Universidad del Bío-Bío

Facultad de Ciencias Empresariales

Departamento de Sistemas de Información

Evaluación de algoritmos de Caching en redes P2P inalámbricas para proteger la privacidad de ubicación

Proyecto de título presentado por Fernando Andrés Vera Catricura

para optar al título de Ingeniero Civil en Informática

Dirigida por Dr. Patricio Galdames Sepúlveda

Septiembre del 2020

Concepción – Chile

Resumen

En los últimos años, los servicios basados en la ubicación (LBSs) han sido participe en las actividades diarias de las personas, ya que estos servicios ofrecen información geográfica acorde a la ubicación que se encuentre el usuario. Para obtener esta información, este debe enviar una consulta basada en la ubicación (LBQ) hacia al servidor LBS a través de su dispositivo móvil. Sin embargo, la ubicación que se expone no está protegida cuando se envía la LBQ, dado que la ubicación se considera un atributo de carácter privado. Por eso, este proyecto de título se implementan dos algoritmos de Caching que ayudan a preservar la privacidad de ubicación del usuario, mitigando el uso del servidor LBS.

La implementación se realizó en un simulador creado por autor de este trabajo, que entrego resultados donde se refleja las ventajas y desventajas de cada algoritmo. Este proyecto de titulo pretende aportar un conocimiento sobre los algoritmos de Caching en la privacidad de ubicación de los usuarios que usan los LBSs.

Palabras Clave – Servicios basados en la ubicación, Caching, privacidad de ubicación, P2P

Índice General

Índice de Ilustraciones

Índice de Tablas

Índice de ecuaciones

1. Introducción

Durante estos últimos años, los servicios basados en la ubicación (LBS) han aumentado su popularidad en la vida diaria de las personas (Arabia, 2017), porque son servicios que ofrecen información geográfica dependiendo de la ubicación que se encuentre el usuario. Para que este pueda conseguir esta información, debe emitir una consulta basada en la ubicación (LBQ) a través de las aplicaciones que ofrecen un LBS, como Google Maps o Yelp. El usuario envía su LBQ hacia al servidor LBS, como por ejemplo una consulta de rango donde se identifiquen todos los puntos de interés (POI) que caben dentro de un área situada en la ubicación del usuario, o bien, los K-POIs más cercanos a su ubicación. Por ejemplo, si un usuario en su horario de colación busca los restaurantes más cercanos de su lugar de trabajo para ir a comer.

Sin embargo, existe un riesgo de la privacidad del usuario cuando utiliza un LBS (Wang, Hu, Sun, & Huang, 2018). Debido a que, la ubicación y la consulta misma (los atributos que conforman la LBQ) declarada en la LBQ, son datos que se pueden correlacionar y le permite al servidor LBS (adversario) identificar e inferir los patrones de comportamiento del usuario, así deducir su estilo de vida. Por ejemplo, si un usuario durante las tardes consulta por pubs o restaurantes, se puede inferir que llega tarde a su hogar y le gusta beber alcohol. Otro ejemplo, es si un usuario realiza su consulta desde un hospital, por tanto, se podría inferir que padece de una enfermedad.

Para abordar el problema de la privacidad del usuario en LBS, existen trabajos previos que dividen la privacidad del usuario en dos dimensiones: privacidad de ubicación y privacidad de consulta. La privacidad de ubicación consiste en preservar la coordenada en donde el usuario envía su LBQ al servidor LBS, logrando que no se descubra la ubicación exacta del usuario. En cambio, La privacidad de consulta consiste en preservar los atributos que componen LBQ, logrando una ambigüedad en los atributos, así sea difícil para el servidor LBS inferir en la información personal del usuario. Por ello, este proyecto de título solo se enfoca en proteger la privacidad de ubicación, implementando dos algoritmos de Caching, con el objetivo de analizar si estos algoritmos logran preservar la ubicación adecuadamente cuando se envían LBQs al servidor LBS.

2. Definición del Proyecto

2.1 Objetivo General

Evaluar e implementar algoritmos de Caching conscientes de la privacidad para el procesamiento de consultas basadas en la ubicación en redes P2P inalámbricas.

2.2 Objetivos Específicos

1. Realizar búsqueda bibliográfica de algoritmos de Caching empleados en redes P2P inalámbricas para proteger la privacidad de los usuarios.

2. Comparar algoritmos de Caching de la revisión bibliográfica, y luego implementar dos escogidos.

3. Desarrollar un simulador para visualizar el efecto que tienen los algoritmos implementados.

2.3 Descripción de la metodología de investigación a utilizar

- Estudiar de forma sistemática la literatura que permita conocer los algoritmos de Caching en servicios basados en la ubicación
- Aplicar criterios de inclusión y exclusión de los primeros trabajos obtenidos de la revisión sistemática.
- Analizar los trabajos seleccionados de la revisión para conocer los algoritmos utilizados.
- Implementar los algoritmos seleccionados en un simulador para que se pueda visualizar los efectos que realiza.
- Comparar los algoritmos implementados en el simulador generando conclusiones claras y concisas.

2.4 Estructura Informe

El proyecto de título se organiza de la siguiente manera:

Capítulo 2. Se define los objetivos y metodología del proyecto de título.

Capítulo 3. Se establece las bases teóricas del proyecto de título como los componentes de los servicios basados en la ubicación y privacidad de ubicación. Se detalla la planificación y resultados de la revisión bibliográfica de los algoritmos de Caching.

Capítulo 4. Se explica los algoritmos de Caching escogidos de la revisión bibliográfica para evaluarlos.

Capítulo 5. Se desarrollan las pruebas de los Algoritmos de Caching en un simulador.

Capítulo 6. Se analiza los resultados de las pruebas de los algoritmos para realizar una comparación detallada con los resultados obtenidos de cada uno.

Capítulo 7. Se realiza una conclusión sobre los algoritmos de Caching evaluados a partir de los análisis y pruebas obtenidas del simulador y se plantea posibles trabajos futuros.

Capítulo 8. Se presentan las referencias ocupadas en el desarrollo de este proyecto.

Capítulo 9. Anexos del proyecto de título.

3. Marco Teórico

3.1 Conceptos Fundamentales

3.1.1 Servicios basados en la ubicación

Los servicios basados en la ubicación (LBS), son servicios que proveen información geográfica a través de consultas basadas en la ubicación (LBQ) que realiza el usuario a estos, utilizando un esquema cliente-servidor para su comunicación (Arabia, 2017)(Liu & Zhou, n.d.). Para enviar un LBQ hacia a LBS, el usuario debe liberar su posición, utilizando el sistema de posicionamiento global (GPS) y tener una conexión a internet mediante una red inalámbrica en su dispositivo móvil. Luego, el usuario a través de una aplicación de servicios LBS instalada en su dispositivo móvil, ingresa el punto de interés (POI) que desea consultar y envía hacia a LBS. Después de la emisión de la consulta, LBS de vuelve la respuesta al usuario, indicando donde se encuentra el POI acorde a la posición en que este usuario, como se muestra en la ilustración 1. Por ejemplo, si una familia desea salir a comer de noche a un restaurante, pero no saben que restaurantes se encuentran abierto, requerirá utilizar estos servicios LBS, enviando un LBQ por medio de sus aplicaciones (Google Maps o Yelp). También, esta familia puede requerir al LBS nuevamente, en el caso que ellos se vayan en automóvil hacia al restaurante, para saber cuál es la ruta menos congestionada para llegar al restaurante o la ruta más corta para llegar al restaurante. Por lo tanto, el LBS es fundamental en la vida diaria de las personas, añadiendo un valor agregado en la calidad de vida.

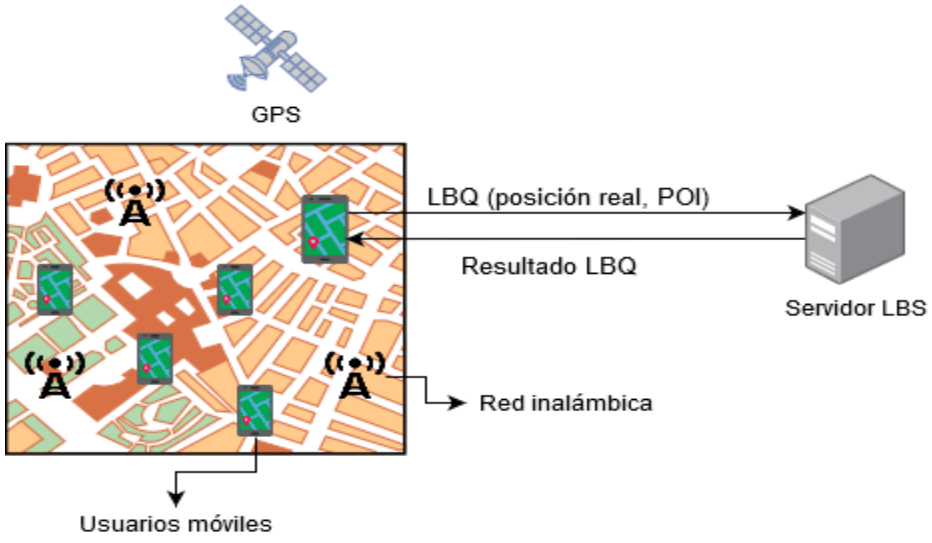


Ilustración 1. Funcionamiento de los servicios basados en la ubicación.

