los tiempos de ejecución se aprecia mucho mejor en los experimentos de mayor categoría, es decir, en los experimentos más difíciles.

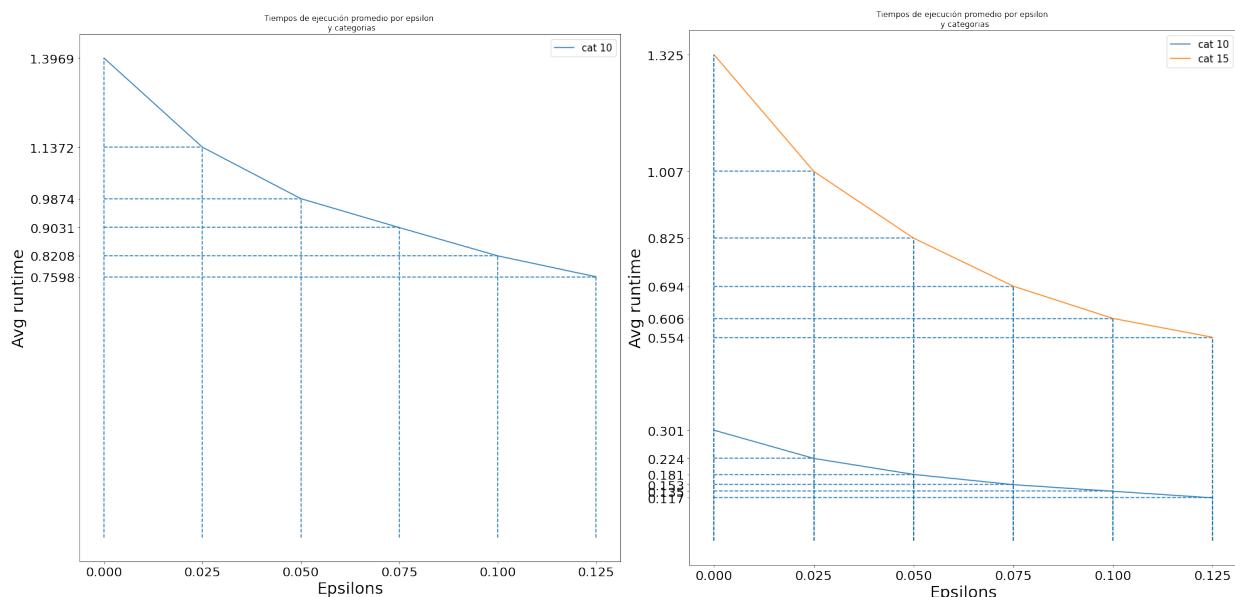


Figura 6.13: Tiempos de ejecución promedio en segundos afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

Cantidad de soluciones

En esta segunda ronda de experimentos se sigue apreciando un brutal impacto del valor de epsilon tanto en las cantidades promedios(ver figuras 6.15 y 6.16) como en las cantidades máximas de soluciones (ver figuras 6.17 y 6.18).

Cantidad promedio de nodos expandidos

En la figura 6.19 se puede apreciar cantidades considerables de nodos expandidos en promedio en ambas ciudades y como es que el factor epsilon provoca la reducción de dichas expansiones a medida que va incrementando su valor.

