

# El Biobjective Shortest Path Problem aplicado al cálculo de rutas seguras

Ponce Pinedo, Víctor    Arrunátegui, Miguel

<sup>1</sup>Facultad de Ciencias  
Universidad Nacional de Ingeniería

Exposición de proyectos, Enero de 2021

# El Biobjective Shortest Path Problem (BSPP)

- 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 consistiría en lo siguiente:

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

# Problema

- Calcular rutas lo más cortas y seguras posibles.

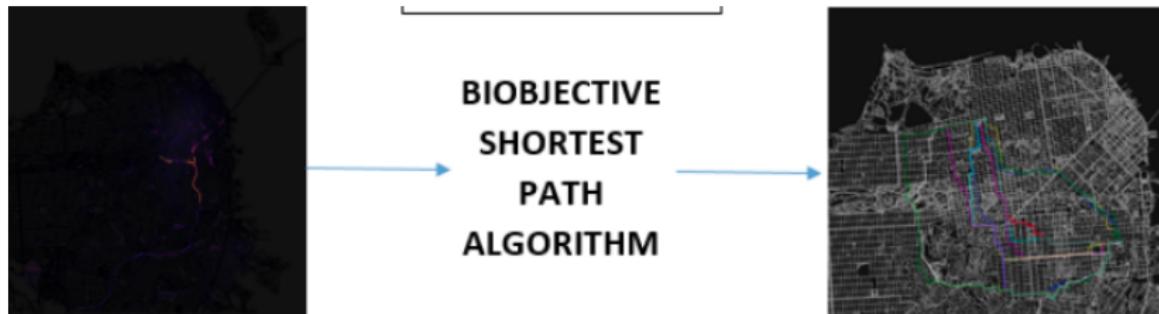


Fig. 1: Cálculo de rutas seguras

# Motivación

- Se pueden evitar quejas y sucesos como los mostrados en la figura 2

The screenshot shows a web browser window with the URL [waze.com/forum/viewtopic.php?t=211805](http://waze.com/forum/viewtopic.php?t=211805). The page is from the 'wazeforum' community. The title of the post is "Waze Taking You To Dangerous Routes" by [natalieholder](#) on Thu Nov 26, 2014 10:37 am. The post content discusses a severe accident involving a woman who was held at gunpoint while traveling on a route recommended by the Waze app. The user describes how their wife had two near-death experiences due to the app's routing decisions.

**Waze Taking You To Dangerous Routes**  
by [natalieholder](#) Thu Nov 26, 2014 10:37 am

Dear Maze,

RE: Maze Routes Responsible For Two Severe Accidents/Hijacking at Gunpoint

I would like to lay a severe complaint.

Your app is the reason myself and my wife had two near death experiences.

1. Your app is the reason my wife recently had a terrible accident and wrote off her car. We had NO medical insurance or car insurance. We lost that car. My wife was navigating from Johannesburg, Randburg to Kimberley, Northern Cape to visit her parents whom had just been released from hospital. Your app took her through back roads and onto a dangerous badly maintained and deserted road. She pulled over on the side of the road and phoned me in panic. She tried to push on as sitting on the side of the road is not safe in South Africa. In desperation in trying to avoid pot holes in the road and no where else to go she hit a terrible pot hole and rolled her car - On a Road that is not suppose to be used when traveling to Kimberley. She called me and it would have taken me nearly 4 hours to get to where she was. She should never have been on that route. Why the fuck would your app send her on such an obscure route? You cannot say that it was to avoid traffic as this was middle of the day and the roads were quiet. Which was even worse as she was stuck in a rolled car down an embankment and little traffic.

2. Secondly, your app is the reason that both my wife and I were held at GUN POINT yesterday and we were in an attempted hijacking. I

Fig. 2: Queja de un usuario de Waze por enrutamiento peligroso.

# 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.

w

# 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.

# Trabajos influyentes en el presente proyecto

- Urban navigation beyond shortest route: The case of safe paths [1].
- A Simple and Fast Bi-Objective Search Algorithm [2].
- Approximate bi-criteria search by efficient representation of subsets of the Pareto-optimal frontier [3].

# Resumen general del trabajo

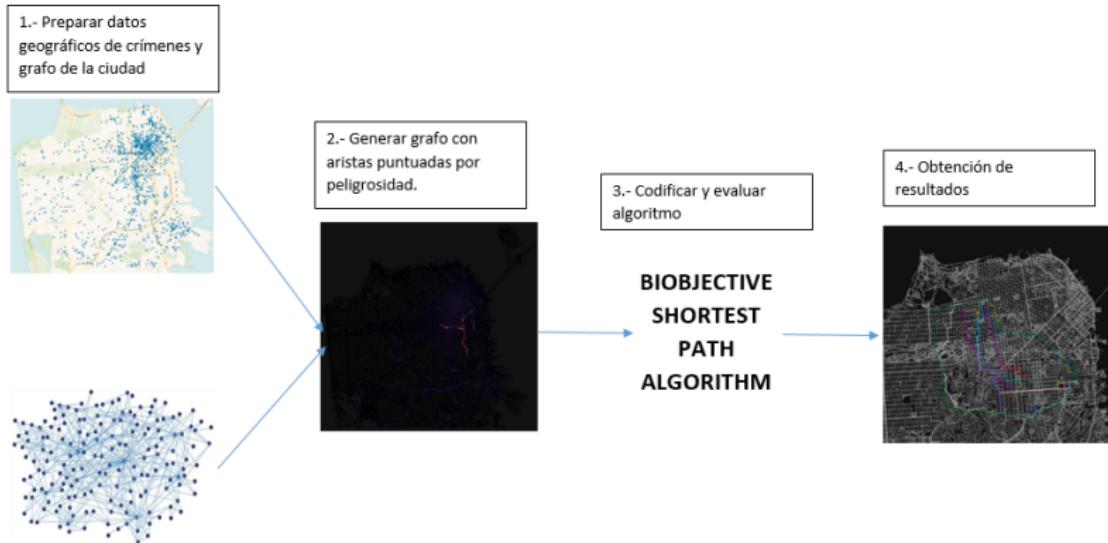


Fig. 3: Resumen general del trabajo

# Frontera de Pareto

La frontera de pareto la constituyen todas las soluciones que no son dominadas por ninguna otra solución.

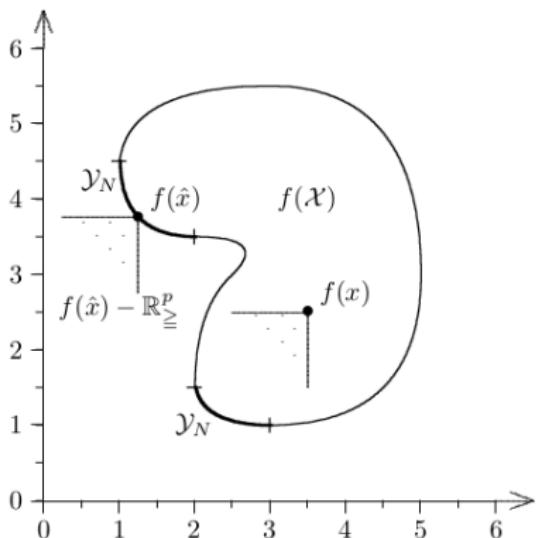


Fig. 4: Ilustración de la frontera de Pareto en un problema de optimización de doble criterio