3.1.2 Privacidad de ubicación

En los trabajos que abordan la privacidad de ubicación en los LBSs, no tienen una definición formal de la privacidad de ubicación, solo se entiende como la preservación de la coordenada en donde el usuario envía su LBQ al servidor LBS, logrando que no se descubra la ubicación exacta del usuario. Pero, un trabajo intenta de definirlo como *:" la capacidad de un individuo de moverse en el espacio público con la expectativa de que, en circunstancias normales, su ubicación no será registrada sistemáticamente ni secretamente para su uso posterior"* (Blumberg & Eckersley, 2009).

Para proteger la privacidad de ubicación, existen trabajos que aplican ciertos mecanismos de protección con enfoque criptográfico, enfoque de anonimato y enfoque colaborativo (Liu & Zhou, n.d.). Los mecanismos con enfoque criptográfico tratan de encriptar la ubicación del usuario. En el trabajo (Mascetti, Freni, Bettini, Wang, & Jajodia, 2011) se propuso un enfoque que notifica a los usuarios cuando sus amigos (llamados buddies) están cerca, pero sin revelar la posición actual del usuario al servidor LBS. Para ello, los autores del trabajo asumen que cada usuario comparte un secreto con cada uno de sus amigos a través de una técnica de encriptación simétrica. Otro enfoque criptográfico propuesto por (Ghinita, Kalnis, Khoshgozaran, Shahabi, & Tan, 2008) utiliza una técnica de Recuperación de Información Privada (PIR) para proporcionar privacidad de ubicación. A través de PIR, un servidor LBS puede responder a las consultas sin aprender o revelar ninguna información sobre la consulta emitida por el usuario. PIR se basa en un supuesto de residuos cuadráticos, establece que es difícil encontrar residuos cuadráticos en la aritmética de módulos de un gran número compuesto para el producto de dos grandes primos. Los mecanismos con enfoque de anonimato consisten en generar ambigüedad en la privacidad de ubicación. El mecanismo más utilizado con este enfoque es k-anonimato (Mellon, 2002), se encarga de escoger k-1 ubicaciones distintas a la ubicación actual del usuario. Normalmente, el k-anonimato es usado junto a otros mecanismos de protección, por ejemplo, con Cloak Region (Niu, Zhu, et al., 2015) o Caching (Zhu et al., 2013). Los mecanismos con enfoque colaborativo consisten en mitigar el uso del servidor LBS, compartiendo información entre los usuarios, logrando preservar la privacidad de ubicación. El Caching o caché (Amini, Hong, Lin, Toch, & Sadeh, n.d.) es un mecanismo que usa este enfoque, ya que cuando un usuario envia una LBQ solicitando un POI al Caching, este solo comparte el POI solicitado cuando es encontrado, sino el usuario debe enviar su LBQ al servidor LBS. Así, generalmente funciona el Caching.

3.2 Revisión sistemática de la literatura

3.2.1 Planificación de revisión sistemática de la literatura

La revisión sistemática de la literatura se realizó siguiendo el protocolo propuesto por (Kitchenham, 2004) . El cual se organiza en 3 etapas: planificación, realización y por último un informe de presentación de los resultados.

El objetivo es conocer el estado del arte sobre el Caching en los servicios basados de la ubicación. A continuación, se definen las preguntas de investigación que se buscan responder al realizar la revisión sistemática de la literatura:

P1: ¿Como los trabajos aplican el Caching en la privacidad de ubicación en LBSs?

Esta pregunta pretende conocer el mecanismo de caché utilizado por los trabajos que protegen la privacidad de ubicación del usuario cuando usa los LBSs.

P2: ¿Cuáles son las arquitecturas de LBS donde los trabajos de Caching funcionan?

Esta pregunta pretende conocer las arquitecturas donde se ejecutan los trabajos de Caching que protegen la privacidad de ubicación del usuario.

P3: ¿Cuáles son las métricas de evaluación que usan los trabajos en los algoritmos de Caching?

Esta pregunta busca conocer las métricas utilizadas por los trabajos de Caching para evaluar su rendimiento.

Los strings de búsqueda utilizados en la revisión para extraer los distintos artículos desde las librerías digitales, fueron los siguientes:

- **S1:** "(Caching LBS OR LBS Caching)"
- **S2:** "(location privacy LBS OR LBS location privacy) AND Caching"

Las librerías digitales seleccionadas para la extracción de trabajos son las siguientes:

- ScienceDirect
- IEEE
- ACM
- Hindawi

3.2.2 Resultados de revisión sistemática de la literatura

En la tabla 1, se muestra los artículos seleccionados de la revisión sistemática de la literatura, después de aplicar criterios de exclusión e inclusión.

Tabla de Trabajos Seleccionados		
		Cantidad de Trabajos Seleccionados
Fuente	IEEE	7
	ScienceDirect	1
	ACM	1
	Hindawi	1
Año	2013	1
	2014	2
	2015	2
	2016	1
	2017	0
	2018	2
	2019	2
N°Trabajo	**Título del Trabajo**	
1	Collaborative Caching Techniques for Privacy- Preserving Location-based Services in Peer-to-Peer Environment (Jung & Park, 2018).	
2	A game theoretic approach for collaborative caching techniques in privacy preserving location-based services (Л.О., 2015).	

3	A Caching and Spatial K-anonymity Driven Privacy Enhancement Scheme in Continuous Location-Based Services(Zhang, Li, Tan, Peng, & Wang, 2019).
4	Hiding in the Mobile Crowd: Location Privacy through Collaboration(Systems, 2011).
5	MobiCache: When k-anonymity meets cache(Zhu et al., 2013).
6	Enhancing privacy through caching in location-based services(Niu, Li, Zhu, Cao, & Li, 2015).
7	Ensuring Privacy Protection in Location-based Services through Integration of Cache and Dummies(Alaradi & Innab, 2019).
8	DeCache: A decentralized two-level cache for mobile location privacy protection(Xiao, Chen, Wang, Zhao, & Chen, 2014).
9	RuleCache: A Mobility Pattern Based Multi-Level Cache Approach for Location Privacy Protection(Yang, 2016).
10	Double Cache Approach with Wireless Technology for Preserving User Privacy (Sen, 2018).