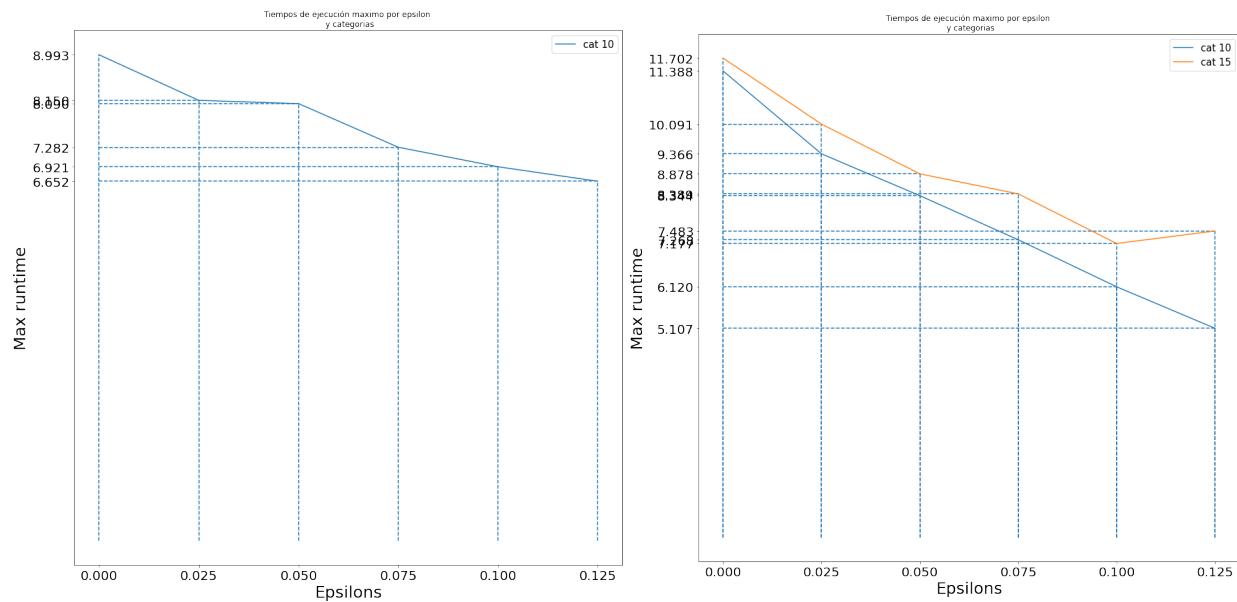


Figura 6.14: Tiempos de ejecución máximos en segundos afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