# Dominancia epsilon

Sean  $x$  e  $y$  dos vectores de coste. La relación  $\epsilon - dominance$  en vectores de coste de  $Z_+^m$  está definido por :

$$x \preceq_\epsilon y \iff x_i \leq (1 + \epsilon)y_i, 1 \leq i \leq m \quad (2)$$

# Ilustración de la dominancia epsilon

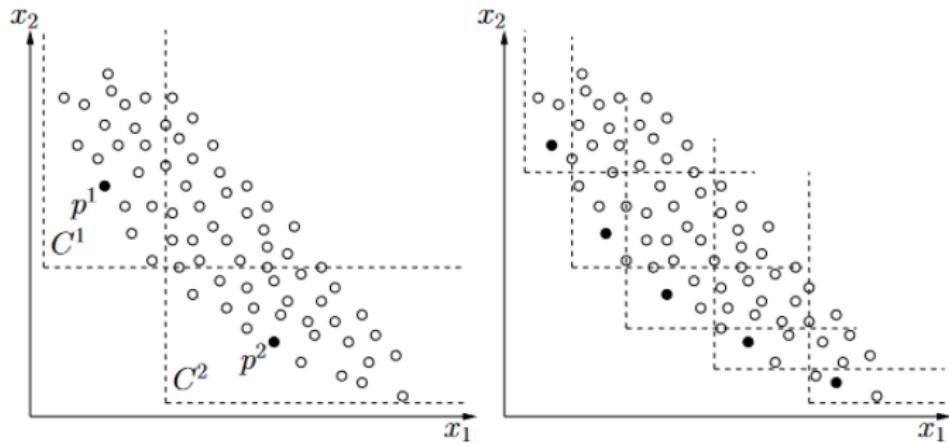


Fig. 5: Ilustración de los efectos de la dominancia epsilon

# Kernel Density Estimation (KDE)

- Técnica estadística que permite estimar una función de densidad de probabilidad de un conjunto de datos.
- La ecuación general viene dada por:

$$f(x) = \frac{1}{n|H|^{\frac{1}{2}}} \sum_{i=1}^n K(H^{\frac{-1}{2}}(x - X_i)) \quad (3)$$

Donde:

- $H$  es la matriz de ancho de banda.
- $n$  es la cantidad de elementos del dataset.
- $K$  es la función kernel.
- $X_i$  es un elemento del conjunto de datos.

# Desarrollo del modelo de riesgo

- Para el uso de la técnica KDE se utilizó la implementación de la librería Scipy codificada en Python.
- La librería usa la **función de distribución normal** como kernel y calcula la matriz de ancho de banda usando **regla de Scott**.
- La librería fue la misma que se utilizó en [1] (se consultó al autor al respecto).

# Ecuaciones del modelo de riesgo

- Denotamos por  $\Gamma(e)$  la geometría de cada una de las aristas del grafo.
- La densidad del crimen en una arista del grafo viene dada por:

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

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

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

# Riesgo total de una ruta

- Para calcular el nivel de riesgo total de una ruta  $P$  los autores de [1] proponen la siguiente ecuación:

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

- Es necesario obtener el riesgo en función de una sumatoria, por lo 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:

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

# Datos de entrada del algoritmo para el BSPP

- El algoritmo llamado BOA\* recibe un grafo dirigido, una función de coste  $c$ , vértices de origen y objetivo y una función heurística consistente.

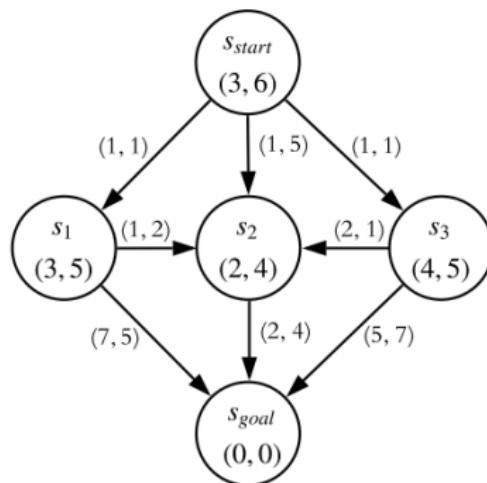


Fig. 6: Grafo de juguete usado en [2]

