Implementación de un Modelo de Datos para Transporte Público

Meraioth Ulloa S.
Departamento de Ingeniería Informática y Ciencias de la Computación
Universidad de Concepción

Profesora Guía: Andrea Rodriguez Tastets

20 de agosto de 2018

Resumen

Las bases de datos de objetos en movimiento (MOD) se centran en la forma de modelar y consultar sobre los movimientos de entidades, como personas, vehículos y barcos. Con el desarrollo de las tecnologías de posicionamiento, como el Sistema de Posicionamiento Global (GPS), ha aumentado el estudio de MOD en los últimos años debido a la amplia aplicación que tienen para servicios basados en ubicaciones de forma online, redes sociales basadas en la localización, etc. En esta memoria de título se hace uso de la implementación de un modelo de datos, es decir, tipos de datos y operadores para estos tipos de datos, el cual utiliza el movimiento restringido de los objetos sobre una red para evitar redundancia en la especificación del posicionamiento de los objetos. Usando el modelo se obtienen resultados similares a utilizar un modelo de espacio libre, esto ocurre en consultas (espaciales y temporales) sobre todo el dominio de los datos, no así en consultas que impliquen utilizar un subconjunto del dominio, aquí el modelo de espacio libre toma ventaja. En cuanto al espacio usado por el modelo, es en peor caso similar a un modelo de espacio libre, pudiendo mejorar su desempeño si los datos son obtenidos con una mayor frecuencia, ya que mapea múltiples puntos a una localización relativa en la red, cuya representación geométrica es independiente.

1. Introducción

Desde que E.F.Codd propuso su modelo relacional [1], el uso de las bases de datos ha sido ampliamente utilizado tanto en ambientes profesionales como en academia. Sin embargo, la magnitud de los datos a procesar ya no es la misma que cuando se implementaron las primeras bases de datos a fines de los 80's por IBM. Es por esto que junto al crecimiento de los datos, también nacieron nuevas consultas, las cuales con un modelo relacional tradicional no se pueden responder de manera eficiente. Así, con el transcurso de los años se han tomado nuevos enfoques, creando nuevas extensiones al clásico modelo relacional, con lo cual se obtienen mejoras en rendimiento y capacidad de expresión con respecto a las consultas.

En el mundo real, más objetos en movimiento, como peatones, automóviles y buses, tienden a moverse sobre redes de transporte subyacentes distintas del espacio geográfico libre. Por esta razón, el tema de modelar objetos en movimiento en las redes ha recibido una creciente atención entre los investigadores. Preguntas relevantes en este contexto son, por ejemplo, ¿A qué hora pasó un vehículo con patente X por el área de neblina en San Joaquín, Santiago?, ¿En qué autopista se encuentra algún camión con 20 toneladas de carga?, ¿Qué parte de la red puede ser alcanzada en un radio de 50 km desde una posición dada?.

1.1. Objetivo General

El objetivo de esta memoria es implementar un modelo de datos de objetos en movimientos sujeto a una red, y comparar su rendimiento en tiempos de consulta y uso espacio con respecto a un modelo de datos de objetos en movimiento en espacio libre, usando datos de la red de transporte de Santiago, y datos reales de objetos en movimiento (buses) obtenidos a través de los GPS que tienen incorporados.

1.2. Objetivos Específicos

- Revisar el estado del arte en el desarrollo de modelos de datos para objetos en movimiento, en particular, los que están restringidas a redes, como las de transporte.
- Extracción, limpieza y procesamiento de datos de distintas fuentes.
- Implementar el modelo de datos de objetos en movimiento sujeto a redes con datos de transporte público.
- Implementar el modelo de datos de objeto en movimiento en espacio libre con datos de transporte público.
- Evaluación cualitativa y cuantitativa. Se espera lograr el mismo poder de expresividad entre los modelos y se desea medir los tiempos de respuestas a distintas consultas, como también el espacio que ocupan los objetos representados en los distintos modelos.

2. Discusión Bibliográfica

El área de las bases de datos de objetos en movimiento [9] ha recibido gran interés de investigación en los últimos años desde los primeros trabajos a mediados de los 90. Como es bien sabido las bases de datos permiten al usuario modelar, almacenar, recuperar y consultar; en el caso de esta memoria, los datos de objetos en movimiento que representan la evolución de desplazamiento de un objeto en el tiempo. En algunos casos, solo las ubicaciones dependientes del tiempo son de interés, y hablamos de puntos en movimiento. Algunos ejemplos son usuarios de teléfonos móviles, animales, barcos, aviones, automóviles, naves espaciales, satélites y misiles. En otros casos, también es necesario manejar la forma o la extensión del área dependiente del tiempo, que puede crecer o reducirse, se habla entonces de regiones en movimiento. Los ejemplos son huracanes, lagos, incendios forestales, derrames de petróleo y la propagación de enfermedades. En algunos casos menos habituales, la forma dependiente del tiempo o la extensión lineal, que puede alargarse o acortarse es de interés, y hablamos de líneas en movimiento. Ejemplos de esto son: congestión vehicular, la frontera que describe un glaciar, y los límites de las regiones en movimiento en general.

Mucho del interés en bases de datos de objetos en movimiento ha sido estimulado por las tendencias actuales en electrónica de consumo, comunicaciones inalámbricas, tecnologías de posicionamiento y servicios basados en la ubicación. En esta línea es necesario señalar las ramas de investigación que han surgido en torno a esta área, se quiere hacer la primera distinción entre bases de datos de objetos en movimientos histórico, y bases de datos de objetos en movimientos predictivos. Las primeras describen la evolución temporal de los objetos espaciales en el pasado y se aprovechan para el análisis espacio-temporal, en cambio las segundas describen la evolución temporal predicha de objetos espaciales en el presente y en el futuro cercano.

2.1. Bases de datos para objetos en movimiento predictivos

El modelo de datos MOST (Moving Objects Spatio-Temporal Model) [7] es uno de los pocos modelos capaz de describir el movimiento actual y futuro cercano o esperado en el contexto de una base de datos. Todos los modelos

específicos de las aplicaciones conocidas son independientes del contexto de una base de datos, por ejemplo, la información no es persistente. El modelo se basa en la observación de que no se deben mantener las posiciones directamente en la base de datos, lo que lleva a un gran volumen de actualizaciones, por lo cual utiliza una representación mediante un vector de movimiento. Solo cuando la posición del objeto predicho por el vector de movimiento se desvía de la posición real en más de un umbral, se debe transmitir una actualización a la base de datos.

La idea fundamental es introducir los llamados atributos dinámicos que cambian sus valores automáticamente con el tiempo. No todos los tipos de atributos son elegibles para ser dinámicos, el tipo de atributo debe tener un valor 0 y una operación de suma. La dinámica viene dada por funciones lineales que describen vectores de movimiento y evitan frecuentes actualizaciones de bases de datos. Ejemplos de esto son los tipos: entero y real dinámico. Lamentablemente, no existe un concepto de tipos de datos espaciales dinámicos, de modo que la única opción para representar un moving point es modelarlo como un par (x: real dinámico, y: real dinámico). Las líneas o regiones dinámicas no se pueden modelar, y no hay ningún concepto de tipos de datos espacio temporales disponibles. Si una consulta se refiere a un atributo dinámico A, su valor dinámico se entiende y se utiliza en la evaluación. Por lo tanto, el resultado depende del momento en que se emite la consulta. Si tal consulta se reevalúa en cada tic del reloj, esta consulta se llama continua. Para este modelo se desarrolló también un lenguaje de consulta tipo SQL, llamado Future Temporal Logic (FTL), incluyendo operadores como until y nexttime.

2.2. Bases de datos para objetos en movimiento históricos

Dentro de esta rama de bases de datos de objetos en movimiento, existen dos campos de investigación.

2.2.1. Objetos en movimiento en espacio libre

Las primeras aproximaciones a bases de datos de objetos en movimiento fueron para objetos espaciales que se mueven libremente en el espacio que

están contenidos, que usualmente es el espacio 2D. Estas surgen con el trabajo de Erwig *et al* [8], donde se presentan definiciones de tipos de datos, por ejemplo, un moving point es la abstracción básica de un objeto físico que se mueve en el plano o un espacio de dimensión superior, para el cual solo la posición es relevante, no así su forma. De igual forma moving region describe una entidad en el plano que cambia su posición, así como su extensión y forma, es decir, una región en movimiento puede no solo moverse, sino también crecer y contraerse. Finalmente, otro tipo de dato interesante a analizar es moving lines, que describen entidades que cambian su largo y su ubicación, esta puede alargarse o acortarse.
Todos estos tipos de datos son un mapeo desde tiempo a espacio. Cuando hablamos de tiempo encontramos los siguientes tipos de datos: instant (similar a timestamp, un punto en una línea de tiempo), period (rango en una línea de tiempo), periods (rangos disjuntos en una línea de tiempo). Por ejemplo tenemos el siguiente mapeo:

$$Moving(\alpha) : time \rightarrow \alpha \tag{1}$$

Esto quiere decir que cuando se hace un mapeo de un moving point desde la dimensión tiempo, obtendremos como resultado un point, geometría conocida e implementada en los gestores de bases de datos actuales. Vale destacar que el elemento α (expresión 1) no solo puede ser un tipo de dato espacial, sino también, entero, booleano, flotante, etc.

