Universidad Autónoma de Querétaro
Facultad de Informática

Optimización de Rutas en Redes de Tráfico Vehicular

Tesis

Que como parte de los requisitos para obtener el grado de

Maestro en Ciencias Computacionales

Presenta

Héctor Manuel Huerta Jiménez

Universidad Autónoma de Querétaro
Facultad de Informática
Maestría en Ciencias Computacionales

Optimización de Rutas en Redes de Tráfico Vehicular

TESIS

Que como parte de los requisitos para obtener el grado de
Maestro en Ciencias Computacionales

Presenta:
Héctor Manuel Huerta Jiménez

Dirigido por:
Dr. Arturo González Gutiérrez

SINODALES

Dr. Arturo González Gutiérrez
Presidente — Firma

Dr. Jaime Rangel Mondragón
Secretario — Firma

M.C. Fidel González Gutiérrez
Vocal — Firma

M.C. Ma. Elena Vázquez Huerta
Suplente — Firma

M.C. Guillermo Diaz Delgado
Suplente — Firma

M.C. Ruth Angélica Rico Hernández
Directora de Informática

Dr. Luis Gerardo Hernández Sandoval
Director de Investigación y Posgrado

Centro Universitario
Querétaro, Qro.
Marzo 2010
México

RESUMEN

En esta tesis abordamos varios problemas de optimización de redes, en particular redes de tráfico vehicular. Se planteo el problema de consulta de planeación de rutas (TPQ por sus siglas en inglés que se refiere al *Trip Planning Query*) analizando e implementando los algoritmos *greedy* del vecino más cercano, mínima distancia y los algoritmos basados en métodos de relajación y restricción. También se analizó el grado de utilización y flujo vehicular de las avenidas del primer cuadro de la ciudad de Querétaro. Para ello se modeló la red de tráfico vehicular mediante un grafo con distancias y posiciones (vértices y aristas).Tales datos provienen de un sistema de información geográfica que provee de datos en coordenadas geográficas UTM (Universal Transversal the Mercator), especificando la distancia entre puntos en metros. El grafo consiste en 412 vértices y 736 aristas, las cuales modelan las avenidas del cuadro principal de Querétaro, delimitado por las avenidas 5 de febrero, Universidad, Constituyentes y Corregidora. Utilizando dicho modelo y funciones especializadas del lenguaje funcional de alto nivel “Mathematica”, se llevaron a cabo evaluaciones experimentales de algunos algoritmos importantes de optimización de rutas así como estudios estructurados de la red de tráfico vehicular a través del grafo dirigido construido. Finalmente, con el propósito de rediseñar la red de tráfico vehicular para el caso de estudio y obtener una red con propiedades estructurales que permitan un mejor funcionamiento del tráfico vehicular, se utilizó la técnica de búsqueda local. Esta técnica permite detectar estados del grafo en términos de redireccionamiento de calles o avenidas que conducen a mejorar la utilización de las mismas.

(Palabras clave: TPQ, Programación lineal, Algoritmos *Greedy*)

ABSTRACT

In this thesis we address several network optimization problems, including vehicular traffic networks. Raised the problem of the Trip Planning Queries(TPQ), analyzing and implementating the greedy algorithm of the nearest neighbor, the minimum distance algorithms, and algorithms based on methods of relaxation and restriction. The use and flow of networks in the main streets in Queretaro was analyzed as well. This network of vehicular traffic was modeled through a graph with distances and positions (vertices, edges). This data comes from a geographic information system that provides acknowledge on geographical coordinates UTM (Universal Transversal the Mercator), specifying the distance between points in meters. The graph is 412 vertices and 736 edges, those shape the avenues of the main square of the city that is limited by the avenues 5 de Febrero, Universidad, Corregidora and Constituyentes. By using the model and specialized functions of functional language of high-level “Mathematica”, were carried out pilot evaluations of algorithms as well as studies structured the network of vehicular traffic through the graph directed built. Finally with the purpose to redesign the network of vehicular traffic for the matter of study and obtain a network better structural properties that this allow a better functioning of vehicular traffic, the local search technique was used this allows states graph detection in terms of redirecting of streets or avenues that lead to improve the use of the ones(techniques).