# El algoritmo BOA\* para el BSPP

---

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

---

**Data:** Un problema de búsqueda ( $S, E, c, s_{start}, s_{goal}$ ) 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       $g(y) \leftarrow g(x) + c(s(x), t)$ 
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
```

# Aspectos generales del BOA\*

- Las funciones  $h_1$  y  $h_2$  son consistentes.
- La lista *Open* esta ordenada lexicográficamente en función de  $f$ .
- Verificación de dominancia en tiempo constante con:  
$$g_2(x) \geq g_2^{\min}(s(x)) \text{ or } f_2(x) \geq g_2^{\min}(s_{goal})$$
- Para cada vértice o estado  $s$  BOA\* mantiene un  $g_2^{\min}(s)$

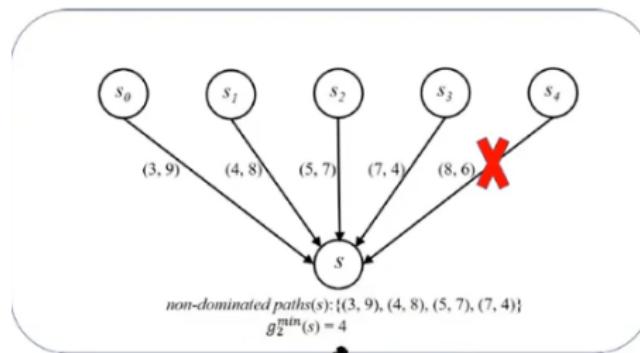


Fig. 8: Ejemplo mostrado por los autores del algoritmo.

# Modificación al algoritmo

- Con la finalidad de obtener un subconjunto de la frontera de pareto en [3] sugieren cambiar las líneas 11 y 22 del algoritmo por:

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

# Descripción de los grafos

- Los grafos fueron obtenidos utilizando la librería OSMNX de Python.

	Vértices	Aristas
San Francisco	41020	116338
Chicago	139225	386477

# Descripción de los datos de crímenes utilizados

- En cuanto a San Francisco se utilizó un dataset que consta de 2306 datos geoespaciales de asaltos del año 2016 en los meses de Noviembre y Diciembre.