En el trabajo presentado en [4] se profundiza más sobre los tipos de datos espacio-temporales, definiendo un conjunto de operadores sobre estos tipos de datos en un ambiente de base de datos. Estos operadores se categorizan en: operadores espaciales, operadores temporales y operadores espacio-temporales. Algunos de ellos son:

- $intersection : mpoint \times mregion \rightarrow mpoint$
- $distance : mpoint \times mpoint \rightarrow mreal$
- $trajectory : mpoint \rightarrow line$

2.2.2. Objetos en movimiento restringido a redes

Muchos objetos espaciales como automóviles, aviones, trenes y personas en los edificios están restringidos en sus movimientos posibles ya que se mueven

en redes espacialmente integradas como carreteras, autopistas, ferrocarriles, líneas de fábricas para productos, edificios o rutas de aviones. Se desprende entonces que las redes espaciales deben tenerse en cuenta en un modelo de datos, y en un lenguaje de consulta de bases de datos para objetos en movimiento. Esto hace posible describir el movimiento relativo a una red en lugar del espacio libre (2D).

Esta temática se aborda en el trabajo realizado por Gütting [5], el cual crea una base de datos espacio-temporal, tratando de abordar el manejo de objetos embebidos en redes de transporte que describen trayectorias, para facilitar las consultas y el rendimiento. Este modelo es una extensión al modelo relacional clásico, y es uno de los pocos artículos que trata de abordar esta perspectiva de objetos en movimientos embebidos en redes, y donde existe una implementación disponible sobre un gestor de bases de datos llamado SECONDO [3].

En [5] se considera una unidad básica llamada route (expresión 2), con ***id*** un identificador entero, ***l*** un real con la longitud de la curva, ***c*** es la línea que describe la curva en el espacio, dos enteros, ***kind*** para almacenar si route es simple o dual (dual quiere decir que tiene dos sentidos) y ***start*** identificando si el punto de finalización de la curva ***c*** es menor o mayor que el comienzo (según el orden x-y lexicográfico en un plano 2D). En base a lo anterior, una autopista o carretera está compuesta por un conjunto de route.

$$Route = \{(id, l, c, kind, start) | id \in int, l \in real, c \in line, kind \in intcon\ \{0 = simple, 1 = dual\}, start \in \{0 = smaller, 1 = larger\}\} \quad (2)$$

La principal característica de este modelo es representar un objeto en movimiento relativo a la route en la que se encuentra, por lo cual para cualquier objeto solo será necesario almacenar el ***id*** de route, y un real representando la posición que ocupa dentro de la curva ***c*** que describe route, esto es llamado route measure (Rmeas). En un Rmeas (expresión 3), ***rid*** es el ***id*** de route, y ***d*** un real que representa la posición desde el inicio de route.

$$Rmeas(R) = \{(rid, d) | rid \in int, d \in real\} \quad (3)$$

Sin embargo, al considerar que route puede ser dual, nace un concepto más general, route location (Rloc), como se señala en la expresión 4.

$$Rloc(R) = \{(rid, d, side) | (rid, d) \in Rmeas(R), side \in \{up, down, none\}, \\ \forall\ (rid, l, c, kind, start) \in R : kind = simple \Leftrightarrow Side = none\} \tag{4}$$

Para complementar la descripción de una red surge el concepto de junction (expresión 5), esta ocurre entre pares de route, y representa la intersección de estas, donde cc indica las características de la intersección en términos de conectividad (conexiones entre el par de route considerando que ambas son duales), y toma un valor codificado en 16 bits como indica la matriz de la figura 1, donde A y B representan rutas duales, usando sufijos u y d para arriba (up) y abajo ($down$) respectivamente (1 para identificar si es posible el movimiento , por ejemplo A_u hacia B_u, y 0 en caso contrario). °

$$Junction(R) = \{(rm1, rm2, cc) | rm1, rm2 \in Rmeas(R), \\ rm1 = (r1, d1), rm2 = (r2, d2), r1 \neq r2, cc \in int\} \tag{5}$$

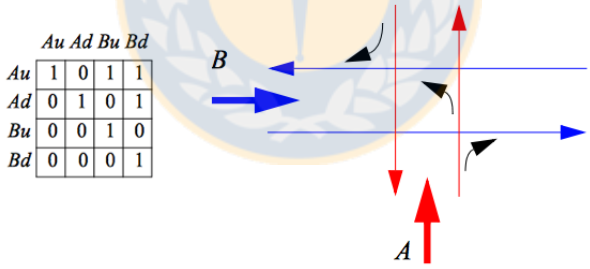

Figura 1: Conectividad entre 2 routes. Fuente: [5]

Finalmente una red se describe como N = (R, J), conjunto R de route, y conjunto J de junctions, como se puede observar en la figura 4.

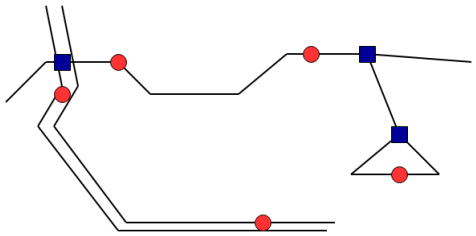

Figura 2: Ejemplo de una red, en cuadrado junctions, en líneas simples routes simples, en líneas dobles routes dual, en círculos route locations. Fuente: elaboración propia.

Todo lo anterior corresponde a la parte espacial del modelo propuesto en [5], para la parte temporal se utiliza la representación basada en porciones o intervalos de tiempo para objetos en movimiento.

Supongamos un objeto en movimiento en una red, empezando en t_0 y terminando en t_n. Para representar este objeto según el modelo, se debe entonces dividir el tiempo en intervalos disjuntos entre si, para cumplir con la restricción que ningún objeto puede estar en distintos lugares en un mismo tiempo como se observa en la figura 3. Para responder a qué tiempo corresponde x posición, dado que solo se tiene intervalos de tiempo, se propone ocupar funciones simples, por ejemplo para puntos en movimientos se ocupan funciones lineales.

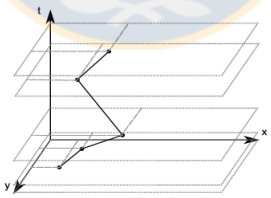

Figura 3: Ejemplo de un objeto en movimiento con representación time-sliced en espacio libre. Fuente : [2]

El siguiente modelo de objetos en movimiento restringido a redes llamado Ordered Relation Graph Representation (en adelante OR), considera la red de transporte como un grafo dirigido $G = (N, E)$ (N conjunto de vértices o nodos de OSM, y E conjunto de arcos dirigidos) desde la información de OSM, permitiendo ver el grafo de una forma relacional (expresión 6). Esta estructura de grafo contiene arcos con un sentido u orientación, a diferencia

del modelo anterior [5], que considera una ruta (simple o dual) como un arco no dirigido más una variable que describe si tiene uno o dos sentidos (kind).

$$Node = \{WayId : int, NodeCounter : int, NodeIdNew : int, Pos : Point\}$$
$$Edge = \{Source : int, Target : int, SourcePos : point, TargetPos : point,$$
$$SourceNodeCounter : int, TargetNodeCounter : int, Curve : curve,$$
$$RoadName : string, RoadType : string, WayId : int\} \tag{6}$$

Otra de las diferencias del modelo OR con [5], es que éste no es capaz de representar otros objetos como puntos o líneas sobre la red, sino que sólo puntos en movimiento. El modelo considera el inicio y termino de la trayectoria de un objeto en movimiento, como nodos del grafo de la red, nunca posiciones ubicadas sobre los arcos.

Por lo tanto, un objeto en movimiento se presenta como una secuencia ordenada de tuplas, que describe el movimiento utilizando como referencia los arcos más un intervalo de tiempo por el cual el objeto estuvo en ese arco de la red. El modelo relacional de la red se describe en la figura 4:

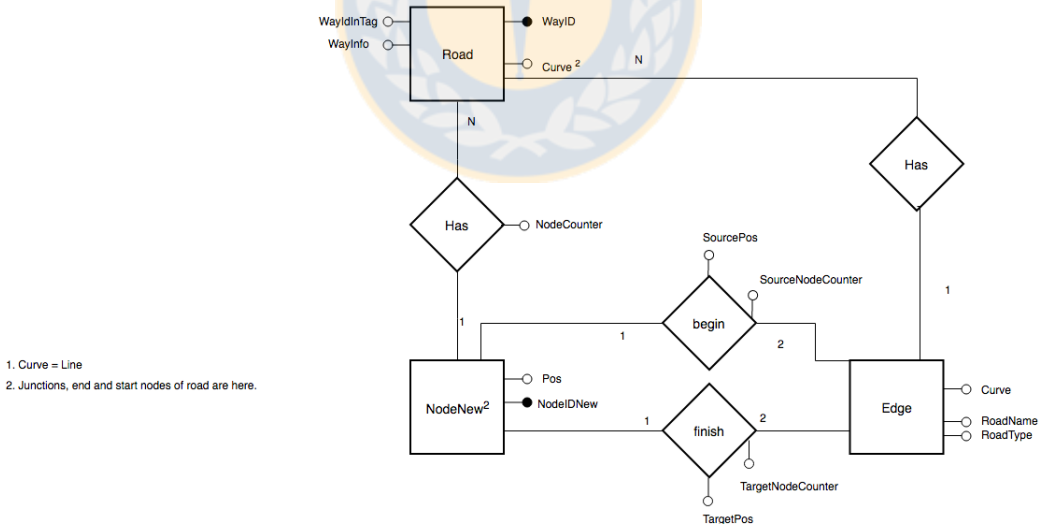

Figura 4: MER de la red del modelo Ordered Relation Graph Representation. Fuente : elaboración propia.