(Key words: TPQ, Lineal Programming, Greedy Algorithms)

AGRADECIMIENTOS

En particular quiero agradecer al Dr. Arturo González Gutiérrez por su colaboración desinteresada así como el conocimiento y orientación aportada que me fue de gran utilidad para la elaboración de la tesis, así como a mis padres que me apoyaron moralmente en cada momento.

Í N D I C E

ÍNDICE DE FIGURAS

I. INTRODUCCIÓN

El problema de la Consulta de Planeación de Rutas (TPQ, por sus siglas en inglés *Trip Planning Queries*) asume que una Base de Datos Espaciales almacena la localización de objetos en el espacio 3D susceptibles de ser clasificados de acuerdo a las categorías dadas un conjunto C. Este problema consiste en dados un punto inicial S y un punto final E en el espacio; un conjunto de categorías R, donde $R \subseteq C$, el objetivo es encontrar una ruta que inicia en el punto S, visita al menos un punto asociado a cada categoría en R y termina en el punto final E, cuya longitud total se la mínima. Una de las variantes del problema TPQ (Feifei et al, 2006) que se podría manejar sería minimizar el tiempo que toma hacer dicho recorrido, en lugar de simplemente la longitud total. Esto lleva el análisis desde una perspectiva de carácter estructural a una de carácter dinámica, ya que el tiempo requerido para recorrer una ruta depende de la velocidad de desplazamiento y esta a su vez de las condiciones de las calles o avenidas a transitar, además del tráfico vehicular en un instante de tiempo determinado.

El estudio de las Bases de Datos Espaciales, así como las técnicas de indexación, consulta y modelado, han sido de gran interés durante la última década (Vazirgiannis y Wolfson, 2004). Durante ese tiempo se han desarrollado algoritmos de aproximación para la planeación de rutas basados en los criterios del vecino más cercano y de la mínima distancia (Berchtold, Bohm, 1997) utilizando los Árboles-R (Beckmann, Kriegel y Schneider, 1990) como estructura de almacenamiento e indexación de los datos. Esta clase de algoritmos pueden utilizarse en sistemas de navegación avanzada y aplicaciones geográficas, como por ejemplo MapQuest, Google Maps y Microsoft Streets & Trips, las cuales soportan consultas partiendo de un punto inicial y especificando un punto destino. Se puede demostrar que el problema TPQ es equivalente computacionalmente hablando, al problema del Agente Viajero TSP (Arora, 1998; Cormen, Leiserson y Rivest, 1997). Asimismo, se ha demostrado que TSP pertenece a la clase de problemas llamados NP-completos (Garey y Johnson's, 1979). La dificultad del problema TPQ radica en que también

pertenece a la clase de problemas NP-completos, es decir, ya que el problema TPQ es una generalización del problema TSP entonces se puede concluir que el problema TPQ también es NP-completo (Feifei et al, 2006).

En la categoría de problemas NP-Completos, ningún problema puede ser resuelto eficientemente. Esto nos lleva a considerar el uso de algoritmos de aproximación para la solución del problema TPQ.

Considere el problema siguiente. El Profesor Perfecto decide llevar a cabo una serie de encargos en el recorrido de su oficina a su casa durante el tiempo que tiene para comer en un día normal de trabajo. El profesor necesita comprar algunos artículos para su despensa, comprar flores para su prometida, enviar una carta por correo ordinario, disponer de dinero y poner gasolina a su auto. En particular, el Profesor Perfecto podría comprar despensa en una tienda de la esquina, como las de la cadena OXXO, o en una tienda en una plaza comercial; comprar flores en una florería o en una plaza comercial; enviar una carta a través de la oficina de correo en algún punto del Servicio Postal Mexicano o utilizar el Servicio de Mensajería; retirar dinero en algún centro comercial, banco o directamente de un cajero automático; y poner gasolina a su auto en cualquier punto de servicio de la ciudad. Ya que el Profesor Perfecto siempre trata de hacer las cosas lo mejor posible, decide crear un plan de cómo llevar a cabo estos encargos en el menor tiempo posible.