- En cuanto a Chicago se utilizó 4865 datos geoespaciales de crímenes del mes de Diciembre del año 2021.

# Grafo resultante de San Francisco

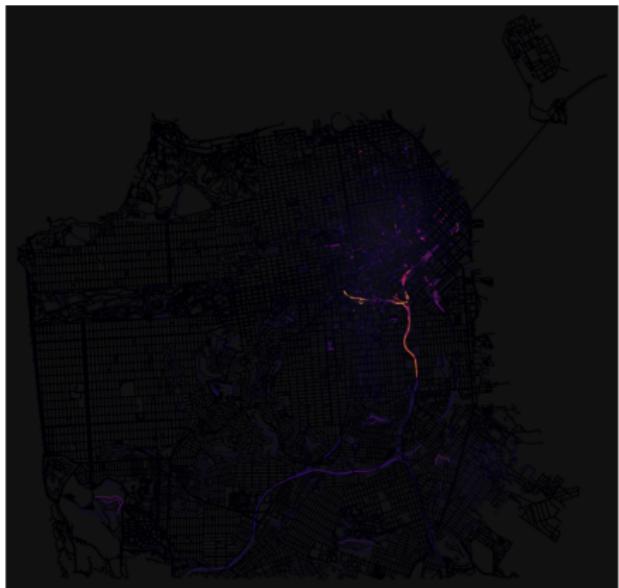
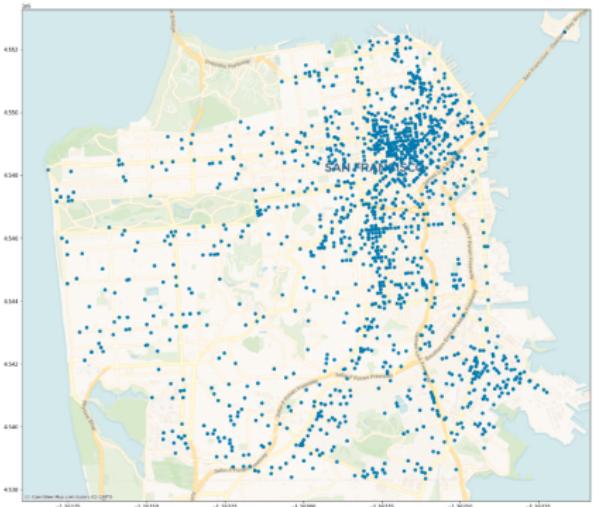


Fig. 9: Resultados en la ciudad de San Francisco

# Grafo resultante de Chicago

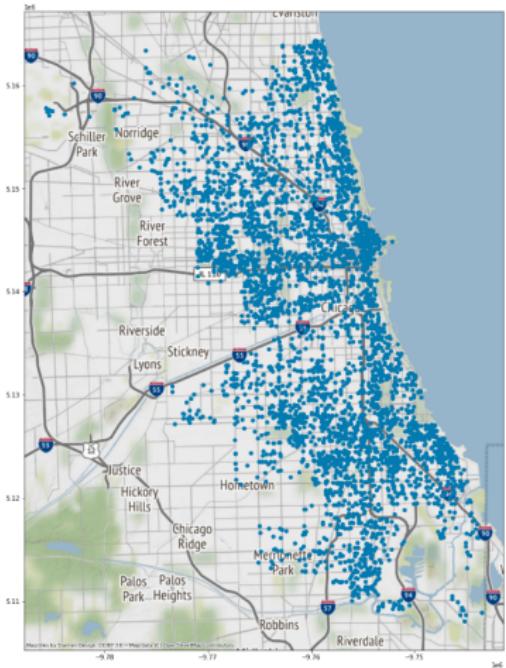


Fig. 10: Resultados en la ciudad de Chicago

# Resultados iniciales

- Inicialmente tanto la codificación del modelo de riesgo como la codificación de los algoritmos del BSPP fueron codificados en Python debido a su sencillez, para visualizar los resultados de manera inmediata y por las facilidades que suelen dar los notebooks para experimentar con las variables.
- Sin embargo, los tiempos de ejecución fueron (como era de esperarse) bastante elevados.

# Experimentos iniciales

- Este experimento calculó la frontera de Pareto completa, obteniendo 2465 soluciones en 107 segundos.

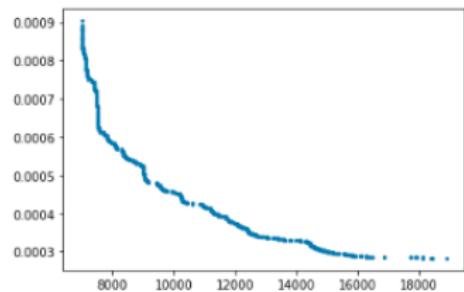


Fig. 11: Experimentos iniciales en San Francisco

# Experimentos iniciales

- Se repitió el experimento pero con un valor de epsilon de 0.1 calculándose finalmente 13 soluciones en 67 segundos.

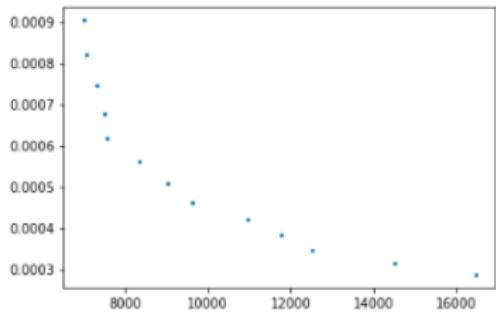


Fig. 12: Experimentos iniciales en San Francisco con un epsilon de 0.1

# Detalles finales de la implementación del algoritmo

- Para ver que tan lejos se puede llegar en términos de tiempos de ejecución, los algoritmos fueron codificados en Java.