3. Modelo de datos implementado

Para completar los objetivos de esta memoria, se analiza un caso real donde se aplican estos modelos de datos. Este caso real contiene los elementos necesarios para la utilización de todos los modelos elegidos, esto es, la red y los objetos en movimiento.

A continuación se describen dos secciones, la primera, donde se detalla el caso real donde se aplican los modelos, y en el siguiente se detalla a cabalidad el modelo utilizado y su implementación.

3.1. Dominio de Aplicación

En esta sección se explica el dominio donde se realizan las comparaciones, esto es, transporte público. Si bien lo que concierne a la red, estos datos son de libre acceso a través de la plataforma Open Street Map. Los datos provistos por Transantiago no son de acceso acceso libre, pero por la condición de contratos con concesionarias privadas del transporte público, deben ser entregadas a la secretaría ministerial, quien a su vez puede permitir el uso de estos datos.

3.1.1. Transantiago

Transantiago es un sistema de buses de transporte público urbano que opera en el área metropolitana de la ciudad de Santiago, capital de Chile. Está integrado tarifariamente con el Metro de Santiago y el servicio de tren suburbano Metrotren Nos. Los usuarios del sistema pueden moverse en cualquiera de los tres modos antes señalados, haciendo hasta dos transbordos por el valor de un solo pasaje. Estos transbordos se pueden realizar en un espacio de tiempo de dos horas (120 minutos) después de haber realizado la primera transacción.

Este sistema de transporte público cubre alrededor de 6,2 millones de usuarios de las 32 comunas que forman parte de Santiago, en un área geográfica de alrededor de 680 km^2 en zonas urbanas. En un día laboral, se efectúan alrededor de tres millones de transacciones al interior de los buses.

Desde el sistema de transporte Transantiago, se obtuvo la información de la demanda, en términos de las transacciones de las tarjetas Bip! (nombre que reciben las smartcards de Transantiago), como también la información relativa a los buses, específicamente el posicionamiento de estos. Para este trabajo sólo se utilizaron los datos del día 23 de mayo, que corresponde a 6.389 buses con un total de 70.820 trayectorias.

Los datos de posicionamiento se almacenan en un archivo que se describe de la siguiente manera:

$$Position = \{BusID : string, Lat : float, Lon : float, TimeGPS : Timestamp, RouteID : string, Speed : float, Ignition : boolean\} \quad (7)$$

3.1.2. Open Street Map

Con respecto a los datos de la red, estos fueron obtenidos desde Open Street Map, la cual es una plataforma colaborativa para crear mapas libres y editables. De esta plataforma se extrajo la información relativa a Santiago Metropolitano, en Chile. El archivo descargado se encuentra en formato XML, con extensión osm. Un archivo osm contiene:

- Nodo: Define una ubicación geográfica en el espacio en términos de longitud y latitud. También tiene un identificador único de nodo.
  Un nodo puede tener asociado una lista tags de nodos. Un tag de nodo es un par (Key, Value) para almacenar información extra sobre el nodo.
- Camino: Describe una geometría (Línea), tiene su propio identificador de camino, y es definido como una secuencia de identificadores de nodos. Un camino puede describir una carretera o el perímetro de un parque o edificio, por ejemplo.
  Observar que las ubicaciones geográficas están presentes solo en nodos, así el encuentro de dos carreteras en una unión deberían compartir un nodo con una única ubicación. Por otro lado, los caminos también pueden tener asociado tags de caminos, vale decir pares (Key, Value) para almacenar información extra sobre el camino.

- Relación: tiene su propio identificador y está definido como una lista de referencia a nodos, caminos o relaciones. Así una relación puede describir un conjunto de cualquier cosa, incluso de una manera anidada. Una relación también puede tener tags de relación.

Para clarificar con mayor detalle se muestra en la figura 5, un modelo entidad relación que describe la información encontrada en un archivo OSM.

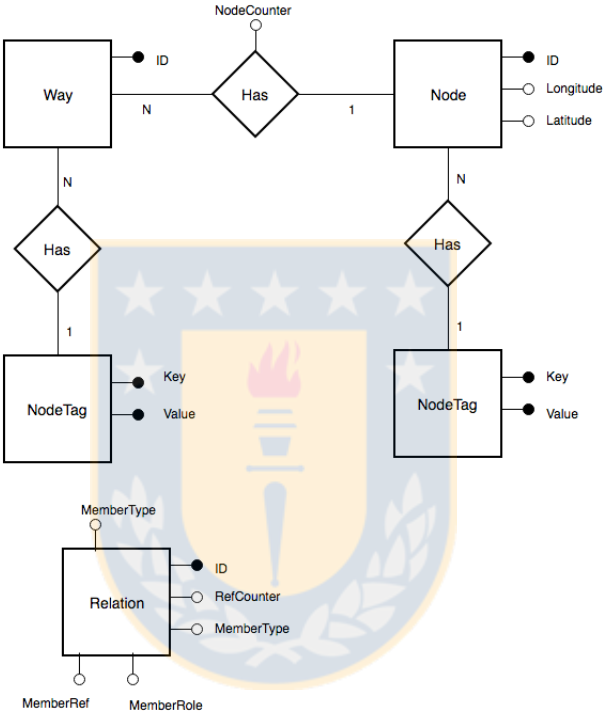

Figura 5: MER de la información de archivo OSM. Fuente: elaboración propia

3.2. Modelos utilizados

Para completar los objetivos de esta memoria se decidió utilizar el gestor de bases de datos Secondo [3], el cual es un ambiente ideal para crear modelos de datos y sus implementaciones ya que estos pueden integrase al sistema a través de módulos llamados Álgebras. Un álgebra en Secondo se compone de tipos de datos y operadores sobre estos, y se utiliza ampliamente para realizar comparaciones, usualmente cuando los investigadores necesitan comparar

sus trabajos, ya sean modelos, índices o estructuras frente a modelos del estado del arte.

El modelo propuesto en [5] explicado en la discusión bibliográfica sección 2.2.2 dispone de una implementación llamada Network, es un álgebra en Secondo que tiene los siguientes tipos de datos:

- Espaciales
  - Red: *network*
  - Punto en la red: *gpoint*
  - Puntos en la red: *gpoints*
  - Línea en la red: *gline*
- Temporales
  - Punto en movimiento sobre la red: *mgpoint*
  - Punto de movimiento en un instante: *igpoint*
  - Punto en movimiento en un intervalo: *ugpoint*

El modelo OR se encuentra implementado en una álgebra llamada OrderedRelation sobre Secondo, como se menciona anteriormente en la sección 2.2.2 un objeto en movimiento se representa como una secuencia ordenada de tuplas, donde cada tupla representa el arco por el cual pasa el objeto, más el intervalo de tiempo durante el cual el objeto estuvo en ese arco. Las tuplas contienen la estructura que describe la expresión 8, donde Source y Target son los identificadores de los nodos del grafo, SourcePos y TargetPos son las posiciones en coordenadas x e y de los nodos del grafo :

$$\begin{gathered} Tuple = \{Source : int, Target : int, SourcePos : point, TargetPos : point, \\ SourceNodeCounter : int, TargetNodeCounter : int, Curve : curve, \\ RoadName : string, RoadType : string, WayId : int, StartTime : Timestamp, \\ StopTime : Timestamp\} \end{gathered} \tag{8}$$

Los modelos a utilizar en esta memoria son:

- Modelo de objetos en movimiento en espacio libre, como una base de comparación.

- Modelo de objetos en movimiento sujeto a redes, con implementación Network[5].
- Modelo Ordered Relation Graph Representation.

Descripción de los Datos

Como se explica anteriormente los datos de la red fueron obtenidos desde OSM y los datos de los objetos en movimiento corresponden a los registros de los buses del transporte público de Santiago. Los datos de OSM son registrados por los usuarios, por lo cual no tiene actualizaciones periódicas ni tampoco se asegura que los datos ingresados son verídicos. En contra parte los datos aportados por Transantiago son obtenidos por dispositivos GPS localizados en los buses, a pesar de que son obtenidos con dispositivos electrónicos y automatizados, sufren de fallas como se muestra en la figura 6, donde se registra para una misma línea (T4TS 00I) distintas trayectorias.

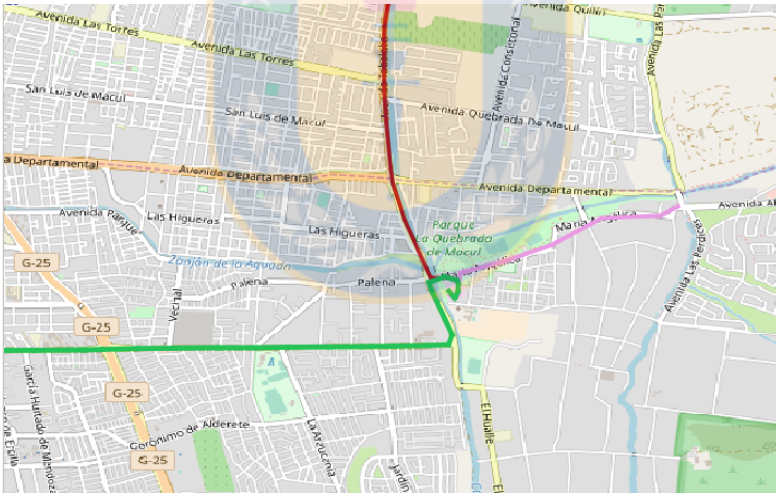

Figura 6: 3 trayectorias de una misma línea de bus, lineas verde, rosada y roja.