Sea T la ruta que el profesor selecciona para llevar a cabo todos sus encargos minimizando la distancia y por lo tanto el tiempo empleado. Considere la ruta indicada en la Figura 1.1 que inicia en la oficina (punto inicial S) del profesor, después pasa al servicio de mensajería para entregar la carta, sigue la ruta hacia la florería más cercana para comprar un ramo de flores inmediatamente después hace la compra de la despensa en la tienda OXXO, pasa al cajero del banco a retirar dinero, pone gasolina a su auto y finalmente arriba a su casa (punto final E).

T={ S,mensajería,florería,OXXO,banco,gasolinería,E}

Figura 1. 1 Ruta T minimizada por el profesor Perfecto

Podríamos extender el problema planteado por el Profesor Perfecto al problema que consiste en dado un punto inicial y un punto final en la ciudad, una lista de encargos y una base de datos que contenga la localización de todos los puntos de servicio empleada, encontrar el recorrido que minimice la distancia total del viaje empleada en el recorrido. La dificultad de este tipo de consultas radica en que para cada categoría puede existir un número muy grande de posibilidades.

Este problema es referido en la literatura como el Problema de Consultas para la Planeación de Rutas (Feifei et al, 2006). El problema TPQ para grafos métricos asociado a una Base de Datos Espaciales (Egenhofer, 1993) consiste en un grafo conexo G(V,E) con n vértices $V=\{v_1,v_2\ldots v_n\}$ y s aristas $E=\{e_1,e_2,\ldots e_s\}$, donde el costo de recorrer un conjunto de puntos $v_i,\ldots,v_j$ está dado por la función $c(v_i,\ldots,v_j) \geq 0$

Asimismo G es un grafo métrico que satisface las siguientes condiciones:

- $c(v_i,v_j) = 0$ si $v_i = v_j$
- $c(vi,v_j) = c(v_j,v_i)$
- $c(v_i,v_k) + c(v_k,v_j) \geq c(v_i,v_j)$ (desigualdad triangular)

El problema TPQ también considera un conjunto de m categorías $C=\{C_1,...,C_m\}$, donde $m \leq n$ y una función de mapeo $\pi: V_i \rightarrow C_j$ donde cada vértice V_i Є V es mapeado a una categoría C_j Є C. Así, el problema TPQ para grafos métricos asociados a Base de Datos Espaciales puede definirse formalmente como sigue:

ENTRADA: Dado un conjunto $P=\{P_1,P_2,\ldots,P_n\}$ de puntos en el espacio y un conjunto de categorías $C=\{C_1,C_2,...,C_m\}$ (donde cada punto P_i está asociado a un subconjunto de categorías de C), un subconjunto $R=\{C_1,C_2,\ldots,C_k\} \subseteq C$, un vértice inicial S, y un vértice final E.
SALIDA: Una ruta $T=\{S,Vt_1,Vt_2,\ldots,Vt_k,E\}$ de costo mínimo que visita al menos un vértice asociado a cada categoría en R.

En este trabajo se lleva a cabo el análisis y la evaluación experimental de los algoritmos de aproximación más relevantes para la planeación de rutas, utilizando grafos sintéticos como base de información, así como la modelación de una red de tráfico urbano con datos reales, modelando dicha red de tráfico vehicular mediante un grafo. Se utilizará un método experimental, en el contexto de los resultados analíticos de la complejidad y del rango de aproximación conocidos, para determinar la eficiencia de los algoritmos de aproximación para el problema TPQ. Para ello se generaran redes de manera sintética, donde el número de categorías y el tamaño de la red se definirán aleatoriamente dentro de un rango de control apropiado. Finalmente, los algoritmos *greedy* se implementarán y se llevarán a cabo una serie de experimentos que ilustren su desempeño en función del número de categorías, aristas y vértices de la red, la evaluación de dichos experimentos se generarán en una computadora con Windows xp, 2Gb en memoria RAM y con un procesador Core 2 Duo a 1.80 GHz.

