Universidad del Bío-Bío
Facultad de Ciencias Empresariales
Departamento de Sistemas de Información

Implementación y simulación de algoritmo de construcción en batch de regiones de encubrimiento para usuarios de redes inalámbricas

---

Proyecto de Título para optar a título profesional de Ingeniería Civil en Informática

Autor

Guillermo Hideo Tobar Hashiguchi

Profesor Guía

Patricio Galdames Sepúlveda

Resumen

El presente proyecto se entrega para dar conformidad a los requisitos establecidos por la Universidad del Bío-Bío en el proceso de titulación de la carrera de Ingeniería Civil Informática.

El proyecto "Implementación y simulación de algoritmo de construcción en batch de regiones de encubrimiento para usuarios de redes inalámbricas" tiene como finalidad el simular diferentes técnicas propuestas de generación de regiones de encubrimiento para usuarios en redes inalámbricas que usan servicios basados en localización, mostrando los distintos resultados y comparándolas entre ellas para su análisis.

La aplicación se desarrolla para simular un ambiente real, mostrando regiones en donde hay una mayor o menor densidad de población y, por consiguiente, una mayor o menor posibilidad de que un usuario busque por productos o servicios en esa área. Las técnicas implementadas toman en cuenta los datos entregados por el usuario y los datos obtenidos desde la base de datos conectada a la aplicación para calcular los sectores más parecidos a la ubicación real y así entregar un conjunto de ubicaciones lo suficientemente parecidas para poder encubrir la ubicación real del usuario.

Abstract

The present proyect is delivered to satisfy the requirements established by the Universidad del Bío-Bío for the professional academic degree of Civil Engineering in Computer Science.

The proyect called “Implementation and simulation of batch construction algorithm of cloacking regions for wireless users” is intended to simulate different proposed techniques of generation of cloacking regions for users in wireless networks that uses location-based services showing the following results and comparing them for analysis.

The application is developed to simulate a real environment, showing regions where there is a greater or less population density and, consequently, a greater or lesser possibility that a user searches for products or services in that area. The techniques implemented consider the data delivered by the user and the data obtained from the database connected to the application to calculate the sectors which are most similar to the real location and thus deliver a set of similar enough locations to cover the user’s real one.

Índice General

Índice de tablas

Índice de Figuras

Capítulo 1: Introducción

Antiguamente, las personas se guiaban por rústicas brújulas, constelaciones o incluso por los cambios del sol, métodos que fueron dejándose de lado a medida que las nuevas tecnologías aparecían y cubrían distintas áreas, entre ellas la geográfica. Hoy en día, saber dónde estamos parados y a dónde queremos llegar solamente requiere de un dispositivo conectado a una red que entregue esa información, permitiendo que la orientación sea mucho más fácil.

Hoy si queremos buscar una estación de servicio, sólo tenemos que sacar un celular, preguntar a un servidor y este nos devolverá los resultados más cercanos de acuerdo a nuestra ubicación, lo que es una gran ventaja, sin embargo esta acción también conlleva algunos riesgos.

Entregar una posición a un servidor implica que el servidor sepa dónde estamos, si sabe nuestra localización puede saber distintas cosas, como donde vivimos, qué nos gusta, en qué lugar trabajamos e incluso cuáles son nuestros panoramas favoritos. Esta información, que a simple vista parece irrelevante, puede llevar a terceros a encontrar nuestras ubicaciones, pudiendo tomar diversas acciones en contra nuestra si es su intención e incluso generar un peligro para nuestras vidas en casos extremos.

Varios investigadores han tomado en cuenta este problema y han formulado distintas alternativas para solucionarlo, tomando en cuenta uno o más de varios enfoques existentes para centrar su trabajo de acuerdo a las distintas consecuencias que puede generar el entregar la ubicación a un servidor.

En este proyecto de título se abordan algunas de estas soluciones propuestas para evitar el problema planteado anteriormente. Abordando los temas de regiones de encubrimiento y grados de anonimato se usarán diferentes técnicas para, dado un punto geográfico simulado, ver qué técnica es la más adecuada y que entregue un mayor grado de privacidad para el usuario.

Los aportes que entrega este trabajo son los siguientes: Primero utiliza datos geográficos reales entregados por OpenStreetMap con los cuales se pueden obtener las coordenadas, generar un mapa real y realizar la simulación de una manera más confiable, además de poder incluir para la simulación a cualquier ciudad del mundo (entre las que se encuentren precargadas en la base de datos al momento de realizar la simulación). Otro punto relevante es que entre las técnicas de simulación se encuentra una nueva técnica desarrollada por el profesor Patricio Galdames, la cual se centra en unificar los enfoques que otras técnicas toman por separado y en la utilización de sistemas de usuarios en batch.

Este informe cuenta de 10 capítulos que definen el proyecto de título. El contenido de estos se detalla a continuación:

Capítulo 1 "Introducción": Se presenta el propósito de este documento, se entrega información sobre los aportes entregados y se detallan los capítulos que el informe contiene.

Capítulo 2 "Definición de la institución": Se presenta la institución para la cual se desarrolla este proyecto de título.

Capítulo 3 "Definición de proyecto": Se da a conocer los objetivos, metodologías, herramientas de apoyo al desarrollo del software y definiciones usadas en este informe.

Capítulo 4 "Estado del arte": Se describen las tecnologías usadas en el proyecto, los trabajos relacionados y las técnicas seleccionadas para ser simuladas.

Capítulo 5 "Ajustes previos para el simulador": Se detallan los pasos previos que se requieren para poder utilizar coordenadas reales y se explica la importancia de la inclusión de la base de datos.

Capítulo 6 "Especificaciones de Requerimientos de Software": Se describen los alcances, objetivos y requisitos mínimos del software, así como la descripción global del producto y los requerimientos específicos.

Capítulo 7 "Simulaciones": Se explican las secciones del software.

Capítulo 8 "Análisis": Se definen los casos de uso y sus especificaciones.

Capítulo 9 Conclusiones: Se entregan las conclusiones del proyecto.

Capítulo 10 Bibliografía: Se detalla la bibliografía utilizada para la elaboración de este proyecto.

Capítulo 2: Definición de la Institución

2.1 Descripción de la Empresa

La Universidad del Bío-Bío, institución para la cual se realiza este proyecto de título, inicia sus funciones el 9 de abril de 1947 como la Universidad Técnica del Estado, bajo la presidencia de Gabriel González Videla.

Si bien inició sus funciones años atrás, no fue hasta 1952 que el senado aprobó su Estatuto Orgánico, permitiendo que la universidad abriera sus puertas oficialmente, para así poder responder a las necesidades y desafíos de la región del Biobío para convertirse en uno de los polos del desarrollo industrial de Chile, tanto a nivel de docencia como a nivel de investigación científica y tecnológica.

En 1981, debido a un cambio radical en el sistema universitario chileno, la Universidad Técnica del Estado (UTE) se convierte en la Universidad del Bío-Bío ya que en ella se impartía la carrera de Arquitectura, una de las doce definidas en ese entonces como universitarias.

Hoy en día la Universidad del Bío-Bío, compuesta por sus 4 campus (Concepción, La Castilla, Fernando May y Andrés Bello), es una de las pocas universidades estatales de Chile y forma parte del Consejo de Rectores de las Universidades Chilenas. Cuenta además con una acreditación de otorgada por la Comisión Nacional de Acreditación (CNA) durante un periodo de 5 años (2014 – 2019) en las áreas de gestión, docencia, investigación y vinculación con el medio.

2.2 Visión

Ser reconocida a nivel nacional e internacional como una Universidad pública, responsable socialmente y regional que, comprometida con su rol estatal, desde la Región del Biobío, forma personas integrales de excelencia y aporta a través de su quehacer al desarrollo sustentable de la región y el país.

2.3 Misión

La Universidad del Bío-Bío, a partir de su naturaleza pública, responsable socialmente y estatal, tiene por misión, desde la Región del Biobío, aportar a la sociedad con la formación de personas integrales, a través de una Educación Superior de excelencia. Comprometida con los desafíos de la región y del país, contribuye a la movilidad e integración social por medio de; la generación y transferencia de conocimiento avanzado, mediante la docencia de pregrado y postgrado de calidad, la investigación fundamental, aplicada y de desarrollo, la vinculación bidireccional con el medio, la formación continua y la extensión. Asimismo, impulsa el emprendimiento y la innovación, el fortalecimiento de la internacionalización y el desarrollo sustentable de sus actividades, basada en una cultura participativa centrada en el respeto a las personas.

Capítulo 3: Definición de Proyecto

3.1 Objetivos del Proyecto

3.1.1 Objetivo General:

- Implementar y comparar con el estado del arte, una nueva técnica de construcción de regiones de encubrimiento para usuarios de redes inalámbricas.

3.1.2 Objetivos Específicos:

- Realizar estudio de las diversas técnicas de construcción de regiones de encubrimiento y seleccionar entre las siguientes tres técnicas: Una que construya una región de encubrimiento para proteger anonimato, una para proteger la privacidad de ubicación y una para la seguridad física de ubicación.
- Desarrollar un ambiente gráfico que muestre distintos mapas según regiones por medio de uso de datos geográficos reales.
- Desarrollar una aplicación que permita gráficamente simular las técnicas propuestas.
- Desarrollar aplicación que evalúe el rendimiento de las técnicas bajo diversos parámetros, como número de solicitudes por región de encubrimiento o distribución geográfica de los usuarios.

3.2 Ambiente de Ingeniería de Software

3.2.1 Metodología de Desarrollo

Para el desarrollo del proyecto se utilizó la metodología RUP. Esta metodología es la más utilizada para el análisis, implementación y desarrollo de sistemas orientados a objetos.

RUP tiene las siguientes fases:

1. Fase de Inicio: Esta fase tiene como propósito definir y acordar el alcance del proyecto, identificar los riesgos asociados este, proponer una visión muy general de la arquitectura de software y producir el plan de las fases y el de iteraciones posteriores.

2. Fase de elaboración: En la fase de elaboración se seleccionan los casos de uso que permiten definir la arquitectura base del sistema y se desarrollaran en esta fase, se realiza la especificación de los casos de uso seleccionados y el primer análisis del dominio del problema, se diseña la solución preliminar.

3. Fase de Desarrollo: El propósito de esta fase es completar la funcionalidad del sistema, para ello se deben clarificar los requerimientos pendientes, administrar los cambios de acuerdo a las evaluaciones realizados por los usuarios y se realizan las mejoras para el proyecto.

4. Fase de Cierre: En este punto se asegura que el software esté disponible para los usuarios finales, se ajustan errores y defectos encontrados en las pruebas de aceptación. Se debe verificar que el producto cumpla con las especificaciones entregadas por las personas involucradas en el proyecto.

3.2.2 Herramientas de apoyo al desarrollo de software

Netbeans IDE 8.2: Netbeans es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) libre y gratuito, hecho principalmente para el lenguaje de programación Java.

Power Designer 16: Power Designer es una herramienta para la administración de metadatos y modelado de objetos.

3.3 Definiciones, siglas y abreviaciones

3.3.1 Definiciones

K-Anonymity [4]: Conocido como K-Anonimato, este término una propiedad característica de los datos anonimizados. El concepto fue introducido por primera vez por Latanya Sweeney en 2002. Se dice que un conjunto de datos publicados tiene la propiedad de k-anonimato (o es k-anónimo) si la información de todas y cada una de las personas contenidas en ese conjunto es idéntica al menos con otras k-1 personas que también aparecen en dicho conjunto.

Geolocalización: La geolocalización es la capacidad para obtener la ubicación geográfica real de un objeto, como un radar, un teléfono móvil o un ordenador conectado a Internet. La geolocalización puede referirse a la consulta de la ubicación, o bien para la consulta real de la ubicación.

OpenStreetMap: Conocido también como “OSM”, OpenStreetMap es un proyecto colaborativo cuya finalidad es crear mapas libres y editables.

PostgreSQL: Sistema de gestión de base de datos relacional de código abierto y orientado a objetos.

Ortofotografías: Representación fotográfica de una zona de la superficie terrestre, en la que todos los elementos presentan la misma escala, libre de errores y deformaciones, con la misma validez de un plano cartográfico.

Shapefile: El formato shapefile es un formato de archivo informático de datos espaciales, creado para ser utilizado por el programa ArcView GIS, pero que actualmente se ha vuelto en el formato estándar para el intercambio de información geográfica.

Adversario: Persona o sistema que busca oponerse a la privacidad de los usuarios recopilando información que podría ser perjudicial para estos.

Entropía: Medida de incertidumbre de una fuente de información.

RAW: Raw es un formato en el cual los datos no se comprimen en absoluto o, de comprimirse, esta compresión de datos no supone una pérdida de información.

3.3.2 Siglas y abreviaciones

IDE: Integrated development Environment o Entorno de Desarrollo Integrado es una aplicación informática que proporciona servicios integrales para facilitar el desarrollo de software.

JDK: Java Development Kit es un software que proporciona un conjunto de herramientas de desarrollo para la creación de programas en java.

JRE: Java Runtime Environment es un conjunto de utilidades que premite la ejecución de programas java en un ordenador.

LBS: Location-Based Service o Servicios basados en localización son servicios que utilizan la ubicación de una persona como referencia para encontrar lugares cercanos de interés para la persona que hace una consulta.

LDS: Location-Depersonalisation Service o Servidor de despersonalización es un servicio por el cual pasan los datos de los usuarios para realizar regiones de encubrimiento de acuerdo a las técnicas que utilicen.