Pre Procesamiento

El primer paso consiste en realizar una extracción, limpieza y transformación de los datos. Desde la fuente OSM se obtiene el archivo de la red como se

menciona en la sección 3.1.2. Este archivo XML es importado en un primer paso con la ayuda de los scripts entregados en [6], para crear la red explicada en la sección 2.2.2. Posteriormente, se obtuvieron los datos del transporte público de Transantiago como se hace mención en la sección 3.1.1.

Luego de obtener la información, se procede a limpiarla y transformarla, esto se realiza bajo los siguientes criterios:

- Se agrupan los datos por patente.
- Se agrupan los datos por recorrido.
- Se separan los viajes, considerando: diferencias de tiempo mayor a 5 minutos, que es el tiempo en promedio en que se empieza un nuevo recorrido una vez finalizado el anterior, y considerando distancias menores a 20 metros entre puntos, debido a que cuando un bus está estacionado sin empezar su recorrido efectivo, son puntos que entorpecen el mapeo hacia la red.

Implementación

Como se menciona en la sección 2.2.2, uno de los modelos a utilizar es Ordered Relation Graph Representation (OR), el cual considera la red como un grafo dirigido, por ende un objeto en movimiento se presenta como una secuencia de tuplas en orden que describe el movimiento utilizando como referencia los arcos y nodos del grafo.

A continuación una breve descripción de cómo se construye el grafo dirigido de la red.

Creación de la Red en Modelo OR

Se procesa el archivo OSM creando una representación relacional como se describe anteriormente, con seis relaciones: Nodes, Tags Nodes, Ways, Way Tags, Relations, y Relation Tags.

Se crea una copia de la relación Nodes tal que latitud y longitud son trasladados a un atributo tipo Point. Luego se crean nuevos identificadores numéricos

que son espacialmente agrupados (números cercanos están en ubicaciones similares), la relación es llamada NodeNew.
Desde las seis relaciones se reconstruye la relación WayNew como una relación anidada, cada tupla de WayNew contiene una sub-relación con la secuencia de nodos definiendo la geometría del camino. La geometría también es añadida como un atributo de tipo Line. Cada tupla de WayNew además contiene una sub-relación para sus tags.

Basado en la información de los tags, se seleccionan las carreteras como un subconjunto de la relación WayNew, y se le asigna el nombre de Road.

Se construye la relación NodeNew2, a partir de los caminos del grafo de la red (relación Road). NodeNew2 son: o bien una unión de dos carreteras distintas o el inicio o fin de una geometría de un camino (también llamados nodos terminales).

Posteriormente se construyen los arcos dirigidos del grafo de la red, el contenido de la relación Edges son trozos de carreteras entre dos nodos de la relación NodeNew2, por ejemplo entre dos uniones, o entre una unión y un nodo terminal. Este procedimiento es hecho en 3 etapas:

- Construcción de los arcos dirigidos en la dirección original del camino. Si una carretera es de un sentido, entonces la dirección de recorrido corresponde al orden de los nodos.
- Construcción de los arcos dirigidos en la dirección contraria si la carretera es de dos sentidos.
- Finalmente se unen los dos conjuntos anteriores.

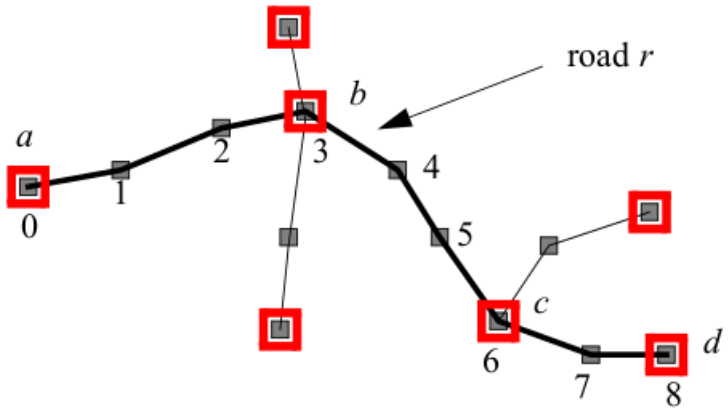

Figura 7: Ejemplo de representación de nodos del grafo. Los cuadrados grises (con números) corresponden a los nodos de OSM, los cuadrados rojos (con letras) corresponden a nodos en la red (uniones o terminales). Fuente : [3]

Teniendo la red creada, se dispone a realizar el map matching de las trayectorias de los buses del Transantiago. Usando nuevamente el álgebra MapMatching con el operador OMapMatchingMHT se lleva a cabo el objetivo, realizando el mapeo de 70.000 trayectorias.

El algoritmo para realizar el mapeo de las trayectorias a la red se describe así:

1. Se almacena en una relación los nombres de los archivos GPX.
2. Se crea una relación donde cada tupla contiene nombre de archivo, ID, y una relación anidada que se genera usando el operador GPXimport. Éste crea la relación {Time: timestamp, Lon: float, Lat: float, Pos: Point} y se le asigna el nombre Raw.
3. Se crea una nueva relación, se le asigna el nombre DayTrip, usando las posiciones de la relación Raw (atributo tipo Point), y el operador approximate del álgebra Temporal [4], se crea una tupla por cada identificador de la relación Raw, y ésta contiene un mpoint.
4. Se crea una última relación llamada MatchedTrips, se obtiene de copiar DayTrip, y de añadir un atributo de tipo relación. En cada tupla de la relación anidada se describe la información del objeto en movimiento en términos de los arcos del grafo. La relación se describe así:

{ Source: int, Target: int, SourcePos: point, TargetPos: point, SourceNodeCounter: int, TargetNodeCounter: int, Curve: curve, RoadName: string, RoadType: string, WayId: int, StartTime: Timestamp, StopTime: Timestamp }

A priori se puede intuir que este último modelo no solo mantiene la misma información que el modelo de espacio libre, sino que añade información extra, lo que puede traer una consecuencia negativa en el uso de espacio.

4. Experimentos y Resultados

En este capítulo se detalla los experimentos realizados con el fin de demostrar las bondades de los modelos basados en movimiento restringido a redes, frente al modelo de espacio libre. Estos experimentos son realizados en una máquina con procesador Intel Xeon CPU E3-1220 v5 3Ghz x 8, memoria RAM de 16 GB y memoria secundaria de 100 GB.

Los experimentos están enfocados en tres grupos. Primero se analiza los tiempos de respuestas de las consultas espaciales, luego temporales y posteriormente espacio-temporales. Por último se realiza un análisis del tamaño que ocupa cada representación como también la calidad en términos de representación de trayectorias.

4.1. Consultas espaciales

Las consultas espaciales se realizaron creando 3 tipos de pruebas, estos corresponden a distintos rangos de consultas (ver sección Anexo). Como se ha explicado anteriormente cada modelo tiene su propia representación de objeto en movimiento, por lo cual las consultas no son las mismas para cada modelo, para realizar comparaciones justas es necesario traducir cada consulta utilizando los operadores que se disponen para cada modelo.

Tomando en consideración el dominio de los datos, la primera consulta llamada *Space L* es sobre un bounding box de la región metropolitana que abarca los recorridos de Transantiago, comprende un rectángulo de 90km en su diagonal. La siguiente consulta *Space M* corresponde a 5 bounding

box tomados de forma aleatoria contenidos en el bounding box de la consulta *Space L*. Estos rectángulos tienen un tamaño de diagonal de 10 km. Finalmente, la última consulta *Space S* corresponde también a 5 bounding box tomados de forma aleatoria contenidos en el bounding box de la consulta *Space L* pero esta vez el tamaño de la diagonal es 400m.

Teniendo estas 3 consultas, se procede a realizarlas sobre los distintos modelos de datos, se realizan pruebas incrementales sobre la cantidad de trayectorias, primero con el conjunto total de trayectorias (70 mil), luego con la mitad del conjunto, y finalmente 10 mil trayectorias, como demuestra en la figura 8, 9 y 10.

A continuación los resultados:

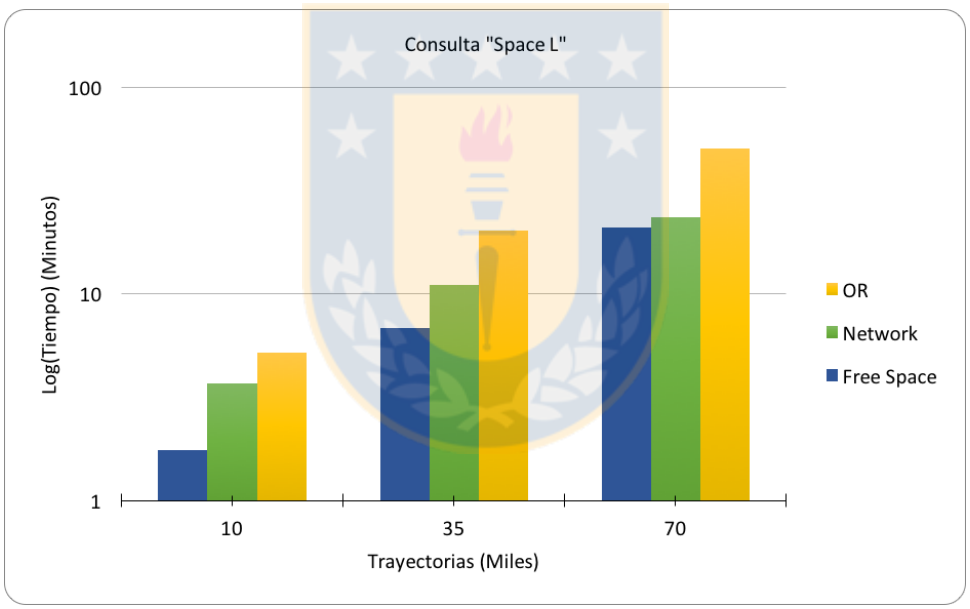

Figura 8: Consulta espacial L.