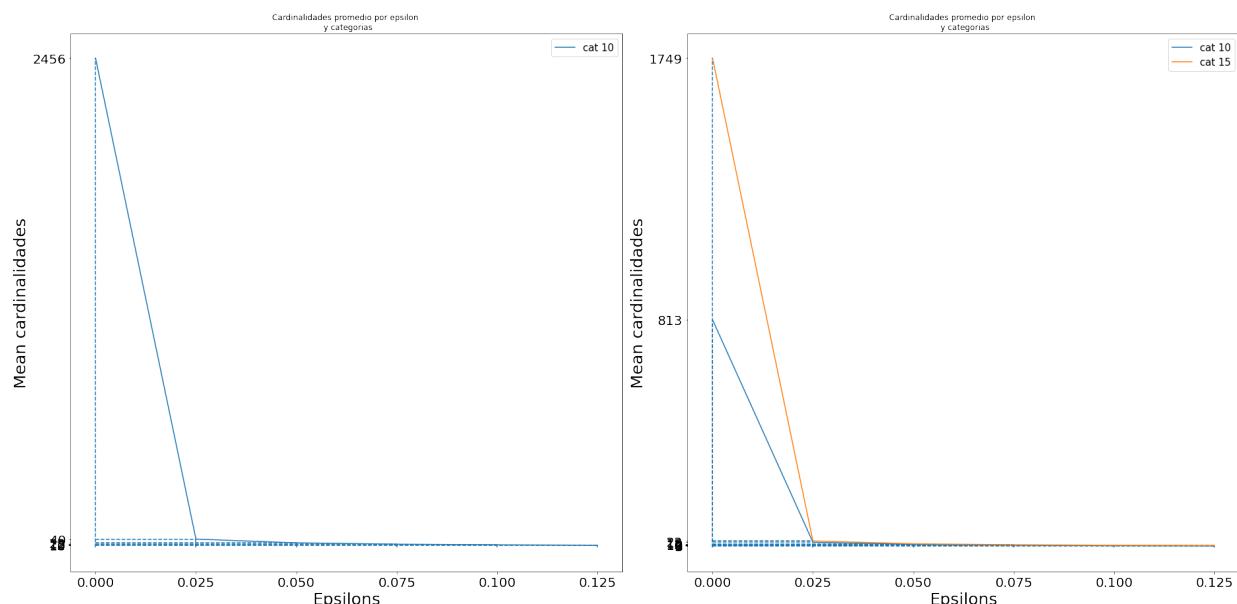


Figura 6.15: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

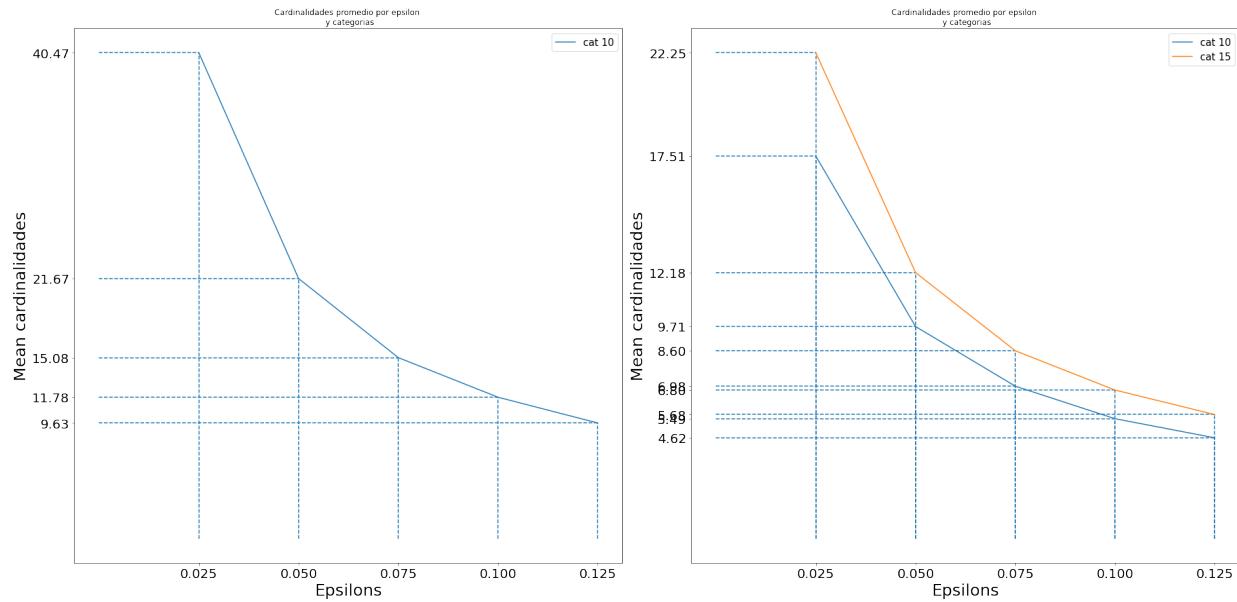


Figura 6.16: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$ en la segunda ronda de experimentos