GPS: Global Positioning System o sistema de posicionamiento global es un sistema que permite determinar en toda la tierra la posición de un objeto con precisión de hasta unos centímetros (dependiendo de la tecnología utilizada).

DLS: Dummy-Location Selection o algoritmo de ubicación de selecciones falsas es un algoritmo creado para alcanzar el anonimato.

CR: Cloacking Region o región de encubrimiento, es el área que forman las ubicaciones compuestas por la ubicación real más las ubicaciones falsas.

MOP: Multi-Objective Optimization Problem o Problema de optimización multi-objetivo.

Capítulo 4: Estado del Arte

En este capítulo se comentarán y definirán las tecnologías y herramientas utilizadas en el simulador, así como también los trabajos relacionados y las técnicas utilizadas para llevar a cabo las simulaciones y la obtención de las coordenadas para desplegar el mapa.

4.1 OpenStreetMap

OpenStreetMap (más conocido como "OMS") es un proyecto colaborativo para crear mapas de todo el mundo que sean libres y editables por la comunidad.

Estos mapas se crean utilizando diferentes medios que entregan información geográfica, tales como dispositivos GPS, ortofotografías y otras fuentes libres a las que pueda tener acceso un usuario. Los datos vectoriales de estos mapas son almacenados en bases de datos y, junto con las imágenes creadas, son distribuidos bajo una licencia llamada "Licencia Abierta de Bases de Datos", por lo que pueden ser accedidos por cualquier persona que esté interesada en aportar a mejorar los mapas, o usuarios que quieran crear aplicaciones en distintas plataformas (tales como Android, PC, plataformas web, entre otras).

OpenStreetMap es fundada en 2004 por Steve Coast en respuesta a los altos cobros que se cobraba en aquellos tiempos por la información geográfica. Dos años más tarde OpenStreetMap se convierte en fundación y comienza a ser tomada en cuenta por más personas y grupos, llegando a obtener información geográfica donada de distintas fuentes (como empresas y entidades gubernamentales), así como autorizaciones de terceros para utilizar sus tecnologías de forma gratuita. Así mismo con el paso de los años OpenStreetMaps empieza a ser incorporada por grandes empresas como Apple y Foursquare (este último abandonando a Google Maps debido a cambios en cobros).

En esta aplicación se utiliza la base de datos de OpenstreetMap para poder generar el mapa y para poder obtener las coordenadas necesarias para medir la densidad de población.

4.2 PostgreSQL

PostgreSQL es un sistema de gestión de bases de datos de código abierto manejado por una comunidad de desarrolladores que trabajan de forma gratuita y desinteresada.

Siendo sus inicios en 1985, PostgreSQL fue liderado por Michael StoneBraker, desarrollador que también lideró en 1982 el proyecto Ingres, el cual fue uno de los primeros intentos en implementar una base de datos relacional. Esta primera experiencia sirvió de base para el proyecto que partió llamándose post-ingres, para después ser llamado simplemente Postgres.

PostgreSQL posee, además de su base de datos, herramientas que permiten extender su funcionalidad, para así poder realizar tareas que comúnmente no podrían ser soportadas por este sistema.

4.2.1 PostGIS

PostGIS es una extensión de PostgreSQL que añade soporte de objetos geográficos al sistema y que es la herramienta utilizada en la aplicación de este proyecto de título para poder procesar los datos geográficos de OpenStreetMap y obtener las coordenadas de estos.

PostGIS es un módulo que añade soporte de objetos geográficos a la base de datos, convirtiéndola en una base de datos espacial, lo cual permite que sea utilizada para sistemas de información geográfica.

4.2.2 Osm2pgsql

Osm2psql es un programa basado en líneas de comandos que convierte datos de OpenStreetMap a bases de datos listas para ser insertadas en un sistema de gestión de bases de datos PostgreSQL que tenga instalado postGIS.

Con esta herramienta se han podido transformar e insertar los datos de las ciudades necesarias para la realización de la simulación.

4.3 Trabajos Relacionados

En el siguiente punto se detalla la información de los trabajos relacionados al tema de este proyecto de título, los cuales fueron la base para escoger las diferentes técnicas de simulación implementadas en la aplicación.

4.3.1 Ben Niu, Qinghua Li, Xiaoyan Zhu, Guohong Cao, Hui Li "Achieving k-anonymity in Privacy-Aware Location-Based Services" [1]:

El trabajo realizado por estos autores detalla la posibilidad de crear una región de encubrimiento a través de dos algoritmos llamados algoritmo DLS (Dummy-Location Selection) y algoritmo DLS mejorado para alcanzar un grado de anonimato en los servidores LBS (Location-Based Services) conocido como k-anonymity [4]. En el informe los autores detallan la problemática al respecto del anonimato en la red, y proponen dos soluciones que distan de otras propuestas. Los autores señalan diferentes puntos a tomar en consideración para la máxima eficiencia de sus algoritmos, un ejemplo es evitar utilizar un servidor como anonimizador para evitar cuellos de botella y a la vez reducir la probabilidad de que un atacante pueda ingresar al servidor anonimizador y así obtener la posición real del usuario.

Los algoritmos que proponen utilizan diferentes puntos geográficos dentro de un área seleccionada para generar diversos usuarios falsos que tengan una probabilidad de hacer una consulta igual o muy parecida a la del usuario real. Si bien el algoritmo DLS utiliza una técnica de conjuntos aleatorios para realizar sus puntos de anonimato, el algoritmo DLS va más allá, al realizar los mismos pasos tomando en cuenta, además de la entropía, la región de encubrimiento.

4.3.2 Marco Gruteser, Dirk Grunwald "Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking [2]:

Marco Gruteser [2] uno de los primeros en plantear el problema de la privacidad en la geolocalización, presentando junto a eso una solución al problema.

El trabajo desarrollado por Gruteser y Grunwald [2] propone una técnica que se concentra en el principio de la recolección mínima. Usando k-anonymity [4] ellos entregan a los LBS sólo datos despersonalizados, lo cual, según el autor, genera un beneficio tanto para el usuario como para el el proveedor del servicio.

Los autores proponen el uso de k-1 usuarios que se encuentren en el área de un usuario que realice una consulta para que, juntando todos, se cree una región de encubrimiento de k usuarios.

4.3.3 Toby Xu, Ying Cai "Feeling-based Location Privacy Protection for Location-based Services" [3]:

En este informe los autores señalan que es problemático el enfoque tradicional de k-anonymity [4] porque la privacidad es acerca de sensaciones, y es incómodo para uno escalar su sentir usando un número, por lo que proponen un modelo de privacidad basado en sensaciones de tranquilidad en un área determinada, omitiendo explícitamente el uso de un número k y usando un sistema de "popularidad de área" para poder traducir esta sensación en una aproximación que los autores llaman "modelado de privacidad basado en sensaciones".

4.3.4 Patricio Galdames "Construcción y procesamiento eficiente de regiones de encubrimiento en redes inalámbricas":

El profesor Patricio Galdames Sepúlveda, docente de la Universidad del Bío Bío, campus Concepción, señala que las técnicas de protección de la privacidad de ubicación de usuarios en redes inalámbricas tienen diversos inconvenientes, tales como que no abarcan todas las exigencias de protección de ubicación que un usuario puede demandar, o que puede generar problemas de escalabilidad al realizar una construcción de encubrimiento para un solo usuario. El autor aborda el problema al desarrollar un algoritmo que construya regiones de encubrimiento en batch para múltiples usuarios, y que a la vez considere tanto la diversidad de requerimientos de despersonalización de los mismos como el impacto que genera en el LBS y LDS la construcción y procesamiento de estas regiones de encubrimiento.

4.4 Clasificación de técnicas de encubrimiento

Los trabajos antes mencionados consideran una o varias categorías de técnicas de encubrimiento para realizar su enfoque. En esta sección se explican las tres categorías tomadas como enfoque para realizar los algoritmos en cada uno de los trabajos.

4.4.1 Uso anónimo de servicios:

Las técnicas en esta categoría buscan prevenir que la información reunida por los servidores de LBS pueda usarse para identificar a un usuario que requiere de un servicio. Cuando un usuario solicita un servicio a un LBS, estos esquemas calculan una región de encubrimiento que contiene al usuario y al menos otros K-1 vecinos. Esta región es luego reportada como la posición del usuario al correspondiente proveedor LBS. Ya que de los K usuarios, uno de ellos es quien solicita el servicio, se debe garantizar que la probabilidad de inferir la posición del solicitante es 1/K.

4.4.2 Protección de la Privacidad de Ubicación:

Las técnicas que usan el enfoque anterior protegen el anonimato del usuario al usar un servicio, pero no su privacidad de ubicación. Un adversario podría identificar a los usuarios en la región de encubrimiento utilizando información pública. Este adversario podría no saber quién solicitó el servicio pero sabe la ubicación de estos usuarios. De hecho, estos estaban en la región al momento de solicitar el servicio, por lo que contando con esta información la ubicación del usuario y este mismo se encontrarán vulnerables a cualquier ataque. Para resolver este problema diversas técnicas se han propuesto que buscan determinar K-1 posiciones que hayan sido tan visitadas como la posición del usuario que solicita protección. Un adversario podría identificar a estos visitantes pero no sabría quién estuvo en esas posiciones al momento que el servicio LBS fue solicitado.

4.4.3 Protección de Seguridad Física de Ubicación:

En este enfoque, al contrario de los anteriores, no predomina el utilizar ubicaciones falsas que estén cerca del usuario, ya que en este caso se toma en cuenta que el adversario no está interesado en encontrar la identidad de una persona en específico, sino que de ver en qué secciones hay una mayor concentración de población para comprometer la seguridad del grupo más grande posible al mismo tiempo.

4.5 Descripción y análisis de técnicas a utilizar

En esta sección se detallan y analizan las técnicas seleccionadas para esta simulación, con el objetivo de comprender en mejor medida cada una de ellas, el por qué fueron seleccionadas y el cómo son ejecutadas.

4.5.1 "Achieving k-anonymity in Privacy-Aware Location-Based Services" [1]:

Los autores Ben Niu, Qinghua Li, Xiaoyan Zhu, Guohong Cao y Hui Li [1] son los investigadores que dieron con esta solución que se encontró una de las más adecuadas para llevar a cabo.

Los autores señalan que los otros algoritmos basados en k-anonimato propuestos anteriormente son deficientes en varios aspectos como la confianza en el servidor LDS, el cual puede sufrir de un ataque cibernético y dejar expuestos a todos los usuarios que estén utilizando el servicio. Otro problema que los autores hacen notar es que el servidor LDS puede generar un cuello de botella si este recibe demasiada información al mismo tiempo para procesar.

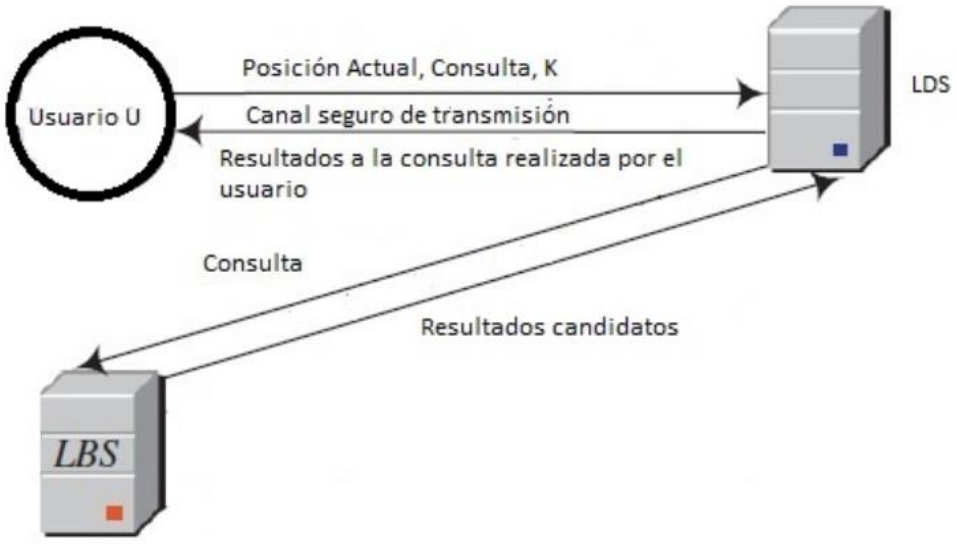


Figura1: Arquitectura tradicional de comunicación entre un LBS, un LDS y un usuario.

Otro problema que señalan es que la mayoría de enfoques existentes no toman en cuenta la posibilidad de que el adversario pudiera tener información complementaria, lo que permitiría al adversario descartar varios puntos falsos en el mapa. En la figura 2 se puede apreciar que el enfoque tradicional de k-anonymity [4] no toma en cuenta la probabilidad del usuario, por lo que si el adversario cuenta con esta información complementaria, puede descartar fácilmente los usuarios falsos, dejando sólo el usuario real (1) y los usuarios con la misma probabilidad (2) y (3), lo cual disminuye en gran medida la eficiencia del algoritmo.