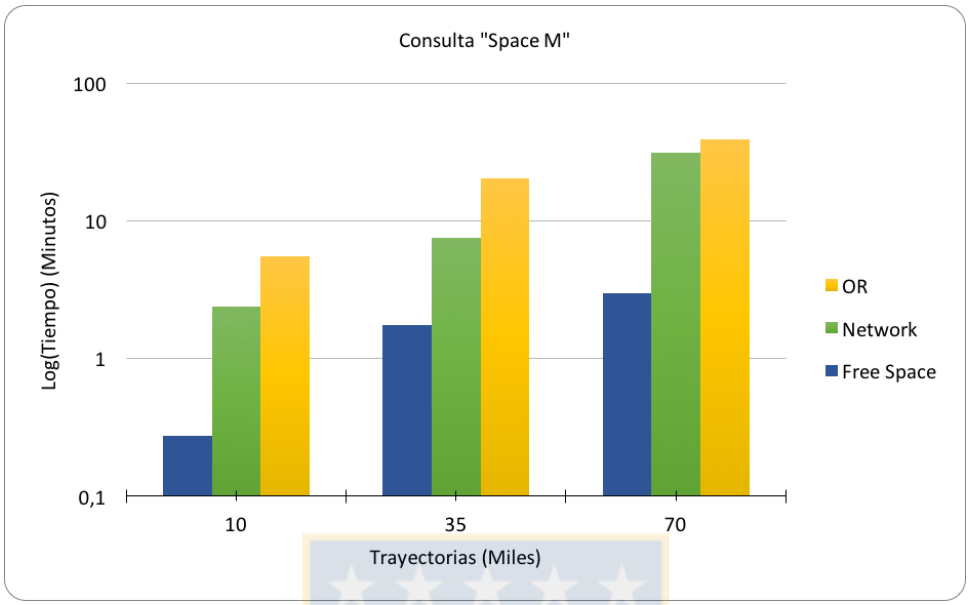

Figura 9: Consulta espacial M.

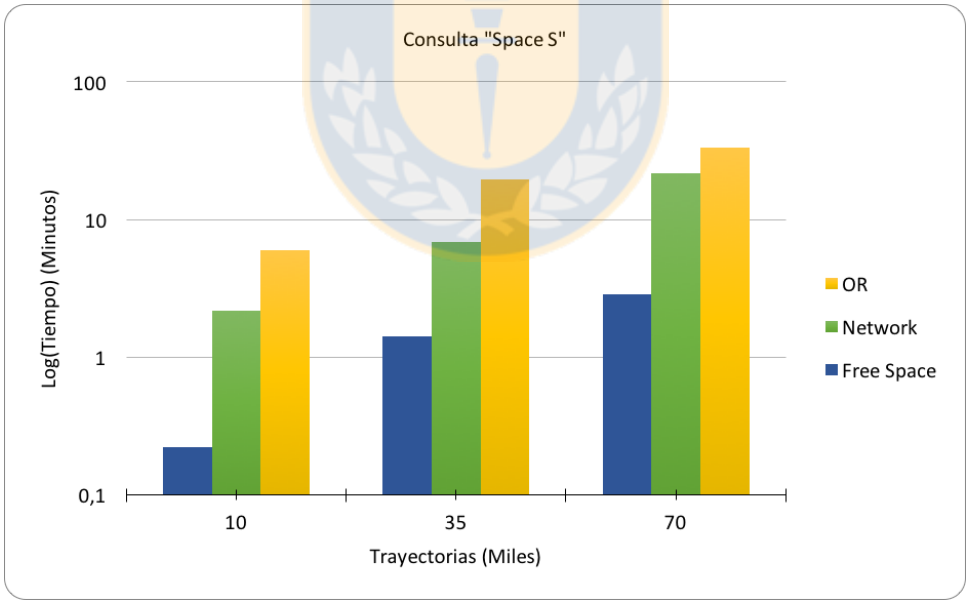

Figura 10: Consulta espacial S.

Se desprende de los resultados que el modelo Free Space es más eficiente en tiempo de consulta que los otros modelos. Se puede observar que mientras

los bounding box de las consultas son más pequeños, Free Space saca una mayor ventaja con respecto a los otros modelos.

4.2. Consultas temporales

Para las consultas temporales, se crean 3 tipos de consultas, siguiendo la estrategia de las consultas espaciales, (1) un intervalo de tiempo de todo el día 23 de mayo de 2016, (2) sobre 5 intervalos aleatorios dentro de las 24 horas, de 3 horas cada uno, y (3) 5 intervalos de 30 minutos. Ninguna de estas consultas considera una región espacial en específico, se consulta sobre todo el dominio de trayectorias. En las figuras 11, 12 y 13 los resultados de las consultas aplicadas sobre los 3 modelos ocupados.

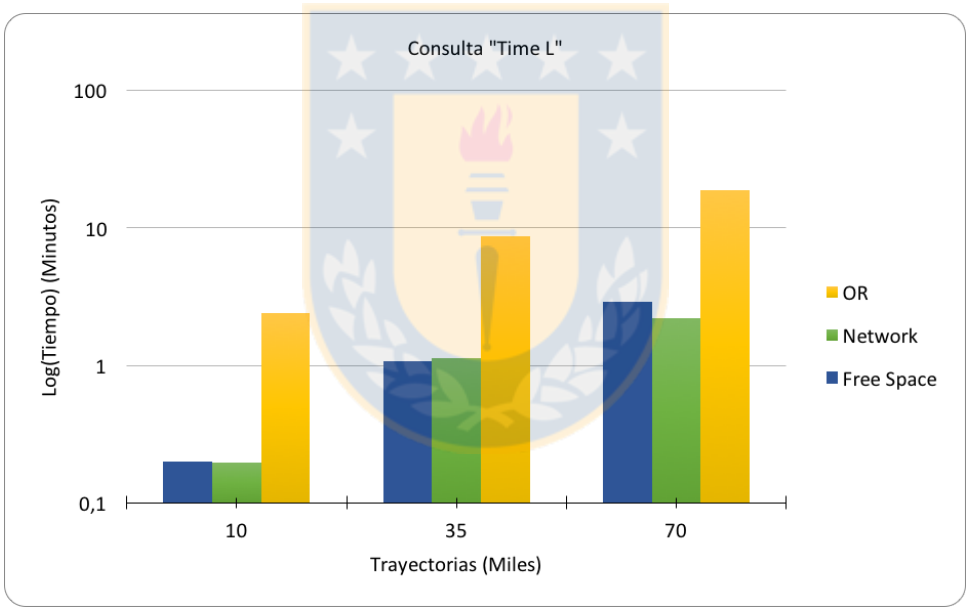

Figura 11: Consulta temporal L.

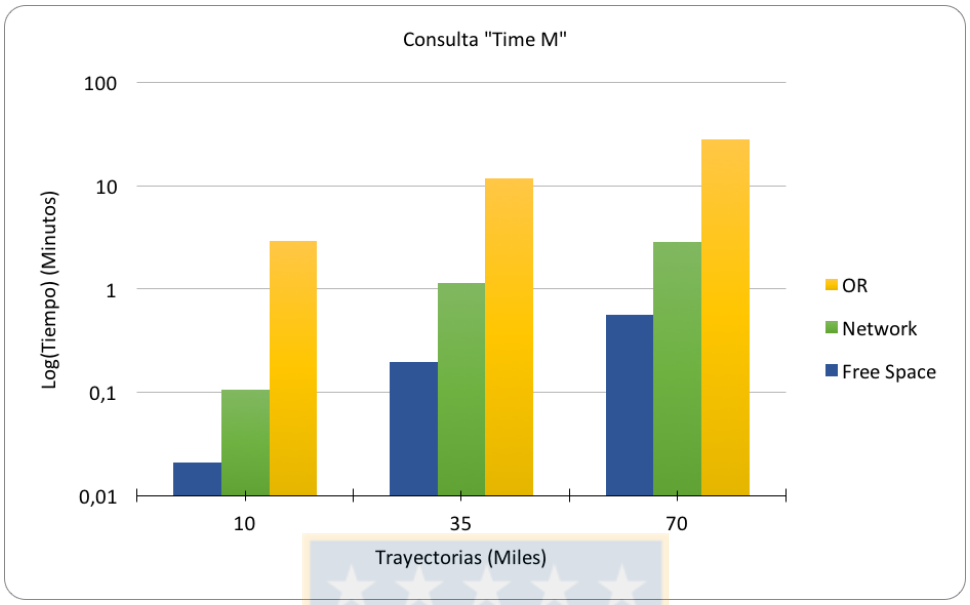

Figura 12: Consulta temporal M.