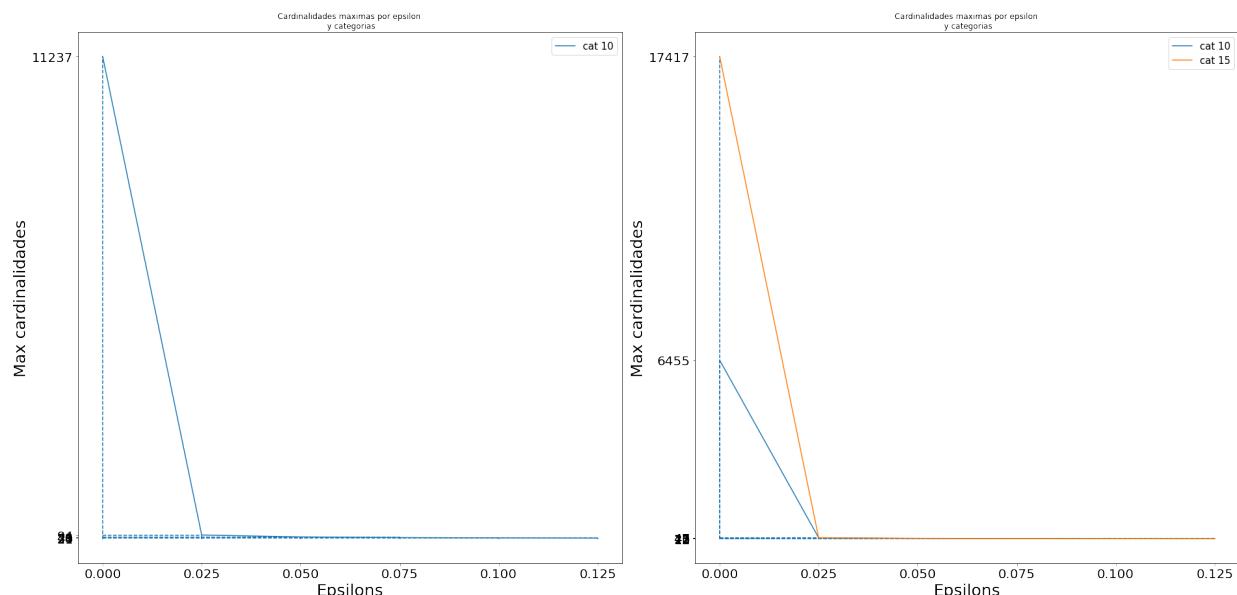


Figura 6.17: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

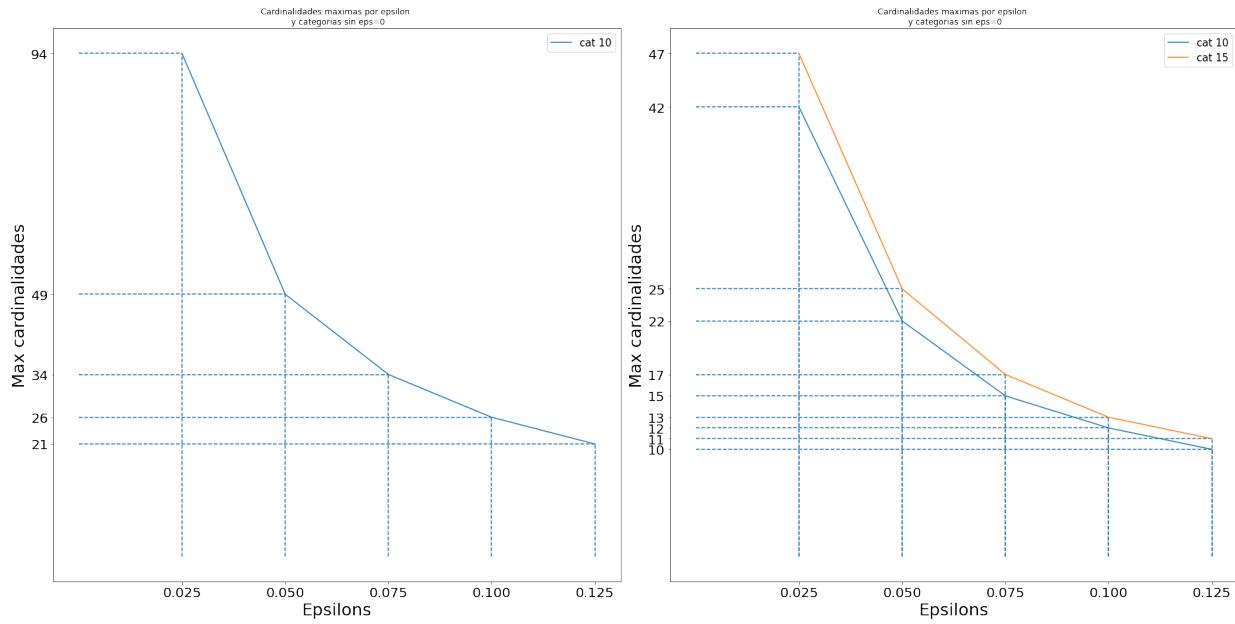


Figura 6.18: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar $\epsilon = 0$ en la segunda ronda de experimentos

