Prototipo de Sistema de Administración y Planificación Automática de Rutas Óptimas para Expresos Escolares de Instituciones Educativas

Cueva, E.[1], Vaca C.[2]
[1-2] Facultad de Ingeniería en Electricidad y Computación
Escuela Superior Politécnica del Litoral - ESPOL,
Campus Gustavo Galindo, Km. 30.5 Vía Perimetral.
Apartado 09-01-5863. Guayaquil, Ecuador
[1] ecueva@fiec.espol.edu.ec
[2] cvaca@fiec.espol.edu.ec

Resumen

El presente proyectodescribe la implementación de un prototipo de aplicación web para la administración y planificación de rutas óptimas para expresos escolares de instituciones educativas. Una aplicación de este tipo permite disminuir sustancialmente el tiempo requerido para la planificación de rutas escolares, a la vez que permite visualizar las rutas generadas.

En el prototipo se generan rutas de menor distancia sin considerar el sentido de las calles. Debido a que no se cuenta con información geocodificada de toda la ciudad de Guayaquil se utiliza para la implementación un área geográfica determinada, para nuestro caso el centro de la ciudad.

En la generación de rutas de menor distancia se utiliza una implementación del algoritmo de TSP, este algoritmo utiliza métodos Greedy con la heurística del vecino próximo.

Para dibujar la ruta se utiliza la implementación del algoritmo A-star, la cual permite definir que intersecciones debe visitar para ir de un punto a otro de acuerdo con el orden de visita dado por el algoritmo TSP.

Palabras Claves: Problema del Viajante, Dijkstra, A-star,*API's d*e Google Maps.

Abstract

This project describes the implementation of a web application prototype for the management and planning of optimal routes to express school educational institutions. One application of this type can substantially reduce the time required for route school planning and at the same time it will allow us to visualize the generated routes.

Using the prototype shorter distance paths are generated without considering the direction of the streets. Because there is no geocoded data for the city of Guayaquil, a predefined geographical area was used, this is the center of the city.

TSP algorithm implementation is used for the generation of shorter distance routes, this algorithm uses greedy methods applying the heuristic of close neighbor.

To draw the generated routes is used A-Start algorithm implementation, that allows you to define which intersections should visit to get from one point to another in accordance with the visitation order given by TSP.

Keywords: Traveling Salesman Problem, Dijkstra, A-star, *API's Google Maps.*

1.Introducción

En la actualidad se ha popularizado el uso de aplicaciones web para la ubicación de direcciones domiciliarias, gigantes de la computación como Google y Yahoo proveen API's para manejo de mapas de forma gratuita. Estas API's pueden ser utilizadas para mostrar rutas a seguir para ir de un punto a otro, una funcionalidad muy útil para cierto tipo de aplicaciones que requieren dibujar rutas por ejemplo: rutas escolares, rutas de entrega de comida, rutas entrega de correo, etc.

Existe, sin embargo, una limitación a los API's mencionados anteriormente: la información de generación de rutas no está disponible para muchos sectores geográficos en nuestro país. En este documento se presenta un prototipo que permite utilizar los mapas provistos por Google en conjunto con una base de datos de direcciones geocodificadas para generar rutas escolares en el sector céntrico de la ciudad de Guayaquil, ese prototipo puede constituir una base para explotar el gran potencial de estas aplicaciones.

Para un sistema de generación y manejo de rutas de expreso escolar, la visualización de las rutas es una parte importante de este tipo de aplicación, pero también es fundamental el uso de algoritmos que permitan obtener rutas optimizadas. En el desarrollo del prototipo se utiliza una implementación del algoritmo TSP que permite obtener las rutas de menor distancia.

2. Sistema de Planeación de Rutas

En toda institución cuya ubicación geográfica es relativamente apartada de los lugares donde se concentra la mayor cantidad de población, se vuelve imprescindible en un momento dado contar con unidades de transporte para dar servicio a empleados o a clientes.

El trabajo de distribuir a personal, clientes, estudiantes en una serie de rutas se vuelve complicado cuando el número de los mismos aumenta y no se tiene un sistema que automatice el proceso de diseño de rutas. Así pues una distribución manual puede llevar a resultados poco eficientes con rutas extremadamente largas y poco equilibradas.Un sistema de planeación de rutas permite automatizar el proceso de diseño de rutas de transporte.