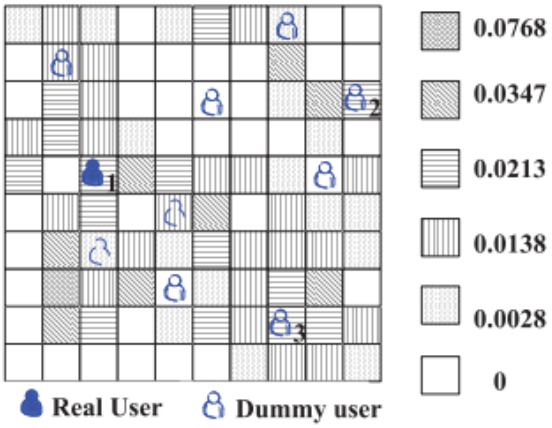


Figura 2, enfoque tradicional

Para resolver este problema los autores plantearon dos algoritmos: DLS y DLS mejorado. Estos algoritmos buscan lograr un enfoque diferente a los algoritmos previos al buscar posiciones falsas de manera cuidadosa, tomando en cuenta la información complementaria que podría ser explotada por los adversarios.

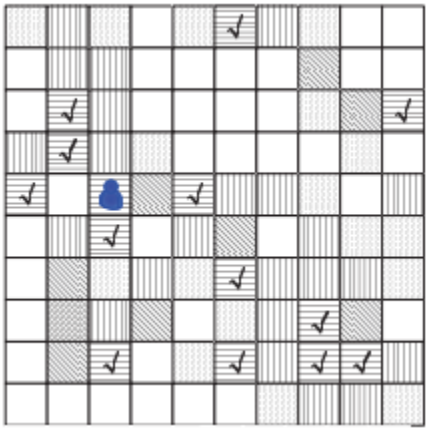


Figura 3, celdas candidatos seleccionadas con el algoritmo DLS.

4.5.1.1 Algoritmo DLS:

El primer algoritmo DLS trabaja con base en la entropía. Suponiendo que el usuario real está en una casilla, se seleccionan 2k usuarios para luego, de forma aleatoria, seleccionar k-1 usuarios y juntarlos con el usuario real, obteniendo así k usuarios con probabilidades similares. En la figura 3 se muestran al usuario real en un tipo de casilla y a 13 candidatos con probabilidades similares, de esta forma el adversario no es capaz de descartar usuarios falsos porque, a su visión, todos tienen la misma probabilidad ser el usuario real.

Esta solución se logra dividiendo el mapa en NxN celdas con igual tamaño, cada celda tiene una probabilidad de consulta basada en su historial de consultas previas, al cual podemos denotar como

$$q_i = \frac{\#\ of\ queries\ in\ cell\ i}{\#\ of\ queries\ in\ whole\ map}, i = 1, 2, \cdots, n^2$$

Donde $\sum_{i=1}^{n^2} q_i = 1.$.

Como primer paso, el usuario debe insertar la posición del usuario real y elegir un grado de anonimato "k", el que será el número de usuarios finales (k-1 usuarios falsos más el usuario real) que conformarán la región de encubrimiento. Además se deberá insertar la cantidad de conjuntos (denotado como "m") que el usuario desea comparar para buscar y elegir al conjunto de k posiciones con mayor Entropía.

Luego de haber hecho esos pasos, el sistema tomará las probabilidades de consultas y las comparará, seleccionando las 2k ubicaciones que tengan las probabilidades más cercanas a la probabilidad que tiene la posición del usuario real. Una vez encontradas se toman m conjuntos aleatoriamente seleccionados con tamaño k-1 y se agrega la ubicación real.

El J_{vo} ($J \in [1, m]$) set se puede denominar como $C_j = [c_{j1}, c_{j2},..., c_{ji},..., c_{jk}]$ y después de haber sacado la probabilidad q_i de cada celda, se puede pasar a normalizarlas con la fórmula

$$p_{ji} = \frac{q_{ji}}{\sum_{l=1}^{k} q_{jl}}, i = 1, 2, \cdots, k,$$

donde la suma de las probabilidades normalizadas es 1.

Finalmente podemos determinar el conjunto C_j óptimo al sacar la entropía "H" de cada conjunto con la fórmula

$$H_j = -\sum_{i=1}^{k} p_{ji} \cdot \log_2 p_{ji},$$

formando la región de encubrimiento suficientemente óptima.

Algoritmo 1: Algoritmo de selección de ubicación falsa

Entrada: Probabilidades de consulta en historial q_i, ubicación real l_{real}, número de sets m, k

Salida: un óptimo set de ubicaciones falsas

1- Ordenar las celdas basadas en su probabilidad de consulta.
2- Escoger 2k candidatos falsos entre los cuales k candidatos están justo antes de L_{real} y k candidatos están justo después de L_{Real} en la lista ordenada.
3- For(j=1;j ≤m;j++)do
   a. Construir set C_j el cual contiene l_{real} y k-1 otras celdas aleatoriamente selecionadas desde los 2k candidatos;
   b. Calcular la propiedad normalizada p_{ji} para cada celda c_{ji} en el set;
   c. $H_j \leftarrow -\sum_{i=1}^{k} p_{ji} \cdot \log_2 p_{ji}$;
4- End
5- Genera arg max H_j

4.5.1.2 Algoritmo DLS mejorado:

Aún cuando el algoritmo DLS obtiene un conjunto de candidatos suficientemente efectivo, el nivel de privacidad puede ser mejorado al considerar, además de entropía, la región de encubrimiento CR. Al usar dos criterios este problema pasa a ser un problema de optimización de multi-objetivos (MOP), ya que a la vez de obtener la mayor entropía posible, también queremos maximizar CR para esparcir las ubicaciones falsas tanto como sea posible.

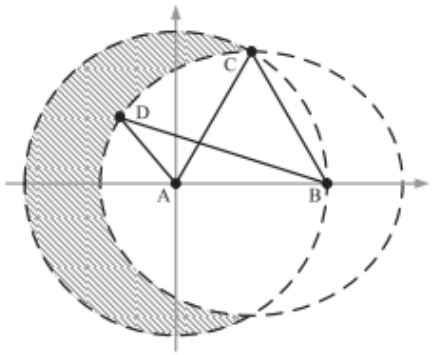


Figura 4: Escenario para DLS mejorado

Para comprender cómo medir CR, tomemos el ejemplo de la figura 4. En este ejemplo la ubicación real es "A", siendo "B" la ubicación más lejana, es escogida como la primera ubicación falsa. Si suponemos que tenemos dos elecciones para la siguiente ubicación falsa, "C" y "D", entonces debemos encontrar un método que nos permita decidir cuál de las dos es la mejor. Para este problema el mejor método es obtener el producto de las distancias entre pares de ubicaciones, lo que en este caso sería CA*CA>DA*DB. Así podemos definir que la siguiente ubicación más óptima es "C".

Así entonces, si tenemos un conjunto C = [C_1, C_2,..., C_k] el cual contiene las ubicaciones falsas y reales, el MOP puede describirse como

$$Max\{-\sum_{i=1}^{k} p_i \cdot \log_2 p_i, \prod_{i \neq j} d(c_i, c_j)\}$$

,

donde C_i, $C_j \in C$, p_i y p_j denota las probabilidades de consulta C_i y C_j, respectivamente.

Para los autores de este algoritmo, la condición básica para elegir un set de ubicaciones falsas para alcanzar la mayor entropía es

$$\mathcal{C} = \arg\max(-\sum_{i=1}^{k} p_i \cdot \log_2 p_i)$$

.

Pasando al planteamiento, lo que se hace primero es seguir las líneas 1-2 del algoritmo 1 para elegir 4k de candidatos como ubicaciones falsas, entonces, al seguir las líneas 3-5 del algoritmo 1, podemos construir un set más pequeño de 2k candidatos desde los 4k originales. Con estos pasos podemos maximizar la parte de entropía del algoritmo, por lo que podemos pasar a la parte de CR.

Luego de haber seguido los pasos anteriores, lo que se hace a continuación es conseguir las k-1 ubicaciones falsas que permitan maximizar CR (C_1, C_2,..., C_{k-1}). Para hacer esto se tiene que tomar la posición real, la cual definiremos como C_{real} e ir escogiendo la posición C_1, C_2, etc. en cada ronda hasta completar k-1 rondas, asignando en cada ronda un peso a los candidatos restantes y eligiendo el siguiente candidato con una probabilidad proporcional a su peso, siendo el peso de cada candidato su distancia con C_{real}.

Así, en la primera ronda se elige al candidato que esté más lejos de C_{real}, en la segunda ronda el peso del candidato es el producto de las distancias entre C_{real} y el primer candidato, y en la tercera ronda es el producto de la distancia entre C_{real} y el segundo candidato escogido, siguiendo así hasta completar el conjunto C.

Algoritmo 2: El algoritmo DLS mejorado

Entrada: Probabilidades de consulta en el historial q_i, la actual celda C_{real}, el número de sets m y k.

Salida: Un óptimo set de ubicaciones falsas.

1. Seguir las líneas 1-2 del algoritmo 1 para escoger 4k candidatos a ubicaciones falsas.
2. Seguir las líneas 3-5 del algoritmo 1 para escoger 2k candidatos a ubicaciones falsas C = {C_1, C_2,…, C_{2k}}.
3. C ← {C_{real}}.
    a. For(i=1;i ≤ k-1;i++)do
    b. Escoger c' como un candidato $c_j \in C^{\wedge}$ de tal manera que c_j es escogido con probabilidad $\frac{\prod_{c_l \in C} d(c_j, c_l)}{\sum_{c_j \in \hat{C}} \prod_{c_l \in C} d(c_j, c_l)}$;
    c. C ← C ∪ {c'};
    d. Remover c' de C^
4. End.
5. Sacar C;

4.5.2 Construcción y procesamiento eficiente de regiones de encubrimiento en redes inalámbricas:

El otro algoritmo que es de interés explicar es el del profesor Patricio Galdames. El enfoque que él desarrolla se basa en aplicar lo mejor posible tanto el uso anónimo de servicios, como la protección de la privacidad de la ubicación y la protección de la seguridad física de ubicación.

El algoritmo del profesor Galdames tiene base en el algoritmo de Ben Niu [1], por lo que inicialmente puede ser parecido, sin embargo en este algoritmo se utiliza un conjunto de usuarios (llamado U_{batch}) para generar una red de encubrimiento para varios usuarios a la vez. Aprovechando de mejor forma los recursos del servidor LDS y cargando de menor manera el LBS.

La idea presentada por el profesor Galdames es realizar, como se ha dicho anteriormente, un algoritmo en primera instancia parecido al de Ben Niu [1], pero con la particularidad de que en este caso es un grupo de usuarios en vez de uno solo. Cada usuario posee su propio valor k deseado, por lo que el autor toma el valor mayor de k (llamado K_{max}) y crea una región de encubrimiento (CR) con la mejor entropía posible, luego revisa todos los usuarios que tengan el mismo valor de k y se crea una región de encubrimiento que satisfaga el caso de tener una alta entropía, pero también que pueda compartir la mayor cantidad posible de elementos en la región de encubrimiento.

Una vez se han obtenido valores óptimos para la región de encubrimiento de todos los usuarios que tengan K_{max}, se busca el nuevo K_{max} (menor en este caso que el anterior) para iniciar nuevamente el proceso de creación de la región de encubrimiento y poder compartir la mayor cantidad de posiciones de la región con los usuarios anteriormente seleccionados. Este proceso se repite hasta que todos los usuarios de U_{batch} tienen su región de encubrimiento.

Para entender mejor el proceso se detallan los algoritmos que componen la técnica explicada:

Algoritmo 0:

- Sea C una lista ordenada de las celdas en las que el espacio de los usuarios es particionado. La lista C es ordenada por la proporción de consultas basadas en la ubicación (q) de cada celda. Luego si C_i es la celda de ID: I, entonces la proporción se denota como q_i.
- Sea C_U: La celda que contiene al usuario U (la posición).
- Sea S(R) un subconjunto de C que contiene a la celda C_U, también contiene las R celdas cuyo q es menor o igual al de la celda C_U, y finalmente también aquellas R celdas cuyo q es mayor o igual al de la celda C_U. Ese conjunto contiene 2R+1 celdas distintas.

Entrada: ID del usuario U que demanda una CR, valor de anonimato (K) de U y rango de búsqueda (R>=K).

Salida: CR (región de encubrimiento) para U.

1. Sea CR= {}. Repetir m veces:
    a. Sea O= {C_U}.
    b. En O agregar k-1 celdas seleccionadas al azar de S(k)\ {C_U}.
    c. Si CR es vacío||la entropía de O es > a la entropía de CR
        i. Hacer CR=O
2. Retornar CR

Variante Batch:

Sea U_{batch} = Lista de usuarios que solicitan una región de encubrimiento.

Sea U(K_{max}) = Lista de usuarios en U_{batch} con K igual al máximo (K_{max}).

Sea G(CR): Lista de usuarios en U(K_{max}) cuya celda está en CR.

Entrada: U_{batch}

Salida: Las CR de cada uno de los usuarios en U_{batch}

1. Hacer:
    a. Averiguar si aún hay usuarios con K_{max}, si no los hay recalcular Kmax.
    b. Seleccionar usuario (U) en U(K_{max}) con K=K_{max}. Si hay varios usuarios con K=K_{max} seleccionar entre ellos uno al azar.
    c. CR=Algoritmo(O(U,K,K).
    d. Asignar CR a todos los usuarios en U_{batch} cuya celda está en G(CR).
    e. U_{batch} = U_{batch} (G(CR))
2. Mientras U_{batch} no sea vacío

Capítulo 5: Ajustes previos para el simulador

5.1 Uso de base de datos postgreSQL y OpenStreetMap [21]

La aplicación cuenta con un conjunto de bases de datos que contiene coordenadas reales obtenidas desde la comunidad OpenStreetMap, esto permite, adaptando estas coordenadas, generar mapas reales de cada región que se encuentre cargada en la base de datos.

