TSCF - Predicción de trayectorias de usuario en el transporte público

Felipe Osimani y Leonardo Piñeyro
Docentes Matías Richart y Jorge Visca

Estructura de la presentación

- Introducción al proyecto
- Algoritmo trabajado
- Enfoque inicial
- Análisis de errores cometidos
- Cambio de foco y nuevas evaluaciones
- Conclusiones y trabajo futuro

Introducción al proyecto

- Recrear la investigación de Fengli Xu Et Al. sobre la recuperación de trayectorias de usuarios a través de datos agregados de torres de celulares
- Generar un ambiente simulado que logre recrear los datos reales
- Recuperar las trayectorias utilizando el algoritmo propuesto y comparar los resultados con el paper
- Utilizando lo aprendido en la investigación, aplicarlo al problema de recuperación de recorridos de usuarios en el sistema de transporte público
- Buscamos poder predecir recorridos de usuarios individuales a partir de datos de entradas/salidas de los ómnibus en cada parada

Algoritmo trabajado

Introducción al trabajo de Fengli Xu Et Al.

- Utilizan datos agregados del sistema de redes celulares o aplicaciones móviles
- Estos datos los presentan como cantidad de usuarios por torre celular en cada unidad de tiempo
- Fundamentándose en que las trayectorias humanas son muy repetitivas y únicas dicen lograr recuperar trayectorias individuales
- La privacidad entonces no se conservaría luego de agregar los datos

Estructura de datos

Los datos se presentan de la siguiente forma

	Torre A	Torre B	Torre C
P^{t0}	5	3	2
P^{t1}	3	4	3
P^{t2}	1	7	2

Algoritmo trabajado

Luego se construye una **distribución** los usuarios en una estructura L, para cada unidad de **tiempo**

$$L^{t_0} = [A_0, A_1, A_2, B_0, B_1, B_2, B_3, B_4, C_0, C_1]$$

En cada iteración del algoritmo se van formando las **trayectorias**

$$S^t = [s_1^t, s_2^t, \quad \dots \quad , s_N^t]$$

$$s_j^t = [q_j^1, q_j^2, \quad \dots \quad , q_j^t]$$

Algoritmo (cont.)

- Esta iteración de hace para todos los **usuarios**, y todos las **entradas de la distribución** en ese **tiempo** particular, logrando una matriz de costos
- En definitiva, para cada usuario se calcula el costo de que ese usuario se mueva a una nueva posición en el siguiente paso.

$$C^t = \{c^t_{i,j}\}_{N\times N},\ N = \#users \qquad C^t = \begin{bmatrix} c^t_{11} & c^t_{12} & \cdots & c^t_{1N} \\ \vdots & \ddots & & \\ c^t_{N1} & c^t_{N2} & \cdots & c^t_{NN} \end{bmatrix}$$

Algoritmo (cont.)

- Con dicha matriz de costos, vamos a calcular una matriz de decisión para elegir un nuevo elemento en cada traza de cada usuario para esta iteración.
- El problema se convierte entonces en un problema de minimización