- Los experimentos mostrados anteriormente tomaron 2.5 y 1.67 segundos respectivamente.
- Dado que el cálculo de las heurísticas no se ven influidas por el factor epsilon fueron omitidas de los resultados, pero de todos modos es importante acotar lo máximo que demoraron dichos cálculos en ambas ciudades. En San Francisco demoró como máximo 0.184 segundos y en Chicago 0.69 segundos.

# Detalles de los experimentos para la evaluación del algoritmo

- Los experimentos están divididos en categorías en función de la distancia entre el origen y el objetivo.
- Un experimento de categoría  $x$  contiene vértices de origen y objetivo que distan entre  $x - 1$  y  $x$  kilómetros.
- Con el fin de evaluar el impacto del factor epsilon, cada experimento se evaluará con 6 valores distintos de dicho factor: 0, 0.025, 0.05, 0.075, 0.1 y 0.125.
- Se harán dos rondas de experimentos que se explicarán posteriormente.
- Se evaluarán los tiempos de ejecución, el tamaño de los conjuntos de solución y la cantidad promedio de nodos expandidos.

# Primera ronda de experimentos

- Se trabajarán con experimentos de las categorías 1 a la 7.
- Para cada ciudad se escogieron 100 vértices al azar que servirán como vértices iniciales para cada experimento.
- A cada vértice inicial se le asignó cuatro vértices objetivo de cada categoría de modo que se tengan 400 experimentos de cada categoría en cada ciudad.