Tabla 1. Trabajos Seleccionados de la revisión sistemática

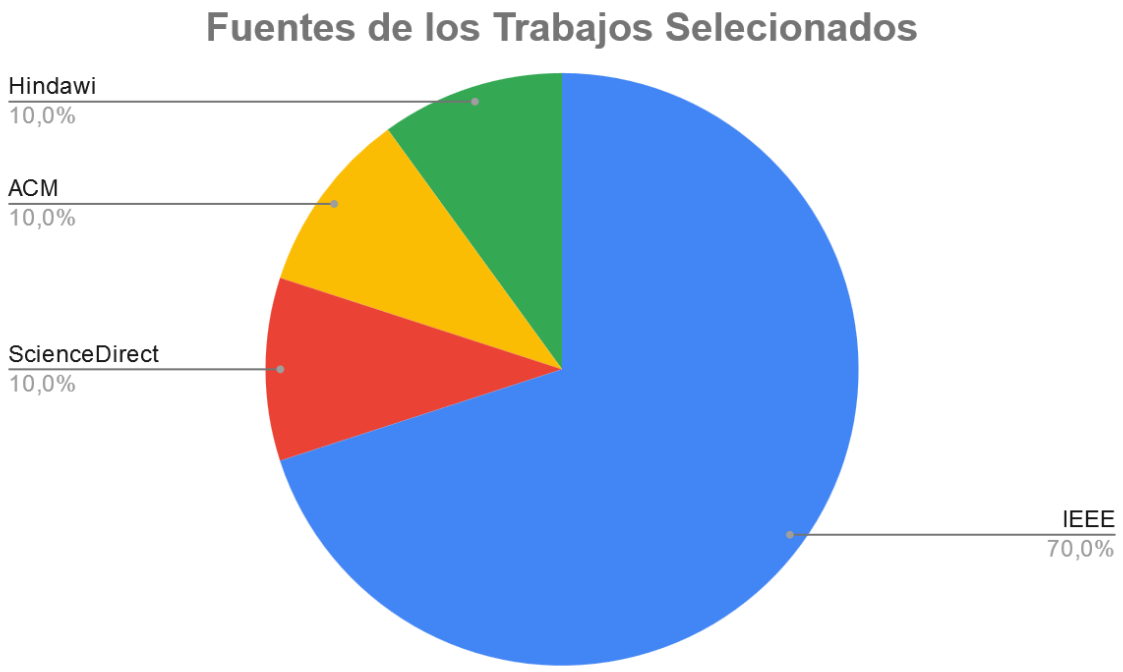


Ilustración 2. Fuentes de los trabajos seleccionados

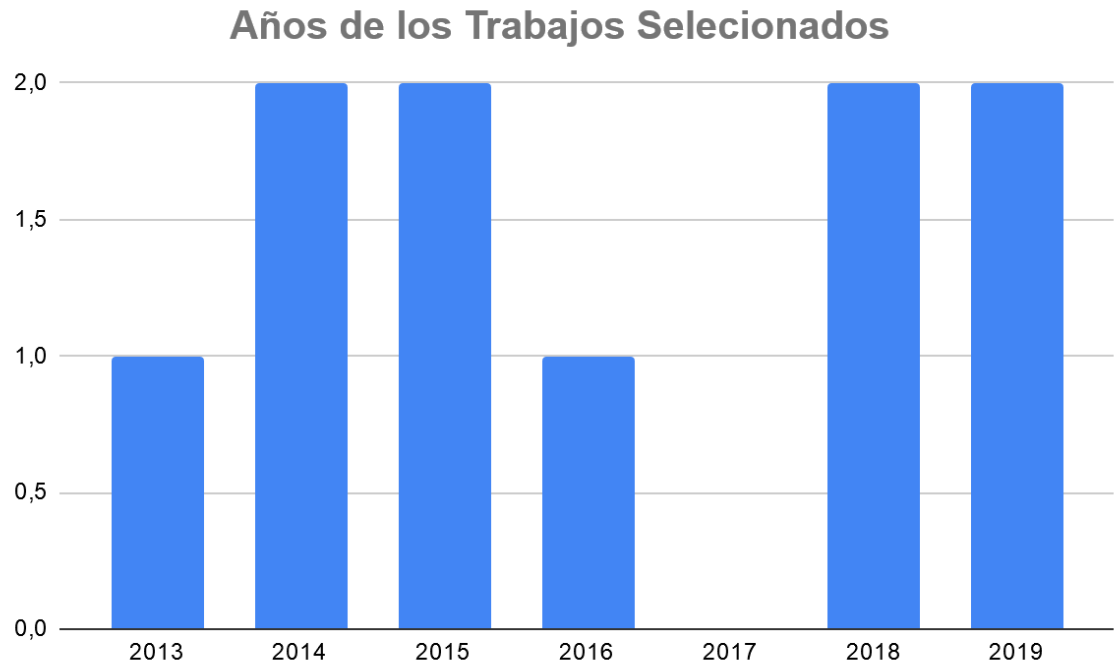


Ilustración 3. Años de los trabajos seleccionados

P1: ¿Como funciona el Caching en la privacidad de ubicación en LBSs?

La mayoría de los trabajos usan el Caching para guardar respuestas de LBQs, es decir, POIs. Pero, algunos de estos trabajos juntan el Caching con otro mecanismo de protección para mejorar la privacidad de ubicación. En la tabla 2 se muestran los trabajos que juntan el Caching con otros mecanismos.

N° de articulo	Mecanismo
5,6 y 7	k-anonimato
3 y 9	Cloak Region
1,2,4, 8 y 10	Ninguno

Tabla 2. Cantidad de Trabajos de Caching junto a otros mecanismos de protección.

P2: ¿Cuáles son las arquitecturas de LBS donde los trabajos de Caching funcionan?

Los trabajos que protegen la privacidad del usuario en los LBSs, funcionan en las siguientes arquitecturas:

Punto a Punto (P2P): Esta arquitectura consiste en resolver las LBQs mediante la colaboración de los usuarios que usan los LBSs, es decir, el usuario antes de enviar la LBQ al servidor LBS, envia su LBQ a sus vecinos para que ellos puedan resolverla. Por ejemplo, el usuario A1 envía su LBQ (LBQ A1) a sus vecinos. Asi que, el usuario A6 responde la LBQ de A1 (R. LBQ A1) como se muestra en la ilustración 4.

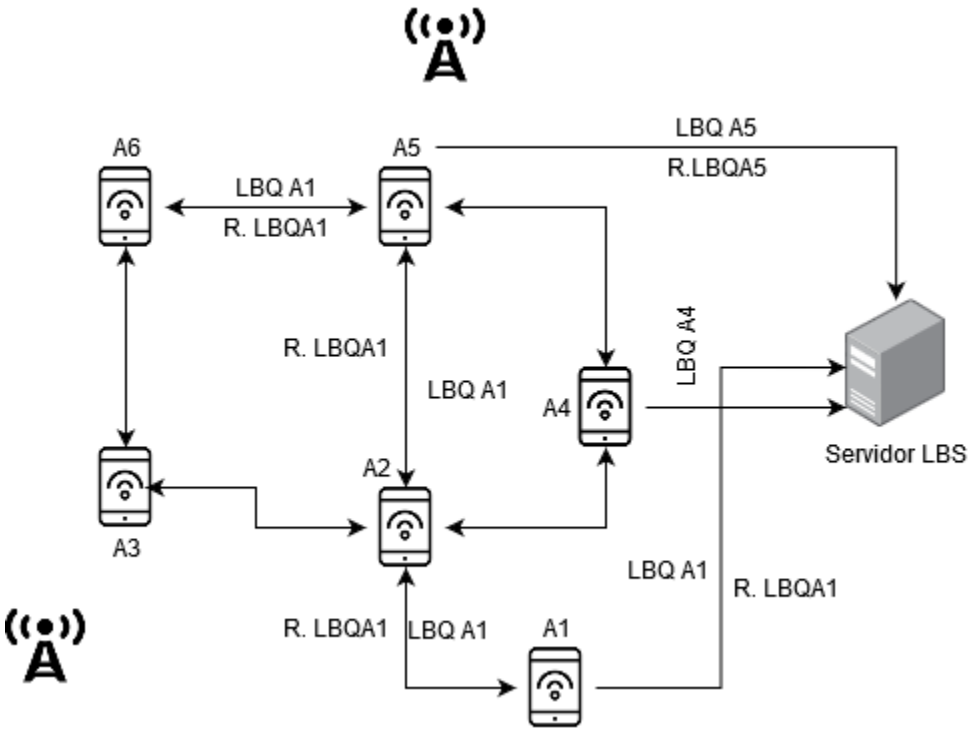


Ilustración 4. Arquitectura P2P

Tercero de Confianza (TTP): Esta arquitectura consiste en añadir un servidor de anonimato como intermediario entre el usuario y el servidor LBS. Todas las LBQs pasan por el servidor de anonimato, ya que este se encarga de aplicar el método de protección, como se muestra en la ilustración 5.

Ilustración 5. Arquitectura TTP

Todos de los trabajos de Caching utilizan una arquitectura P2P para procesar las LBQs enviadas por los usuarios. En la tabla se muestra los trabajos de Caching que usan arquitectura P2P y TTP.

N° de articulo	Arquitectura
todos	P2P
Ninguno	TTP

Tabla 3. Cantidad de trabajos de Caching con arquitectura P2P Y TTP

P3: ¿Cuáles son las métricas de evaluación que usan los trabajos en los algoritmos de Caching?

Las métricas más comunes usadas en los trabajos para evaluar el Caching son las siguientes:

- Tasa de aciertos en caché: Esta métrica trata mide la cantidad de veces que el caché responde a una LBQ.
- Costo de Almacenamiento: Esta métrica mide la cantidad de respuestas guardadas en caché. La unidad de medida de esta métrica puede ser en Kilobytes o Bytes.
- Costo de Comunicación: Esta métrica mide la cantidad de LBQs que fueron enviadas al servidor LBS.

4. Desarrollo de Algoritmos de Caching

En este proyecto de título, se escogieron dos algoritmos de Caching para implementar y comparar, que fueron los siguientes: *Double Cache Approach* (DCA) (Sen, 2018) y *Caching-aware Dummy Selection Algorithm* (CaDSA) (Niu, Li, et al., 2015). Los aspectos que se consideraron para desarrollar estos dos algoritmos de Caching fueron los siguientes:

- El funcionamiento del Caching de ambos algoritmos es distinto. Así que, se quiere visualizar el efecto de cada algoritmo en como aporta a la privacidad de ubicación del usuario.
- Los trabajos que presentan los algoritmos de Caching escogidos, son más actuales que el resto y son autores diferentes, pues se quiere evaluar el enfoque de cada autor en como observan el Caching en la privacidad de ubicación.

4.1 Caching-aware Dummy Selection Algorithm

Este trabajo busca que el Caching guarde POIs del usuario cuando envía LBQs junto a los dummies. Los dummies son ubicaciones distintas a la ubicación actual del usuario, que funcionan en base al concepto de k-anonimato, por ejemplo, si el usuario escoge un k-anonimato igual a 3, se escogen k-1 ubicaciones aparte de la ubicación actual del usuario como se muestra en la ilustración 6. El objetivo general del trabajo es preservar la privacidad de ubicación con estas dummies, es decir, enviar la LBQ del usuario desde la ubicación real y desde las dummies hacia al servidor LBS, y luego se recibe las respuestas de las LBQs enviadas para guardarlas en el Caching. Por ello, se propone Caching-aware Dummy Selection Algorithm (CaDSA).

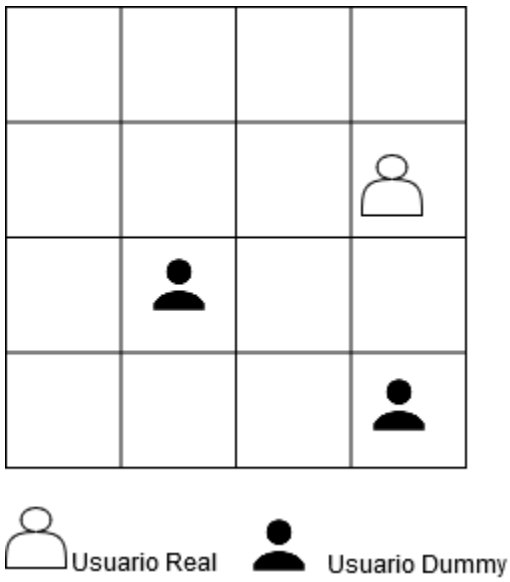


Ilustración 6. muestra un k-anonimato igual a 3 en un mapa de NxN

Este algoritmo propone escoger distintas ubicaciones (dummies) a la ubicación actual del usuario. Para escoger a estas, se debe cumplir dos condiciones: tener una probabilidad de consulta similar que la ubicación actual del usuario y deben contribuir al Caching.