$$Minimize \sum_{i=1}^{N} \sum_{j=1}^{N} c_{i,j}^{t} \times x_{i,j}^{t}$$

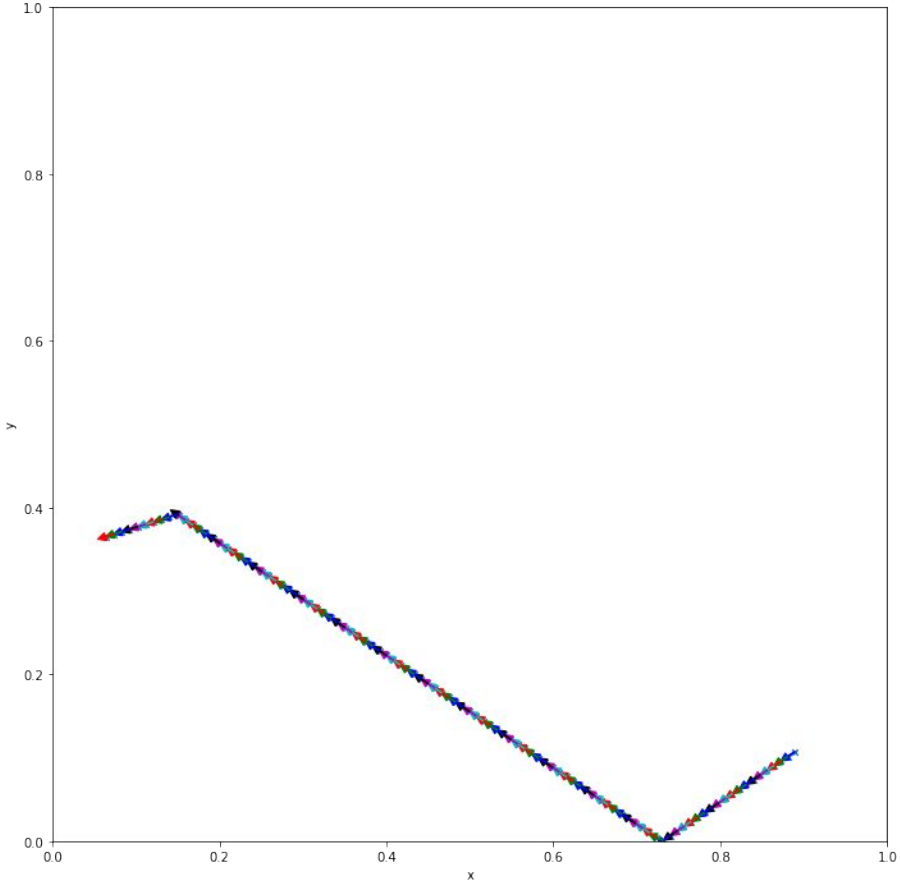
- Este tipo de problema ya es conocido como un **problema de asignación** y lo solucionaremos con el **algoritmo del Húngaro** que es el mejor algoritmo hallado para la resolución de estos problemas.