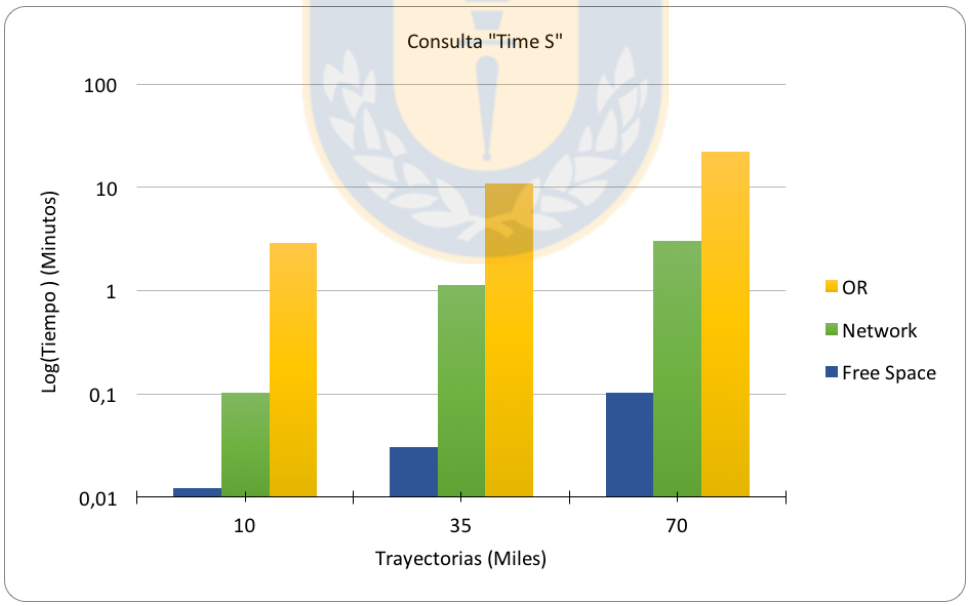

Figura 13: Consulta temporal S.

En la consulta temporal L tanto Free Space como Network alcanzan resultados similares, esto puede ser debido a que no hay descarte de los conjuntos de

trayectorias consultados, es decir, para consulta temporal L teniendo 70.000 trayectorias, retorna las 70.000 trayectorias como resultado. A medida que los rangos temporales son más pequeños, Free Space toma ventaja. En todas las consultas OR es significativamente peor que los otros modelos.

4.3. Consultas espacio-temporales

Las consultas espacio-temporales, abarcan 3 tipos de consultas, provenientes de la combinación de las consultas espaciales y temporales mencionadas anteriormente, vale decir una conjunción de *Space* X y *Time* X, con $X \in \{L, S, M\}$, como se señalan en las figuras 14, 15 y 16.

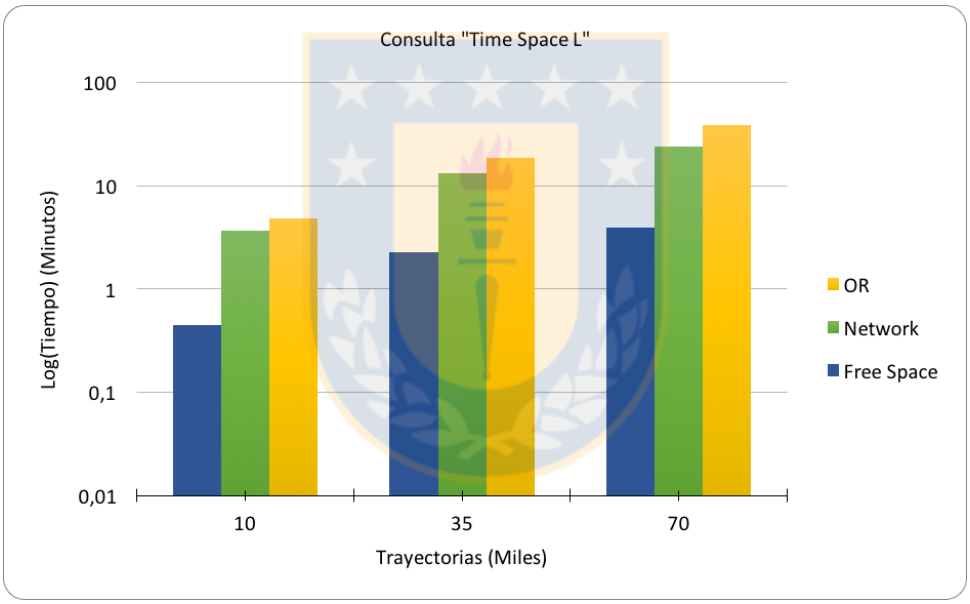

Figura 14: Consulta espacio-temporal L.

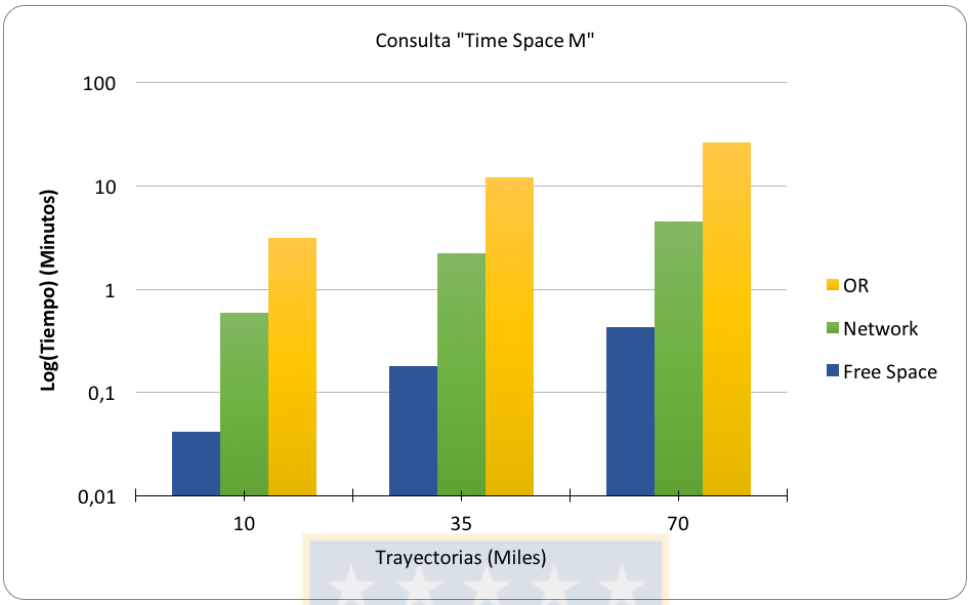

Figura 15: Consulta espacio-temporal M.

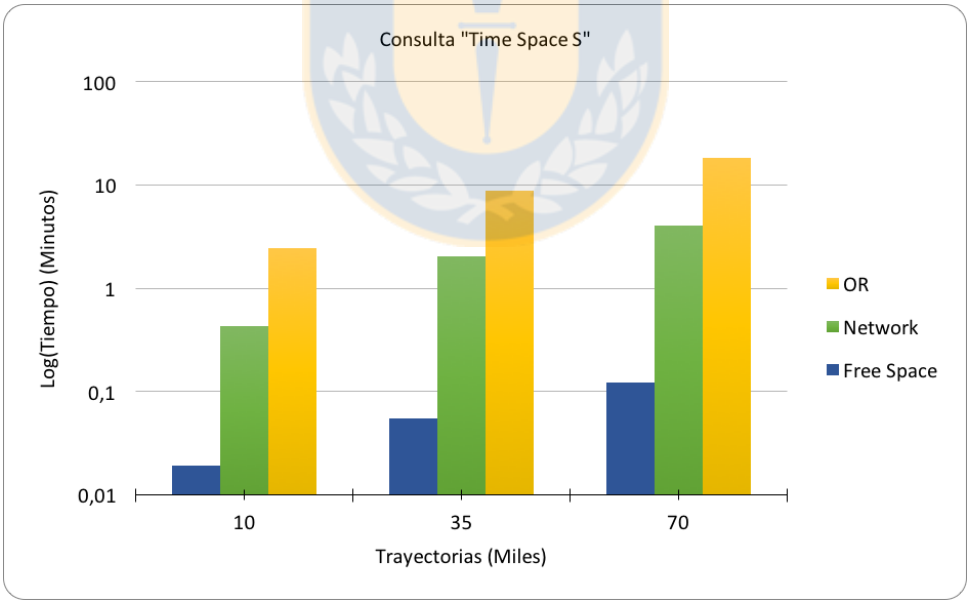

Figura 16: Consulta espacio-temporal S.

4.4. Uso de espacio

En esta sección se hace un análisis del espacio utilizado por cada modelo, el modelo Free Space utiliza un tamaño de $1GB$ para representar 70,000 trayectorias. Se hace necesario señalar que de todos los modelos utilizados en este trabajo, éste es el único que no utiliza una red subyacente.

Los modelos Network y OR sí ocupan una red subyacente para ser representados, es por esto que en el tamaño que utilizan estos modelos no solo se considera el espacio utilizado por las trayectorias, si no que también el tamaño de la representación de la red según corresponde. El modelo Network utiliza $0,36GB$ para representar la red, y $1GB$ para representar el conjunto de trayectorias. Finalmente el modelo OR utiliza $0,8GB$ para representar la red subyacente y $10,4GB$ para el conjunto de trayectorias, debido a que no solo se almacenan los identificadores de los nodos del grafo, sino que la ubicación espacial de los nodos, la curva que describe el arco, entre otros.