Para lograr esto se precisó de una serie de pasos, los cuales son enumerados a continuación:

1. **Aprender sobre los formatos de mapas de OpenStreetMap:**

   OpenStreetMap tiene varios formatos para distintos tipos de aplicaciones, por lo que hay que elegir el tipo de formato que se adapte a los requerimientos para esta aplicación y descargar la región del mapa deseado.

2. **Preparar el entorno adecuado para cargar el mapa (instalación de postgreSQL):**
   Los mapas no vienen en formato de archivo de texto, por lo que para su correcta lectura es necesario un programa externo que gestione el mapa en forma de base de datos. En este punto entra en juego PostgreSQL, el cual, gracias a su extensión PostGIS, es capaz de soportar y gestionar objetos geográficos, creando el medio necesario para extraer los datos de manera rápida. Una vez se tiene instalado PostgreSQL junto a su extensión, se puede proceder a crear la base de datos para luego cargar el mapa deseado.

3. **Transformación de archivos (uso de osm2pgsql):**
   Como se dijo anteriormente, existen varios tipos de archivos. Lo más fácil es cargar datos a Postgres con formato shapefile, pero este archivo no posee todos los datos y podría dejarnos sin lo que necesitamos. Es por eso que es recomendado usar datos raw. Sin embargo para ingresar estos archivos raw en la base de datos creada en el punto anterior, es necesario convertirlos con la herramienta osm2pgsql, la cual además sirve para importar los datos necesarios en la base de datos.

4. **Testeo de la base de datos:**
   Una vez los datos están importados a la base de datos, podemos testear la integridad de la misma a través del programa QGIS, este programa usa la base de datos para cargar el mapa que esta contiene a través de la generación de polígonos. Con esto podemos ver que la base de datos haya cargado correctamente las coordenadas.

En definitiva, el uso de coordenadas reales hace necesario el manejo de una base de datos para cargarlos y leerlos, por lo que aún si la aplicación en sí no crea, modifica o guarda información en la base de datos, aún debe conectarse a esta para lograr cargar las coordenadas y mostrar el mapa.

Capítulo 6: Especificaciones de Requerimientos de Software

6.1 Alcances

El objetivo del proyecto a desarrollar es crear una aplicación que permita simular técnicas de encubrimiento de geolocalización ya propuestas y compararlas con una nueva técnica propuesta al hacer consultas a un servidor LBS.

La aplicación permite:

- Conectarse a una base de datos PostgreSQL para cargar datos necesarios.
- Generar mapas reales basados en coordenadas geográficas entregadas por la comunidad de OpenStreetMap.
- Simular técnicas de encubrimiento en un área geográfica real.
- Mostrar una grilla diferenciando el área en donde está el usuario real.

La aplicación no permite:

- No genera mapas ficticios.
- No se ingresa más de un usuario real en la búsqueda al mismo tiempo.
- Generar más usuarios falsos que el número de áreas existentes en el mapa.
- Cargar datos geográficos que no se encuentren en la base de datos.

6.2 Objetivos del Software

6.2.1 Objetivo general

La comparación entre diferentes técnicas de encubrimiento de geolocalización a través del tiempo para usuarios de redes inalámbricas.

6.2.2 Objetivos específicos

- La aplicación muestra un mapa real cargado por base de datos PostgreSQL.

- El usuario elige una técnica de ocultamiento entre las disponibles en la aplicación para realizar la simulación.

- La aplicación ubica los puntos ficticios y el punto real en el mapa mediante íconos.

- La aplicación muestra en pantalla los resultados de la simulación.

-La aplicación guarda los resultados en un archivo compatible con GNUplot.

- La aplicación compara los resultados de las diferentes técnicas para ver cuál es la más eficiente en un punto determinado.

6.3 Requisitos mínimos del software

En esta sección se indica los requisitos mínimos de software que debe cumplir para correr la aplicación:

6.3.1 Requisitos de sistema operativo

La aplicación se ha construido en el lenguaje de programación Java, pero además necesita conexión con una base de datos, por lo que tomando en cuenta lo antes mencionado, la aplicación se puede ejecutar en los siguientes sistemas operativos:

- Linux (todas las distribuciones recientes)
- Windows (vista y posteriores)
- Solaris 10 y posteriores
- MacOS X 10.8.3 y posteriores.

6.3.2 Requisitos de software:

Como se ha mencionado anteriormente, la aplicación está escrita en java y utiliza una base de datos, por lo que se necesita software adicional al sistema operativo en donde se instala, para hacerlo funcionar. Estos programas son:

- JRE 8: Máquina virtual necesaria para correr código java en cualquiera de los sistemas operativos mencionados anteriormente.

- PostgreSQL 9.6: Base de datos en donde se sube la información sobre los mapas que el programa renderiza.

6.3.3 Requisitos de Hardware

Para la aplicación se necesita un computador que sea compatible con cualquiera de las versiones de sistema operativo listadas en el punto 5.3.1, ya que la aplicación y los programas a instalar para correr esta son lo suficientemente livianos como para operar en cualquier computador de los últimos años.

6.4 Descripción Global del Producto

6.4.1 Interfaz de Hardware

Al ser un simulador, la aplicación solamente utiliza las interfaces de hardware normales, lo que viene siendo teclado, mouse y pantalla.

6.4.2 Interfaz de Software

La aplicación utiliza JRE y PostgreSQL, el primero para que el computador pueda interpretar y correr el lenguaje java, mientras que el segundo es la base de datos elegida que permite entregar la información de las coordenadas a la aplicación.

6.4.3 Interfaces de Comunicación

La aplicación, al ser un simulador, no se comunica con ningún otro dispositivo.

6.5 Requerimientos específicos

6.5.1 Requerimientos funcionales del sistema

ID	Nombre	Descripción
1	Elegir mapa	La aplicación permite elegir al usuario el mapa precargado a simular.
2	Elegir técnica a usar	La aplicación permite al usuario elegir la técnica a simular.
3	Colocar parámetros	La aplicación permite al usuario ingresar los parámetros que se usarán para hacer la simulación.
4	Ejecutar simulación	La aplicación corre la simulación con los parámetros seleccionados.
5	Entregar datos	La aplicación entrega los datos de la simulación en pantalla.
6	Cancelar simulación	La aplicación puede cancelar la simulación para iniciar otra nueva.
7	Simulación finalizada	La aplicación envía un mensaje cuando la simulación está lista.

6.5.2 Interfaces externas de entrada

ID	Nombre del ítem	Detalle de datos del ítem
1	Mapa a seleccionar	El mapa a seleccionar para la simulación
2	Técnica a utilizar	La técnica a seleccionar para realizar la simulación
3	Posición real	La posición en la que el usuario real se encuentra.
4	Cantidad de usuarios falsos	La cantidad de usuarios falsos que usará la aplicación para la simulación.
5	Cantidad de conjuntos	Se ingresa la cantidad de conjuntos a crear para la elección de la mejor entropía.

Capítulo 7: Simulaciones y experimentos

En este capítulo se detallan las simulaciones y resultados hechos en la aplicación:. Para ello simularemos la arquitectura estándar para proveer privacidad de ubicación que se muestra en la Figura 1. Los algoritmos de batch son ejecutados por el servidor LDS de la figura.

7.1 Interfaz del software

El software inicia con una ventana la cual pide al usuario ingresar la ciudad de su preferencia.

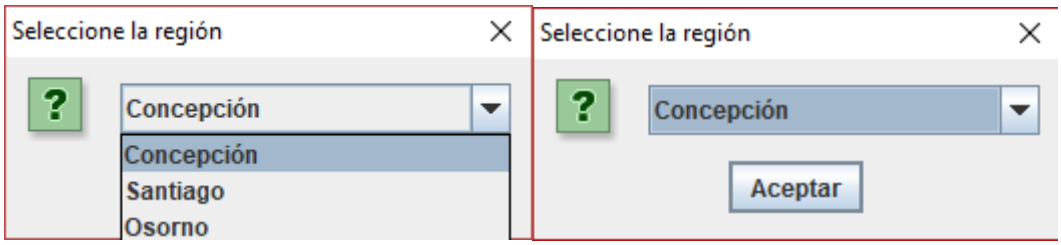


Figura 5: JOptionPane mostrando la selección inicial

Una vez se ha seleccionado el mapa, este aparecerá junto con las opciones principales para generar la simulación.

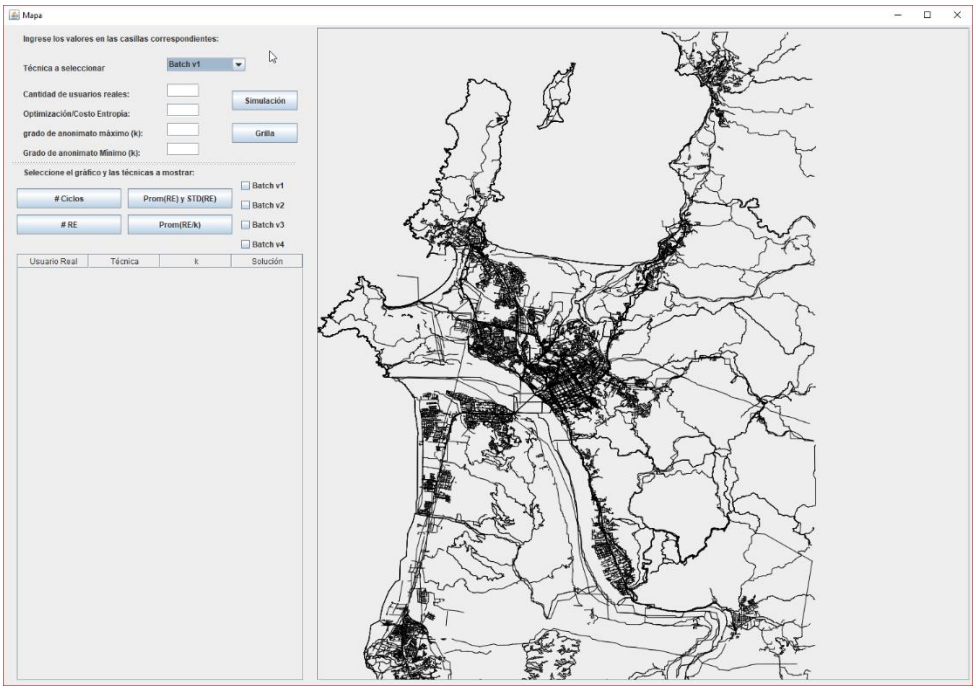


Figura 6: Sección principal del simulador

En la sección superior izquierda nos encontramos con los valores que hay que ingresar y seleccionar, además de la sección post simulación, la cual permite graficar los resultados obtenidos en diferentes simulaciones de la comparación entre técnicas batch, tal cual se observan en la figura 7.

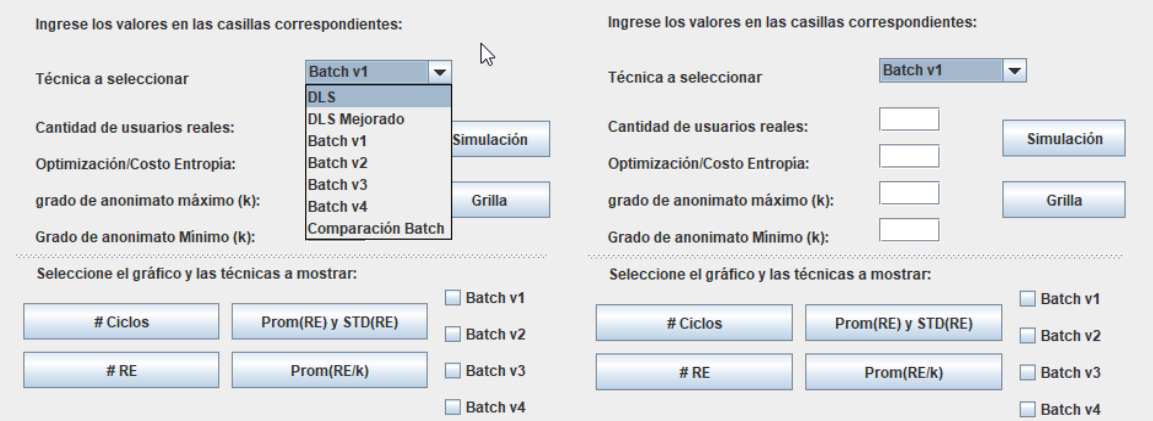


Figura 7: Sección de ingreso de variables

En la sección derecha se puede apreciar el mapa de la ciudad cargado.

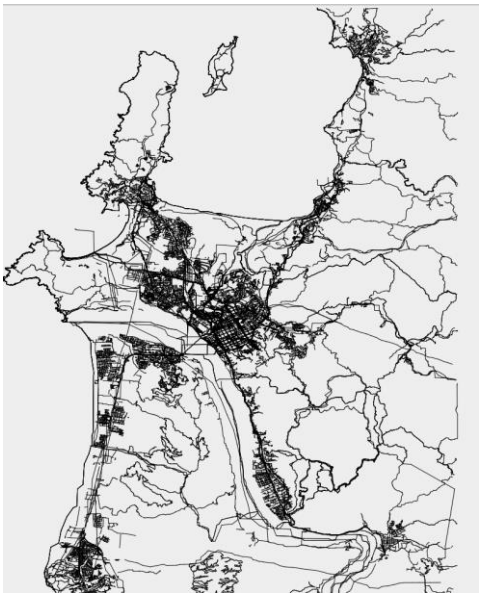


Figura 8: Mapa general de Concepción

7.2 Ejecución de pruebas la simulación

Una vez la simulación se haya ejecutado, aparecerán los resultados, mostrados de dos formas:

- Tabla:

En esta sección se muestran los sectores seleccionados para la región de encubrimiento, al seleccionar uno, se podrá mostrar gráficamente en el mapa, los sectores que componen la región de encubrimiento, así como el punto en donde se encontraba el usuario real en ese momento.