Enfoque inicial

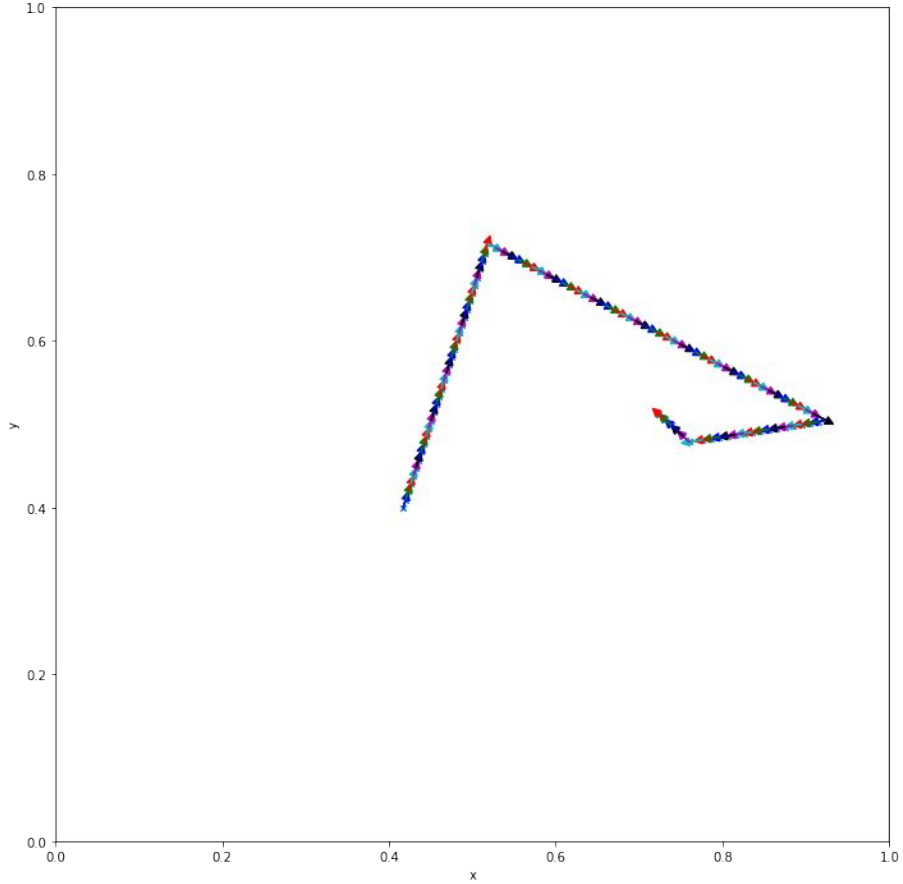
Enfoque inicial

- Estudiamos e implementamos el algoritmo descrito por Fengli Xu Et Al.
- Creamos un generador de **simulaciones** de usuarios utilizando modelos de movilidad estándares.
- Generamos gran cantidad de datos sobre el rendimiento del algoritmo en diferentes escenarios
    - Variamos **cantidad de usuarios**, **cantidad de torres**, **velocidad de los usuarios** y **modelo de movilidad**.
- El proceso entero consta de 3 pasos: **simulación**, **recuperación**, y **evaluación**.