La primera condición trata de la probabilidad de consulta, esta es la probabilidad de que un usuario realice una LBQ desde una área o punto en específico dentro de un mapa NxN, la idea es que los dummies se ubiquen en áreas que tengan una probabilidad de consulta similar (✓) al área que se encuentra el usuario como se muestra en la ilustración 7. Con esto, se consigue generar una ambigüedad en donde el usuario envía su LBQ solicitando un POI al servidor LBS. En cambio, la segunda condición trata de la contribución al Caching, básicamente no es guardar

un dummie repetido en el Caching para no generar un aumento de almacenamiento innecesario. Por lo tanto, CaDSA escoge varios conjuntos de dummies de tamaño k-1 (depende k-anonimato ingresado), y el mejor conjunto que cumpla las dos condiciones será escogido para enviar la LBQ junto con el usuario.

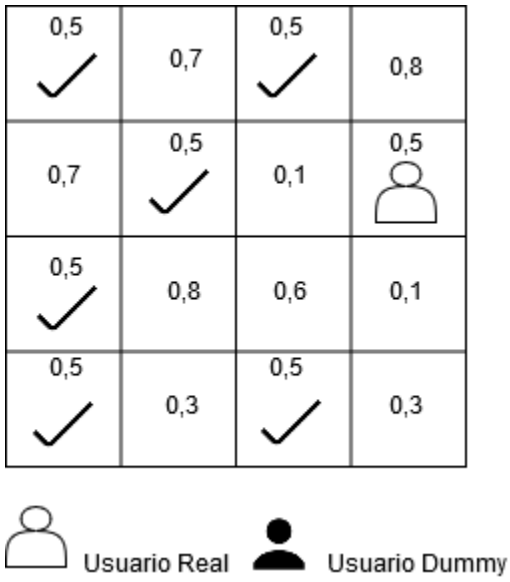


Ilustración 7. Probabilidades de consulta similares a la posición real del usuario.

4.1.1 Pseucodigo de CaDSA

Como se explicó anteriormente, CaDSA escoge un conjunto de dummies que cumpla las siguientes dos condiciones: tener una probabilidad de consulta similar al área donde se ubica el usuario y que tenga mejor contribución al Caching, respectivamente. Previamente a la explicación del algoritmo, las dos condiciones principales, se basan en la ecuación 1 denominada Multi-Objective Optimization Problem (MOP) por los autores del trabajo:

$$C_{dummy} = \arg\max \left\{ -\sum_{i=1}^{k} p_i * \log_2 p_i \, , \sum_{i=1}^{k} \delta_i \right\} (1)$$

Ecuación 1. Multi-Objective Optimization Problem

La ecuación 1 realiza las dos condiciones que se necesita para escoger al conjunto de dummies final, pero según los autores del trabajo es difícil realizar la ejecución de la ecuación 1 en un solo paso, por eso se realiza la ejecución en dos pasos. El primer paso se encarga de ejecutar la primera condición, que es escoger los dummies ubicados en un área del mapa que tenga una probabilidad de consulta similar al área que está ubicado el usuario y agruparlos en conjuntos de tamaño k-1, esta condición se representa en la ecuación 2. Luego, se ejecuta la segunda

condición, que es escoger al mejor conjunto que contribuya al Caching de los conjuntos escogidos en el primer paso, esta condición se representa en la ecuación 3. Por lo tanto, la ecuación 1 se realizó por parte, en las ecuaciones 2 y 3.

$$C_c = \arg\max\left(-\sum_{i=1}^{k} p_i * \log_2 p_i\right) (2)$$

Ecuación 2. Probabilidad de consulta similar

$$C_{dummy} = \arg\max \sum_{i=1}^{k} \delta_i \ (3)$$

Ecuación 3. Contribución al Caching

En la tabla 4, se encuentra el pseudocodigo de CaDSA y se explica de la siguiente manera:

- Desde la línea 1 hasta 3 se trabaja con las probabilidades de consulta de cada celda del mapa NxN, cada celda representa un área específica. Primero, se ordena las probabilidades de cada celda de forma creciente en una lista, luego acorde a la celda que se encuentre el usuario (C_r) se eligen 4k celdas desde la lista ordenada, específicamente, 2k celdas del lado derecho y 2 celdas del lado izquierdo desde celda que se encuentra el usuario en lista. Desde esas 4k celdas elegidas, se escogen 2k celdas de forma aleatoria. Esas celdas escogidas aleatoriamente se denominan celdas candidatas (C_c). En el trabajo los autores sacaron las probabilidades de consulta a través de una API de Google, pero en la implementación de este algoritmo no se puedo conseguir esas probabilidades, asi que se usaron distribuciones de probabilidades, en el capítulo 5 hay más detalle sobre esto.
- La línea 5 hay una combinatoria, que se resuelve con el k-anonimato ingresado por el usuario, debe ser mayor a 1. Si el resultado de la combinatoria es menor o igual a s (parámetro de sistema), pasa a línea 6. En esta línea se crea un conjunto de probabilidad de consulta ($\hat{C}_c$) que tiene subconjuntos. Cada subconjunto que compone a $\hat{C}_c$ tiene C_c, y son de tamaño k-1. En cambio, si la combinatoria es mayor a s, se debe crear s subconjuntos de C_c en $\hat{C}_c$ y cada subconjunto es de tamaño k-1 escogidos de manera aleatoria desde C_c.
  Al terminar de añadir los subconjuntos a $\hat{C}_c$, a cada subconjunto se le denomina dummies.

- Desde la línea 12 hasta 14, se realiza un ciclo for para recorrer $\hat{C}_c$. Cabe destacar que $\hat{C}_c$ en cada posición tiene un subconjunto y cada elemento del subconjunto es una probabilidad de consulta (p) llamado dummie, se multiplica por g ($contribucion\ del\ dummie$) para calcular la contribución al Caching (ecuación 3). El valor de g puede ser 0 o 1, cuando es 0 quiere decir que el dummie ya está en el Caching y cuando es 1 quiere decir que el dummie no esta en el Caching, esto aporta que no se ocupe almacenamiento extra en el Caching.
- En la línea 15 se retorna el subconjunto máximo de $\hat{C}_c$, que cumpla con las dos condiciones principales, o sea, C_{dummy}.

Caching-aware Dummy Selection Algorithm
Input: q (query probability of each of cell), c_r (real location), s (a system parameter)
1. sort cells based on their query probability q 2. choose 4k cells (2k cells are right before c_r and 2k cells are right after c_r in the sorted list) 3. randomly select 2k cells out of them as the candidate set C_c 4. $\hat{C}_c = \emptyset$ 5. **If** $\binom{2k}{k-1} \leq$ s **then** 6. $\hat{C}_c = \{C' \mid C' \subset C_c \ \& \ (\lvert C' \rvert = k-1)\}$ 7. **End** 8. **Else** 9. Generate s subsets of C_c with k – 1 random dummies in each subset; 10. Add these subsets to $\hat{C}_c$ 11. **End** 12. **For** each C' in $\hat{C}_c$ **do** 13. Compute $\sum_{i=1}^{k} \delta_i$ for the k-1 dummies in C' 14. **End** 15. $C_{dummy} = \arg max_{C' \subset \hat{C}_c} \sum_{i=1}^{k} \delta_i$
Output: C_{dummy}

Tabla 4. Pseudocodigo de CaDSA

4.1.2 Implementación de CaDSA

Básicamente, el funcionamiento de este algoritmo es primero preguntar al Caching por el POI que está en la LBQ, si encuentra la respuesta se envía al usuario. Sino se envía la LBQ junto al conjunto de dummies.

La implementación se realizó en un simulador creado por el autor de este proyecto de título. Pero, durante su implementación en el simulador se consideraron los siguientes aspectos:

- El Caching se encuentra en un punto de acceso, al igual que el trabajo que propone CaDSA.
- El Caching guardar los POIs de LBQs y los dummies que fueron usados para enviar LBQs al servidor LBS.
- Para la simulación de este algoritmo no se realizó en un mapa de NxN, solo realizo en un plano cartesiano que representa una celda de un mapa de NxN.
- Para obtener la probabilidad de consulta que se necesita para escoger a las dummies se usaron distribuciones de probabilidad, como la distribución normal y distribución Cauchy.
- Los usuarios y POIs son escogidos aleatoriamente para cada consulta procesada.
- El Caching tiene un tamaño fijo.

4.2 Double Cache Aproach

Este trabajo busca que el Caching no solamente guarde POIs de LBQs pasadas, sino que participe en la gestión de la LBQ con los usuarios, además se quiere conseguir cooperación entre los usuarios para proteger su privacidad del servidor LBS. Por eso, se propone el algoritmo Double Cache Approach (DCA).

Este algoritmo propone dos caches para procesar las LBQs enviadas por los usuarios, estas se definen como: caché de respuesta y caché de consulta.

- Caché de respuesta: Esta caché tiene POIs guardadas de LBQs pasadas. Si un usuario envía su LBQ a esta, puede conseguir donde se encuentra el POI solicitado en la LBQ. En el caso de no encontrar POI en la caché, la LBQ se dirige a caché de consulta.
- Caché de consulta: Esta caché funciona cuando la caché de respuesta no puede responder a la LBQ del usuario. Su función es gestionar el intercambio de la LBQ con otro usuario, y las respuestas de sus LBQs se guardan en la caché de respuesta, para que los usuarios accedan nuevamente a caché de respuesta y consigan el POI.