2.1. Problema del camino más corto

Los seres humanos por naturaleza minimizan esfuerzo, sobre todo cuando se trata de moverse. Cuando tienen la oportunidad, siempre tratarán de elegir el camino más corto para ir de un lugar a otro.

El transporte, como una actividad económica, reproduce este proceso de minimización, en particular tratando de reducir al mínimo la distancia entre localidades. Estrategias para obtener rutas con tiempos más cortos y costos más bajos son constantemente analizadas por individuos y empresas. Para un individuo, a menudo es sólo una cuestión de conveniencia, pero para una sociedad es de importancia estratégica como un costo monetario directo. En tales circunstancias, no es sorprendente que numerosos métodos hayan sido desarrollados para hacer frente a la compleja cuestión de selección de rutas. Uno de esos clásicos es el problema del vendedor viajero, cuando el camino más corto tiene que ser seleccionado de un conjunto de numerosas combinaciones de posibles trayectorias.

En teoría de grafos, el problema del camino más corto es el problema de encontrar un camino entre dos nodos, de tal manera que la suma de los costos de sus enlaces sea mínima; siendo los enlaces el segmento que une los nodos y el costo de cada enlace es el número que representa tal enlace [3].

El problema del camino más corto se lo puede clasificar en: el camino más corto desde un origen a un destino y el camino más corto desde un origen a muchos destinos.

En secciones posteriores se describen los algoritmos de ordenamiento (TSP – Traveling Salesman Problem) y de distancias mínimas (A-star), los cuales permiten obtener rutas optimizadas, es decir rutas en las cuales el camino es el más corto.

En nuestro caso se trata de generar rutas optimizadas de tal manera que se llegue por el camino más corto.de una dirección domiciliaria a otra, en este caso, los nodos representan las direcciones domiciliarias y los enlaces representan la distancia y se ponderan por la distancia existente para ir de un punto a otro.

2.2. Geocodificacción de Direcciones Domiciliarias

La geocodificación de una dirección domiciliaria consiste en tomar la información dada usando zonas y nombres de calles y representarla como un punto geográfico expresado en unidades de latitud y longitud.

Existen dos métodos para geocodificar direcciones domiciliarias, de los cuales se ha elegido el API de Google Maps.

2.2.1. Uso del API de Google Maps: Google Maps provee un API que permite incluir mapas en una página web utilizando el lenguaje de programación JavaScript. Es posible añadir contenido al mapa a través de una variedad de servicios, lo que permite crear aplicaciones robustas de un sitio web.

Google Maps permite usar el API de forma gratuita, siempre y cuando el sitio web que lo utilice ofrezca servicios sin costo para los consumidores. Google Maps ofrece una serie de utilidades para la manipulación de mapas (desplazamiento en el mapa, zoom, etc.), mapas del mundo entero, fotos satelitales, la ruta más corta entre diferentes ubicaciones y muchas características interesantes. Google Maps es fácilmente integrable con cualquier sitio web.

Para hacer uso del API es necesario completar el registro, luego de lo cual se obtiene una clave necesaria para utilizar los scripts que mostrarán los mapas en las páginas web. La guía para desarrolladores y las referencias del API están publicadas en el web y pueden ser obtenidas de forma gratuita[1].

2.3. Problema del Viajante (Traveling Salesman Problem)

---

[1] http://code.google.com/apis/maps/

2.3.1. Introducción: El problema del viajante consiste en encontrar una ruta de un número dado de nodos, visitando cada nodo exactamente una vez y retornando al nodo de partida donde la longitud de esta ruta es minimizada.

El problema del vendedor viajante puede ser declarado matemáticamente como sigue:

Dado un grafo de costos G = (V, E) donde el costo C_{ij} sobre el enlace entre los nodos i y j es un valor no negativo, encontrar la ruta de todos los nodos que tenga el mínimo costo total.

El problema puede ser convenientemente modelado por un grafo ponderado, con los nodos del grafo representando las ciudades y los costos de los enlaces especificando las distancias. Entonces el problema puede ser declarado como el problema de encontrar el circuito Hamiltoniano más corto del grafo [1].

2.3.2. Aplicaciones: El problema del viajante tiene diferentes aplicaciones en el mundo real, haciendo de ello un problema muy interesante de resolver. Por ejemplo, algunas instancias del problema de ruteo de vehículos puede ser modelado como un problema del viajante (bus escolar, atención de llamadas de emergencia, servicio de correo expreso). Para este ejemplo el problema es encontrar cual cliente debe ser visitado por cual vehículo y el número mínimo de vehículos necesitados para visitar a cada cliente. Hay diferentes variaciones de este problema incluyendo encontrar el tiempo mínimo para visitar a todos los clientes.