Random Direction

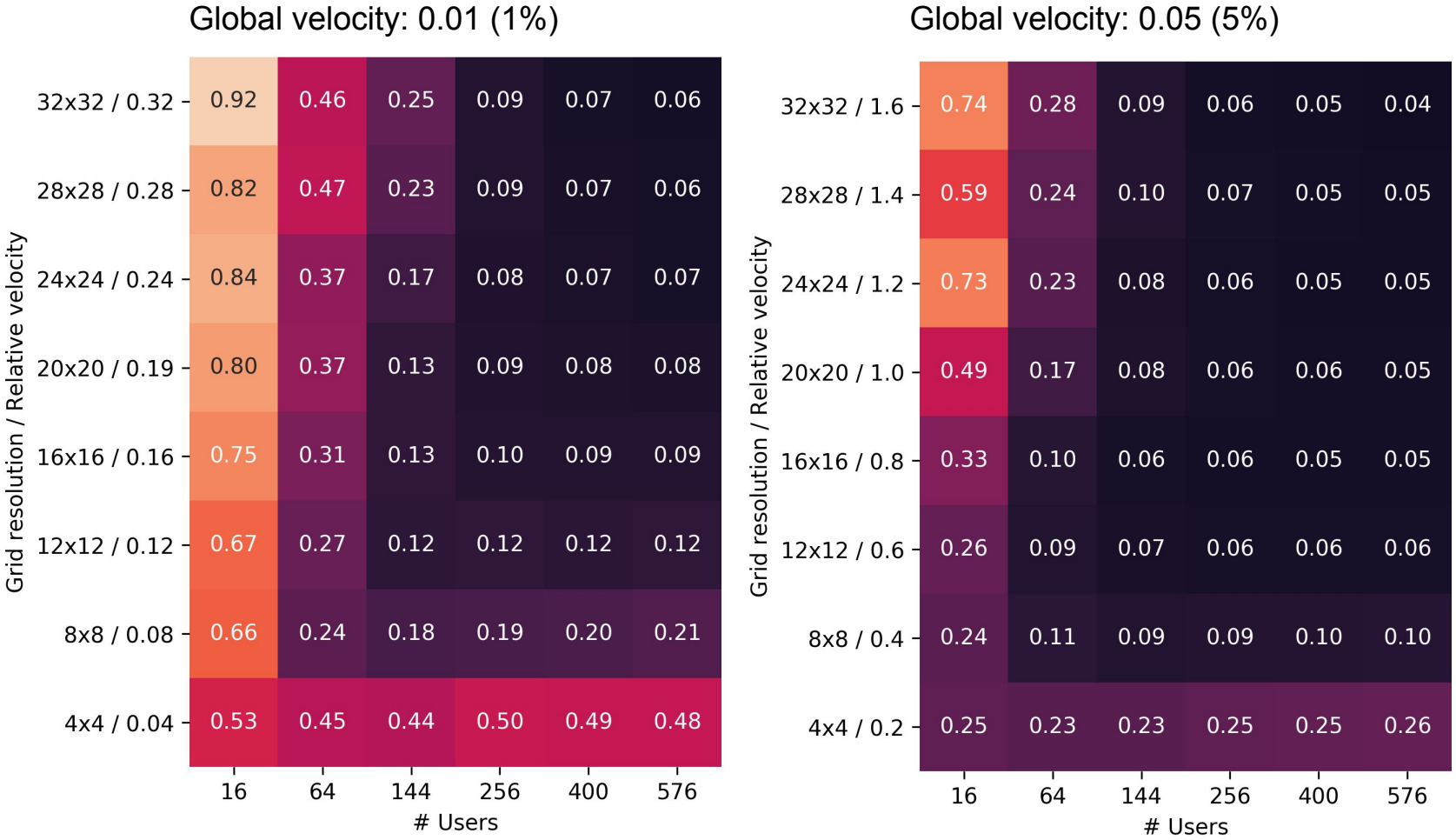


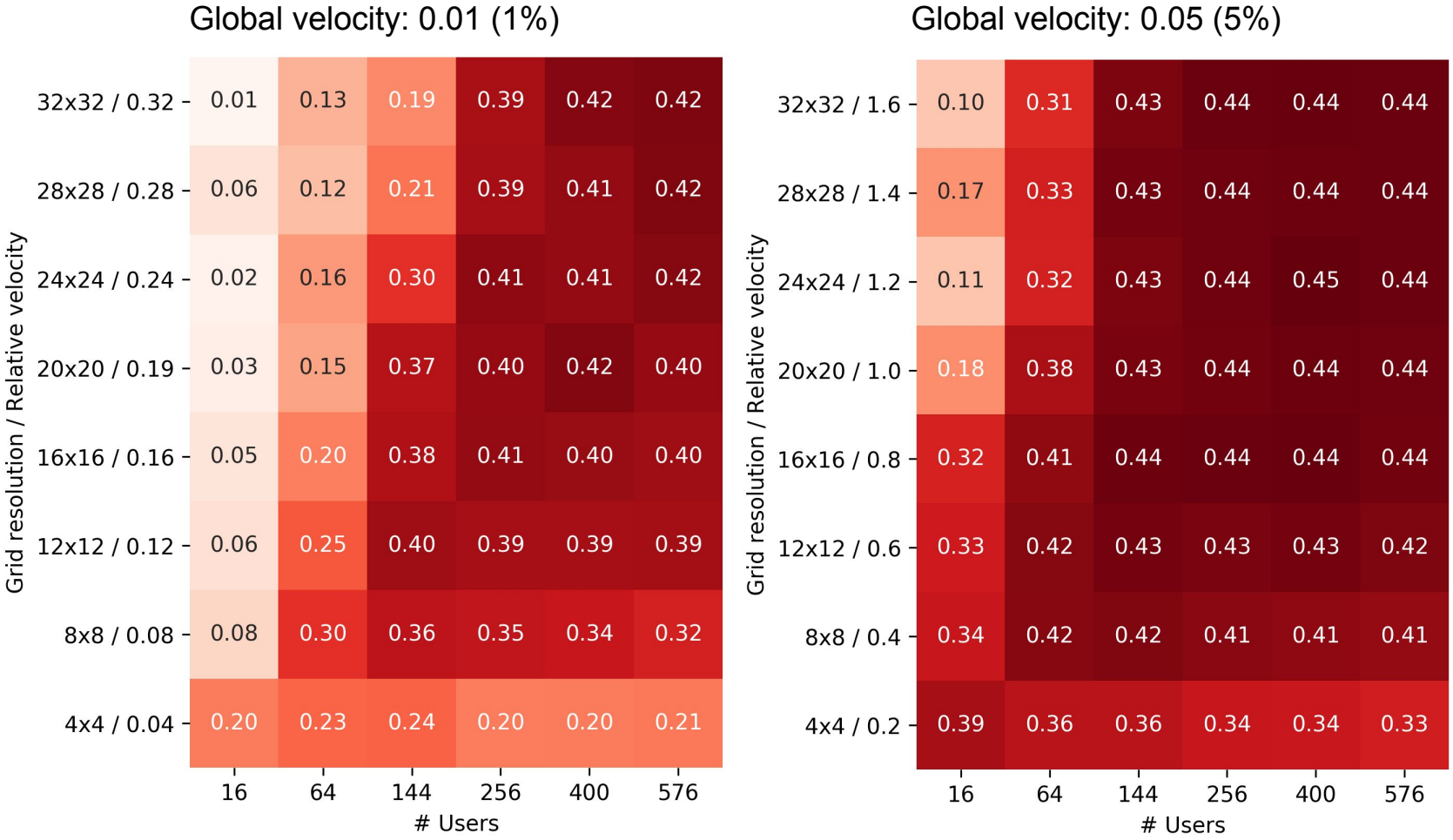
Random Waypoint



Evaluación inicial

- Resultados cambiando **velocidad global**, número de **usuarios** y **torres**
- También se corrieron pruebas utilizando diferentes modelos de movilidad: *random direction* y *random waypoint*.