Usuario Real	Té...	k	Solución
Prob: 459.0 Pos: (134.1369945...	B...	4	[Prob: 390.0 sector: (10,4)] | [Pr...
Prob: 390.0 Pos: (705.9928141...	B...	8	[Prob: 360.0 sector: (7,0)] | [Pro...
Prob: 5651.0 Pos: (257.180883...	B...	4	[Prob: 4363.0 sector: (4,6)] | [Pr...
Prob: 13743.0 Pos: (364.40373...	B...	8	[Prob: 5651.0 sector: (3,4)] | [Pr...
Prob: 780.0 Pos: (550.3746969...	B...	9	[Prob: 723.0 sector: (8,13)] | [Pr...
Prob: 459.0 Pos: (194.1930243...	B...	6	[Prob: 369.0 sector: (3,12)] | [Pr...
Prob: 4124.0 Pos: (107.140019...	B...	4	[Prob: 3753.0 sector: (7,5)] | [Pr...
Prob: 2004.0 Pos: (150.795609...	B...	8	[Prob: 2149.0 sector: (3,7)] | [Pr...
Prob: 408.0 Pos: (583.1339549...	B...	7	[Prob: 390.0 sector: (10,4)] | [Pr...
Prob: 728.0 Pos: (475.6589043...	B...	4	[Prob: 723.0 sector: (8,13)] | [Pr...
Prob: 459.0 Pos: (134.1369945...	B...	4	[Prob: 459.0 sector: (2,7)] | [Pro...
Prob: 390.0 Pos: (705.9928141...	B...	8	[Prob: 494.0 sector: (2,11)] | [Pr...
Prob: 5651.0 Pos: (257.180883...	B...	4	[Prob: 12776.0 sector: (5,8)] | [P...
Prob: 13743.0 Pos: (364.40373...	B...	8	[Prob: 6654.0 sector: (4,7)] | [Pr...
Prob: 780.0 Pos: (550.3746969...	B...	9	[Prob: 680.0 sector: (3,3)] | [Pro...
Prob: 459.0 Pos: (194.1930243...	B...	6	[Prob: 494.0 sector: (2,11)] | [Pr...
Prob: 4124.0 Pos: (107.140019...	B...	4	[Prob: 4124.0 sector: (1,7)] | [Pr...
Prob: 2004.0 Pos: (150.795609...	B...	8	[Prob: 2033.0 sector: (7,12)] | [P...
Prob: 408.0 Pos: (583.1339549...	B...	7	[Prob: 494.0 sector: (2,11)] | [Pr...
Prob: 728.0 Pos: (475.6589043...	B...	4	[Prob: 900.0 sector: (2,2)] | [Pro...
Prob: 459.0 Pos: (134.1369945...	B...	4	[Prob: 459.0 sector: (2,7)] | [Pro...
Prob: 390.0 Pos: (705.9928141...	B...	8	[Prob: 352.0 sector: (8,14)] | [Pr...
Prob: 5651.0 Pos: (257.180883...	B...	4	[Prob: 5651.0 sector: (3,4)] | [Pr...
Prob: 13743.0 Pos: (364.40373...	B...	8	[Prob: 3774.0 sector: (2,3)] | [Pr...
Prob: 780.0 Pos: (550.3746969...	B...	9	[Prob: 795.0 sector: (5,14)] | [Pr...
Prob: 459.0 Pos: (194.1930243...	B...	6	[Prob: 464.0 sector: (2,12)] | [Pr...
Prob: 4124.0 Pos: (107.140019...	B...	4	[Prob: 5651.0 sector: (3,4)] | [Pr...
Prob: 2004.0 Pos: (150.795609...	B...	8	[Prob: 2004.0 sector: (2,10)] | [P...
Prob: 408.0 Pos: (583.1339549...	B...	7	[Prob: 464.0 sector: (9,10)] | [Pr...
Prob: 728.0 Pos: (475.6589043...	B...	4	[Prob: 771.0 sector: (3,11)] | [Pr...
Prob: 459.0 Pos: (134.1369945...	B...	4	[Prob: 352.0 sector: (8,14)] | [Pr...
Prob: 390.0 Pos: (705.9928141...	B...	8	[Prob: 352.0 sector: (8,14)] | [Pr...
Prob: 5651.0 Pos: (257.180883...	B...	4	[Prob: 5651.0 sector: (3,4)] | [Pr...
Prob: 13743.0 Pos: (364.40373...	B...	8	[Prob: 13743.0 sector: (5,7)] | [P...
Prob: 780.0 Pos: (550.3746969...	B...	9	[Prob: 864.0 sector: (5,2)] | [Pro...
Prob: 459.0 Pos: (194.1930243...	B...	6	[Prob: 352.0 sector: (8,14)] | [Pr...
Prob: 4124.0 Pos: (107.140019...	B...	4	[Prob: 13743.0 sector: (5,7)] | [P...
Prob: 2004.0 Pos: (150.795609...	B...	8	[Prob: 2057.0 sector: (2,13)] | [P...
Prob: 408.0 Pos: (583.1339549...	B...	7	[Prob: 464.0 sector: (9,2)] | [Pro...

Figura 9: Tabla con los resultados de los experimentos.

- Gráfico:

En esta sección se muestran gráficamente los sectores seleccionados que conforman la región de encubrimiento para poder distinguir claramente en qué posición del mapa se encuentra el usuario real en ese instante de tiempo y cuáles fueron los sectores finalmente escogidos para encubrir su posición.

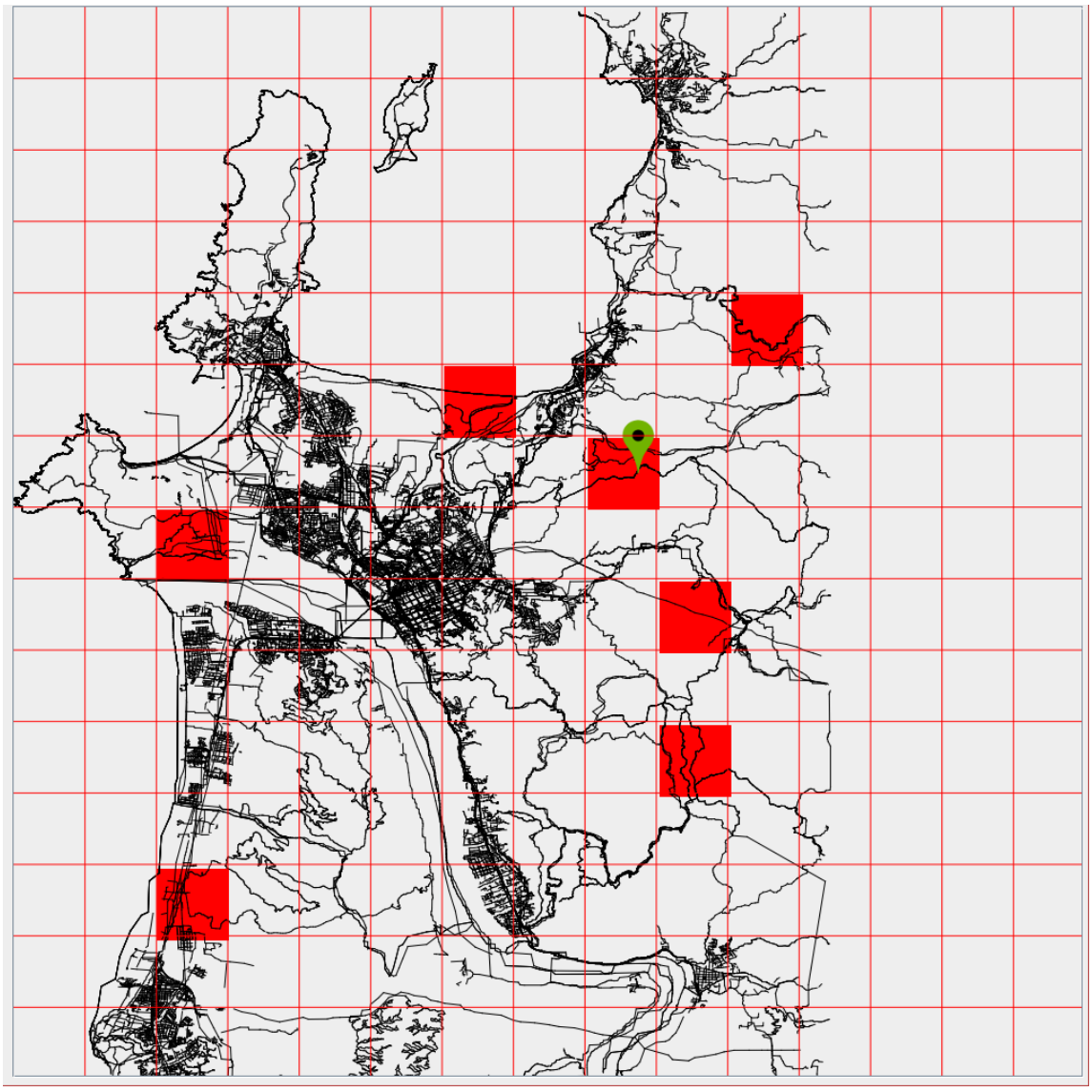


Figura 10: Resultados de los experimentos mostrados de manera gráfica

7.3 Gráficos de experimentos según diversas métricas

En esta sección se muestran algunos de los gráficos producidos por la comparación de los diferentes algoritmos en batch. Estos gráficos comparan, bajo diferentes métricas, las distintas versiones de batch.

Gráfico 1:

En el gráfico mostrado a continuación se puede observar la diferencia de ciclos (ciclos "for" y ciclos "while") que requiere cada una de las 4 técnicas para poder generar las mismas regiones de encubrimiento.

Como se puede apreciar en este gráfico, la versión 1 de la técnica en batch muestra ser la más costosa en cuanto a ejecución de ciclos, esto debido a que debe crear más regiones de encubrimiento al comparar sólo usuarios con el mismo grado de anonimato.

Así mismo podemos apreciar, como lo muestra la imagen, que cuando incluimos la técnica DLS en la comparativa, todas las técnicas demuestran tener una superior administración de recursos del servidor, esto al optimizar la creación de regiones de encubrimiento y su paso por las distintas secciones del código.

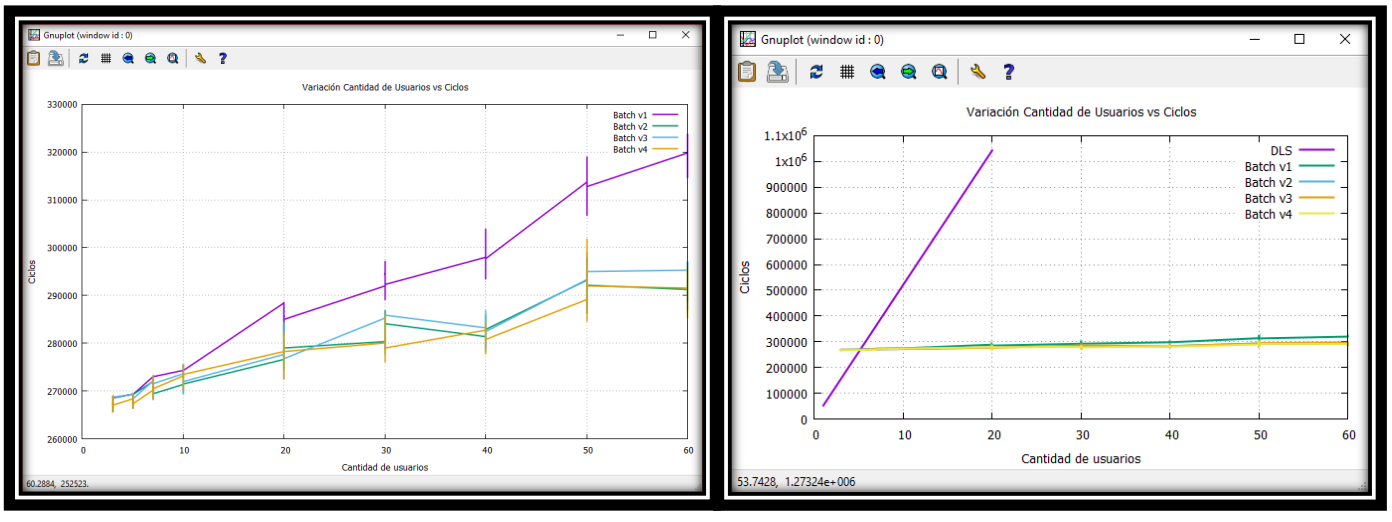


Figura 11: Gráfico Ciclos vs Cantidad de Usuarios

Gráfico 2:

En este gráfico se muestra la diferencia entre la versión 1 y la versión 4 de batch, en la cual se aprecia el tamaño promedio de las regiones de encubrimiento creadas por cantidad de usuarios.