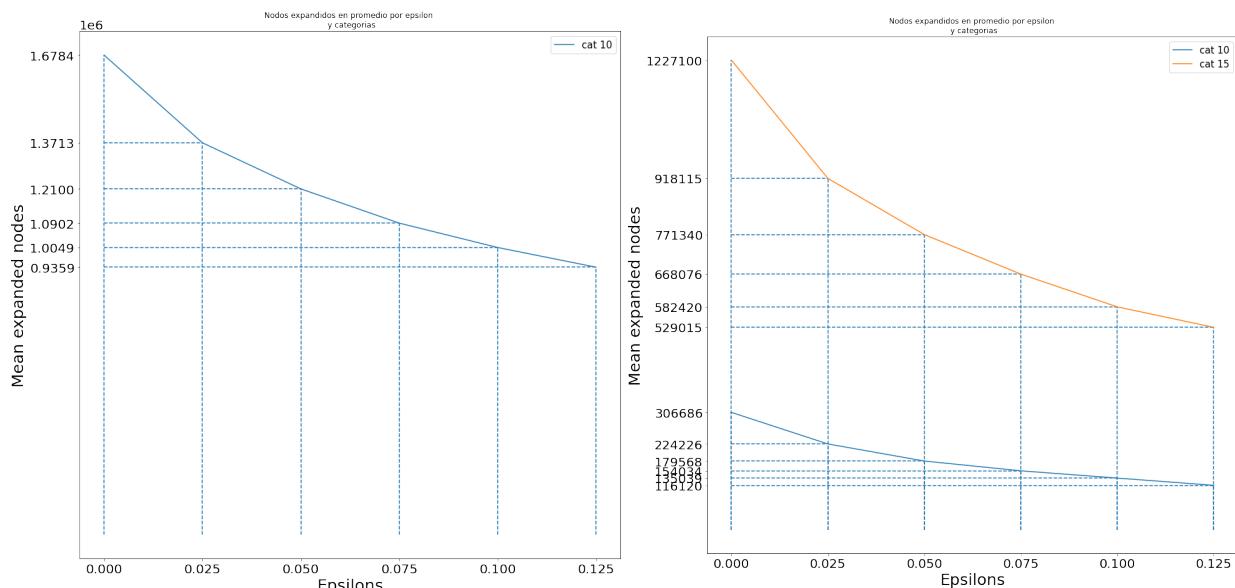


Figura 6.19: Nodos expandidos en promedio afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

Capítulo VII

Conclusiones y trabajo futuro

7.1. Conclusiones

- Se logró la obtención de la representación de dos ciudades en forma de grafo para su manipulación y posterior procesamiento que nos permitió evaluarlos adecuadamente para el algoritmo.
- Las mejoras del epsilon en los tiempos de ejecución se lograron apreciar mejor en los tiempos promedio y sobre todo en los experimentos de mayor categoría. En el caso de los tiempos máximos se exhiben comportamientos un poco erráticos pero esto puede deberse a que ciertas operaciones computacionales suelen tomar distinta cantidad de tiempo en una ejecución y en otra.
- Se logró apreciar como el aumento del valor de epsilon favorece a reducir la cantidad promedio de nodos expandidos lo cual nos da una idea de la cantidad de cálculo y memoria ahorrados en los experimentos.
- Los efectos del valor de epsilon sobre la cantidad final de soluciones son bastante significativos. Bastaba con darle al epsilon un valor de 0.025 para empezar a ver una reducción drásticas tanto en las cantidades promedio como en las cantidades máximas de soluciones.

- En líneas generales el factor epsilon tiene un efecto positivo sobre los experimentos haciendo los tiempos de ejecución y la cantidad de soluciones mucho más manejables. Sin embargo, considero que debe realizarse un estudio más detallado sobre la cantidad de soluciones y realizar estudios de manera más detallada en una ciudad determinada para corroborar si es práctico o no su uso en dicha ciudad debido a que la estructura de las ciudades y la distribución de las peligrosidades pueden provocar resultados muy variados y por ello puede que cada ciudad necesite una manera distinta de calibrar el factor epsilon.

7.2. Trabajo futuro

- Profundizar más en el estudio del modelo de riesgo para desarrollar un modelo más robusto.
- Analizar propuestas modernas que han estado realizándose para mejorar el algoritmo propuesto en [28] y evaluar si se puede aplicar la sugerencia planteada en [21] y mejorar aún más los tiempos de ejecución.
- Investigar de manera más detallada el impacto del factor epsilon en una ciudad determinada y su impacto en la cantidad de soluciones conseguidas para determinar si es posible realizar su calibración adecuadamente.

REFERENCIAS

- [1] Urban navigation beyond shortest route: The case of safe paths. *Information Systems* 57 (2016), 160–171.
- [2] BOEING, G. Osmnx: A python package to work with graph-theoretic openstreetmap street networks. *Journal of Open Source Software* 2, 12 (2017), 215.
- [3] CHAINY, S., AND RATCLIFFE, J. *GIS and Crime Mapping (Mastering GIS: Technology, Applications and Management)*, first ed. Wiley, 2005, pp. 145–148.
- [4] CHICAGO GOVERNMENT. Chicago's data about several topics . <https://data.cityofchicago.org/>, 2016.
- [5] CNN NEWS. Waze app directions take woman to wrong Brazil address, where she is killed. <https://edition.cnn.com/2015/10/05/americas/brazil-wrong-directions-death/index.html>, 2015.
- [6] DE SOUZA, A. M., BRAUN, T., BOTEGA, L. C., CABRAL, R., GARCIA, I. C., AND VILLAS, L. A. Better safe than sorry: a vehicular traffic re-routing based on traffic conditions and public safety issues. *Journal of Internet Services and Applications* 10, 1 (Sep 2019), 17.
- [7] DE SOUZA, A. M., BRAUN, T., BOTEGA, L. C., VILLAS, L. A., AND LOUREIRO, A. A. F. Safe and sound: Driver safety-aware vehicle re-routing based on spatiotemporal information. *IEEE Transactions on Intelligent Transportation Systems* 21, 9 (2020), 3973–3989.