Como se muestra en la ilustración 8, el usuario A1 si no encuentra POI en la caché de respuesta, la LBQ (LBQ A1) se dirige a caché de consulta para que se gestione el intercambio de LBQ con otro usuario, en este caso será con A4. La caché de consulta realiza el intercambio con el usuario A4 (LBQ A4), luego ambos envían la LBQ del otro hacia el servidor LBS. Finalmente, el servidor LBS guarda el POI solicitado en la LBQ en caché de respuesta, para que el A1 y A4 accedan nuevamente a conseguir POI correspondiente, es decir, el POI_{A1} y POI_{A4} las desde un comienzo.

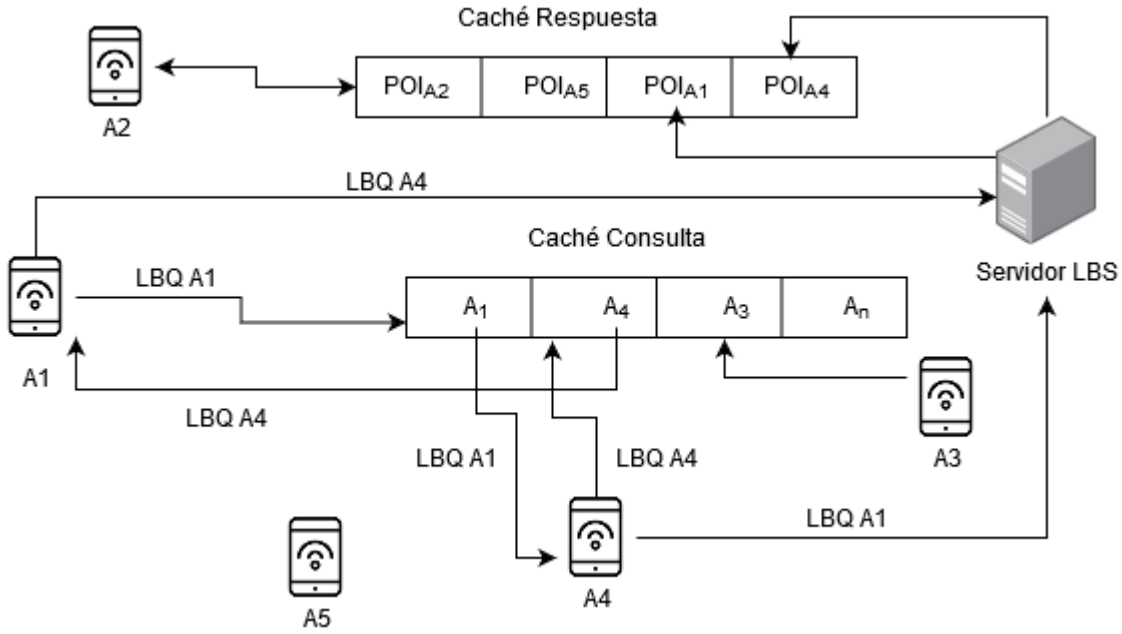


Ilustración 8. Funcionamiento de DCA

4.2.1 Pseucodigo de DCA

En la Tabla 5, se encuentra el pseudocodigo de CaDSA y se explica de la siguiente manera:

- En la línea 1 pregunta si el POI de la LBQ del usuario existe en Cache1 (caché de respuesta), si existe se retorna el POI como indica la línea 3. Sino pasa a la línea 5.
- Desde la línea 5 hasta la línea 10 se ejecuta la gestión de Cache2 (caché de consulta). Primero la LBQ es ingresada a Cache2 para gestionar el intercambio con otro usuario (usuario desconocido), luego en la línea 6 el usuario original recibe la LBQ del usuario desconocido para que el envié la LBQ al servidor LBS, y el usuario desconocido haga lo mismo con la LBQ del usuario original. Cuando ambos reciben respuestas de la LBQs por parte del servidor, serán guardadas en Cache1.

- Al estar guardadas las respuestas de las LBQs del usuario original y desconocido en Cache1, en la línea 11 se llama nuevamente a la función del algoritmo DCA para que LBQ del usuario original pregunte de nuevo a Cache1 y reciba el POI correcto.

Double Cache Approach
Input: Query (q) of user A
Function Search 1. **If** ($q \in$ Query in Cache1) **then** 2. $q_{ID} =$ (MAX_{ID} in Cache1) 3. Return q_{ANS} 4. **End** 5. **Else** 6. Insert (q in Cache2) 7. Get first $q\sim$ of Cache2 // Unknow User 8. Insert $q\sim_{ANS}$ to Cache1 with MAX_{ID} 9. Query with MIN_{ID} in the Cache1 will be deleted 10. **End** 11. Recall Search Function 12. End Function
Output: Answers of q

Tabla 5. Pseudocodigo de DCA

4.2.2 Implementación de DCA

Básicamente, el funcionamiento de este algoritmo es primero preguntar al caché de respuesta por POI que envían por la LBQ, si encuentran la respuesta se envía al usuario. Sino se envía la LBQ a la caché de consulta para que gestione el intercambio con otro usuario, y envíen las LBQs al servidor LBS.

La implementación se realiza en un simulador creado por el autor de este proyecto de título. Pero durante su implementación en el simulador se consideraron los siguientes aspectos:

- La caché de respuesta y caché de consulta se encuentra en un punto de acceso, al igual que el trabajo que propone DCA.
- Caché de respuesta guardar los POIs de las LBQs pasadas.

- Para la simulación de este algoritmo no se realizó en un mapa de NxN, solo realizo en un plano cartesiano que representa una celda de un mapa de NxN.
- Para cada consulta que se procesa en el simulador pueden ocurrir dos cosas:
  - Si la LBQ del primer usuario es respondida por caché de respuesta, termina el procesamiento de esta.
  - Si la LBQ del primer usuario no es respondida por caché de respuesta, se realiza el intercambio de LBQ con un segundo usuario en caché de consulta. Para realizar el intercambio se busca un segundo usuario de forma aleatoria que quiera enviar una LBQ. Primero, la LBQ de este segundo usuario pregunta en caché de respuesta, si es respondida se busca a otro segundo usuario hasta que la LBQ de este no sea respondida y pase a caché consulta para que realice el intercambio con el primer usuario, así termine el procesamiento de la consulta.
- Los usuarios y POIs son escogidos aleatoriamente para cada consulta procesada.
- El caché de respuesta es de tamaño fijo.

5.Experimentación

5.1 Métricas

Para la evaluación de los algoritmos de Caching se utilizaron las métricas encontradas en la revisión sistemática de la literatura, las cuales son:

- Tasa de aciertos en caché: Esta métrica indica cuantos LBQs se respondieron en el Caching, entre mayor sea el resultado es mejor, porque se preserva la privacidad de ubicación del usuario.
- Costo de Almacenamiento: Esta métrica indica la cantidad de memoria utilizada en el Caching, debido a que almacena los POIs con sus posiciones para responder a las LBQs. En este proyecto de título se mide bytes.
- Costo de Comunicación: Esta métrica indica la cantidad de LBQs enviadas al servidor LBS o conexiones solicitadas a este.

5.2 Metodología de experimentación

- Se desarrollo un simulador Python 3.7.3 para evaluar los algoritmos de Caching escogidos.
- Se ingresan los datos de entrada como cantidad de LBQs que se enviaran, la cantidad de usuarios que componen la red y la cantidad de pasos que cada usuario dará en el plano del simulador para visualizar el efecto de cada algoritmo y compararlos.
- Se muestra los resultados que entrega cada métrica al finalizar la ejecución de cada algoritmo, para realizar la comparación y sacar conclusiones de los experimentos.

5.3 Simulador

5.3.1 Componentes

Se realizo un simulador en Python 3.7.3 para llevar a cabo la experimentación. Tiene los siguientes componentes:

- Un plano cartesiano que representa una celda de un mapa NxN. Cada lado de la celda tiene 300 metros. Dentro del plano cartesiano se muestran los puntos de interés y un punto de acceso para conectarse y enviar las consultas. En este punto de acceso se ejecuta el algoritmo de Caching que se desea, ya que en los trabajos que proponen los algoritmos de Caching funcionan en este, además se ubica en el centro de la celda (0,0).
- Los datos de entradas son el número de usuarios que participaran en la simulación, el número de pasos que caminaran los usuarios después de cada consulta, el número de consultas que se procesaran y el algoritmo de Caching. Para los pasos de cada usuario se ocupo el modelo de movimiento Random Walk (Nosofsky & Palmeri, 1997).
- Hay otros datos de entradas solo para el CaDSA. Uno es el k-anonimato y otro la distribución de probabilidad para dar valores a las probabilidades de consulta de cada punto del plano, ya que puntos son posibles ubicaciones de los usuarios.
- Los botones necesarios del simulador son: “Simular” para la ejecución de la simulación del algoritmo y “Limpiar” borra los resultados de la simulación, en el caso que se quiera hacer otra.

En la ilustración 9, se muestra el simulador cuando inicia su ejecución donde hay ciertas figuras como:

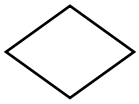
Figura	Descripción
	Esta figura representa un punto de interés. Como se muestra en la ilustración 9, donde hay 24 puntos de interés en el plano.
	Esta figura representa el punto de acceso. Como se muestra en la ilustración 9, donde se ubica en el centro del plano.
	Esta figura representa un usuario. Como se muestra en la ilustración 10, la ejecución de un algoritmo con los usuarios en el plano.