. Este trabajo modela una red de tráfico vehicular de un sector del centro histórico de la ciudad de Querétaro, utilizando funciones en el lenguaje de alto nivel Mathematica (Wolfram Research, Inc, 2007) basadas en el algoritmo de Dijkstra y dicha red de tráfico que contiene distancias en metros, se analizarán las rutas óptimas y por medio de técnicas de búsqueda local se manejarán cambios en los sentidos de las avenidas para intentar mejorar el flujo y utilización de las mismas.

.

El resto de esta tesis está estructurado como sigue. En el Capítulo 2 de analiza e implementa en Mathematica el algoritmo *greedy* del vecino más cercano, así como el análisis del funcionamiento del algoritmo de la mínima distancia. En el Capítulo 3 se analiza e implementa el algoritmo de aproximación utilizando Programación Lineal. En el Capítulo 4 se modela una red de tráfico vehicular, tomando como caso de estudio un sector del centro de la ciudad de Querétaro, definiendo la conectividad entre puntos de intersección de calles y avenidas de acuerdo a sentidos del tráfico vehicular. En el Capítulo 5 describe la implementación del modelo de red vehicular basado en un levantamiento de vértices y definición de aristas, obteniendo la longitud de las aristas y las rutas óptimas entre dos puntos, así como la frecuencia de uso de dichas rutas. En el Capítulo 6 se hace una evaluación experimental con los datos obtenidos, verificando la frecuencia de uso de aristas al cambiar algunos de los sentidos de las avenidas más importantes, y se utiliza la técnica de Búsqueda local para mejorar el promedio de utilización de las avenidas. Finalmente en el Capítulo 7 se dan conclusiones importantes del trabajo.

II. ANÁLISIS DE ALGORTIMOS *GREEDY* PARA PLANEACIÓN DE RUTAS

Existen varios algoritmos de aproximación que pueden resolver el problema para planeación de rutas (TPQ), en esta sección analizaremos dos algoritmos de aproximación también conocidos como *Greedy*, el algoritmo del vecino más cercano y el algoritmo de la mínima distancia (Dumitrescu y Mitchell. 2001). Estos algoritmos tienen un rango de aproximación en términos del número m de categorías.

Dado un grafo G el cuál contiene una arista por cada par de vértices V_i,V_j donde ($1 \leq i,j \leq n$), dicha arista representa el costo del camino más cercano de V_i a V_j. T_k representa el camino que ha visitado k vértices y lo definimos de la siguiente manera, $T_k=\{Vt_o,Vt_1,\ldots,Vt_k\}$, estos algoritmos se basan en el número de categorías existentes relacionadas en cada vértice.

II.1 Algoritmo del Vecino más cercano

El algoritmo del vecino más cercano aproxima el problema de TPQ generando una ruta T_k, esta ruta es generada iterativamente seleccionando el vecino más cercano a partir del último vértice Vt_k del conjunto de vértices para las categorías en R que no han sido seleccionadas aún, iniciando en el vértice inicial S. Básicamente tenemos una ruta T_k con $k<m$ donde m representa el número de categorías.

T_{k+1} se obtiene insertando el vértice Vt_{k+1} ,el cual es el vecino más cercano de Vt_k del conjunto de vértices en R de las categorías que no han sido seleccionadas. Una vez cubiertas todas la categorías el algoritmo termina conectando Vt_m al vértice final E.

Para ejemplificar este algoritmo tenemos como punto de partida un punto inicial S, m=4 que representa el numero de categorías (rojo, azul, verde y gris), varios miembros por cada categoría y un punto final E, Figura 2.1