- [8] DUQUE, D., LOZANO, L., AND MEDAGLIA, A. L. An exact method for the biobjective shortest path problem for large-scale road networks. *European Journal of Operational Research* 242, 3 (2015), 788–797.
- [9] ECK, J., CHAINY, S., CAMERON, J., LEITNER, M., AND WILSON, R. Mapping crime: Understanding hot spots. *Eck, J.E. and Chainey, S. and Cameron, J.G. and Leitner, M. and Wilson, R.E. (2005) Mapping crime: understanding hot spots. Research report. National Institute of Justice, US.* (08 2005), 4–10.
- [10] EHRGOTT, M. *Multicriteria Optimization*. Springer-Verlag, Berlin, Heidelberg, 2005.
- [11] GARCÍA-PORTUGUÉS, E. *Notes for Nonparametric Statistics*. 2021. Version 6.5.1. ISBN 978-84-09-29537-1.
- [12] GEOPANDAS. Adding a background map to plots . https://geopandas.org/en/stable/gallery/plotting_basemap_background.html.
- [13] GOOGLE. Maps JavaScript API . <https://developers.google.com/maps/documentation/javascript/reference/directions#DirectionsService>, 2018.
- [14] KALINIC, M., AND KRISP, J. Kernel density estimation (kde) vs. hot-spot analysis - detecting criminal hot spots in the city of san francisco.
- [15] LEVY, S., XIONG, W., BELDING, E., AND WANG, W. Y. Saferoute: Learning to navigate streets safely in an urban environment.
- [16] MATA, F., TORRES-RUIZ, M., GUZMÁN, G., QUINTERO, R., ZAGAL-FLORES, R., MORENO-IBARRA, M., AND LOZA, E. A mobile information system based on crowd-sensed and official crime data for finding safe routes: A case study of mexico city. *Mobile Information Systems* 2016 (Mar 2016), 8068209.
- [17] MOLENDIJK, A. L. Finding extreme supported solutions to the bi-objective shortest path problem.

- [18] OPENSTREETMAP CONTRIBUTORS. Planet dump retrieved from <https://planet.osm.org> .
<https://www.openstreetmap.org>, 2017.
- [19] PERNY, P., AND SPANJAARD, O. Near admissible algorithms for multiobjective search.
pp. 490–494.
- [20] RAITH, A. Speed-up of labelling algorithms for biobjective shortest path problems.
- [21] SALZMAN, O. Approximate bi-criteria search by efficient representation of subsets of the pareto-optimal frontier. In *ICAPS* (2021).
- [22] SANTOS, R. B. *Crime Analysis with Crime Mapping*, fourth ed. Sage Publications, 2016,
pp. 32–33.
- [23] SCOTT, D. W. *Multivariate Density Estimation: Theory, Practice, and Visualization*, 2 ed.
Wiley Series in Probability and Statistics. Wiley, 2015.
- [24] SEDEÑO-NODA, A., AND COLEBROOK, M. A biobjective dijkstra algorithm. *European Journal of Operational Research* 276, 1 (2019), 106–118.
- [25] SHAH, S., BAO, F., LU, C.-T., AND CHEN, I.-R. Crowdsafe: crowd sourcing of crime incidents and safe routing on mobile devices. pp. 521–524.
- [26] SIRIARAYA, P., WANG, Y., ZHANG, Y., WAKAMIYA, S., JESZENSZKY, P., KAWAI, Y.,
AND JATOWT, A. Beyond the shortest route: A survey on quality-aware route navigation
for pedestrians. *IEEE Access* 8 (2020), 135569–135590.
- [27] SONI, S., GAURI SHANKAR, V., AND SANDEEP, C. Route-the safe: A robust model for
safest route prediction using crime and accidental data. 1415–1428.
- [28] ULLOA, C. H., YEOH, W. G. S., BAIER, J. A., ZHANG, H., SUAZO, L., AND KOENIG,
S. A simple and fast bi-objective search algorithm. In *ICAPS* (2020).