Tabla 6. Descripción de las figuras del simulador

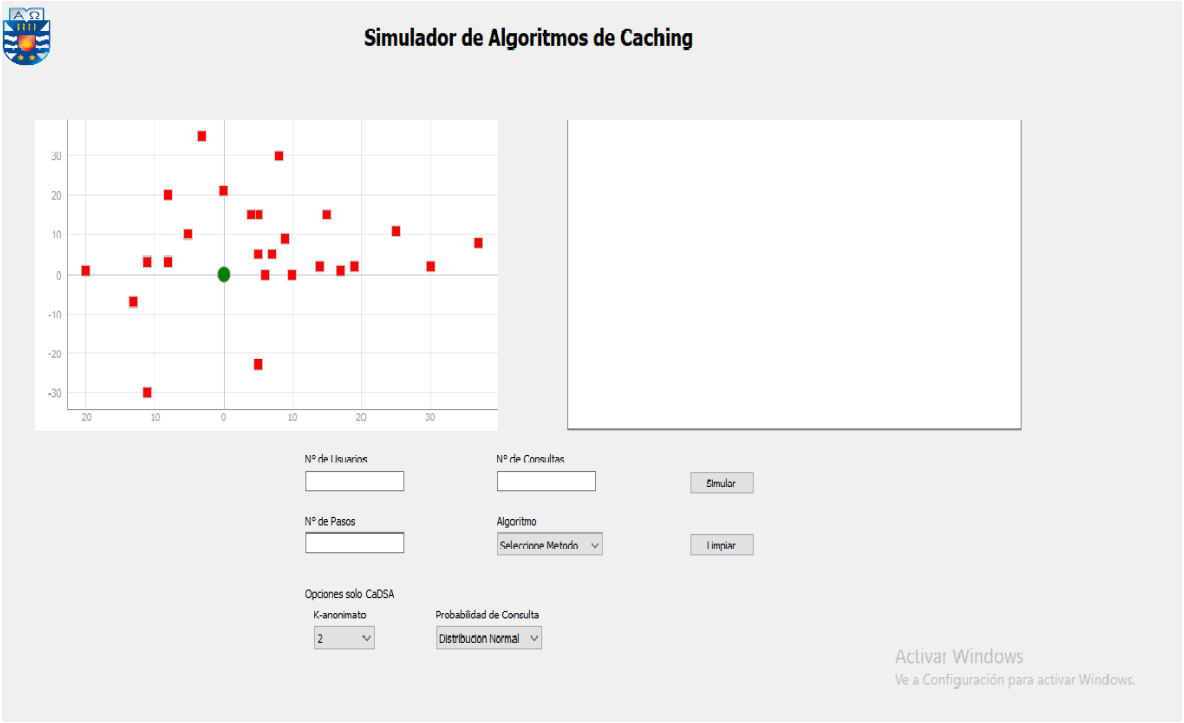


Ilustración 9. Simulador

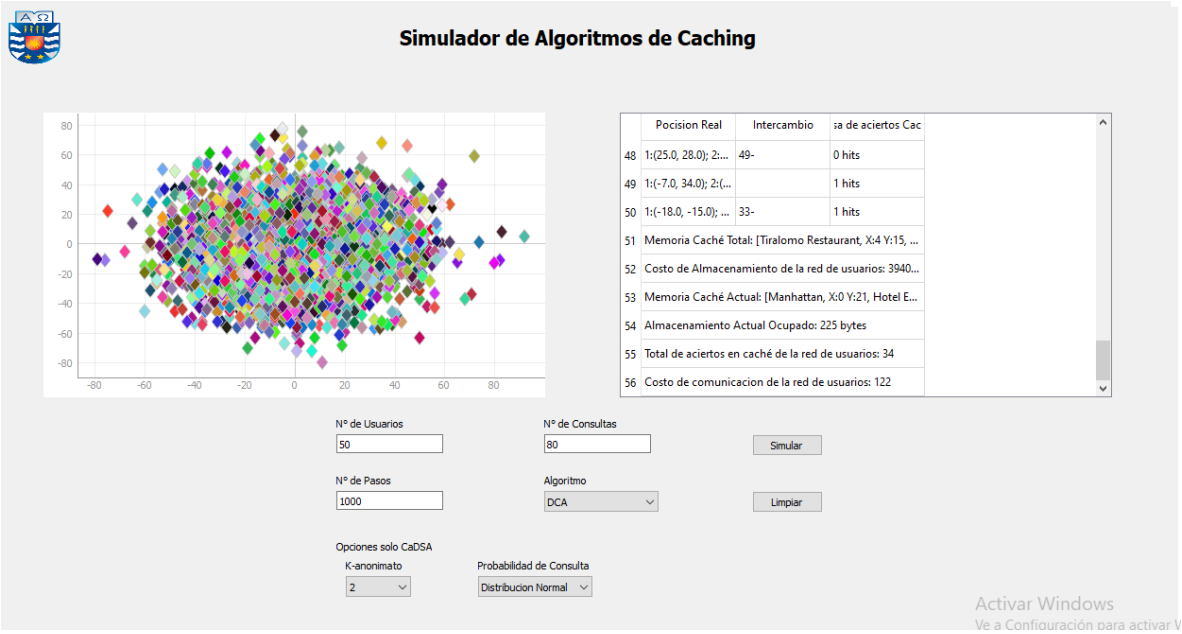


Ilustración 10. Simulador procesando LBQs

5.3.2 Librerías utilizadas

Como se mencionó el simulador fue desarrollado Python 3.7.3, y el código fuente del simulador fue escrito en el editor de texto Visual Studio Code 1.41.1, para el desarrollo del código fuente se usaron las siguientes librerías de Python:

- Numpy 1.16.1: Esta librería le agrega mayor soporte para vectores y matrices, constituyendo una biblioteca de funciones matemáticas de alto nivel para operar con esos vectores o matrices. Se adjunto enlace de instalación: https://numpy.org/install/
- Scipy.stats 1.5.2: Esta librería contiene un gran número de distribuciones de probabilidad, así como una creciente biblioteca de funciones estadísticas. Esta ayudo para obtener la probabiliades de consulta en CaDSA. Se adjunta la documentación: https://docs.scipy.org/doc/scipy/reference/stats.html
- PyQt5: Esta librería permite crear la interfaz gráfica del simulador de pruebas. Se adjunta la documentación: https://doc.qt.io/qtforpython/
- PyqtGraph 1.0: Esta librería permite grafica cada objeto en el plano del simulador. Se adjunta documentación: http://www.pyqtgraph.org/
- Sys, os y subprocess: Esos son módulos estándares de Python que sirven para la manipulación de archivos y memoria. Solo se utilizó el módulo sys, ya que este ayudo a obtener al cantidad de memoria en Bytes del Caching con su función *getsizeof()*.

6 Resultados y Análisis de las Experimentaciones

6.1 Resultados CaDSA

Se realizo la experimentación a CaDSA en el simulador con 50 usuarios en el plano, 1000 pasos por cada usuario en cada consulta. Los datos de entrada que cambian son la cantidad de consulta y el k-anonimato, también se consideró el tiempo de ejecución de cada prueba. La experimentación dio los siguientes resultados:

Usuarios	**Pasos**		**K-anonimato**	
50	**1000**		**2**	
	Consultas			
	20	40	60	80
Tasa de aciertos	2	8	21	22
Costo de Comunicación	36	64	78	116
Costo de Almacenamiento (bytes)	1446	2541	3098	4464
Tiempo (segundos)	273,3926	1027,5539	2102,4789	4454,5428

Tabla 7. Resultado CaDSA con k-anonimato igual a 2

Usuarios	**Pasos**		**K-anonimato**	
50	**1000**		**3**	
	Consultas			
	20	40	60	80
Tasa de aciertos	4	15	16	17
Costo de Comunicación	48	75	132	189

Costo de Almacenamiento (bytes)	1606	2537	4336	6342
Tiempo (segundos)	289,4945	1300,7626	2192,9506	4510,8269

Tabla 8. Resultado de CaDSA con k-anonimato igual a 3

Usuarios	**Pasos**		**K-anonimato**	
50	**1000**		**4**	
	Consultas			
	20	40	60	80
Tasa de aciertos	3	9	16	17
Costo de Comunicación	68	124	176	252
Costo de Almacenamiento (bytes)	2087	3856	5234	7423
Tiempo (segundos)	310,8791	1023,8827	2209,3681	4881,2902

Tabla 9. Resultado de CaDSA con un k-anonimato igual a 4

Usuarios	**Pasos**		**K-anonimato**	
50	**1000**		**5**	
	Consultas			
	20	40	60	80
Tasa de aciertos	5	6	16	18
Costo de Comunicación	75	170	220	310

Costo de Almacenamiento (bytes)	2440	4849	6212	8777
Tiempo (segundos)	294,4161	1062,3420	2221,5224	5881,6372

Tabla 10. Resultado de CaDSA con un k-anonimato igual a 5

Usuarios	**Pasos**		**K-anonimato**	
50	**1000**		**6**	
	Consultas			
	20	40	60	80
Tasa de aciertos	4	8	12	20
Costo de Comunicación	96	192	288	360
Costo de Almacenamiento (bytes)	2701	4811	7763	10571
Tiempo (segundos)	380,8700	1062,3420	2553,2305	6711,8231

Tabla 11. Resultado de CaDSA con un k-anonimato igual a 6.