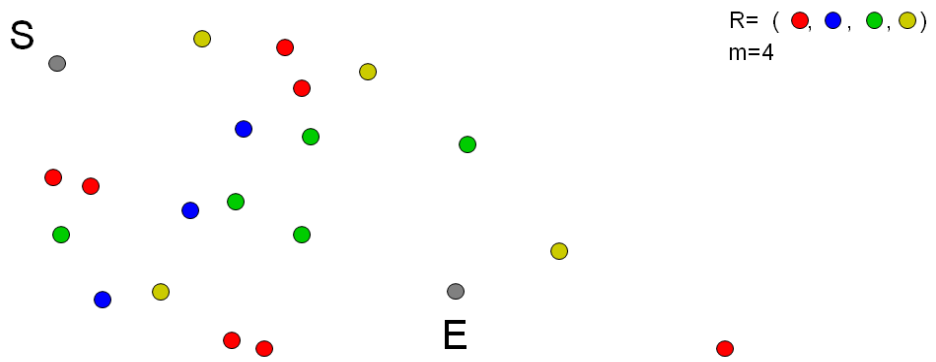


Figura 2. 1 Puntos Iniciales algoritmo del vecino más cercano

El algoritmo comienza en el punto inicial S, a partir de ese punto se busca el vecino más cercano calculando la distancia hacia cada vértice el cual no pertenezca a una categoría seleccionada con anterioridad, en este caso como es el punto inicial S se obtiene el vecino más cercano calculando la distancia de todos los vértices de todas las categorías, Figura 2.2

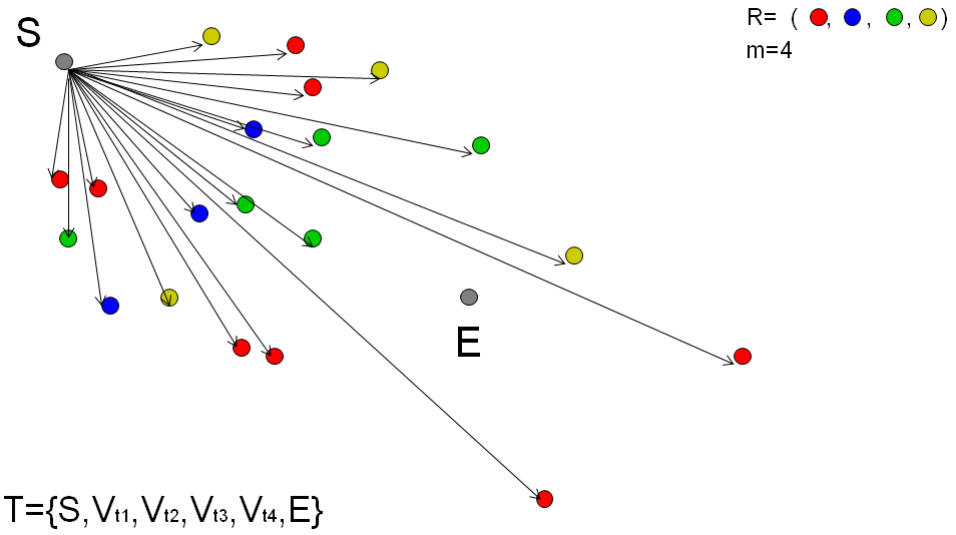


Figura 2. 2 Análisis del vecino más cercano a todos los vértices

Para el ejemplo el vértice Vt_1 iluminado de color rojo es el vecino más cercano con respecto al vértice inicial S, Figura 2.3

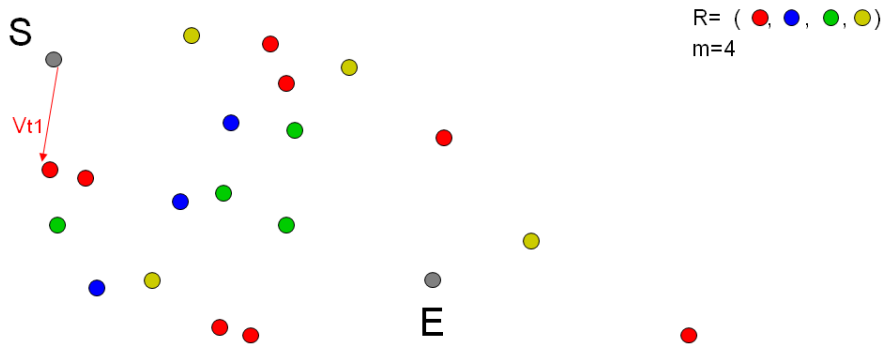


Figura 2.3 Vértice Vt_1 rojo primer vecino más cercano

En la siguiente iteración el algoritmo selecciona el vecino más cercano a partir del vértice Vt_1 de color rojo, calculando la distancia mínima a cada uno de los vértices que no pertenecen a la categoría del color rojo ya seleccionado, Figura 2.4