- Se evaluaron tiempos de ejecución promedio y máximos, cantidad de soluciones promedio y máximas y finalmente cantidad promedio de nodos expandidos

# Tiempos de ejecución promedio

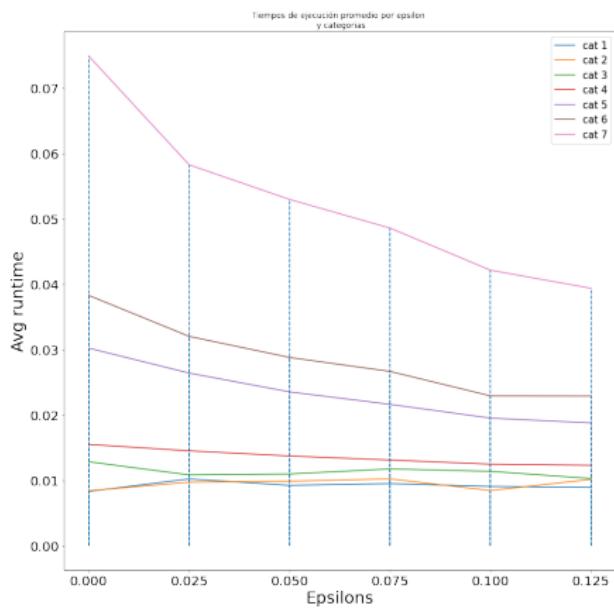
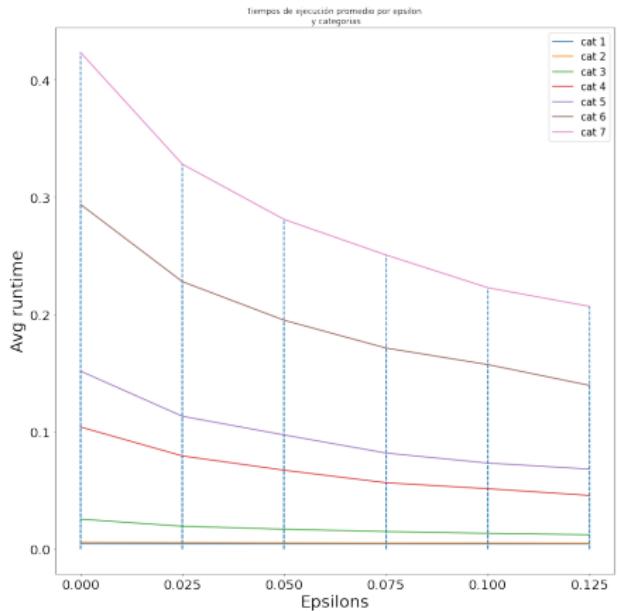


Fig. 13: Tiempos de ejecución promedio en segundos afectados por el epsilon en San Francisco y Chicago

# Tiempos de ejecución máximos

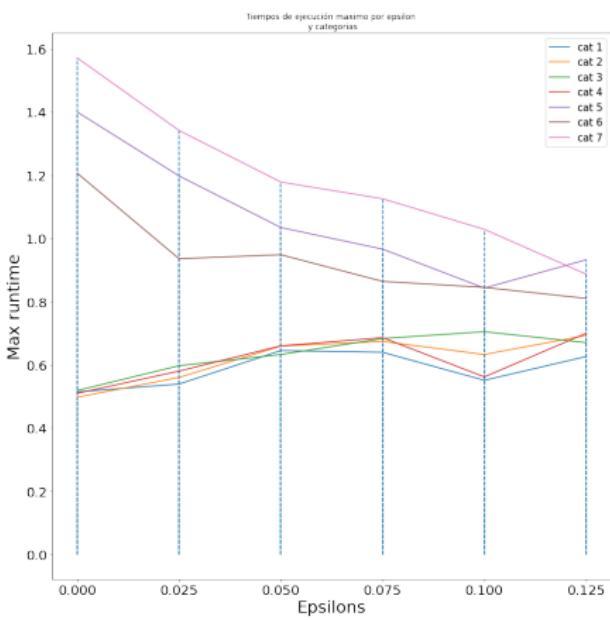
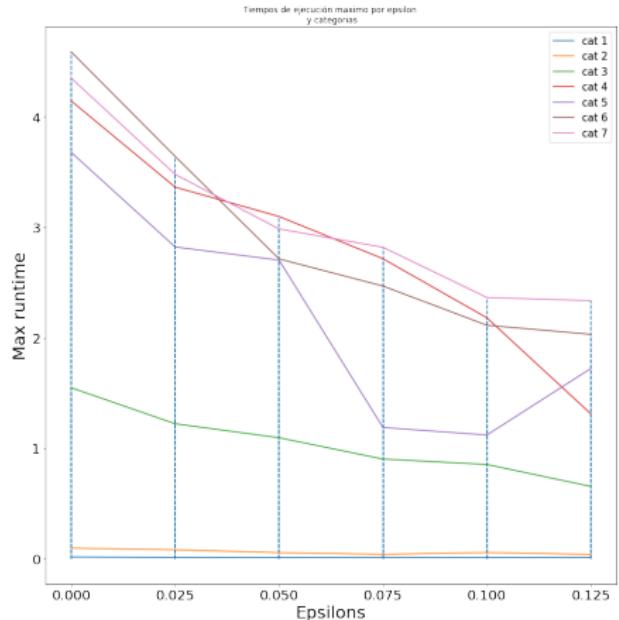


Fig. 14: Tiempos de ejecución máximos en segundos afectados por el epsilon en San Francisco y Chicago

# Cantidad de soluciones promedio

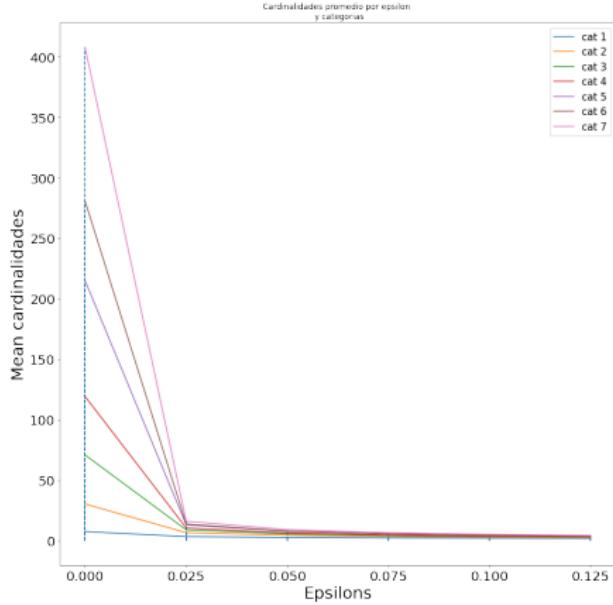
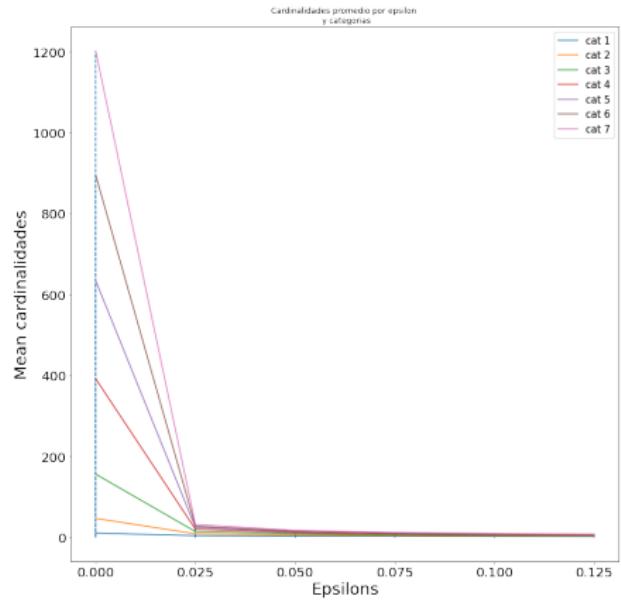


Fig. 15: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago

# Cantidad de soluciones promedio sin epsilon igual a cero

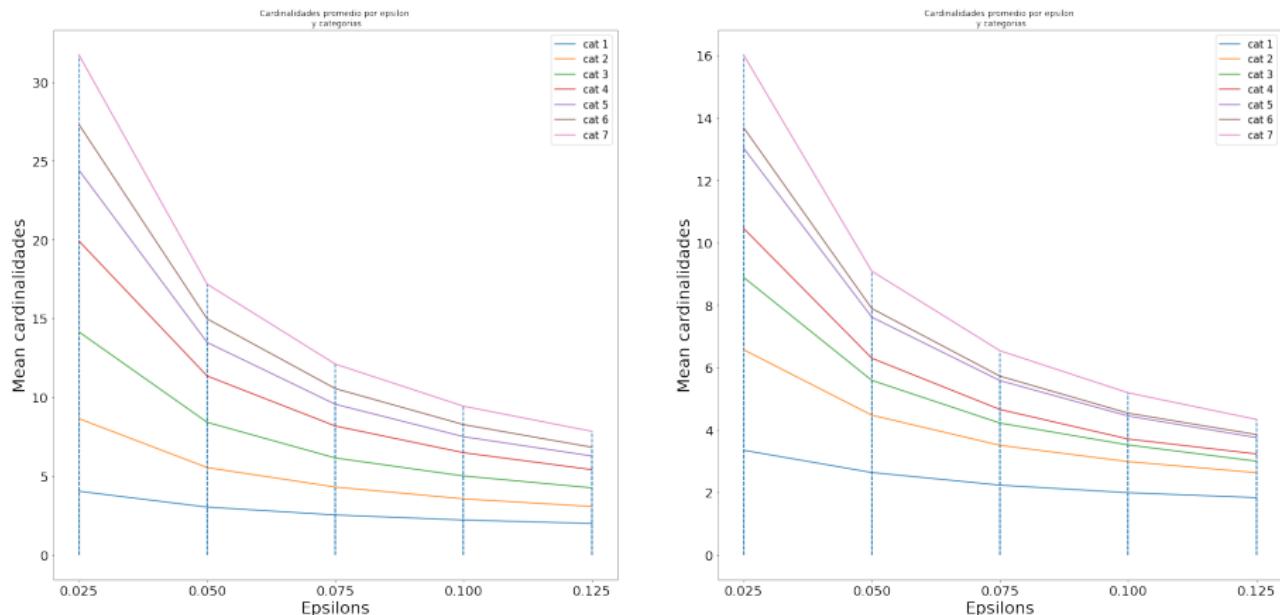


Fig. 16: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar  $\epsilon = 0$

# Cantidad de soluciones máximas

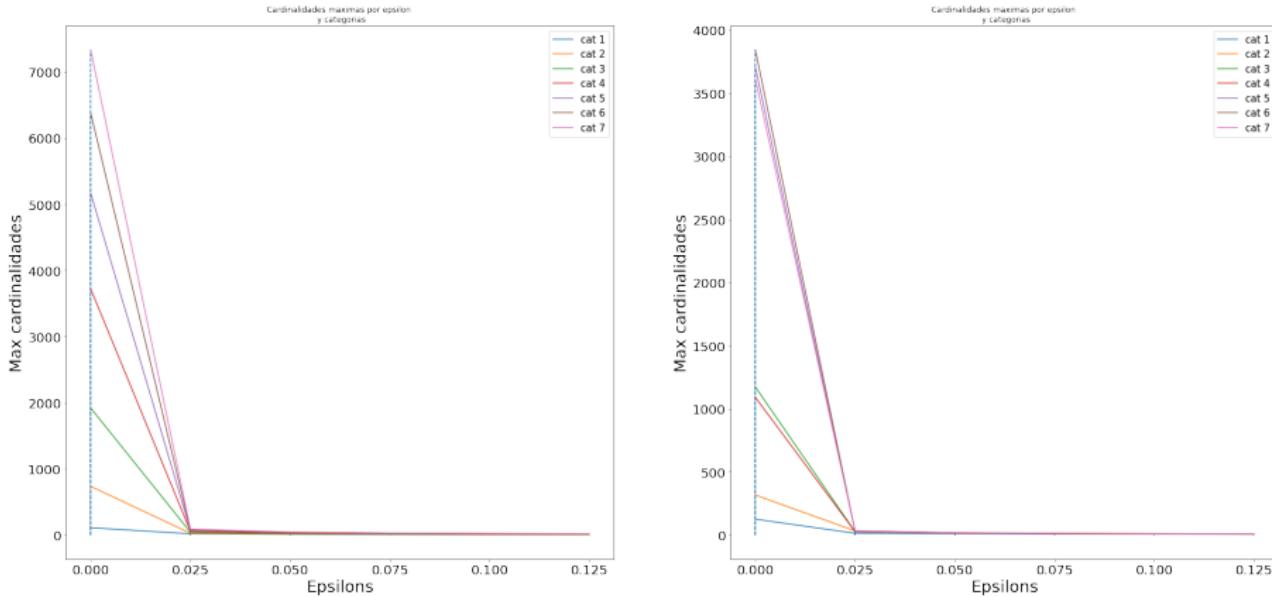


Fig. 17: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago

# Cantidad de soluciones máximas sin epsilon igual a cero

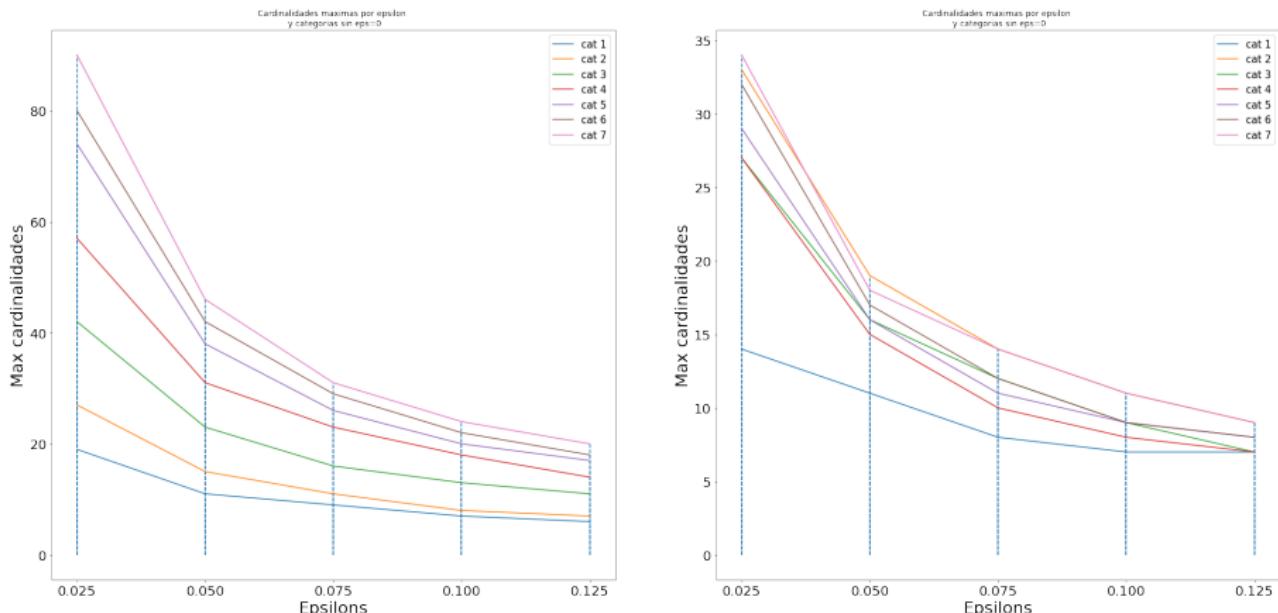


Fig. 18: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar  $\epsilon = 0$

# Cantidad promedio de nodos expandidos

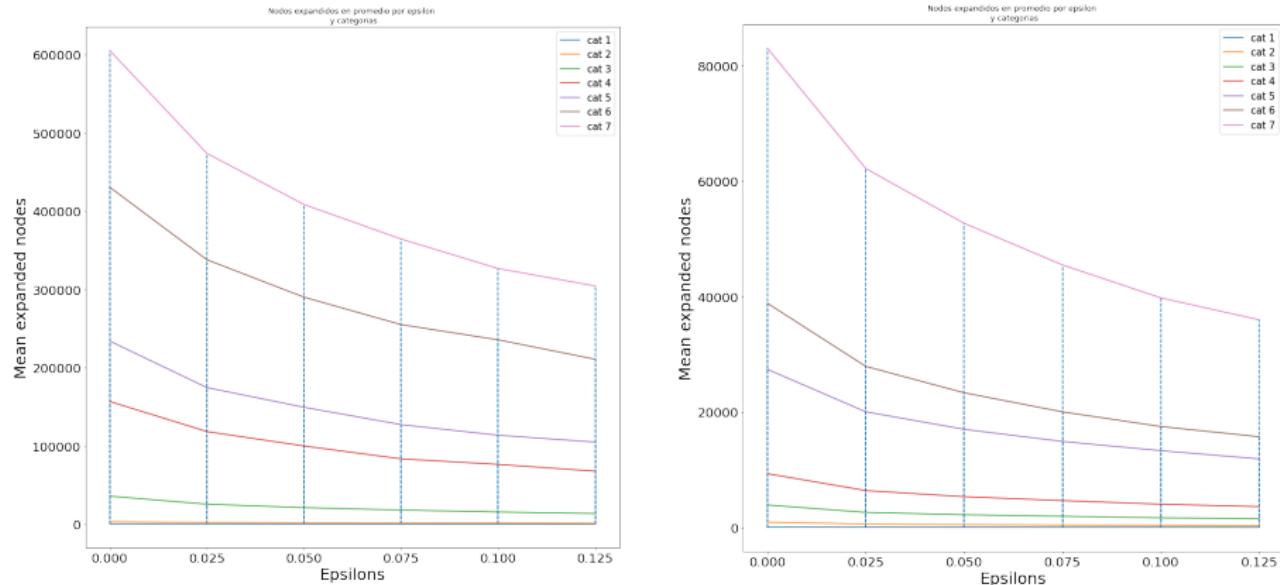