Global velocity: 0.01 (1%)
Grid resolution / Relative velocity
32x32 / 0.32
28x28 / 0.28
24x24 / 0.24
20x20 / 0.19
16x16 / 0.16
12x12 / 0.12
8x8 / 0.08
4x4 / 0.04
0.92 0.46 0.25 0.09 0.07 0.06
0.82 0.47 0.23 0.09 0.07 0.06
0.84 0.37 0.17 0.08 0.07 0.07
0.80 0.37 0.13 0.09 0.08 0.08
0.75 0.31 0.13 0.10 0.09 0.09
0.67 0.27 0.12 0.12 0.12 0.12
0.66 0.24 0.18 0.19 0.20 0.21
0.53 0.45 0.44 0.50 0.49 0.48
16 64 144 256 400 576
Users
Global velocity: 0.05 (5%)
Grid resolution / Relative velocity
32x32 / 1.6
28x28 / 1.4
24x24 / 1.2
20x20 / 1.0
16x16 / 0.8
12x12 / 0.6
8x8 / 0.4
4x4 / 0.2
0.74 0.28 0.09 0.06 0.05 0.04
0.59 0.24 0.10 0.07 0.05 0.05
0.73 0.23 0.08 0.06 0.05 0.05
0.49 0.17 0.08 0.06 0.06 0.05
0.33 0.10 0.06 0.06 0.05 0.05
0.26 0.09 0.07 0.06 0.06 0.06
0.24 0.11 0.09 0.09 0.10 0.10
0.25 0.23 0.23 0.25 0.25 0.26
16 64 144 256 400 576
Users


Global velocity: 0.01 (1%)
Grid resolution / Relative velocity
32x32 / 0.32
28x28 / 0.28
24x24 / 0.24
20x20 / 0.19
16x16 / 0.16
12x12 / 0.12
8x8 / 0.08
4x4 / 0.04
0.01 0.13 0.19 0.39 0.42 0.42
0.06 0.12 0.21 0.39 0.41 0.42
0.02 0.16 0.30 0.41 0.41 0.42
0.03 0.15 0.37 0.40 0.42 0.40
0.05 0.20 0.38 0.41 0.40 0.40
0.06 0.25 0.40 0.39 0.39 0.39
0.08 0.30 0.36 0.35 0.34 0.32
0.20 0.23 0.24 0.20 0.20 0.21
16 64 144 256 400 576
Users
Global velocity: 0.05 (5%)
Grid resolution / Relative velocity
32x32 / 1.6
28x28 / 1.4
24x24 / 1.2
20x20 / 1.0
16x16 / 0.8
12x12 / 0.6
8x8 / 0.4
4x4 / 0.2
0.10 0.31 0.43 0.44 0.44 0.44
0.17 0.33 0.43 0.44 0.44 0.44
0.11 0.32 0.43 0.44 0.45 0.44
0.18 0.38 0.43 0.44 0.44 0.44
0.32 0.41 0.44 0.44 0.44 0.44
0.33 0.42 0.43 0.43 0.43 0.42
0.34 0.42 0.42 0.41 0.41 0.41
0.39 0.36 0.36 0.34 0.34 0.33
16 64 144 256 400 576
Users

Enfoque inicial - Conclusiones generales

- Claramente, cuanto más usuarios y torres agreguemos, menos la accuracy y mayor el error
  - La cantidad de usuarios afecta más que la cantidad de torres
- Las variables utilizadas afectaban en varias medidas al algoritmo porque no son independientes.
- Esto parece seguir los resultados obtenidos por los autores
  - En el paper, presentan una accuracy de 73% al 92% con torres agrupadas en “distritos” y números mucho peores para distritos más chicos
  - Esto es replicable a nuestra variación de cantidad de torres. 16 torres pueden ser vistas como 1600 torres en 10 distritos ➜ **ERROR!!!**
- Pensar que la velocidad del usuario es un concepto equivalente al sampling de datos ➜ **ERROR!!!**