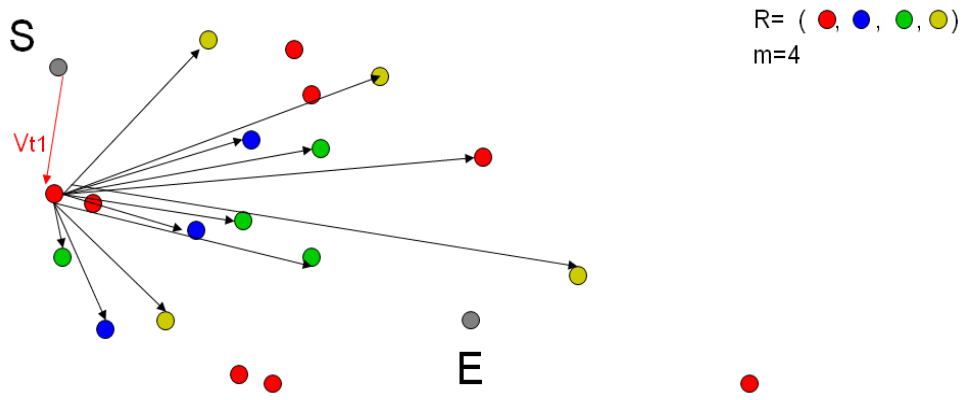


Figura 2.4 Vecino más cercano a partir del vértice Vt_1

El algoritmo selecciona el vértice como el vecino más cercano que contenga la mínima distancia a partir del vértice rojo Vt_1, para este ejemplo se selecciona el vértice Vt_2 de color verde, a partir de este punto la siguiente iteración selecciona el vecino más cercano analizando solo los vértices pertenecientes a las categorías restantes hasta que se llegan a cubrir todas las categorías en R, la última categoría cubierta para el ejemplo es el vértice Vt_4 de color azul, el algoritmo finaliza conectando el vértice final Vt_4 que pertenece a la última categoría seleccionada con el vértice final E, Figura 2.5. La ruta T_k generada contiene los vértices que minimizan el costo cubriendo todas las categorías.

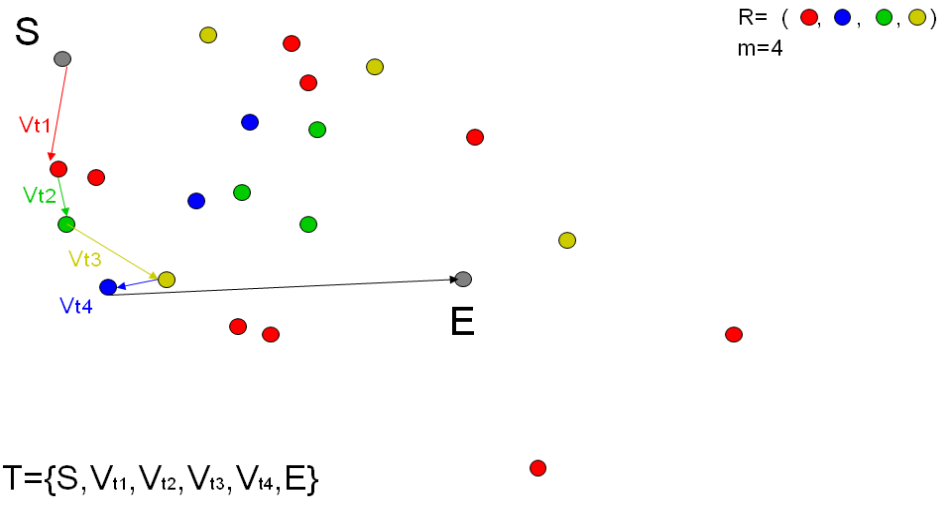


Figura 2.5 Ruta T aproximada obtenida del algoritmo del vecino más cercano

En resumen para el algoritmo del vecino más cercano se tiene:

Un punto inicial S y un punto final E.