En este gráfico se observa que la versión 4 de batch genera regiones de encubrimiento de mayor tamaño que la versión 1, lo que puede incurrir en una mayor carga para el LBS.

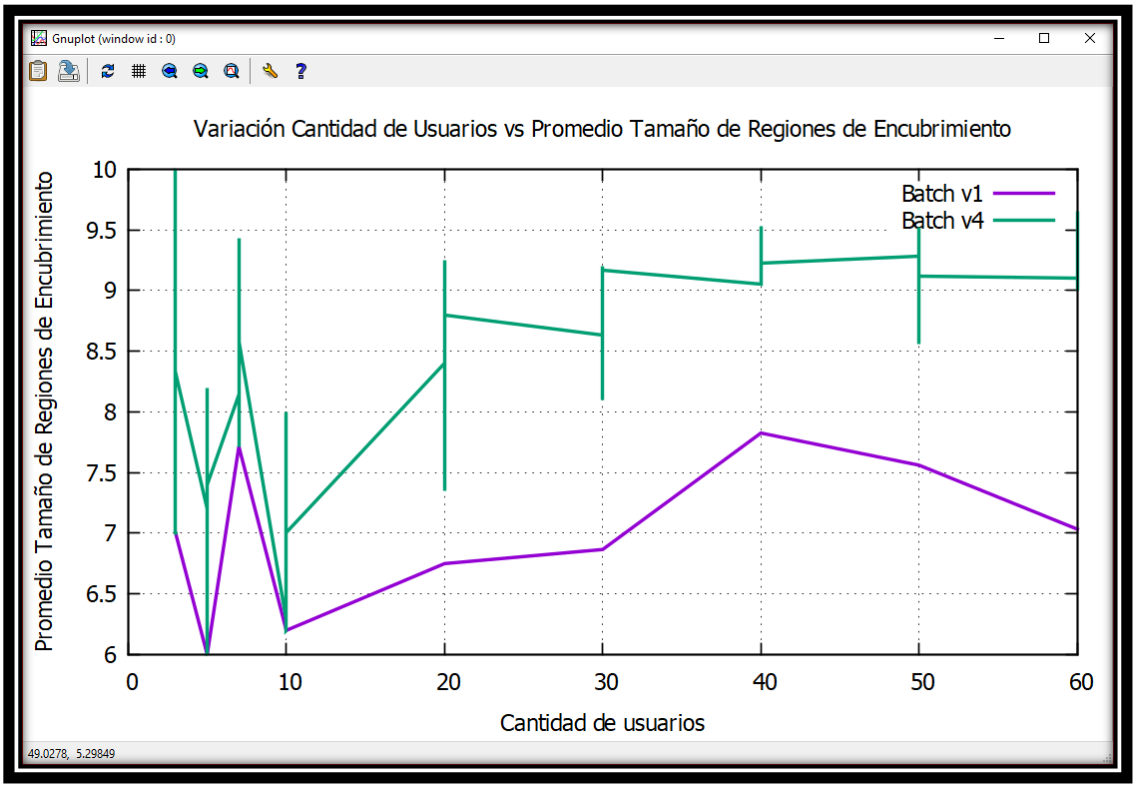


Figura 12: Gráfico Promedio Tamaño de Regiones de encubrimiento vs Cantidad de Usuarios

Gráfico 3

En el gráfico siguiente se aprecia la Cantidad de Regiones de encubrimiento creadas vs la cantidad de usuarios, en donde queda demostrado que la técnica 1 es la más costosa en cuanto a recursos necesarios, esto es nuevamente debido a que sólo toma en cuenta los usuarios con igual grado de anonimato para la optimización de las regiones de encubrimiento. Así mismo se aprecia la diferencia que se va generando entre las técnicas en batch y el algoritmo DLS trabajando de forma independiente, pudiéndose apreciar un significativo aumento de regiones de encubrimiento creadas mientras más usuarios soliciten su red de anonimato.

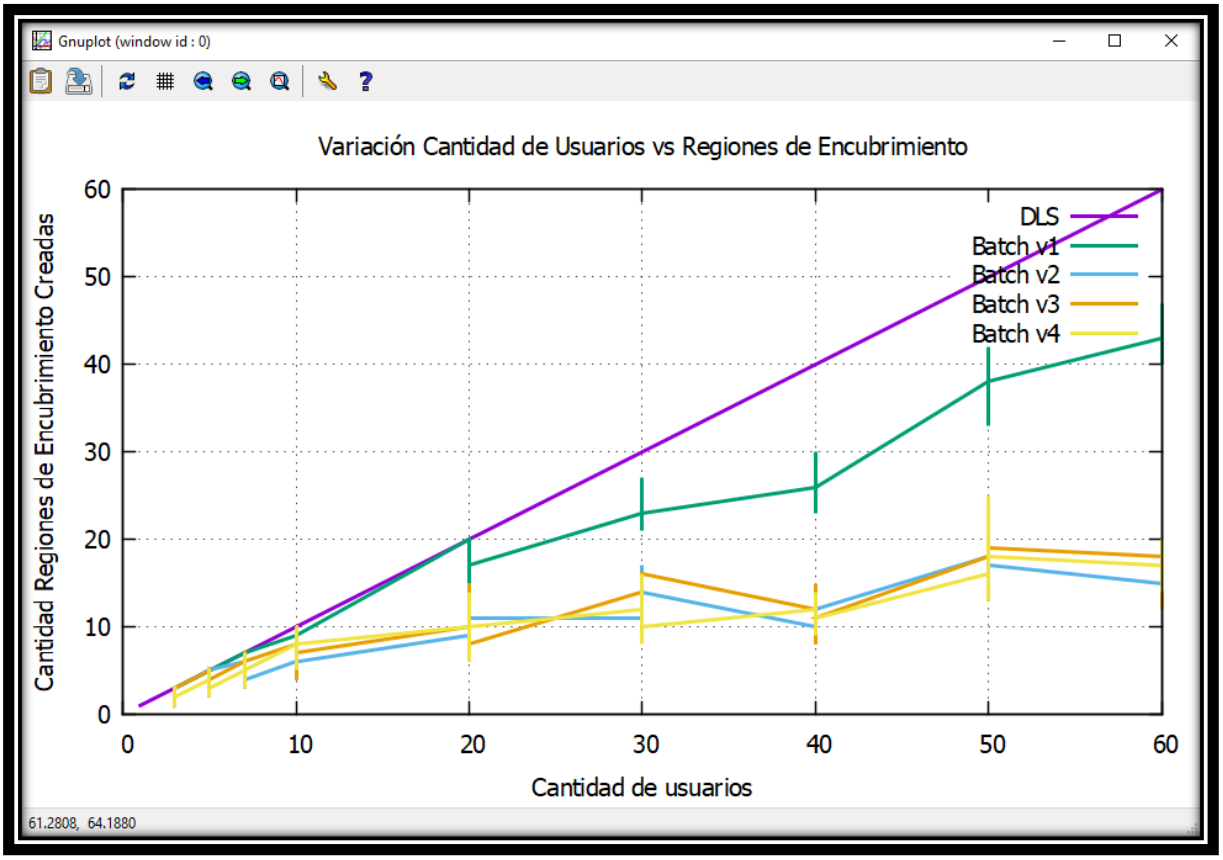


Figura 13: Gráfico Cantidad Regiones de encubrimiento Creadas vs Cantidad de Usuarios

Capítulo 8: Análisis

8.1 Casos de uso

8.1.1 Actores

Usuario:

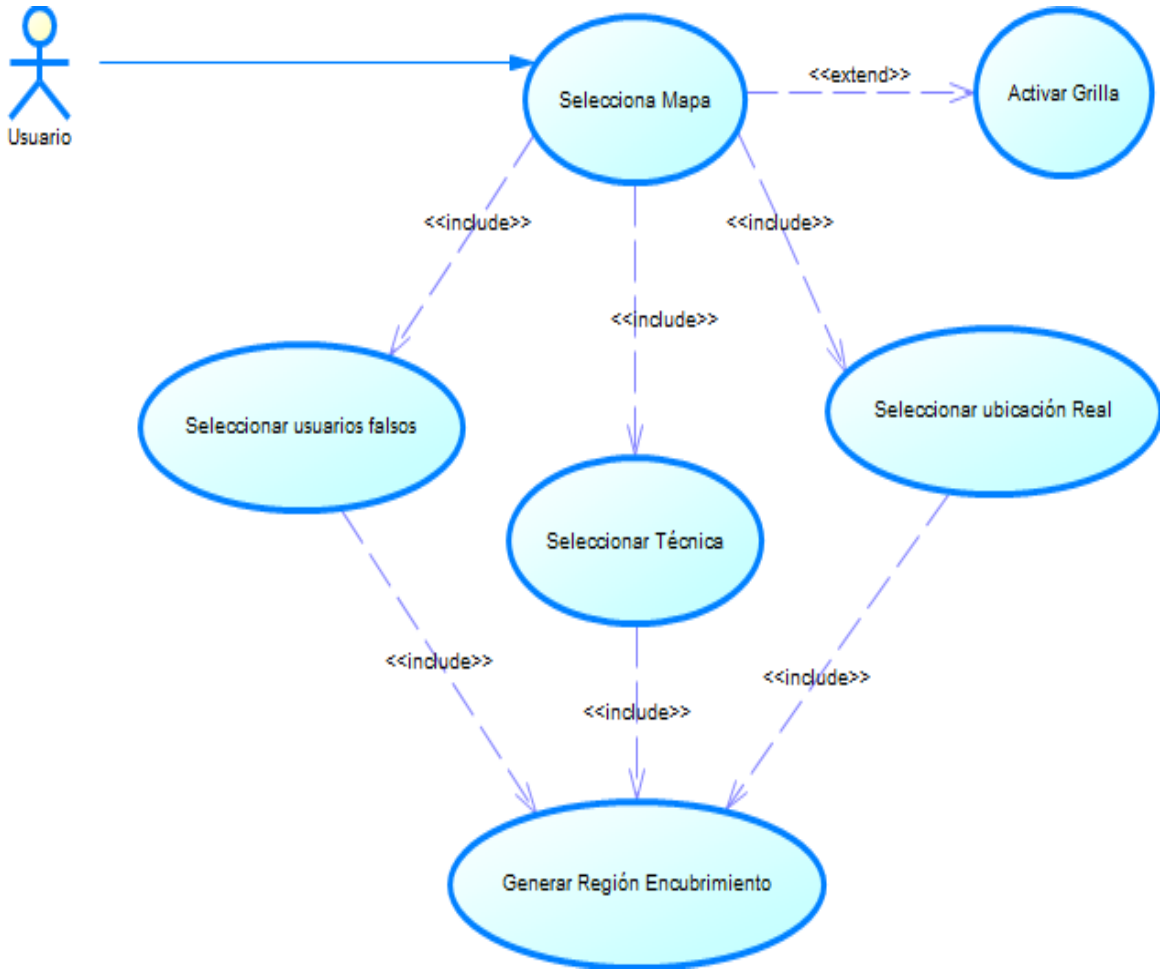
El usuario es la persona que usará el software desarrollado, este tiene facultades para ingresar todos los datos requeridos y no tiene limitaciones en el simulador. Este actor no requiere de preparación previa para utilizar la aplicación, ya que solamente necesita ingresar datos dentro de un rango establecido y seleccionar entre las opciones disponibles.

8.1.2 Casos de uso y descripción

El usuario es el que maneja la totalidad del simulador, este selecciona un mapa entre los disponibles y, una vez generado el mapa, este procede a seleccionar el área del mapa en donde quiere que el usuario real se encuentre, así como la cantidad de usuarios falsos y la técnica a utilizar.

No se necesita ningún sistema de entrada por contraseña, por lo que el usuario puede acceder a todas las funcionalidades desde un principio.

Diagrama de Casos de Uso



8.1.3 Especificación de los casos de uso

8.1.3.1 Caso de Uso: Seleccionar Mapa

Descripción:

Se selecciona un mapa para su generación en el simulador.

Pre condiciones:

El usuario debe elegir dentro de los mapas disponibles.

Flujo de eventos básicos:

Usuario	**Sistema**
1.- El usuario selecciona un mapa entre los disponibles.	2.- Se carga el conjunto de coordenadas desde la base de datos
	3.- Se genera y muestra el mapa en pantalla.

Flujo de eventos alternativos:

Usuario	**Sistema**
	1.- Se informa al usuario que la opción seleccionada no contiene coordenadas para cargar.

Post condiciones:

Se genera un mapa con las coordenadas entregadas y se muestra en pantalla.

8.1.3.2 Caso de uso: Seleccionar usuarios falsos

Descripción:

Se establece la cantidad de usuarios falsos a generar.

Pre condiciones:

El usuario debe ingresar la cantidad de usuarios falsos.

Flujo de eventos básicos:

Usuario	**Sistema**
1.- El usuario ingresa un número de usuarios falsos a generar.	2.- Se establece el número de usuarios falsos.

Flujo de eventos alternativos:

Usuario	**Sistema**
	1.- Se informa al usuario que el valor de usuarios falsos debe ser mayor que 0.

Post condiciones:

Se establece el número de usuarios falsos que el sistema simulará.

8.1.3.3 Caso de uso: Seleccionar técnica

Descripción:

Se selecciona la técnica de ocultamiento a utilizar para la simulación.

Pre condiciones:

El usuario debe seleccionar la técnica entre las opciones disponibles.

Flujo de eventos básicos:

Usuario	**Sistema**
1.- El usuario selecciona una de las técnicas de ocultamiento entre las que hay disponibles.	2.- Se establece la técnica de ocultamiento a utilizar para la simulación.