Análisis de errores cometidos

Análisis de errores cometidos

- Pensar que reducir la cantidad de torres es equivalente a tomar distritos es un error
  - Modifica la calidad de la función de costos ya que los saltos entre torre y torre, los saltos que la función de costos predice, son más grandes
- Si el usuario se mueve lento, al momento de saltar de una torre a otra va a generar un salto más grande
- Pensar en distritos sólo cambia el accuracy del algoritmo, no su forma de actuar y predecir

Análisis de errores cometidos (cont)

- Los autores adecuaron sus datos a su algoritmo con diferentes sampling rates
  - Esto es, tomaron datos cada media hora, una hora, etc. y utilizaron los que les daba mejor
- Necesitamos una forma de evaluar nuestro algoritmo de la mejor forma posible

Cambio de foco y nuevas evaluaciones

Cambio de foco

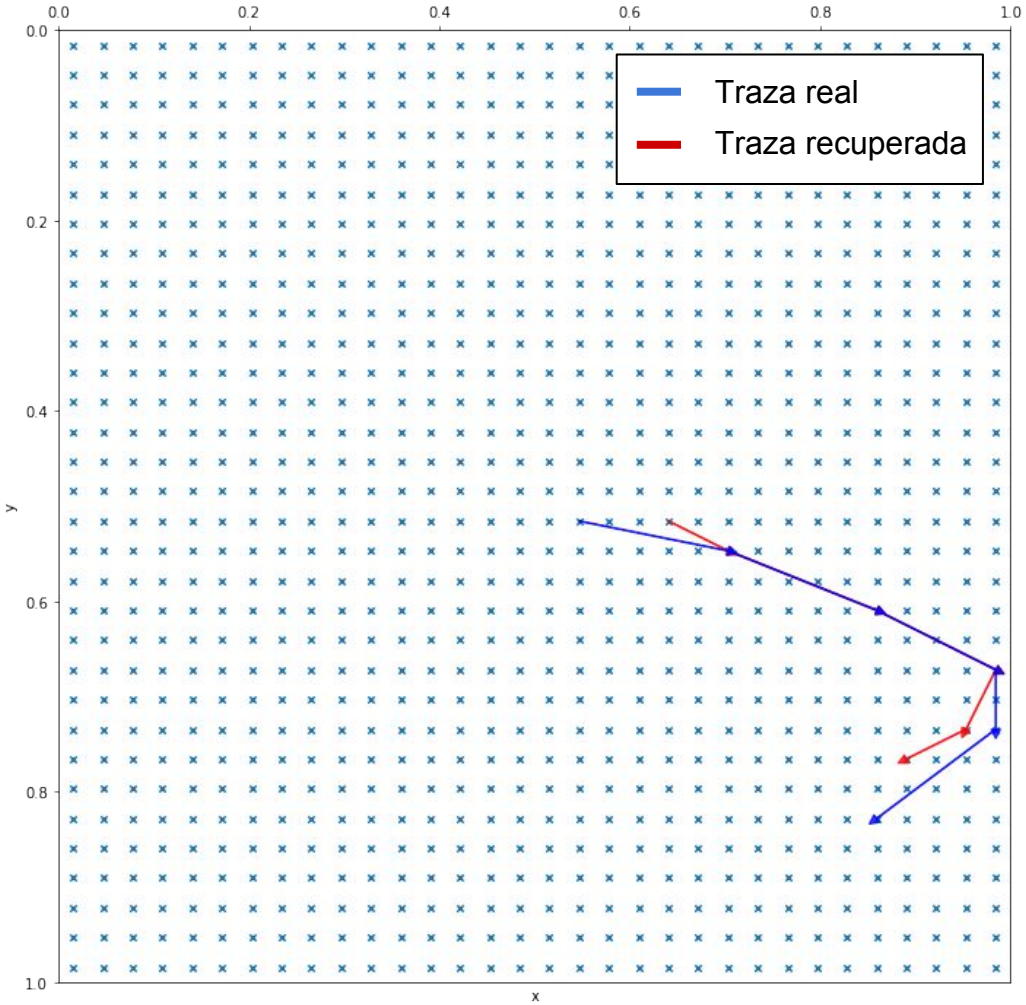
- Elegir un escenario particular de todas nuestras simulaciones
  - Algo que parezca similar a lo utilizado por los autores
- Correrlo con más ciclos
  - 96 en vez de 24 que fue el valor inicial utilizado
- Sobre la misma simulación, cambiar el sample rate
  - Tomamos samples de a 1, 2, 3, 4, 8 y 16
- Sobre cada una de estas simulaciones, evaluarla con distritos de diferentes tamaños
  - Medimos con distritos de tamaño 1 (original), 2, 4 y 8