Una ruta T_k con k<m.

T_{k+1} → insertando Vt_{k+1} (vecino más cercano de Vt_k de un conjunto de vértices en R).

E→Vt_m.

El algoritmo del vecino más cercano tiene un rango de aproximación con respecto a la solución óptima de 2^{m+1} -1, se puede describir el algoritmo mediante la siguiente codificación:

Algoritmo VMC (G,R,S,E)

V=S, I ={1,…,m}, T_k ={S}
Para n=1 to m hacer
    V= el más cercano NMC(v,R_i) para todo i $\in$ I
    $T_k \leftarrow$ {v}
    I $\leftarrow$ I – {v}
Fin Para
$T_k \leftarrow$ {E}

II.2 Implementación del algoritmo del Vecino más cercano

En la siguiente sección se implementa el algoritmo del vecino más cercano utilizando el lenguaje Mathematica. Generamos un grafo aleatorio que contiene 15 vértices, 30 aristas y 5 categorías representadas por los colores azul, amarillo, morado, rojo y verde donde cada una de ellas es asignada a un vértice distinto

Inicializamos los parámetros de entrada (número de vértices, aristas y categorías).

```
{nv,ne,nc}={15,30,5};
```

Se generan coordenadas aleatorias para la creación de los vértices y conexión de las aristas.

```
v=RandomReal [{-1,1},{nv,2}];
f=Flatten [(Thread[{First[#1],Last[#1]}]&)/@DelaunayTriangulation[v],1];
g=Union [Sort/@f];
tg=ToUnorderedPairs[ShortestPathSpanningTree[FromUnorderedPairs[g],1]];
k=Join[tg,Take[Complement[g,tg],ne-Length[tg]]];
```

La lista *col* contiene las categorías a la que pertenece cada vértice.

```
col=Table[If[i≤nc,i,RandomInteger[{1,nc}]],{i,nv}];
```

Basándonos en los vértices y aristas creadas aleatoriamente, se grafica el grafo con sus respectivas categorías por vértice Figura 2.6.

```
Graphics
  [{Line/@({v[[First[#1]]],v[[Last[#1]]]}&)/@k,
     EdgeForm[Thin],MapIndexed[{Hue[col[[#2[[1]]]]/nc],Disk[#1,0.035]}&,v]},
   }PlotRange→1.1
  ]
```

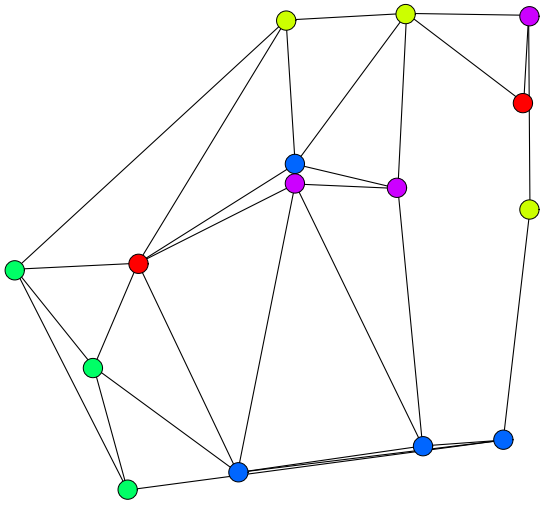


Figura 2.6 Grafo con 15 vértices, 30 aristas y 5 categorías

Se agrupan los vértices por categoría para el ejemplo los vértices 1,6 y 14 pertenecen a la categoría 1 los vértices 2,8 y 15 a la categoría 2, almacenando los resultados en la lista *clusterList.*

```
clusterList[clustering_]:=
Module[{m=Max[clustering]},
          Map[Flatten[Position[clustering,#]]&,Range[m]]
        ];
```

```
col
```

```
{1,2,3,4,5,2,3,1,3,2,2,3,1,3,4}
```

```
clusterList[col]
```

```
{{1,6,14},{2,8,15},{3,7,10,12},{4,9,13},{5,11}}
```