Flujo de eventos alternativos:

Usuario	**Sistema**
	1.- Se informa al usuario que no existe una técnica seleccionada.

Post condiciones:

Se establece el número de usuarios falsos que el sistema simulará.

8.1.3.4 Caso de uso: Seleccionar ubicación real

Descripción:

Se establece ubicación en donde se encuentra el usuario real.

Pre condiciones:

El usuario debe ingresar la ubicación en el mapa.

Flujo de eventos básicos:

Usuario	**Sistema**
1.- El usuario ingresa la ubicación real en el mapa.	2.- Se establece la ubicación real del usuario.

Flujo de eventos alternativos:

Usuario	**Sistema**
	1.- Se informa al usuario que no se ha seleccionado ninguna ubicación para el usuario real.

Post condiciones:

Se establece el lugar en donde el usuario real se encuentra.

8.1.3.5 Caso de uso: Generar región encubrimiento

Descripción:

Se genera la región de encubrimiento a partir de la simulación de la técnica seleccionada.

Pre condiciones:

El usuario debe haber establecido el número de usuarios falsos, el lugar donde está la ubicación real y la técnica a utilizar, luego de eso debe apretar el botón “Simulación”.

Flujo de eventos básicos:

Usuario	**Sistema**
1.- El usuario presiona el botón “simulación”.	2.- Se realiza la simulación de la capa de encubrimiento.
	3.- Se muestra en pantalla el resultado de la simulación.

Flujo de eventos alternativos:

Usuario	**Sistema**
	1.- Se informa al usuario que alguno de los valores anteriores no ha sido ingresado correctamente, por lo que la simulación ha fallado.

Post condiciones:

Se muestra en pantalla la región de encubrimiento resultante.

8.1.3.6 Caso de uso: Activar grilla

Descripción:

Se activa la grilla en el mapa, para la correcta visualización de los sectores generados.

Pre condiciones:

El usuario debe haber seleccionado el mapa a utilizar y debe haber presionado el botón “generar grilla”.

Flujo de eventos básicos:

Usuario	**Sistema**
1.- El usuario presiona el botón “generar grilla”.	2.- Se despliega en pantalla la grilla.

Post condiciones:

La grilla se muestra en pantalla para la correcta visualización de las áreas del mapa.

Capítulo 9: Conclusiones

Hoy en día podemos decir que existen múltiples beneficios cuando un usuario entrega su posición, ya que diferentes servicios entregan mejores resultados cal conocer la ubicación del usuario, el poder acceder a información casi al instante sencillamente nos hace la vida más fácil, sin embargo no nos damos cuenta que al liberar nuestra posición también podemos exponer nuestra identidad a terceros, esto porque el usuario, dependiendo de dónde hace la consulta, puede mostrar lugares de importancia para su vida diaria, como su hogar, trabajo o lugares frecuentados. Para poder resolver este dilema y romper la relación existente entre la identidad del usuario y la posición es que se han propuesto diversas técnicas de encubrimiento. En este trabajo se implementan una serie de algoritmos que en modo batch y de forma eficiente intentan construir un gran número de regiones de encubrimiento para un gran número de usuarios a la vez.

En este trabajo se llevaron a cabo los siguientes objetivos para poder cumplir con lo mencionado anteriormente:

- Se realizó estudio de técnicas de construcción de regiones de encubrimiento escogidas.
- Se desarrolló un ambiente gráfico que carga mapas obtenidos de fuentes de datos geográficos reales.
- Se desarrolló una aplicación que permite gráficamente simular las técnicas propuestas.
- Se desarrolló aplicación que compara el rendimiento de las técnicas bajo diversos parámetros.

Como ítems deseables, sería interesante abordar otras métricas para ver la efectividad de estos algoritmos, como por ejemplo medir la entropía o la efectividad de los algoritmos a través del tiempo. Además sería deseable incluir otros algoritmos de batch que sean enfocados en la seguridad física del usuario.

Uno de los puntos a destacar de este simulador es que permite entregar datos más cercanos a la realidad, esto gracias a que puede generar caminos y rutas reales basándose en los mapas y coordenadas de las ciudades escogidas.

Finalmente se ha podido concluir que las técnicas de encubrimiento estudiadas en este informe son un gran paso para recuperar la privacidad y la seguridad que se han perdido con los avances tecnológicos.

Capítulo 10: Bibliografía

[1] Ben Niu, Qinghua Li, Xiaoyan Zhu, Guohong Cao, Hui Li. (27 Abril 2014). Achieving k-anonymity in Privacy-Aware Location-Based Services. Proceedings IEEE INFOCOM 2014, 754-762

[2] M. Gruteser and D. Grunwald. (2003). Anonymous Usage of Location-based Services through Spatial and Temporal Cloaking. In ACM MobiSys'03, 31–42.

[3] T. Xu and Y. Cai. (Noviembre 2009). Feeling-based Location Privacy Protection for Location-based Services. ACM Conference on Computer and Communications Security (CCS'09), 348–357.

[4] "K-Anonymity". Fuente: https://es.wikipedia.org/wiki/K-anonimato

[5] "Geolocalización". Fuente: https://es.wikipedia.org/wiki/Geolocalizaci%C3%B3n

[6] "OpenStreetMap". Fuente: https://es.wikipedia.org/wiki/OpenStreetMap

[7] "PostgreSQL". Fuente: https://es.wikipedia.org/wiki/PostgreSQL

[8] "Ortofotografía". Fuente: https://es.wikipedia.org/wiki/Ortofotograf%C3%ADa

[9] "Shapefile". Fuente: https://es.wikipedia.org/wiki/Shapefile

[10] "Entropía". Fuente: http://definicion.de/entropia/

[11] "raw". Fuente: https://es.wikipedia.org/wiki/Raw_(formato)

[12] "Adversario". Fuente: http://www.wordreference.com/definicion/adversario

[12] "IDE". Fuente: http://tecnologia.glosario.net/terminos-tecnicos-internet/ide-860.html

[13] "JDK". Fuente: http://www.esi.unav.es/Asignaturas/Informat2/Clases/Clases9899/Clase01/JavaEntorno/tsld003.htm

[14] "JRE". Fuente: http://www.internetglosario.com/1099/JRE.html

[15] “LBS”. Fuente: https://es.wikipedia.org/wiki/Servicio_basado_en_localizaci%C3%B3n

[16] migue85.wordpress.com, “Conectando PostgreSQL con NETBEANS JAVA”. Fuente: https://migue85.wordpress.com/2008/07/11/conectando-postgresql-con-netbeans-java/

[17] “GPS”. Fuente: https://es.wikipedia.org/wiki/Sistema_de_posicionamiento_global

[18] stackoverflow , “Drawing close range GPS coordinates on a JPanel”. Fuente: https://stackoverflow.com/questions/16364291/drawing-close-range-gps-coordinates-on-a-jpanel

[19] stackexchange, “How to get coordinates from geometry in PostGIS?”. Fuente: https://gis.stackexchange.com/questions/42970/how-to-get-coordinates-from-geometry-in-postgis

[20] “MOP”. Fuente: https://en.wikipedia.org/wiki/Multi-objective_optimization

[21] learnOSM, “OpenStreetMap Data”. Fuente: http://learnosm.org/en/osm-data/

Anexo

En este anexo se detalla el código que se ha trabajado para el proyecto.

Clase Main:

```
package mapa;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class main extends JFrame{
    JButton Grilla;
    JButton T1;
    JButton T2;
    JButton T3;
    JButton T4;
    JLabel Label;
    JLabel Text2;
    JLabel Text3;
    JLabel Text4;
    JLabel Text5;
    JTextField UReal;
    JTextField m;
    JTextField k;
    v VT= new v();
    Tecnicas t = new Tecnicas();
```

```
  public static void main(String[] args){
     String [] Regiones= {"Concepción", "Santiago", "Tokyo"};
     JComboBox R=new JComboBox(Regiones);
     JOptionPane.showMessageDialog(null,R, "Seleccione la región",JOptionPane.QUESTION_MESSAGE);
     if(R.getSelectedItem()=="Concepción"){
       main m = new main("Concepción");
     }else{
        if(R.getSelectedItem()=="Santiago"){
           main m = new main("Santiago");
        }else{
           main m = new main("Tokyo");
        }
     }
  }
public main(String a){
     lienzo l=new lienzo();
     JFrame frame = new JFrame();
     JScrollPane scroll = new JScrollPane(l);
     Grilla = new JButton("Activar Grilla");
     T1 = new JButton("Simulación");
     T2 = new JButton("Técnica 2");
     T3 = new JButton("Técnica 3");
     T4 = new JButton("Técnica 4");
     Label = new JLabel("Ingrese los valores en las casillas correspondientes:");
     String [] nombresT= {"DLS", "DLS Mejorado", "Batch"};
```

```
        JComboBox tecnica=new JComboBox(nombresT);
        Text2 = new JLabel("Ubicación real: ");
        Text3 = new JLabel("Número de sets m: ");
        Text4 = new JLabel("Cantidad de usuarios falsos: ");
        Text5 = new JLabel("Técnica a seleccionar");
        UReal = new JTextField();
        m = new JTextField();
        k = new JTextField();
scroll.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_
NEVER);

scroll.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_NEVER
);
        scroll.getVerticalScrollBar().setUnitIncrement(16);
        scroll.getHorizontalScrollBar().setUnitIncrement(16);
        scroll.setBounds(400, 4, 1000, 1000);
        Grilla.setBounds(100, 700, 150, 30);
        T1.setBounds(125,300,100,30);
        T2.setBounds(200,300,100,30);
        T3.setBounds(50,400,100,30);
        T4.setBounds(200,400,100,30);

        Text2.setBounds(30, 90, 250, 30);
        Text3.setBounds(30, 120, 250, 30);
        Text4.setBounds(30, 150, 250, 30);
        Text5.setBounds(30, 180, 250, 30);
        UReal.setBounds(250, 90, 90, 20);
```

```
m.setBounds(250, 120, 90, 20);
k.setBounds(250, 150, 90, 20);
tecnica.setBounds(250, 180, 90, 20);
T1.addActionListener(VT);
Label.setBounds(30, 30, 350, 30);
//scroll.setPreferredSize(new Dimension(5000,5000));
JPanel Contenedor = new JPanel(null);
//Contenedor.setPreferredSize(new Dimension(500,500));
Contenedor.add(scroll);
Contenedor.add(tecnica);
Contenedor.add(Grilla);
Contenedor.add(Text5);
Contenedor.add(T1);
Contenedor.add(Label);
Contenedor.add(Text2);
Contenedor.add(Text3);
Contenedor.add(Text4);
Contenedor.add(UReal);
Contenedor.add(m);
Contenedor.add(k);
frame.setContentPane(Contenedor);
frame.pack();
frame.setTitle("Mapa");
frame.setSize(1500, 1050);
frame.setVisible(true);
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
}
class v implements ActionListener{
@Override
public void actionPerformed(ActionEvent e) {
      int valor1=Integer.parseInt(UReal.getText());
         int valor2=Integer.parseInt(m.getText());
         int valor3=Integer.parseInt(k.getText());
   if (e.getSource()==T1) {
         t.tecnica1(valor1,valor2,valor3);
      }
      if (e.getSource()==T2) {
         t.tecnica2(valor1,valor2,valor3);
      }
      if (e.getSource()==T3) {
         t.tecnica3(valor1,valor2,valor3);
      }
      if (e.getSource()==T4) {
         t.tecnica4(valor1,valor2,valor3);
      }
   }
}
}
```

Clase Mapa:

```
package mapa;
import java.sql.*;
public class Mapa {
   Connection connect =null;
   Connection miconexion;
   ResultSet rsDatos;
   Statement stSentencias;
   PreparedStatement psPrepararSentencias;
   ResultSetMetaData rsm;
   public Mapa() throws SQLException, ClassNotFoundException
   {
   try{
      Class.forName("org.postgresql.Driver");
      String url="jdbc:postgresql://localhost:5432/osm";
      miconexion=DriverManager.getConnection(url,"postgres","cazador1");
      if(miconexion!=null){
         System.out.println("Conexión exitosa");
      }

   }catch(ClassCastException | SQLException ex){
      throw ex;
   }
   }
```

```
   public ResultSet consulta(String sql) throws SQLException{
      try{
         rsDatos=stSentencias.executeQuery(sql);
      }catch(SQLException ex){
         throw ex;
      }
      return rsDatos;
   }
}
```