Fig. 19: Nodos expandidos en promedio afectados por el epsilon en San Francisco y Chicago

## Segunda ronda de experimentos

- Para San Francisco se trabajó con experimentos de categoría 10 pero solo se escogieron 36 nodos que funcionarán nodos iniciales. Se le asoció a cada uno 4 nodos objetivos teniendo finalmente **144 experimentos**.
- Para Chicago se trabajaron con experimentos de las categorías 10 y 15 del mismo modo que en la primera ronda de experimentos. Se tienen finalmente 400 experimentos tanto en la categoría 10 como en la 15.

# Tiempos de ejecución promedio

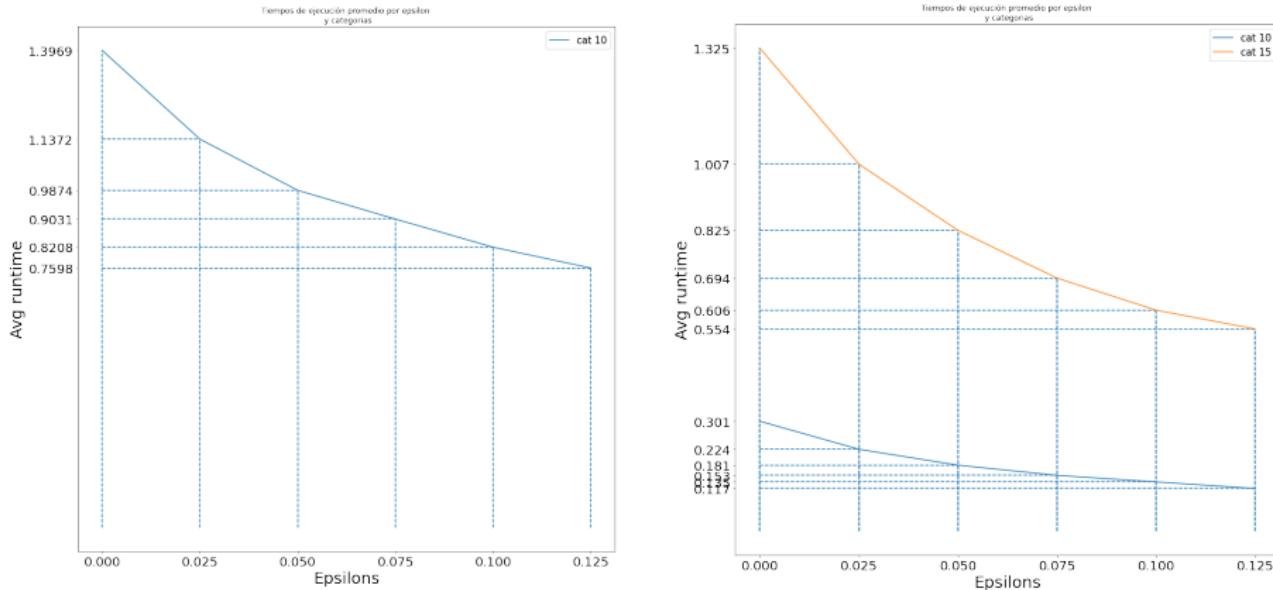


Fig. 20: Tiempos de ejecución promedio en segundos afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

# Tiempos de ejecución máximos

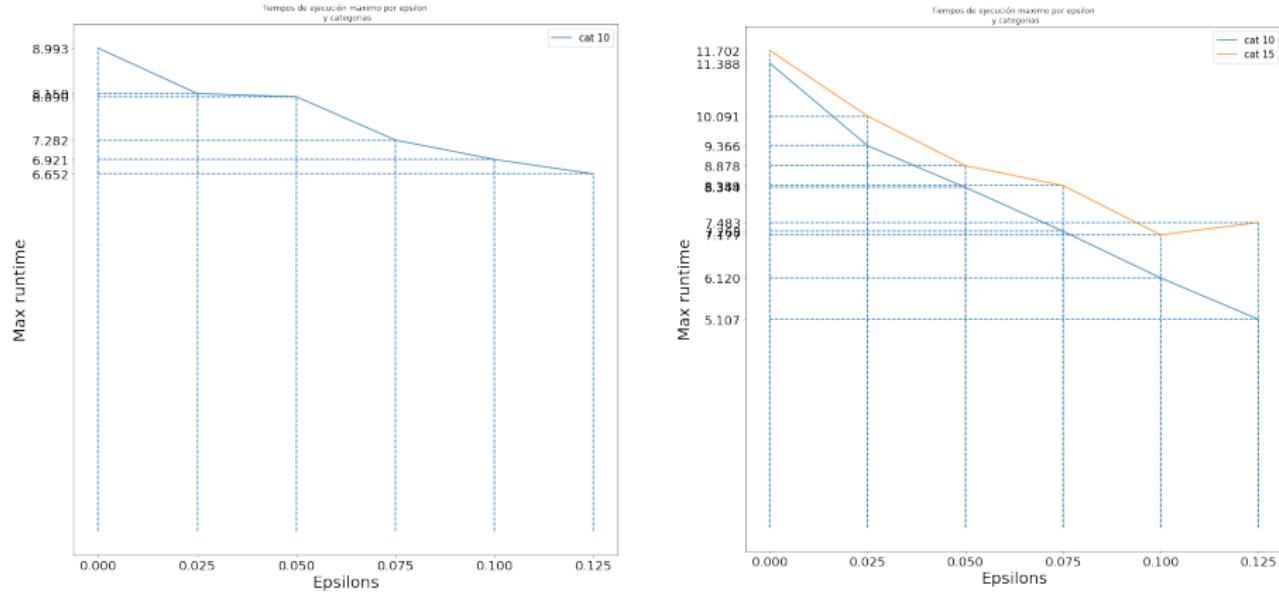


Fig. 21: Tiempos de ejecución máximos en segundos afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

# Cantidad de soluciones promedio

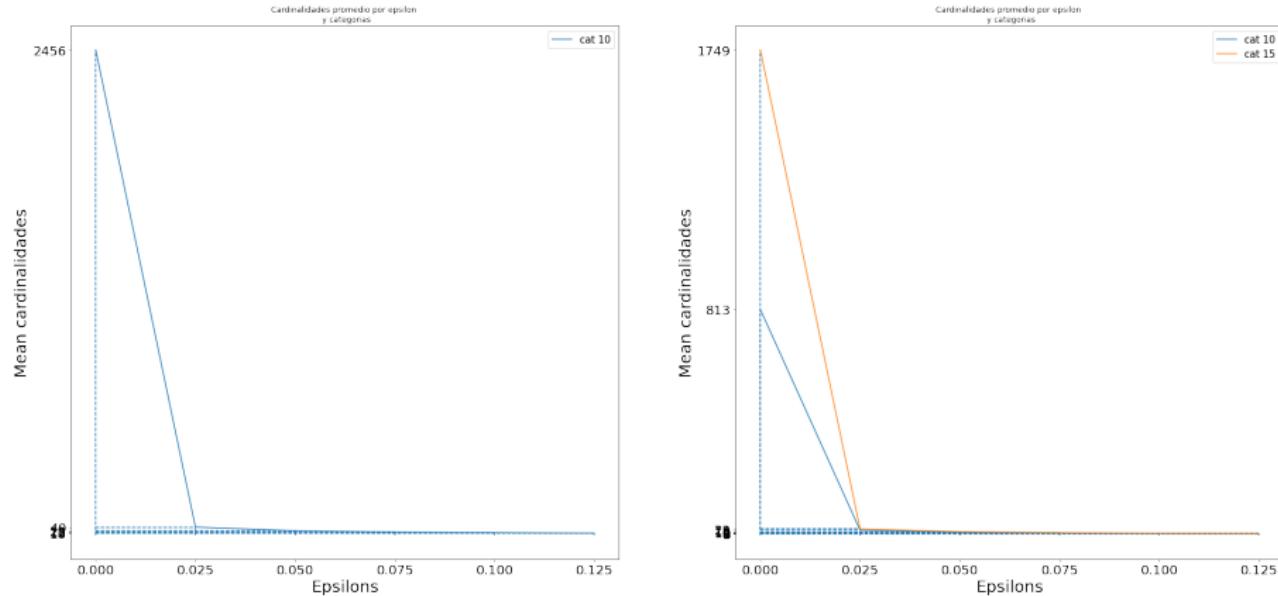


Fig. 22: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

# Cantidad de soluciones promedio sin epsilon igual a cero

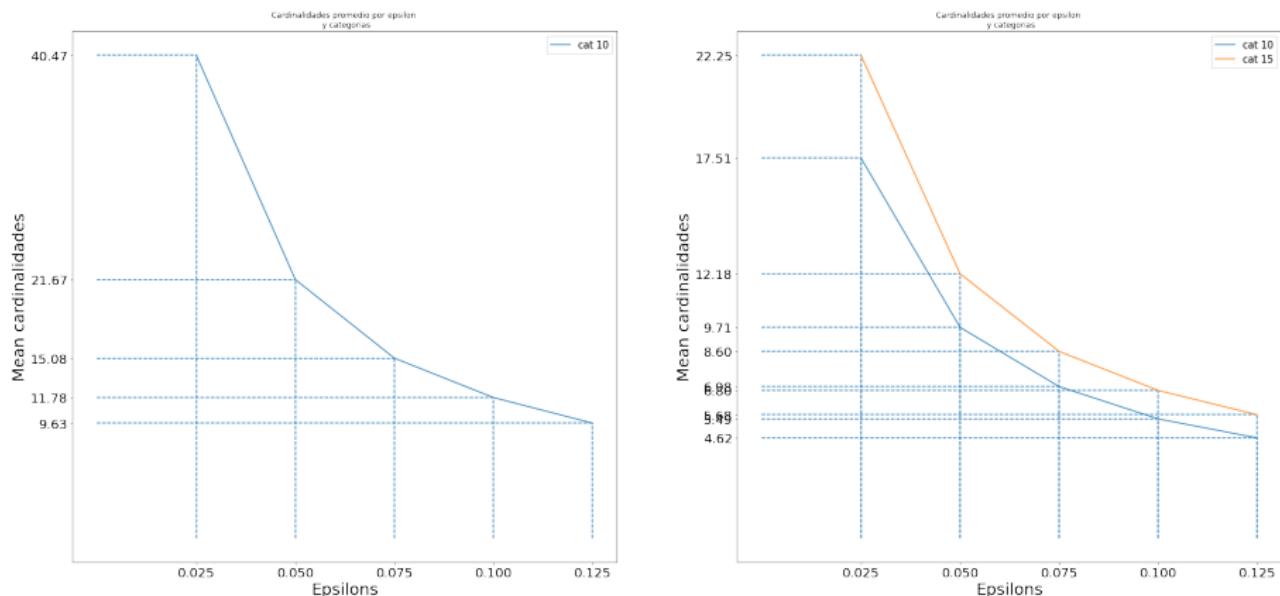


Fig. 23: Tamaños promedio del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar  $\epsilon = 0$  en la segunda ronda de experimentos

# Cantidad de soluciones máximas

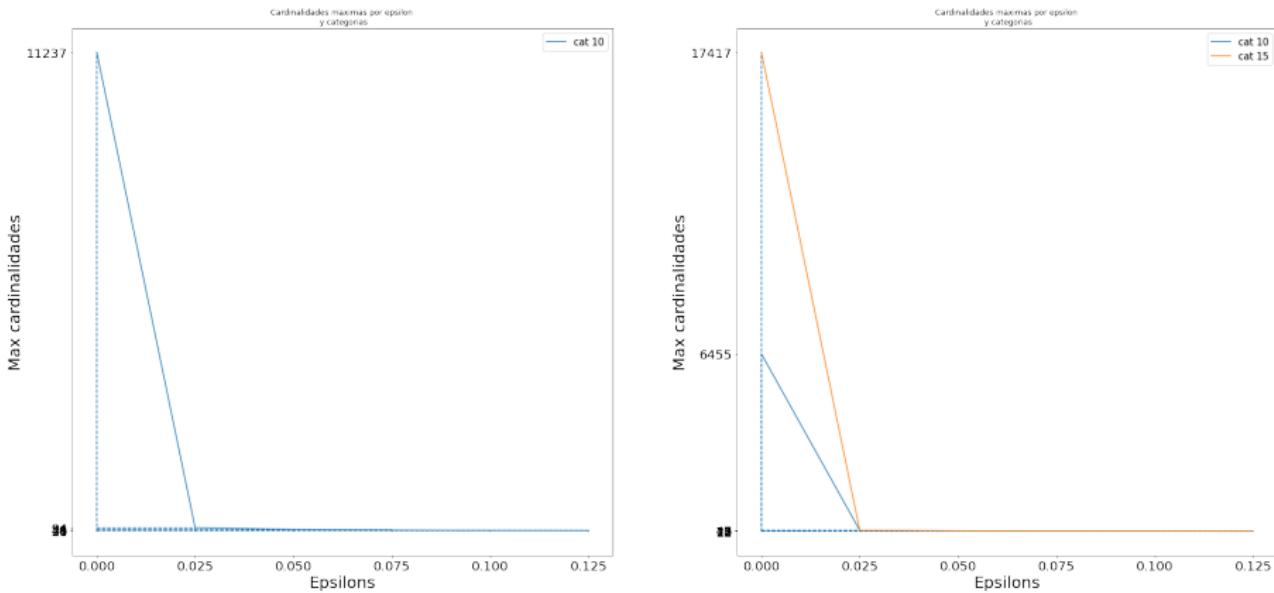