6.2 Resultados DCA

Se realizo la experimentación a CaDSA en el simulador con 50 usuarios en el plano, 1000 pasos por cada usuario en cada consulta. Los datos de entrada que cambian son la cantidad de consulta, también se consideró el tiempo de ejecución de cada prueba. La experimentación dio los siguientes resultados:

Usuarios	**Pasos**						
50	**1000**						
	Consultas						
	20	30	40	50	60	70	80
Tasa de aciertos	13	8	10	24	32	33	34
Costo de Comunicación	36	58	62	82	86	108	122
Costo de Almacenamiento (bytes)	1208	1865	2029	2626	2782	3425	3940
Tiempo (segundos)	284,07	531,17	960,48	1428,57	2087,36	2915,35	3591,17

Tabla 12. Resultado general de DCA

6.3 Análisis de los Resultados

Matriz Comparativa	
CaDSA	
Ventajas	**Desventajas**
CaDSA ofrece un grado de privacidad de ubicación muy alto, debido a que los dummies generan una ambigüedad en donde se envía la LBQ realmente.	Según los resultados de la experimentación, CaDSA al ofrecer un k-anonimato provoca que el costo de comunicación sea alto, ya que cada dummie envía la misma LBQ que el usuario real, y el servidor LBS debe responder a esa misma LBQ al usuario real y a cada dummie. Según los resultados de la experimentación, CaDSA al ofrecer un k-anonimato provoca que el costo de almacenamiento sea alto, ya que cada dummie ocupado es guardado en el

	caché que esta en el punto de acceso. Así que, entre más dummies se ocupan más almacenamiento se necesita. El efecto del k-anonimato aparte de generar un alto costo de comunicación y almacenamiento, también genera un tiempo de ejecución alto. Por ejemplo, en la ilustración 11 muestra el tiempo de ejecución de cada k-anonimato en 80 LBQs procesadas.
DCA	
Ventajas	**Desventajas**
DCA se destaca en que no usa el caché solo para guardar POIs como CaDSA, sino como parte del procesamiento de la LBQ mediante el caché de consulta. Según los resultados de la experimentación, DCA tiene un costo menor en comunicación que CaDSA, ya que solo envia dos consultas como máximo, debido al intercambio. Además, tiene un costo menor en almacenamiento, ya que solo almacena POIs en caché de respuesta.	DCA quizás no ofrezca un grado de privacidad de ubicación alto comparado a CaDSA, ya que la protección que ofrece es solo intercambiar consultas entre usuarios y enviarlas para que servidor LBS no se entere cual es la información verdadera de cada usuario.
Similitudes de CaDSA y DCA	
Según los resultados de la experimentación, en la tasa de aciertos del caché ambos algoritmos obtuvieron resultados parecidos, logrando responder LBQs y no exponer la posición de los usuarios al LBS en gran parte de las consultas procesadas.	

Tabla 13. Matriz Comparativa de los algoritmos de Caching

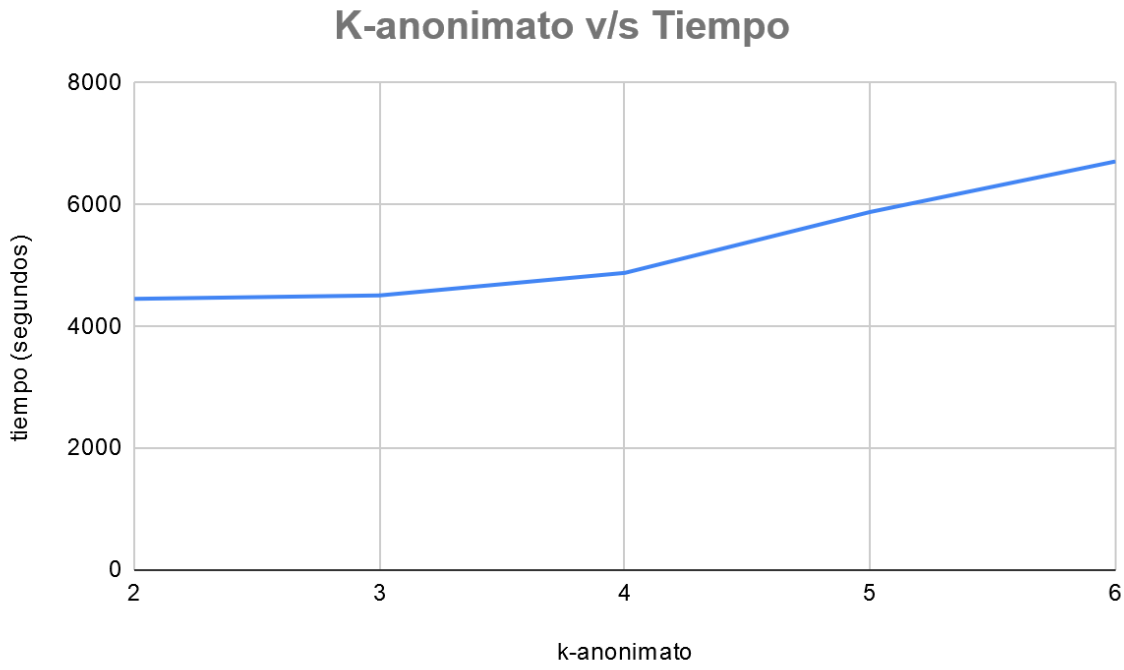


Ilustración 11. K-anonimato v/s Tiempo de ejecución

7 Conclusiones y Trabajos Futuros

7.1 Conclusiones

La privacidad de los datos se está considerando como algo importante en la vida diaria de los usuarios que usan la internet, ya que ellos exponen cualquier tipo de dato personal cuando usan algún servicio provee este, por ejemplo, los LBSs. Al usar LBSs, los usuarios deben exponer su ubicación para obtener información geográfica que necesitan, pero la ubicación es un dato privado y con la ayuda de los algoritmos de Caching pueden preservar su ubicación.

Dado los resultados de la experimentación de los algoritmos, se demuestra que pueden ser un aporte para mitigar el uso del servidor LBS y preserva la ubicación, ya que utilizando DCA y CaDSA en el mundo real del LBS, los usuarios confiaran más en estos servicios, debido a que la integridad y confidencialidad de los datos de los usuarios estará un poco más seguros, si estos algoritmos son aplicados.

7.2 Trabajos Futuros

El Caching todavía no ha sido explotado en su totalidad, y la comunidad de las ciencias de la computación puede crear posibles futuros trabajos con Caching que tenga una aplicación en el mundo real, por ejemplo:

- Anadir el uso del Caching como una extensión de apps móviles o webs basadas en la ubicación.
- Combinar mecanismos de protección existentes con Caching para diseñar nuevos algoritmos.
- Proponer métricas mas especificas hacia al Caching, ya que las existentes están hace mucho tiempo.

Referencias

Alrahhal, M. S., Khemakhem, M., & Jambi, K. (2017). A SURVEY ON PRIVACY OF LOCATION-BASED SERVICES: CLASSIFICATION, INFERENCE ATTACKS, AND CHALLENGES. *Journal of Theoretical & Applied Information Technology*, *95*(24)

Wang, S., Hu, Q., Sun, Y., & Huang, J. (2018). Privacy preservation in location-based services. *IEEE Communications Magazine*, *56*(3), 134-140.

Liu, B., Zhou, W., Zhu, T., Xiang, Y., & Wang, K. (2018). *Location Privacy in Mobile Applications* (pp. 1-101). Springer.

Blumberg, A. J., & Eckersley, P. (2009). On locational privacy, and how to avoid losing it forever. *Electronic frontier foundation*, *10*(11), 1-7.

Mascetti, S., Freni, D., Bettini, C., Wang, X. S., & Jajodia, S. (2011). Privacy in geo-social networks: proximity notification with untrusted service providers and curious buddies. *The VLDB journal*, *20*(4), 541-566.

Ghinita, G., Kalnis, P., Khoshgozaran, A., Shahabi, C., & Tan, K. L. (2008, June). Private queries in location based services: anonymizers are not necessary. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data* (pp. 121-132).

Sweeney, L. (2002). k-anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, *10*(05), 557-570.

Niu, B., Zhu, X., Li, W., Li, H., Wang, Y., & Lu, Z. (2015, February). A personalized two-tier cloaking scheme for privacy-aware location-based services. In *2015 International Conference on Computing*, *Networking and Communications (ICNC)* (pp. 94-98). IEEE.

Zhu, X., Chi, H., Niu, B., Zhang, W., Li, Z., & Li, H. (2013, December). Mobicache: When k-anonymity meets cache. In *2013 IEEE Global Communications Conference (GLOBECOM)* (pp. 820-825). IEEE.

Amini, S., Lindqvist, J., Hong, J., Lin, J., Toch, E., & Sadeh, N. (2011, June). Caché: caching location-enhanced content to improve user privacy. In *Proceedings of the 9th international conference on Mobile systems, applications, and services* (pp. 197-210)

Kitchenham, B. (2004). Procedures for performing systematic reviews. *Keele, UK, Keele University*, *33*(2004), 1-26.

Jung, K., & Park, S. (2017, December). Collaborative caching techniques for privacy-preserving location-based services in peer-to-peer environments. In *2017 IEEE International Conference on Big Data (Big Data)* (pp. 4497-4506). IEEE.

Jung, K., Jo, S., & Park, S. (2015, February). A game theoretic approach for collaborative caching techniques in privacy preserving location-based services. In *2015 International Conference on Big Data and Smart Computing (BIGCOMP)* (pp. 59-62). IEEE.

