Miembros: Martin Flores y Matías Fernández
Repositorio GIT: https://github.com/mfloresuy/VecDefTEA

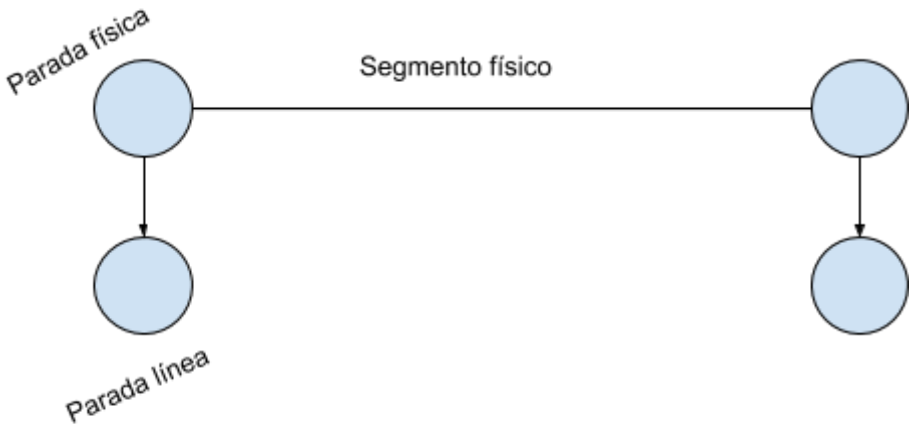
Resumen de diseño de algoritmo para TEA

Para la implementacion del desafio TEA nos basamos el reporte de proyecto de David Nissimoff que se adjunta en el siguiente link http://cs229.stanford.edu/proj2016/report/Nissimoff-RealTimeLearningAndPredictionOfPublicTransitBusArrivalTimes-report.pdf

Basados en este proyecto nuestro algoritmo se resume en tres etapas.

Etapa 1: Armado del grafo de segmentos.

Para el armado de este grafo se utiliza el servicio de trayectos provisto por la máquina virtual de la letra del desafío. En este proceso se crean y cargan tres entidades. Las pardas físicas, las paradas de línea, y el segmento físico.



De esta manera se crea un grafo de paradas físicas unido por segmentos físicos que conectan paradas como muestra la imagen. Las paradas físicas tienen varias paradas de líneas que son la representación de la parada para una línea particular.

Cada segmento guardará tiempo estimado de recorrido (calculado en Etapa 2). De la parada física se guarda la posición X e Y ya que como en el estudio de David Nissimoff, la longitud, latitud se proyectan en un par de ejes cartesianos X,Y. De esta manera distintas líneas pueden compartir el tiempo estimado de los segmentos que comparten.

La parada línea guardará la línea a la que pertenece la parada y el ordinal, por lo que una lista de parada líneas ordenadas por ordinal en una línea definirán un trayecto sobre el grafo físico.

Etapa 2: Suscripción Orion y lectura de datos.

Con la documentación provista de Orión, se conecta la aplicación para recibir las lecturas de los buses.

Para cada lectura recibida del Orion se guarda el bus y su posición.
Para determinar en qué segmento físico de su trayecto se encuentra se busca la parada física más cercana al bus de su línea, luego de los dos segmentos conectados a la parada se elige el más cercano. De esta manera siempre se tiene ubicado a un bus a un segmento físico.

En cada segmento se va guardando cuánto tiempo le llevó a los buses pasar por el. Cuando se detecta que un bus cambió de segmento se registra el tiempo que le tomó. Además se va calculando el promedio del tiempo y un promedio móvil ponderado del tiempo (utilizado como dato principal de la respuesta de la solución). Para calcular el TEA usamos la suma de los promedios ponderados desde el bus hasta la parada.

Estos datos se utilizarán en la Etapa 3.