2.3.2.1. Métodos para Resolver el TSP: Homaifar [2] afirma: “un enfoque que encontraría la solución de cualquier TSP es la aplicación de enumeración exhaustiva y evaluación. El procedimiento consiste en generar todas las posibles rutas y evaluar las longitudes correspondientes. La ruta con la longitud más pequeña es seleccionada como la mejor, la cual es garantizada a ser óptima”.

Debido al tiempo que se necesita para procesar todas las rutas, se hace necesario encontrar un algoritmo que nos dé una solución en un período de tiempo más corto. El problema del viajante es un NP (no polinomial)-duro, lo cual significa que no hay un algoritmo conocido para resolver este problema en tiempo polinomial. De esta manera probablemente se tiene que sacrificar optimización en orden a lograr una buena respuesta en un tiempo más corto. Muchos algoritmos han sido tratados para el problema del viajante, entre ellos:

Algoritmos Greedy

El algoritmo Greedy da soluciones factibles, sin embargo no son siempre buenas. El enfoque Greedy sugiere construir una solución a través de una secuencia de pasos, cada expansión una solución parcial construida hasta el momento, hasta que una solución completa del problema es alcanzada. En cada paso, y este es el punto central de esta técnica, la elección hecha debe ser: factible, localmente óptima e irrevocable.

Existen varios algoritmos Greedy propuestos para resolver TSP, entre ellos tenemos: el algoritmo del vecino más próximo y el algoritmo del árbol de expansión mínimo. En nuestro caso el prototipo desarrollado utiliza el algoritmo Greedy con la heurística del vecino más próximo.

2.4. Algoritmos de Rutas Mínimas

Para encontrar rutas mínimas es necesario calcular el camino más corto desde un nodo de origen a un nodo destino. Existen dos algoritmos importantes para el camino más corto entre dos nodos: Dijkstra y A-star. En nuestro caso el prototipo desarrollado utiliza el algoritmo de rutas mínimas A-star, el cual a continuación se describe.

2.4.1. A-star: A * (pronunciado como “A star”) es uno de los mejores algoritmos de búsqueda de grafos que encuentra el camino de menor costo de un nodo inicial a un nodo final (de uno o más objetivos posibles).

Procedimiento A*

1. Inicialice Q a ser el conjunto conteniendo el par simple (s, 0).
2. S ← 0.
3. Mientras Q no es vacío y S no contiene una asignación a g:
4. Remover un par (n, w) de Q minimizando w + h(n) (sobre todos los elementos de Q).
5. Si S no contiene un costo para n, por ejemplo, si S no contiene cualquier par de la forma (n, w’), entonces:
6. Añada (n, w) a S.
7. Para cada enlace (n, m) con costo w’ en el grafo G añada el par (m, w + w’) a Q.
8. Si S contiene una asignación a g entonces termina con éxito (una ruta ha sido encontrada) sinotermina con fracaso (no hay ruta).

3. Diseño del Prototipo

3.1. Arquitectura del Sistema

Las herramientas utilizadas para desarrollar la aplicación web son las siguientes:

- Visual JSF.
- Java Server Pages (JSP).
- Netbeans.

- Base de Datos MySql.
- JavaScript.

Para implementar cada una de las funciones del aplicativo web y administrar los datos requeridos de una manera organizada, adecuada y eficiente se ha procedido a crear módulos que a la vez permiten obtener un aplicativo de fácil mantenimiento. Los módulos implementados del aplicativo web son:

- Módulo de geocodificación de direcciones.
- Módulo de ingreso de alumnas.
- Módulo de asignación de direcciones.
- Módulo de generación de rutas.
- Módulo de mantenimiento de rutas.
- Módulo de mantenimiento de datos.
- Módulo de reportes.

A continuación se describen dos de estos módulos.

Módulo de Geocodificación de Direcciones.- En este módulo se realiza la geocodificación de las intersecciones de las calles del área geográfica indicada como alcance en la implementación del prototipo.

Para lograr obtener los datos de las intersecciones de calles en términos de latitud y longitud se utilizan las funciones de geocodificación que nos facilita el API de Google Maps. La obtención de los datos geocodificados de las intersecciones de las calles del área geográfica a considerar se realiza una sola vez, para de esta manera contar con el repositorio de datos geográficos necesarios para la obtención de rutas de transporte escolares óptimos.