Zhang, S., Li, X., Tan, Z., Peng, T., & Wang, G. (2019). A caching and spatial K-anonymity driven privacy enhancement scheme in continuous location-based services. *Future Generation Computer Systems*, *94*, 40-50.

Alaradi, S., & Innab, N. Ensuring Privacy Protection in Location-Based Services Through Integration of Cache and Dummies.

Xiao, C., Chen, Z., Wang, X., Zhao, J., & Chen, C. (2014, July). DeCache: A decentralized two-level cache for mobile location privacy protection. In *2014 Sixth International Conference on Ubiquitous and Future Networks (ICUFN)* (pp. 81-86). IEEE.

Niu, B., Li, Q., Zhu, X., Cao, G., & Li, H. (2015, April). Enhancing privacy through caching in location-based services. In *2015 IEEE conference on computer communications (INFOCOM)* (pp. 1017-1025). IEEE.

Yang, Q., & Kong, P. (2016, December). RuleCache: A Mobility Pattern Based Multi-Level Cache Approach for Location Privacy Protection. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)* (pp. 448-455). IEEE.

Sen, A. A. A., Eassa, F. B., Yamin, M., & Jambi, K. (2018). Double Cache Approach with Wireless Technology for Preserving User Privacy. *Wireless Communications and Mobile Computing*, *2018*.

Nosofsky, R. M., & Palmeri, T. J. (1997). An exemplar-based random walk model of speeded classification. *Psychological review*, *104*(2), 266.

Anexos

Anexo 1: Carta Gantt acorde a los objetivos específicos.

Carta Gantt del Anteproyecto de titulo						
Obj.	Actividades	2019-2				
		Septiembre	Octubre	Noviembre	Diciembre	Enero
1	Búsqueda de los trabajos relacionados con Caching en los LBSs en distintas librerías digitales (IEE, ACM, Sciendirect, Hindawi).					
	Búsqueda de tesis sobre Caching en distintos repositorios digitales de universidades.					
2	Selección de los trabajos encontrados y analizar sus algoritmos.					
	Escoger los algoritmos a implementar, acorde ciertos criterios.					
	Planificar sobre el contenido del simulador donde se van realizar los algoritmos de Caching.					

Tabla 14. Carta Gantt del Anteproyecto de Titulo

Carta Gantt del Proyecto de titulo						
Obj.	Actividades	2020-1				
		Abril	Mayo	Junio	Julio	Agosto
3	Implementar los algoritmos en el simulador.					
	Realizar la interfaz gráfica del simulador.					
	Realizar la experimentación de los algoritmos de Caching en el simulador.					
	Desarrollar el informe del proyecto.					

Tabla 15. Carta Gantt del Proyecto de Titulo

Anexo 2: Puntos de interés usados en el simulador

Puntos de interés de Concepción
"Hospital Regional"
"Hospital Higueras"
"Hospital Clínico del Sur"
“Universidad de Concepción”
"Terminal Collao"
"Mall del Centro"
“Plaza de la Independencia de Concepción"

"Sumo Sushi Bar"
“Furai Delivery Concepcion”
“Hotel Diego de Almagro Concepción”
"Loft Single Rent Apartment"
"Hotel Camilo Henríquez"
“Hotel Atton Concepción”
"Hotel Eclipse"
"La Cevichería Concepción"
"Tiralomo Restaurant"
"BurgerBar"
"Faro Belén Restorán"
"La Pasta de la Nonna"
"Catus Pizza"
“Fuente Penquista"
"Manhattan"
"Quispe & Nisei"

Tabla 16. Puntos de interés

Anexo 3: Manual del Simulador

A continuación, se describe como se utiliza el simulador que se uso para realizar la experimentación.

1. Primero, en la ilustración 12 se muestra el simulador en el inicio de su ejecución. Que tiene un plano que representa una celda de un mapa de NxN que esta los POIs y el punto de acceso, y al lado derecho un cuadro blanco que muestra los resultados obtenidos luego de la simulación. Abajo, se muestra los datos de entrada que recibe el simulador.

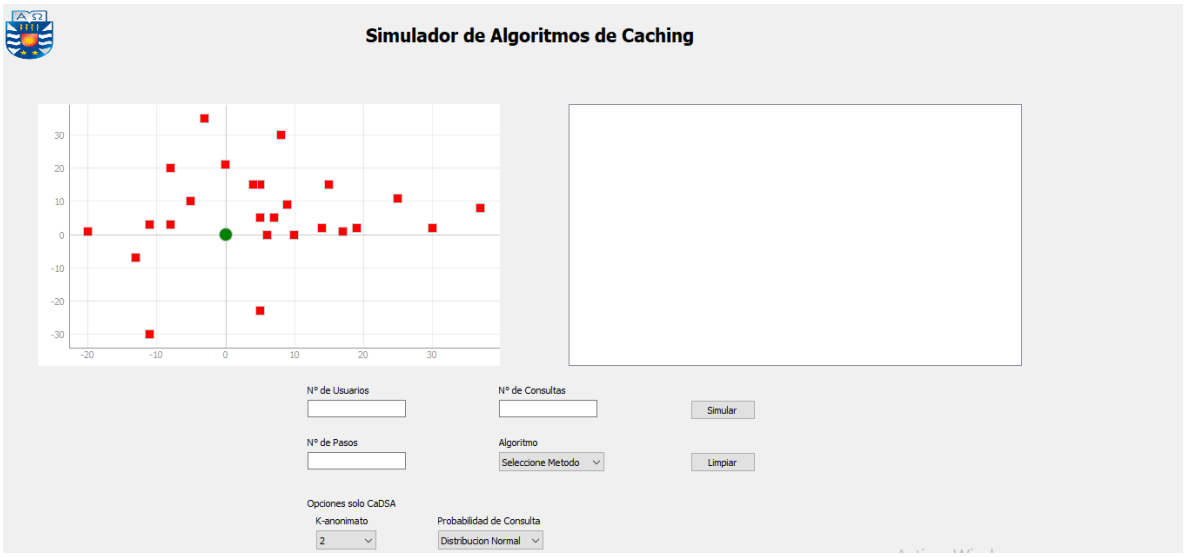


Ilustración 12. Inicio del Simulador

2. Luego, se debe ingresar los datos de entrada como se muestra en la ilustración 13 que son los siguientes:
    - N° de usuarios: Este dato es la cantidad de usuario que quieren enviar una LBQ para preguntar sobre un POI.
    - N° de consultas: Este dato es la cantidad de LBQs que serán procesadas.
    - N° pasos: Este dato es la cantidad de pasos que darán cada usuario por cada LBQ procesada, es decir, cada usuario estará en una posición diferente en cada consulta. Para lograr esto, se uso el modelo de movimiento Random Walk.
    - Algoritmo: Este dato es el algoritmo que será simulado, acorde a los datos de entrada mencionados anteriormente. Existen dos algoritmos que CaDSA y DCA.

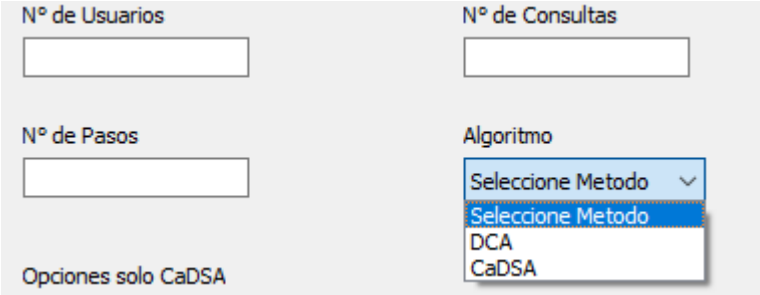


Ilustración 13. Datos de entrada principales

También, existe datos de entrada opcionales solamente para CaDSA como se muestra en la ilustración que son los siguientes:

- Distribución de probabilidad: Este dato es la distribución que se ocupara para obtener las probabilidades de consulta de cada punto del plano, ya que en cualquier punto puede ser una ubicación de un usuario. Existe distribución Normal y Cauchy.
- K-anonimato: Este dato indica la cantidad de posiciones distintas a escoger, entre mayor sea la cantidad, el grado de privacidad de ubicación es mejor. Empieza como un mínimo de 2.

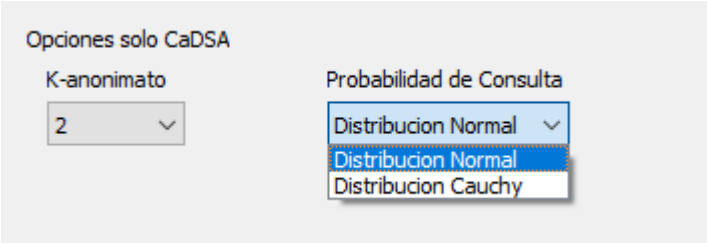


Ilustración 14. Opciones de CaDSA

3. Una vez ingresado los datos de entrada, se debe ejecutar la simulación del algoritmo de Caching escogido, se debe presionar el botón "Simular", este será encargado de ejecutar el algoritmo, luego se muestren los resultados en el cuadro blanco del lado derecho del plano. Si se quiere realizar otra simulación de un algoritmo, se necesitará borrar los resultados anteriores del cuadro de resultado y para eso se debe presionar el botón "Limpiar" como se muestra en la ilustración.

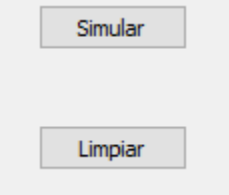


Ilustración 15. Botones del Simulador