Etapa 3: Consulta de TEA

Al invocar el servicio se busca la parada solicitada y luego se recorre todo el trayecto en sentido inverso hasta encontrar el primer bus en el trayecto. Al encontrar el bus, se sabe en qué segmento está por la Etapa 2, luego se suman todos los promedios ponderados desde donde se encuentra el bus hasta la parada y se devuelve este resultado como respuesta.

Ejecución y configuración de la solución entregada

La solución tiene tres variables de entorno a configurar, la variable donde se encuentra el servicio de trayectos **TRAYECTOS_URL**, la variable de configuración de Orion requerida para la suscripción **ORION_URL**, y la variable de callback donde el Orion invocara para enviar las lecturas de buses **ORION_CALLBACK_URL**.

Ejemplo de configuración local de variables y valores:

TRAYECTOS_URL=http://192.168.1.46/api/trayectosporlinea
ORION_URL=http://192.168.1.46:1026/v2/subscriptions
ORION_CALLBACK_URL=http://192.168.1.48:8080/bus_evento

La solución está implementada en SpringBoot, compatible con JAX-RS y con el servidor web embebido spring-boot-starter-jersey para ser más liviano, si es requerido utilizar tomcat 8 puede cambiarlo en el pom.xml por spring-boot-starter-tomcat.

La base de datos de la solución es una base de datos en memoria persistible a archivo H2 para simplificar configuración y tener soporte de datos simple.

A continuación se deja un procedimiento para descargar y ejecutar la solución luego de configuradas las variables de entorno.

git clone GIT_URL
cd VecDefTEA
mvn package
VecDefTEA
java -jar target/vecdefTEA-0.0.1-SNAPSHOT.jar

Si todo esta correcto debería ver en terminal el servidor levantado con un mensaje parecido a este ***Started VecdefTeaApplication in 3.851 seconds (JVM running for 4.395)***

Comentarios de la solución:

Durante la ejecución de la simulación de entrenamiento, aparecieron algunos buses que no tenían líneas asociadas en el servicio de trayectos. No se consideran las lecturas de estos buses en al solución ya que al momento de diseño del algoritmo se supuso que no había buses que no tuvieran línea asociada sobre los datos provistos.

Al ejecutar la solución con otra simulación puede ver en log líneas como ***"No se encontraron paradas para la línea"*** , en caso que se de la misma situación.

También se vieron lecturas de buses donde cambiaba la posición pero no el timestamp enviado, al utilizara este timestamp para determinar la duración del bus en el segmento, estas lecturas que no varían la duración en el segmento también son descartadas.

En cuanto al cálculo se dan dos medidas, la fundada en promedio, y el promedio ponderado que va en el valor requerido por el documento de planteo del problema. Se dan dos ya que el punto donde se calcula esto en base al histórico de lectura es fácilmente configurable y parecía interesante tener dos TEA.

El algoritmo puede ser mejorado, aumentando la granularidad de los segmentos, y si las lecturas son de mayor cantidad y no cada 15 segundos como se dio en el simulador provisto, esto sería una manera fácil de aumentar la precisión de la solución.

También hay mejoras en el cálculo de los tiempos, descartando lecturas que parecen de trayectos fuera de la línea, tomar radio de algunas cuadras alrededor de las paradas del trayecto puede ser una fácil solución de descarte de lecturas para determinar situaciones anómalas como ir al taller.

Tambien tendria sentido purgar los datos, descartando datos viejos después de tanto tiempo, ya que mientras más viejo se va volviendo el dato, menos peso tiene en el cálculo del promedio ponderado, además de que va aumentando el tiempo de cálculo por lectura de bus al aumentar la cantidad de datos.

Nos parece importante destacar que lo interesante es que se guarda un histórico de los tiempos agrupados por segmentos. Esto hace que todas las líneas que pasen por un mismo segmento aporten su información y la compartan entre ellas. A partir de este histórico compartido entre líneas se podría elegir otra forma de calcular un estimador de tiempo para cada segmento.