Módulo de Generación de Rutas - Este módulo tiene las funciones que detallamos a continuación:

- Ordenamiento - Este módulo se encarga de colocar el orden de visita de los puntos geográficos a ser visitados. Este módulo es utilizado tanto para el ordenamiento inicial en la generación de rutas como para el ordenamiento de cada una de las rutas generadas. El primer ordenamiento va a permitir obtener rutas de distancias mínimas es decir rutas balanceadas. El segundo ordenamiento nos permite de igual manera hacer que cada una de las rutas generada sea óptima.
- Determinación de Menor Distancia - Este módulo se encarga de encontrar cuáles son las intersecciones de calles a ser visitados de tal manera que se visite cada uno de los puntos que conforman la ruta en el orden asignado por el módulo de ordenamiento, es decir que encuentra el camino más corto para visitar todos los puntos de la ruta. En la obtención de estas intersecciones no se considera el sentido de las calles debido a que no se tiene esta información en formato digital.
- Visualización de la Ruta - Este módulo nos permite visualizar cada una de las rutas generadas. Para lo cual hace uso de los puntos devueltos por la función que calcula la ruta de menor distancia. Este módulo emplea las funciones del API de Google Maps para mostrar la ruta.

3.2. Diseño de Pruebas

3.2.1. Asignación Correcta de Nodos a Rutas: Para verificar el grado de certeza de la generación de rutas se realiza una prueba llevando a cabo los siguientes pasos:

1.) Se generan las rutas, se visualizan
2) Luego se agrega un nodo nuevo cuya ubicación sea obvia en una de las rutas obtenidas.
3) Se generan rutas nuevamente para verificar que el nodo haya sido asignado de forma correcta.

3.2.2. Medición de Tiempo Usando TSP: Es importante determinar el orden de tiempo requerido para generar rutas según el número de nodos a utilizar. Para esta prueba se corre el algoritmo de generación de rutas usando la técnica Greedy en 3 escenarios que difieren solamente en el número de nodos. Al final de cada corrida se determina el tiempo empleado en la selección y ordenamiento de nodos.

3.2.3. **Medición de Tiempo Usando A-Star:** Se mide el tiempo para determinar las intersecciones que se deben visitar para ir de un domicilio a otro dentro de una ruta.

4. Implementación delPrototipo

4.1. Algoritmo para Determinar Orden de Visita de los Nodos

TSP es el algoritmo utilizado para determinar en qué orden se visita cada uno de los nodos que conforman las rutas.La técnica que el algoritmo TSP utiliza es la técnica Greedy con la heurística del vecino más próximo.

El Costo de un punto a otro está dado por la distancia que existe entre ellos en línea recta.

El proceso de ordenamiento empieza en el punto de llegada/partida que en nuestro caso es la dirección geográfica de la institución educativa. A partir de ahí se busca el siguiente punto a visitar seleccionando aquel que represente el de menor costo. Este proceso descrito se continúa hasta que se evalúen todos los nodos del grupo.

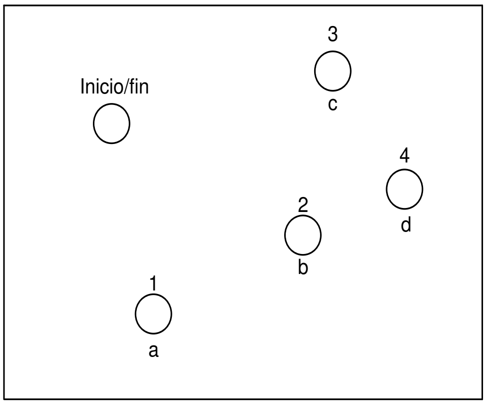

Figura 4.1: Ordenamiento de nodos usando TSP.

4.2. Determinar Distancias Mínimas entre Nodos

De acuerdo con lo indicado en el diseño, para el módulo de visualización de rutas generadas se hace uso de la implementación del algoritmo A Star que permite obtener la distancia mínima entre dos nodos. Para determinar la distancia mínima entre dos puntos:

- Se visitan todos los nodos vecinos de un punto y se determina aquel cuyo costo sea menor.
- Se agrega este punto a la ruta y se verifica si se ha llegado a la meta.
- Si aún no se llega a la meta se continúa con el nodo siguiente.
- En el algoritmo se utilizan estructuras de datos adecuadas para el manejo de los nodos y sus respectivos costos.
- Para obtener los nodos que conforman la distancia o ruta mínima del punto inicio al punto meta se crea una lista en el que se añadirá como primer elemento el nodo meta y de reverso obtendremos el nodo previo empezando en el nodo meta, estos nodos que se obtiene se almacenan en la lista creada que contendrá la ruta más corta de un punto a otro.