Fig. 24: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

# Cantidad de soluciones máximas sin epsilon igual a cero

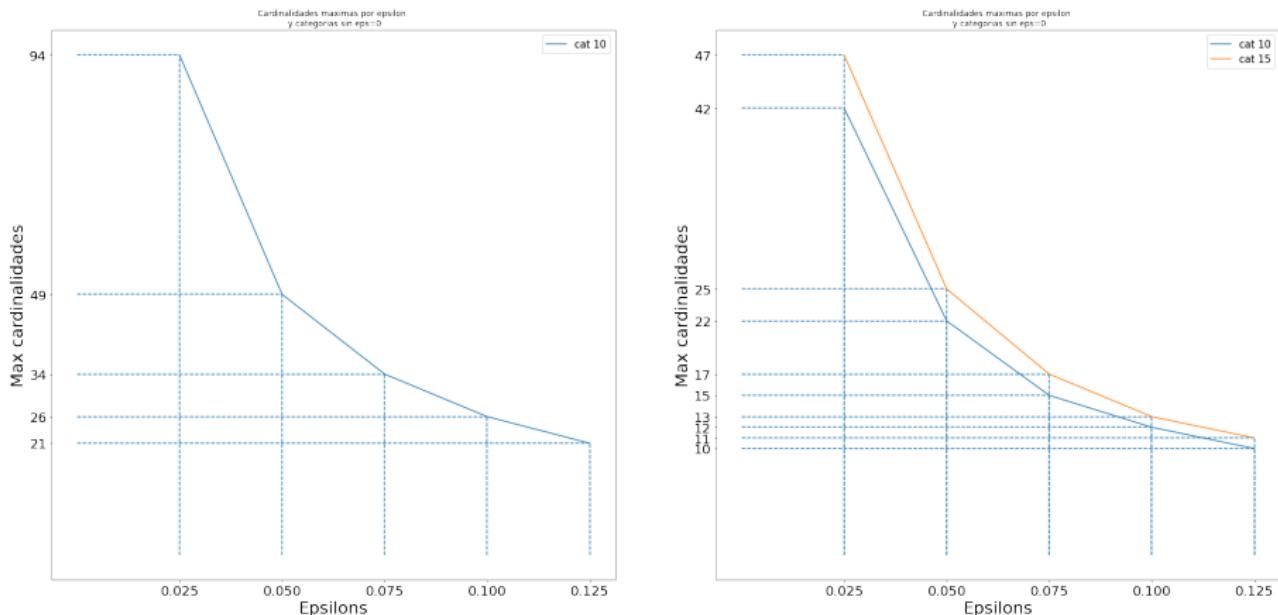


Fig. 25: Tamaños máximos del conjunto solución afectados por el epsilon en San Francisco y Chicago sin contar  $\epsilon = 0$  en la segunda ronda de experimentos

# Cantidad promedio de nodos expandidos

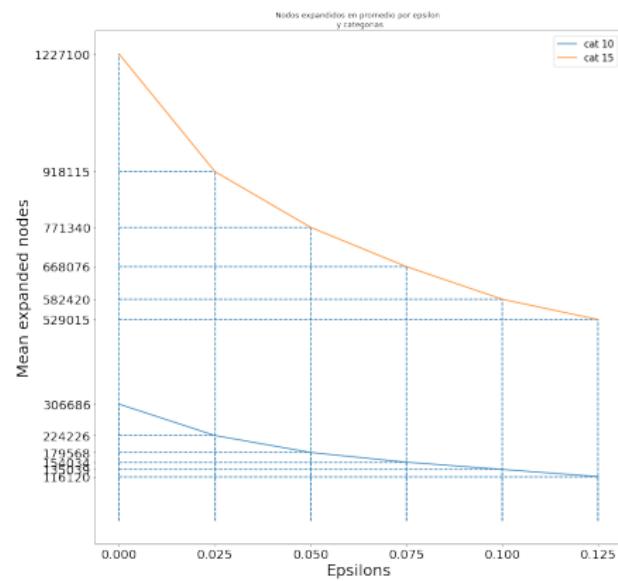
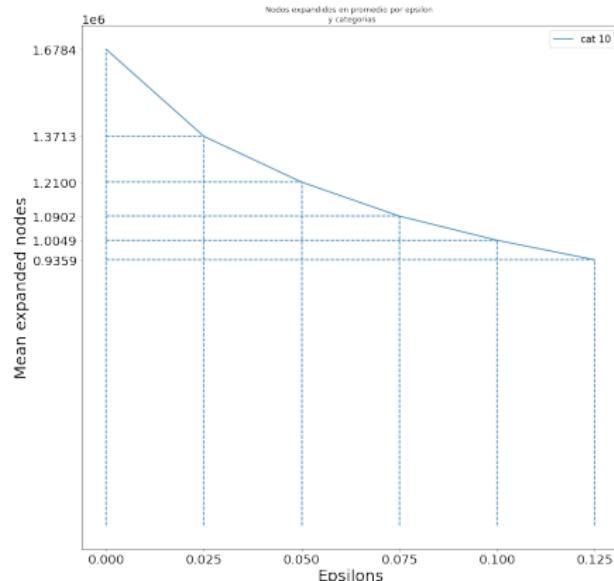


Fig. 26: Nodos expandidos en promedio afectados por el epsilon en San Francisco y Chicago en la segunda ronda de experimentos

# 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.

# Conclusiones

- 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.

# 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 [2] y evaluar si se puede aplicar la sugerencia planteada en [3] 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.

# FIN DE LA EXPOSICIÓN

## SECCIÓN DE PREGUNTAS Y SUGERENCIAS

# References I

-  **Urban navigation beyond shortest route: The case of safe paths.**  
*Information Systems*, 57:160–171, 2016.
-  Carlos Hernández Ulloa, William G. S. Yeoh, Jorge A. Baier, Han Zhang, Luis Suazo, and Sven Koenig.  
A simple and fast bi-objective search algorithm.  
In *ICAPS*, 2020.
-  Oren Salzman.  
Approximate bi-criteria search by efficient representation of subsets of the pareto-optimal frontier.  
In *ICAPS*, 2021.