En la figura 17 se señalan los espacios totales utilizados por cada representación.

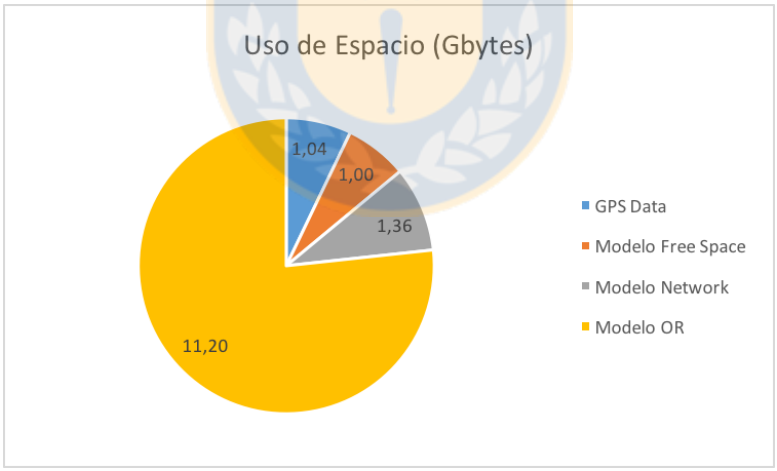

Figura 17: Uso de espacio de los 3 modelos utilizados y de la información de los buses en bruto.

Se puede observar que OR no es competitivo con los otros modelos en el uso de espacio, puesto que utiliza al menos 10 veces el espacio que utiliza Network. Los modelos Network y Free Space utilizan espacio en disco bastante

similares, se debe tener en cuenta que Network además de almacenar las trayectorias, almacena la red subyacente.

4.5. Calidad de representación y uso de Map Matching

Es preciso mencionar que los datos utilizados de los objetos en movimiento se encuentran en un formato de espacio libre (coordenadas geográficas) por lo cual el modelo Free Space mantiene una similitud casi exacta con las trayectorias de los objetos, ya que este modelo también es representado en coordenadas geográficas, a diferencia de los modelos OR y Network, ya que ambos ocupan algoritmos de Map Matching, sabiendo que el problema de Mapear objetos a una red es un problema abierto, no siempre el algoritmo entrega buenos resultados, el análisis de la calidad del algoritmo de Map Matching utilizado está fuera del alcance de esta memoria de titulo.

5. Conclusiones

Tomando en cuenta los resultados de los experimentos se puede observar que el modelo OR no es competitivo con ninguno de los otros dos modelos, tanto en consultas espaciales, como temporales y espacio-temporales. Esto debido a que este ocupa una representación de tuplas por cada arco que recorre en el grafo, y para realizar este procedimiento es necesario consultar sobre todas las tuplas del conjunto para chequear, por ejemplo, intersección con algún intervalo de tiempo o algún bounding box como es el caso de las consultas de este trabajo.

Los modelos Network y Free Space alcanzan resultados similares solo cuando se consulta sobre los dominios totales, tanto espacial ($Consulta\ "SpaceL"$) como temporal ($Consulta\ "TimeL"$). Para el caso temporal, aun cuando existe el mismo operador para ambos modelos (atperiods), y tomando en cuenta que ambos modelos comparten la misma representación temporal, el álgebra Free Space está mejor implementada que para Network [6]. Para el caso espacial, es necesario mencionar que Network solo dispone de operadores para consultas de rango temporales y no espaciales, por lo cual es estrictamente necesario mapear un punto en movimiento sobre la red a un punto en movimiento sobre el espacio libre para realizar las consultas espaciales, esto hace que las consultas sean más costosas ya que debe pasar de un dominio a otro.

En cuanto a lo que concierne a espacio utilizado para cada representación, OR no es competitivo con los otros modelos, puesto que utiliza al menos 10 veces el espacio que utiliza Network. Como punto favorable se puede mencionar que OR tiene un mayor poder de expresividad, ya que está conformado por tuplas con tipos de datos simples (point, curve, integer). En consecuencia si se quisiese realizar consultas más elaboradas, por ejemplo consultas agregadas, no sería necesario crear operadores para esto, ya que Secondo dispone de álgebras para tratar los tipos de datos simples, a diferencia de los otros dos modelos, que para nuevas consultas es necesario crear otros operador en las álgebras correspondientes.

Los modelos Network y Free Space utilizan espacio en disco bastante similares, a pesar de que las trayectorias de Network deben tener la red subyacente en la base de datos. Si se considera un dominio más grande que el registro uti-

lizado (1 día), por ejemplo, un semestre, el tamaño que utiliza la red tiende a ser despreciable, lo que lleva a que ambos modelos utilizan un espacio aproximadamente iguales. En este punto se hace necesario resaltar las ventajas de Network, ya que como se menciona en la sección de Pre-procesamiento los datos fueron limpiados en una etapa temprana, antes de realizar el mapeo, lo que nos lleva a puntos separados temporal y espacialmente, lo suficiente para que el modelo no demuestre su mejor faceta.

Supongamos lo siguiente, un bus viaja por la red de Transantiago obteniendo datos del gps cada medio segundo (actualmente los datos son obtenidos cada 30 segundos). Suponiendo que viaja a una velocidad de 50 km/h, recorre 13.9 metros cada segundo, y se encuentra en una calzada (sin intersecciones) de longitud de 5 km, este cálculo nos lleva a que el GPS recolecta 720 veces la ubicación del bus, Free Space necesitaría de un moving point con 720 componentes, a diferencia de Network que solo necesitaría 2 componentes para representar dicho movimiento (posición inicial y final con respecto a la ruta, más tiempo inicial y tiempo final del movimiento).

Finalmente se puede decir que en cuanto a espacio Network es al menos tan bueno como Free Space, siendo mejor si los datos son obtenidos con una mayor frecuencia. En cuanto a rendimiento de consultas, Free Space obtiene notorias ventajas en todos los tipos de consultas realizadas en este trabajo, siendo más notorio cuando se consulta un subconjunto del dominio espacial y/o temporal. Esto puede deberse a que Free Space es un modelo más maduro, y con operadores mejor implementados (más eficientes), por lo cual no se descarta que Network eventualmente pudiese acercarse en rendimiento.

Referencias

[1] E.F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, June 1970.

[2] Christian Düntgen, Thomas Behr, and Ralf Hartmut Güting. Berlinmod: A benchmark for moving object databases. *The VLDB Journal*, page 1335, 2009.

[3] R. H. Güting, V. T. de Almeida, D. Ansorge, T. Behr, Z. Ding, T. Höse, F. Hoffmann, M. Spiekermann, and U. Telle. Secondo: An extensible dbms platform for research prototyping and teaching. *ICDE*, page 1115–1116, 2005.

[4] R. H. Gütting, M.H. Bolen, M. Erwig, C.S. Jensen, N. A. Lorentzos, M. Schneider, and M. Vazirgiannis. A foundation for representing and querying moving objects. *ACM Transactions on Database System*, 25(1):1–42, 2000.

[5] R. H. Gütting, V. T. de Almeida, and Z. Ding. Modeling and querying moving objects in networks. *The VLDB Journal*, 15(2):165–190, 2006.

[6] Simone Jandt. *Secondo DBMS and Network Constrained Moving Objects*. FernUniversiät in Hagen, June 2013.

[7] Sistla A. P., Wolfson O, Chamberlain S, and Dao S. Modeling and querying moving objects. *Int. Conf. on Data Engineering*, page 422–432, 1997.

[8] Erwig M.and Güting R.H. and Schneider M. et al. Spatio-temporal data types: An approach to modeling and querying moving objects in databases. *GeoInformatica*, 3:269–296, 1999.

[9] Güting R.H and Schneider M. *Moving Objects Databases*. Morgan Kaufmann Publishers, 2005.

6. Anexo

6.1. Consultas Realizadas

En esta sección se muestra las consultas realizadas, los rangos temporales y los bounding box espaciales, tomando un ejemplo de cada tipo consulta, considerando un conjunto de 70.000 trayectorias.

- Creación Rangos Temporales para modelo Free Space y Network
  - *let time_L* = [*const periods value* (("2016 − 05 − 23 − 00 : 00 : 01" "2016 − 05 − 24 − 00 : 01" *TRUE FALSE*))]
  - *let time_M* = [*const periods value* (("2016 − 05 − 23 − 01 : 00" "2016 − 05 − 23 − 04 : 00" *TRUE FALSE*))]
  - *let time_S* = [*const periods value* (("2016 − 05 − 23 − 19 : 16" "2016 − 05 − 23 − 19 : 46" *TRUE FALSE*))]
- Creación Rangos Temporales para modelo OR
  - *let begin_time_L* = [*const instant value* "2016 − 05 − 23 − 05 : 00"]
  - *let begin_time_M* = [*const instant value* "2016 − 05 − 23 − 05 : 00"]
  - *let begin_time_S* = [*const instant value* "2016 − 05 − 23 − 19 : 46"]
  - *let duration_1d* = [*const duration value* (1 0)]
  - *let duration_3h* = [*const duration value* (0 10800000)]
  - *let duration_30m* = [*const duration value* (0 1800000)]
- Consultas Temporales
  - Modelo Free Space:

    Query Time_L: *query DayTrips feed addid filter*[.*TID* < 70000] *extend*[*N* : *no_components*(.*DayTrip atperiods time_L*)] *filter*[.*N* # 0] *count*