Clase Técnicas:

```
package mapa;
import java.util.*;
public class Tecnicas {
   String mensaje;
   lienzo L = new lienzo();

   public int[][] calculo(){
      //creando variables e inicializándolas si corresponde
      ArrayList<String[]> cOriginales;
      ArrayList<String> coordString;
      ArrayList<Double>  cooX, cooY, cooXFinal = new ArrayList<>(), cooYFinal = new ArrayList<>();
      int [][]Matriz = new int[15][15];
      int x, y;
```

```
//rellenando la matriz con 0
for (int i = 0; i < 15; i++) {
   for (int j = 0; j <15; j++) {
      Matriz[i][j]=0;

   }
}
//obteniendo datos desde clase lienzo
cOriginales=L.cargarDatos();
for (int i = 0; i < cOriginales.size(); i++) {
   coordString=L.transformarDatos(i, cOriginales);
   cooX=L.partirDatos(coordString, 0);
   cooY=L.partirDatos(coordString, 1);
   cooXFinal.addAll(L.cambiarCoordenadas(cooX, 37.0189803960268,
1005.0));
   cooYFinal.addAll(L.cambiarCoordenadas(cooY, 73.2194969804201,
0.0));
}
System.out.println("La cantidad de coordenadas finales es:
"+cooXFinal.size()+"------------------------------------------------------------");

for (int i = 0; i < cooXFinal.size(); i++) {

   x = (int) Math.round(cooXFinal.get(i)/66.66666);
   y = (int) Math.round(cooYFinal.get(i)/66.66666);
   if(x<15&&y<15){
```

```
            Matriz[x][y]++;
        }else{
            if(x>=15&&y<15){
                x--;
                Matriz[x][y]=Matriz[x][y]+1;
            }else{
                if(x<15&&y>=15){
                    y--;
                Matriz[x][y]++;
                }else{
                    x--;
                    y--;
                Matriz[x][y]++;
                }
            }
        }


    }
    int sumamatriz=0;
    for (int i = 0; i < 15; i++) {
        for (int j = 0; j < 15; j++) {
            sumamatriz=sumamatriz+Matriz[i][j];
        }
    }
    System.out.println("la cantidad de elementos en la matriz es: "+sumamatriz);
    return Matriz;
```

```
}
public void comprobacionMatriz(int[] valor){

   int fff=0;
   System.out.print("\t");
   for (int i = 0; i < 15; i++) {
      if(i!=14){
         System.out.print(i+"\t");
      }else{
         System.out.println(i);
         System.out.println("");
      }
   }
   int bandera=1;
   for (int i = 0; i < valor.length; i++) {
      if(i==0){
         System.out.println("");
         System.out.print(fff+"\t"+valor[i]+"\t");
         fff++;
      }else{
      if(bandera%15==0){
         System.out.println(valor[i]);
         System.out.println("");
         System.out.println("");
         if(fff!=15){
          System.out.print(fff+"\t");
```

```
            }
            fff++;

        }else{
            System.out.print(valor[i]+"\t");
        }
    }
        bandera++;
    }
}
```

```
public double[][] construirCj(int valor[], double Cj[][], int posF, int k, int
UReal){

    int aux=10000, ai=0,flag, l=0;
    int [] escogidos=new int[(2*k)];
    for (int i = 0; i < valor.length; i++) {
        if(aux>Math.abs(UReal-valor[i])){
            aux=Math.abs(UReal-valor[i]);
            ai=i;
        }
    }
    if(ai+k<valor.length){
        if(ai-k>=0){
            flag=0;
            for (int i = ai-k; i < ai+k+flag; i++) {
```

```
        if(i==ai-k){
          for(int j = ai-k; j < ai+k; j++){
            if(valor[j]==UReal){
              flag++;
            }
          }
        }
        if(valor[i]!=UReal){
          escogidos[l]=valor[i];
          l++;
        }
      }
    }
    else{
      flag=0;
      for (int i = 0; i < ai+k+Math.abs(ai-k)+flag; i++) {
        if(i==0){
          for(int j = 0; j < ai+k+Math.abs(ai-k); j++){
            if(valor[j]==UReal){
              flag++;
            }
          }
        }
        if(valor[i]!=UReal){
          escogidos[l]=valor[i];
          l++;
```

```
        }
        }
      }
}else{
   flag=0;
   for (int i = valor.length-(2*k); i < valor.length; i++) {
      if(valor[i]==UReal){
            flag++;
            }
   }
   for (int i = valor.length-(2*k)-flag; i < valor.length; i++) {
      if(valor[i]!=UReal){
            escogidos[l]=valor[i];
            l++;
        }
   }
}

//creando conjunto de números aleatorios
int numero, f=0, compr=0;
int[] aleatorios=new int[k-1], orden=new int[k];
do{
   numero = (int) (Math.random()*2*k);
   if(f==0){
      aleatorios[f]=numero;
      f++;
```

```
   }else{
      for (int i = 0; i < aleatorios.length; i++) {
         if(aleatorios[i]==numero){
            compr=1;
         }
      }
      if(compr==0){
         aleatorios[f]=numero;
         f++;
      }
      compr=0;
   }

}while(f<k-1);
for (int i = 0; i < k-1; i++) {
      Cj[posF][i]=escogidos[aleatorios[i]];
   }
   Cj[posF][k-1]=UReal;
   double aux2;
   for (int i = 0; i < k; i++) {
      for (int j = 0; j < k; j++) {
      numero= (int) (Math.random()*(k-1));
      aux2=Cj[posF][j];
      Cj[posF][j]=Cj[posF][numero];
      Cj[posF][numero]=aux2;
      }
```

```
        }

    return Cj;
  }

  public double[][] convertirCj(double [][] Cj, int consultasTotales, int m, int
k){
    double [][] CjC=new double[m][k];
    for (int i = 0; i < m; i++) {
      for (int j = 0; j < k; j++) {
        CjC[i][j]=Cj[i][j]/consultasTotales;
      }
    }
    return CjC;
  }

  public double [][] creandoPj(double [][] Cj, int m, int k){
    double [][] Pj=new double[m][k];
    double total;
    for (int i = 0; i < m; i++) {
      total=0;
      for (int j = 0; j < k; j++) {
        total=total+Cj[i][j];
      }
      System.out.println("El total de la fila "+i+" es: "+total);
      for (int j = 0; j < k; j++) {
        Pj[i][j]=Cj[i][j]/total;
```

```
      }
    }

    return Pj;
}

public double [] entropia(double [] H, double [][] Pj, int m, int k){
    double suma;
    for (int i = 0; i < m; i++) {
       suma=0;
         for (int j = 0; j < k; j++) {
          suma=suma+(Pj[i][j]*(Math.log(Pj[i][j])/Math.log(2)));
       }
         H[i]=-suma;
    }
    return H;
}
public void tecnica1 (int UReal, int m, int k){
    //primero se cargan los datos del mapa en una matriz
    int[][] Matriz=calculo();
    int [] posX=new int[225];
    int [] posY=new int[225];
    int [] valor=new int[225];
    double [][] Cj=new double[m][k], Pj, CjC;
    double []H = new double[m];
    double C=-1000;
```

```
int l=0, aux,aY, aX, consultasTotales=0, indiceH=0;
for (int i = 0; i < 15; i++) {
   for (int j = 0; j < 15; j++) {
      posX[l]=i;
      posY[l]=j;
      valor[l]=Matriz[i][j];
      consultasTotales=consultasTotales+Matriz[i][j];
        l++;
   }
}
//función para comprobar el valor de la matriz 15x15
comprobacionMatriz(valor);

//ordenando la lista:

for (int i = 0; i < valor.length; i++) {
   for (int j = 0; j < valor.length; j++) {
   aux=valor[i];
   aX=posX[i];
   aY=posY[i];
   if (i<valor.length) {
      if(aux<valor[j]){
      valor[i]=valor[j];
      valor[j]=aux;
      posX[i]=posX[j];
      posX[j]=aX;
```

```
        posY[i]=posY[j];
        posY[j]=aY;
        }
    }
    }
}
for (int i = 0; i < m; i++) {
    Cj= construirCj(valor, Cj, i, k, UReal);
}
System.out.println("los valores de Cj son:");
for (int i = 0; i < m; i++) {
    for (int j = 0; j < k; j++) {
        System.out.print(Cj[i][j]+"\t");
    }
    System.out.println("\n");
}
CjC=convertirCj(Cj, consultasTotales, m, k);
System.out.println("los valores de Cj convertidos son:");
for (int i = 0; i < m; i++) {
    for (int j = 0; j < k; j++) {
        System.out.print(Cj[i][j]+"\t");
    }
    System.out.println("\n");
}
Pj=creandoPj(CjC,m,k);
```

```
        System.out.println("los valores de Pj son:");
        for (int i = 0; i < m; i++) {
            for (int j = 0; j < k; j++) {
                System.out.print(Pj[i][j]+"\t");
            }
            System.out.println("\n");
        }
        H=entropia(H, Pj, m, k);
        for (int i = 0; i < H.length; i++) {
            if (C<H[i]) {
                C=H[i];
                indiceH=i;
            }
        }
        for (int i = 0; i < k; i++) {
            System.out.println("valores de Cj con la mayor entropía: "+Cj[indiceH][i]);
        }

        System.out.println("Finalmente la mayor entropía es: "+C);
    }
```

Clase lienzo:

```
package mapa;
import java.awt.*;
import java.awt.geom.*;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.regex.Pattern;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.table.DefaultTableModel;
public class lienzo extends JPanel{

  public lienzo(){
    this.setPreferredSize(new Dimension(1000,1000));
  }


  public ArrayList<String[]> cargarDatos(){

    ArrayList<String[]> cOriginales=new ArrayList<>();
    DefaultTableModel dtm;
    ResultSet rsd;
```

```
    try{


      Mapa miconexion=new Mapa();
      String sentenciaInsert="SELECT ST_AsText(st_transform(way, 4326)) FROM planet_osm_line";

miconexion.psPrepararSentencias=miconexion.miconexion.prepareStatement(sentenciaInsert);
      //JOptionPane.showMessageDialog(null,"los datos se cargaron correctamente");
      rsd=miconexion.psPrepararSentencias.executeQuery();
      miconexion.rsm=rsd.getMetaData();
      while(rsd.next()){
        String[] rows=new String[miconexion.rsm.getColumnCount()];
        for (int i = 0; i < rows.length; i++) {
          rows[i]=rsd.getString(i+1);
          cOriginales.add(rows);
        }


      }
    }catch(SQLException | ClassNotFoundException | HeadlessException e){
      System.out.println(e.getCause());
      System.out.println("pasé por acá");
    }
    return cOriginales;
  }
  public ArrayList<String> transformarDatos(int valor, ArrayList<String[]> cOriginales){
```

```
ArrayList<String> datosString =new ArrayList<>();
String auxiliar;
String [] auxiliar2;
//System.out.println("La cantidad de datos en coordenadas es: "+coordenadas.size());
  auxiliar=Arrays.toString(cOriginales.get(valor));
  auxiliar2=auxiliar.split(" ");
  datosString.addAll(Arrays.asList(auxiliar2));


 //System.out.println("La cantidad de datos obtenida es: "+datosString.size());
int x=0;
String a=new String();
String b;
for (int i = 0; i < datosString.size(); i++) {
auxiliar=datosString.get(i);
  if (auxiliar.contains("(")) {
    x++;
    auxiliar2=auxiliar.split(Pattern.quote("("));
    a=auxiliar2[1];
    datosString.set(i, a);
  }
  if(auxiliar.contains(")")){
    auxiliar2=auxiliar.split(Pattern.quote(")"));
    b=auxiliar2[0]+","+a;
```

```
            datosString.set(i, b);
          }
        }
        return datosString;




    }
    //-------------------------------------------------------------------------------------------
--------------------------------//
    public ArrayList<Double> partirDatos(ArrayList<String> datosString, int
valor ){
        String auxiliar;
        String [] auxiliar2;
        ArrayList<String> Local=new ArrayList<>();
        ArrayList<Double> coord=new ArrayList<>();

        for (int i = 0; i < datosString.size(); i++) {
          auxiliar=datosString.get(i);
          if(auxiliar.contains(",")){
            auxiliar2=auxiliar.split(",");
            Local.add(auxiliar2[valor]);
          }
        }
        for (int i = 0; i < Local.size(); i++) {
          coord.add(Double.parseDouble(Local.get(i)));
        }
```

```
        return coord;
    }
    //-------------------------------------------------------------------------------------
-------------------------------//
    public ArrayList<Double> cambiarCoordenadas(ArrayList<Double> coord,
Double v, Double z){


        ArrayList<Double>C=new ArrayList<>();



        //System.out.println("el menor v es:"+v);
        for (int i = 0; i < coord.size(); i++) {
            if (z!=0.0) {
                C.add(z-((coord.get(i)+v)*2293.6938934486));
            }else{
                C.add((coord.get(i)+v)*2293.6938934486);
            }


        }
        return C;
    }
    public void paintComponent(Graphics gg){
        super.paintComponent(gg);
        gg.setColor(Color.black);
        Graphics2D g = (Graphics2D) gg;
    g.setRenderingHint(RenderingHints.KEY_ANTIALIASING,
RenderingHints.VALUE_ANTIALIAS_ON);
```

```
   g.setRenderingHint(RenderingHints.KEY_STROKE_CONTROL,
RenderingHints.VALUE_STROKE_PURE);
   Line2D.Double shape;
   ArrayList<String[]> cOriginales;
   cOriginales=cargarDatos();
   ArrayList<String> coordenadasListas;
   ArrayList<Double> cooX, cooY, cooXFinal, cooYFinal;
      for (int i = 0; i < cOriginales.size(); i++) {
            coordenadasListas=transformarDatos(i, cOriginales);
            cooX=partirDatos(coordenadasListas, 0);
            cooY=partirDatos(coordenadasListas, 1);
            cooXFinal=cambiarCoordenadas(cooX,          37.0189803960268,
1005.0);
            cooYFinal=cambiarCoordenadas(cooY, 73.2194969804201, 0.0);
            for (int j = 0; j < cooXFinal.size()-2; j++) {
         Double x=cooXFinal.get(j);
         Double y=cooYFinal.get(j);
         Double xNext=cooXFinal.get(j+1);
         Double yNext=cooYFinal.get(j+1);
            shape = new Line2D.Double(y,x,yNext,xNext);
         g.draw(shape);
      }
      }
   }
}
```