Nueva evaluación - Nuevas simulaciones

- Resultados con cantidad de torres y usuarios fija: 576 usuarios y 1024 torres
- Introducimos a las pruebas los conceptos de **distritos** y **muestreo**
- Notar que el muestreo está involucrado en el proceso de recuperación, mientras que el uso de distritos de usa en el proceso posterior de evaluación del algoritmo.

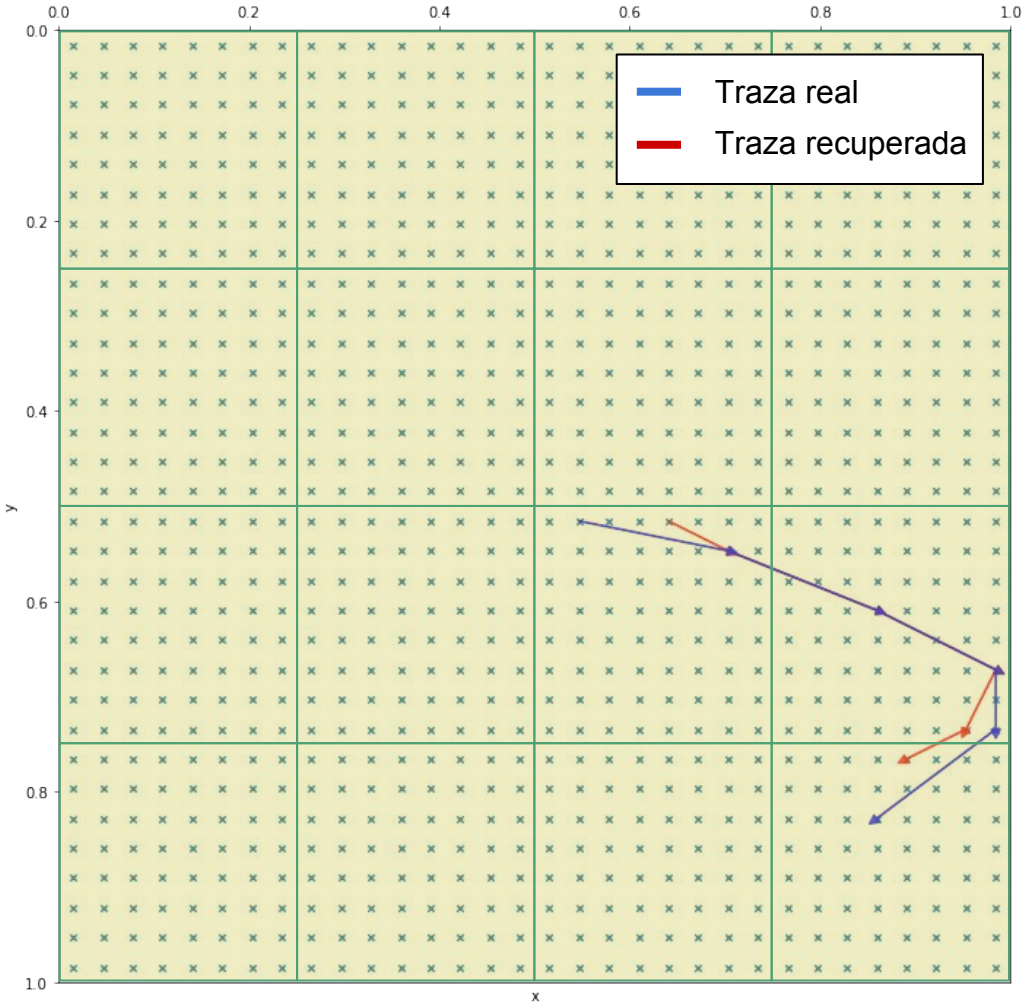
Distritos: 1x1

Accuracy: 50%

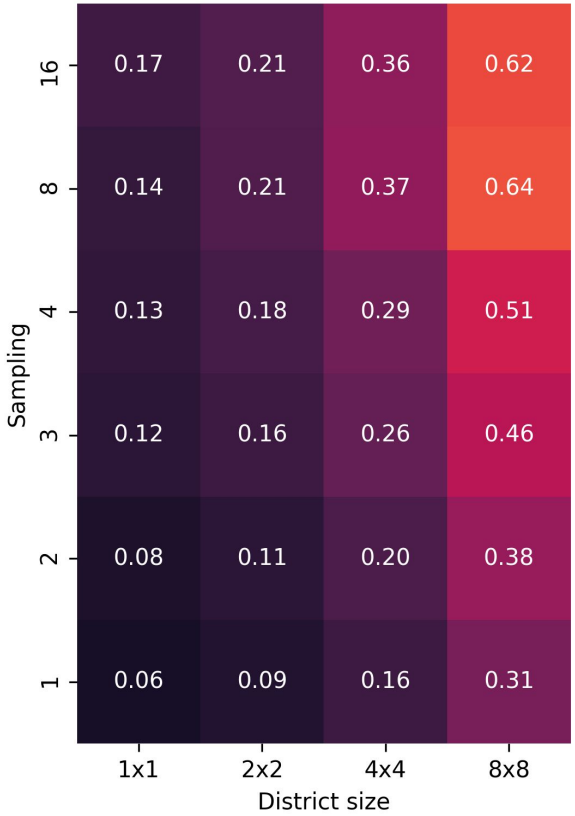


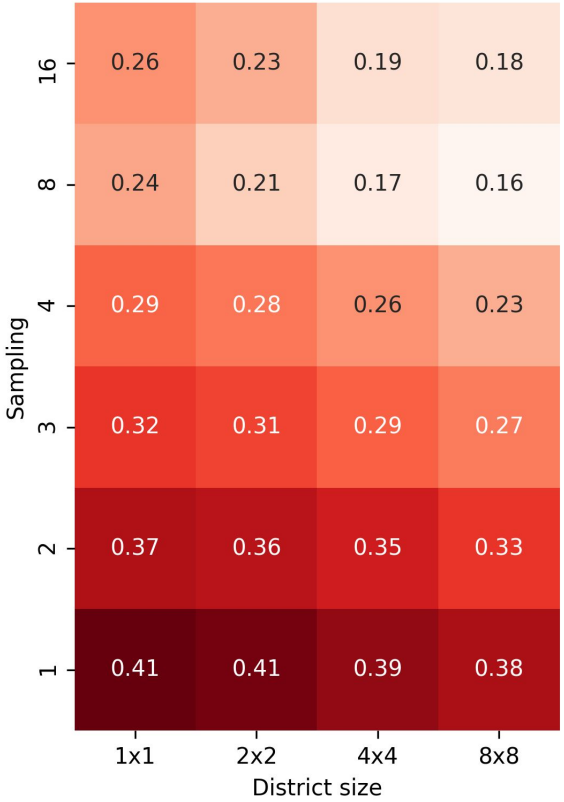
Distritos: 8x8

Accuracy: 100%



Usuarios: 576 **Torres:** 1024 **Velocidad global:** 0.01 **Pasos de simulación:** 96





Conclusiones y trabajo futuro

Conclusiones

- El **sampling** afecta directamente el **accuracy** porque su valor está ligado a la definición de nuestra función de costos.
- Un **sampling** que se asemeje a lo que nuestra función de costos espera del movimiento de un usuario, mejora la precisión del algoritmo.
- Que el sampling sea muy alto afecta mínimamente en comparación a un sampling muy bajo.
- Los **distritos** cumplen la función de “suavizar” la exactitud de la precisión con la que se evalúa al algoritmo.
    - Cuanto más grandes los distritos, más rápido es el cambio en el rendimiento del algoritmo al cambiar el sampling rate

Conclusiones (cont.)

- Con tamaños de distritos suficientemente grandes logramos obtener resultados cercanos a los del paper estudiado
- Si el objetivo es lograr una precisión y granularidad alta el algoritmo no es una buena solución
- Si no se sabe la forma que se mueven los usuarios, la función de costos podría no adaptarse bien

Trabajo a futuro

- Probar nuestras conclusiones con modelos de movilidad más complejos, o mejor aún, con datos reales
- Investigar si esta función de costos podría aplicarse al problema de predicción de trayectorias de usuarios en el sistema de ómnibus