En la figura 4.2, se ilustra cómo se determina la distancia o ruta mínima entre dos puntos.

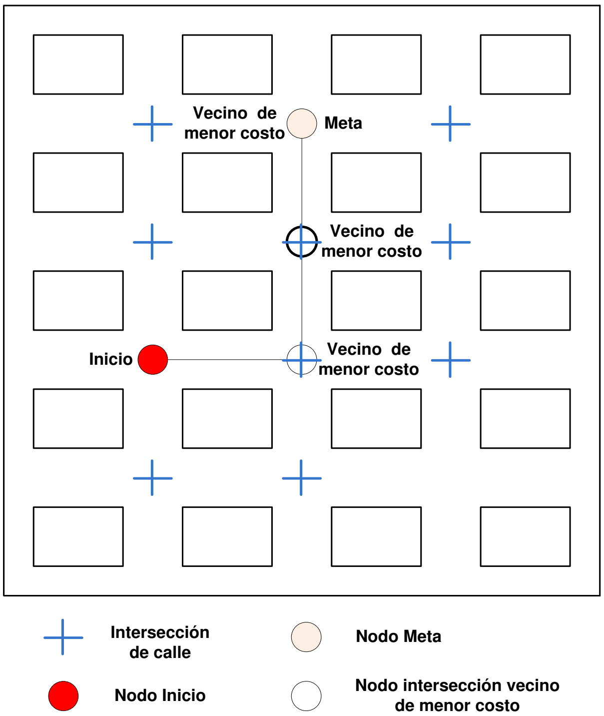

Figura 4.2: Distancia mínima de un punto a otro.

Lo anteriormente descrito corresponde a determinar la distancia mínima de un punto a otro. Para poder determinar la distancia mínima de una determinada ruta, se procede a aplicar la implementación de distancia mínima de un punto a otro varia veces hasta que se consideren todos los puntos que conforman la ruta, el orden en que van tomando los puntos es de acuerdo con el ordenamiento obtenido a través de la implementación del algoritmo de TSP en sentido descendente.

4.3. Visualización de la Ruta Generada en Google Maps.

A través del módulo de generación de rutas se obtienen las diferentes rutas requeridas, en este módulo primeramente se ordenan todos los puntos que serán considerados en la generación para luego proceder a generar grupos de acuerdo con la capacidad de bus que ha sido definido como un valor constante. Las intersecciones que conforman las distancias mínimas de un punto a otro son almacenadas en la base.

Al acceder a la interfaz de consulta de rutas se leerá de la base todas las intersecciones que conforman la distancia mínima de una ruta las cuales han sido previamente almacenadas en la base, al leerlos se los mantiene ordenados por el orden registrado en un campo de la tabla.

Una vez que los puntos se han cargado en una colección en memoria, se recorre la misma para dibujar fragmentos de línea que en su conjunto representan la ruta.

En la figura 4.3 se muestra la visualización de una ruta de transporte escolar.

Figura 4.3: Visualización de ruta generada usando funciones del API's de Google Maps.

5. Pruebas y Análisis de Resultados

5.1. Asignación Correcta de Nodos a Rutas Generadas

Los pasos para llevar a cabo esta prueba son los siguientes:

a) Generar y visualizar rutas.
b) Añadir punto.
c) Volver a generar y visualizar rutas.

Para la implementación de esta prueba se ha considerado un set de datos de 22 puntos geográficos que corresponden a las direcciones domiciliarias de las alumnas, las cuales a su vez se relacionan con los datos geográficos de la intersección de calle más próxima a la dirección domiciliaria de la alumna.

Como conclusión de esta prueba se puede decir que la agrupación de los puntos es realizado de acuerdo a lo que se esperaba, es decir considerando la cercanía de los mismos. Cuando se añade un punto, las rutas son reagrupadasy el nuevo punto es colocado en la ruta más cercana al punto.