    Query Time_M: *query DayTrips feed addid filter*[.*TID* <

70000] $extend[N : no\_components(.DayTrip\ atperiods\ time\_M)]$ $filter[.N\ \#\ 0]$ *count*

Query Time_S: *query DayTrips feed addid* $filter[.TID\ <$ 70000] $extend[N : no\_components(.DayTrip\ atperiods\ time\_S)]$ $filter[.N\ \#\ 0]$ *count*

- Modelo Network:

  Query Time_L: *query Matched feed addid* $[.TID\ <\ 70000]$ $extend[N\ :$ $no\_components(.Matched\ atperiods\ time\_L\ ]$ $filter[.N\ \#\ 0]$ *count*

  Query Time_M: *query Matched feed addid* $[.TID\ <\ 70000]$ $extend[N\ :$ $no\_components(.Matched\ atperiods\ time\_M\ ]$ $filter[.N\ \#\ 0]$ *count*

  Query Time_S: *query Matched feed addid* $[.TID\ <\ 70000]$ $extend[N\ :$ $no\_components(.Matched\ atperiods\ time\_S\ ]$ $filter[.N\ \#\ 0]$ *count*

- Modelo OR:

  Query Time_Space_L: *query MatchedTrips feed addid* $filter[.TID\ <$ 70000] $extend[I\ : .Matched\ a feed\ filter[.StartTime\ >\ begin\_time\_L]$ $filter[.EndTime\ <\ begin\_time\_L\ +\ duration\_1d]$ *count*] $filter[.I\ \#\ 0]$ *count*

  Query Time_Space_M: *query MatchedTrips feed addid* $filter[.TID\ <$ 70000] $extend[I\ : .Matched\ a feed\ filter[.StartTime\ >\ begin\_time\_M]$ $filter[.EndTime\ <\ begin\_time\_M\ +\ duration\_3h]$ *count*] $filter[.I\ \#\ 0]$ *count*

  Query Time_Space_S: *query MatchedTrips feed addid* $filter[.TID\ <$ 70000] $extend[I\ : .Matched\ a feed\ filter[.StartTime\ >\ begin\_time\_S]$ $filter[.EndTime\ <\ begin\_time\_S\ +\ duration\_30m]$ *count*] $filter[.I\ \#\ 0]$ *count*

- Creación Bounding Box

  - *let space_L* = [*const rect value* (−70,9477 −70,2634 −33,6841 − 33,3145)]

  - *let space_M* = [*const rect value* (−70,6469 −70,5571 −33,5707 −

33,5184)]

- *let space_S* = [*const rect value* (−70,6478 −70,6243 −33,4669 −33,4532)]

- Consultas Espaciales
  - Modelo Free Space:

    Query Space_L: $query\ DayTrips\ feed\ addid\ filter\ [.TID\ <\ 70000\ ]\ extend[N:\ never(.DayTrip\ inside\ space\_L\ rect2region)]\ filter[.N\ \#\ TRUE]\ count$

    Query Space_M: $query\ DayTrips\ feed\ addid\ filter\ [.TID\ <\ 70000\ ]\ extend[N:\ never(.DayTrip\ inside\ space\_M\ rect2region)]\ filter[.N\ \#\ TRUE]\ count$

    Query Space_S: $query\ DayTrips\ feed\ addid\ filter\ [.TID\ <\ 70000\ ]\ extend[N:\ never(.DayTrip\ inside\ space\_S\ rect2region)]\ filter[.N\ \#\ TRUE]\ count$

  - Modelo Network:

    Query Space_L: $query\ Matched\ feed\ addid\ filter[.TID\ <\ 70000]\ filter[rectproject(mgpbbox(.Matched),1,2)\ intersects\ space\_L]\ filter[\ no\_components(mgpoint2mpoint(.Matched)\ inside\ space\_L\ rect2region)\ \#\ 0]\ count$

    Query Space_M: $query\ Matched\ feed\ addid\ filter[.TID\ <\ 70000]\ filter[rectproject(mgpbbox(.Matched),1,2)\ intersects\ space\_M]\ filter[\ no\_components(mgpoint2mpoint(.Matched)\ inside\ space\_M\ rect2region)\ \#\ 0]\ count$

    Query Space_S: $query\ Matched\ feed\ addid\ filter[.TID\ <\ 70000]\ filter[rectproject(mgpbbox(.Matched),1,2)\ intersects\ space\_S]\ filter[\ no\_components(mgpoint2mpoint(.Matched)\ inside\ space\_S\ rect2region)\ \#\ 0]\ count$

  - Modelo OR:

    Query Space_L: $query\ MatchedTrips\ feed\ addid\ filter[.TID\ <$

70000] $filter[.Matched\ afeed\ filter[not(isempty($
$intersection\_new(toline(.Curve), space\_L\ rect2region)))]\ count \# 0]\ count$

Query Space_M: $query\ MatchedTrips\ feed\ addid\ filter[.TID <$
70000] $filter[.Matched\ afeed\ filter[not(isempty($
$intersection\_new(toline(.Curve), space\_M\ rect2region)))]\ count \# 0]\ count$

Query Space_S: $query\ MatchedTrips\ feed\ addid\ filter[.TID <$
70000] $filter[.Matched\ afeed\ filter[not(isempty($
$intersection\_new(toline(.Curve), space\_S\ rect2region)))]\ count \# 0]\ count$

- Consultas Espacio-Temporales
  - Modelo Free Space:

    Query Time_Space_L: $query\ DayTrips\ feed\ addid\ filter[.TID <$
    70000] $extend[N : no\_components(.DayTrip\ atperiods\ time\_L)]\ filter[.N \# 0]$
    $extend[M : never(.DayTrip\ inside\ space\_L\ rect2region)]$
    $filter[.M\ \#\ TRUE]\ count$

    Query Time_Space_M: $query\ DayTrips\ feed\ addid\ filter[.TID <$
    70000] $extend[N : no\_components(.DayTrip\ atperiods\ time\_M)]\ filter[.N \# 0]$
    $extend[M : never(.DayTrip\ inside\ space\_M\ rect2region)]$
    $filter[.M\ \#\ TRUE]\ count$

    Query Time_Space_S: $query\ DayTrips\ feed\ addid\ filter[.TID <$
    70000] $extend[N : no\_components(.DayTrip\ atperiods\ time\_S)]\ filter[.N \# 0]$
    $extend[M : never(.DayTrip\ inside\ space\_S\ rect2region)]$
    $filter[.M\ \#\ TRUE]\ count$

  - Modelo Network:

    Query Time_Space_L: $query\ Matched\ \ feed\ addid\ filter[.TID <$
    70000] $extend[N : no\_components(.Matched\ atperiods\ time\_L)]$
    $filter[.N \# 0]\ filter[rectproject(mgpbbox(.Matched), 1, 2)\ intersects\ space\_L]$
    $filter[\ no\_components(mgpoint2mpoint(.Matched)\ inside\ space\_L\ rect2region\ )$
    $\# 0]\ count$

    Query Time_Space_M: $query\ Matched\ \ feed\ addid\ filter[.TID <$
    70000] $extend[N : no\_components(.Matched\ atperiods\ time\_M)]$

$filter[.N \# 0]$ $filter[rectproject(mgpbbox(.Matched), 1, 2)$ $intersects$ $space\_M]$ $filter[$ $no\_components(mgpoint2mpoint(.Matched)$ $inside$ $space\_M$ $rect2region$) $\# 0]$ $count$

Query Time_Space_S: $query$ $Matched$ $feed$ $addid$ $filter[.TID <$ 70000] $extend[N :$ $no\_components(.Matched$ $atperiods$ $time\_S)]$ $filter[.N \# 0]$ $filter[rectproject(mgpbbox(.Matched), 1, 2)$ $intersects$ $space\_S]$ $filter[$ $no\_components(mgpoint2mpoint(.Matched)$ $inside$ $space\_S$ $rect2region$) $\# 0]$ $count$

- Modelo OR:

Query Time_Space_L: $query$ $MatchedTrips$ $feed$ $addid$ $filter[.TID <$ 70000] $extend[I : .Matched$ $afeed$ $filter[.StartTime >$ $begin\_time\_L]$ $filter[.EndTime <$ $begin\_time\_L + duration\_1d]$ $count$] $filter[.I \# 0]$ $filter[.Matched$ $afeed$ $filter[not(isempty(intersection\_new(toline(.Curve),$ $space\_L$ $rect2region)))]$ $count \# 0]$ $count$

Query Time_Space_M: $query$ $MatchedTrips$ $feed$ $addid$ $filter[.TID <$ 70000] $extend[I : .Matched$ $afeed$ $filter[.StartTime >$ $begin\_time\_M]$ $filter[.EndTime <$ $begin\_time\_M + duration\_3h]$ $count$] $filter[.I \# 0]$ $filter[.Matched$ $afeed$ $filter[not(isempty(intersection\_new(toline(.Curve),$ $space\_M$ $rect2region)))]$ $count \# 0]$ $count$

Query Time_Space_S: $query$ $MatchedTrips$ $feed$ $addid$ $filter[.TID <$ 70000] $extend[I : .Matched$ $afeed$ $filter[.StartTime >$ $begin\_time\_S]$ $filter[.EndTime <$ $begin\_time\_S + duration\_30m]$ $count$] $filter[.I \# 0]$ $filter[.Matched$ $afeed$ $filter[not(isempty(intersection\_new(toline(.Curve),$ $space\_S$ $rect2region)))]$ $count \# 0]$ $count$