5.2. Medición de Tiempo Usando TSP

Para la medición del tiempo de repuesta del algoritmo de ordenamiento de nodos TSP que nos permite generar rutas óptimas, se ha procedido a realizar pruebas variando de manera progresiva el número de nodos. En la tabla 5.1 se describen los resultados obtenidos y en la figura 5.9 se muestra la relación de tiempo vs número de nodos. Como se puede observar el tiempo de respuesta aumenta rápidamente para un conjunto pequeño de nodos, pero a partir de los 10 nodos la curva se suaviza.

Tabla 5.1: Resultados de tiempos TSP con diferente número de puntos.

Número de Puntos	Tiempo (Segs.)
5	0,43
10	2,39
30	3,28

Figura 5.1: Relación número de nodos Vs. tiempo para el algoritmo TSP.

5.3. Medición de Tiempo Usando A-Star

Para medir el tiempo de respuesta del algoritmo de rutas mínimas A-star, se ha procedido de la misma manera que para el algoritmo TSP, se han realizadopruebas aumentando el número de nodos de manera progresiva.

En la tabla 5.2 se muestra los resultados obtenidos para el tiempo de respuesta para diferente número de nodos. También se muestra un gráfico que define la relación de número de nodos vs. tiempo. Como se puede observar en la tabla el tiempo de respuesta del algoritmo de distancias mínimas aumenta rápidamente para un conjunto pequeño de nodos, la curva tiende a suavizarse a partir de los 10 nodos.

Tabla 5.2: Resultados de tiempos A-Star con diferente número de puntos

Número de Puntos	Tiempo (Segs.)
5	0,38
10	1,31
30	1,81

.

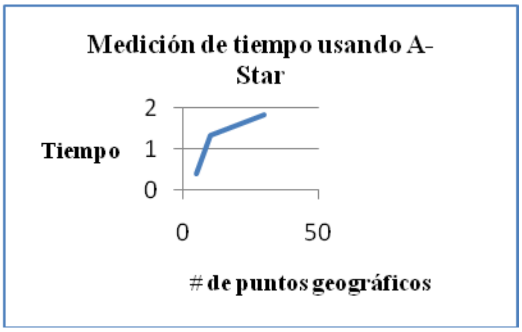

Figura 5.2: Relación número de nodos Vs. tiempo para el algoritmo A-Star.

6. CONCLUSIONES

Una vez finalizada la implementación y realizadas las pruebas se pudo concluir que:

- El API de Google Maps facilitó la creación del repositorio de datos de direcciones domiciliarias geocodificados para el área geográfica definida en el alcance, y utilizando implementaciones previamente propuestas para algoritmos de rutas mínimas fue posible construir un aplicativo web que permite obtener rutas de expreso escolar óptimas.

- El algoritmo TSP utilizado para el agrupamiento de puntos en rutas presentó resultados en los cuales se observa que un punto geográfico que se agrega a un set de datos de direcciones geocodificadas es colocado en la ruta que contiene los puntos más cercanos al mismo.

- El algoritmo A-Star permitió obtener las distancias mínimas de un punto a otro, este algoritmo se aplica para cada punto de la ruta de acuerdo con el orden de visita, los cuales son utilizados luego para graficar estos segmentos de líneas que en su conjunto representan la ruta. Los resultados obtenidos para este algoritmo si representan la distancia mínima de un punto a otro.

- La elección de un área geográfica definida para implementar el prototipo fue de gran importancia pues no habría sido posible crear un repositorio de datos de direcciones geocodificadas para toda la ciudad de Guayaquil con los recursos de que se disponían.

- Para implementar una aplicación se requiere de mucha más información geográfica tales como: sentido de las calles e información domiciliaria geocodificada de un área más amplia. La obtención de esta información geocodificada para esta área más amplia representaría una inversión de tiempo significativo. En el caso de tener un área geográfica de alcance mayor, los algoritmos de generación de rutas considerados en el presente trabajo podrían ser aplicados ya que los tiempos de respuestas de acuerdo a lo mencionado son aceptables para una aplicación.

7. Referencias

[1] AnanyLevitin, Introduction to the Design & Analysis of Algorithms,Pearson Education, Agosto 2002.

[2] AbdollahHomaifar, Shanguchuan Guan, and Gunar E. Liepins, Schema Analysis of the traveling salesman problem using genetic algorithms. SociedadMatemática Americana, 1992.

[3] Pilar Bravo, Juan Carlos Ferrando y Ana Martínez, Complementos de Matemática Discreta Curso Práctico, Universidad Politécnica de Valencia - Servicio de Publicaciones, 1994.