La función myShortestClusterP calcula el camino más corto que existe del punto inicial al vecino más cercano.

```
myShortestClusterP[g_,sv_,cluster_]:=
Module[{p,c,pos},
          p=Map[ShortestPath[g,sv,#]&,cluster];
          c=Map[N[CostOfPath[g,#]]&,p];
          pos=Flatten[Position[c,Min[c]]]{p[[First[pos]]],c[[First[pos]]]}
        ];
```

Con la siguiente función calculamos el camino más cercano de un vértice a otro hasta que se llegan a cubrir todas las categorías.

```
shPath[edgSet_,vertSet_,clusterList_,sv_]:=
Module[

{path={sv},auxClusterList=clusterList,k,svertex=sv,res,cand,c,g=Graph[edgSet,vertSet]},
          auxClusterList=DeleteCases[auxClusterList,{___,svertex,___}
         ];
While[ auxClusterList→{},
        {k=Map[myShortestClusterP[Graph[edgSet,vertSet],svertex,#] &,auxClusterList];
        res=First[Flatten[Position[k,Min[Map[Last[#]&,k]]]]];
        path=Append[path,Last[First[k[[res]]]]];
        cand=Last[First[k[[res]]]];
        auxClusterList=DeleteCases[auxClusterList,{___,cand,___}];
        svertex=Last[First[k[[res]]]]; }
     ];

path=Flatten[Join[{sv},Rest/@Map[ShortestPath[g,First[#],Last[#]]&,Partition[path,2,1]]]];
{path,CostOfPath[g,path]}]
```

En la lista *e* se generan las aristas y el peso de cada una de ellas

```
e=Map[{#,EdgeWeight→Norm[v→First[#]]-v→Last[#]]]}&,k]
```

```
{{{2,13},EdgeWeight→0.271622},{{3,8},EdgeWeight→0.565924},{{1,4},EdgeWeight
→0.600687},{{5,11},EdgeWeight→0.499576},{{1,6},EdgeWeight→0.299697},{{7,8},
EdgeWeight→0.895165},{{1,8},EdgeWeight→0.239364},{{1,9},EdgeWeight→0.471497
},{{6,10},EdgeWeight→0.327947},{{6,11},EdgeWeight→0.408854},{{4,12},EdgeWei
ght→0.685666},{{1,13},EdgeWeight→0.219072},{{4,14},EdgeWeight→0.990388},{{1
1,15},EdgeWeight→0.595626},{{2,6},EdgeWeight→0.367894},{{2,11},EdgeWeight→0
.591374},{{2,12},EdgeWeight→0.224878},{{2,15},EdgeWeight→0.224509},{{3,5},E
dgeWeight→0.503149},{{3,7},EdgeWeight→0.398434},{{3,10},EdgeWeight→0.218635
},{{3,11},EdgeWeight→0.379697},{{4,9},EdgeWeight→0.425548},}}
```

En la lista *solution* se generan los vértices que corresponden a la solución del algoritmo del vecino más cercano.

```
solution=shPath[e,v,clusterList[col],1]//Timing
{0.14,{{1,7,4,5,2},1.63913}}
```

Con los vértices solución obtenidos, se grafica el grafo que muestra la ruta que representa la solución a nuestro problema, generándose el camino aproximado pasando por cada uno de los vértices de diferente categoría, partiendo del vértice inicial 1 (amarillo), pasando por el vértice de color azul, seguidos por los vértices de color morado, rojo y verde, como se muestra en la Figura 2.7

```
Graphics
        [{Line/@({v[[First[#1]]],v[[Last[#1]]]}&) /@k,RGBColor[0,0,0],Disk[v[[First[path]]],0.05],
          Disk[v[[Last[path]]],0.05],Black,PointSize[0.02],EdgeForm[Thin],
          MapIndexed[{Hue[col[[#2[[1]]]] /nc],Disk[#1,0.03]}&,v],Thickness[0.007],
          Arrowheads[0.04],Arrow/@Partition[v[[path]],2,1]
         },PlotRange→1.1
       ]
```

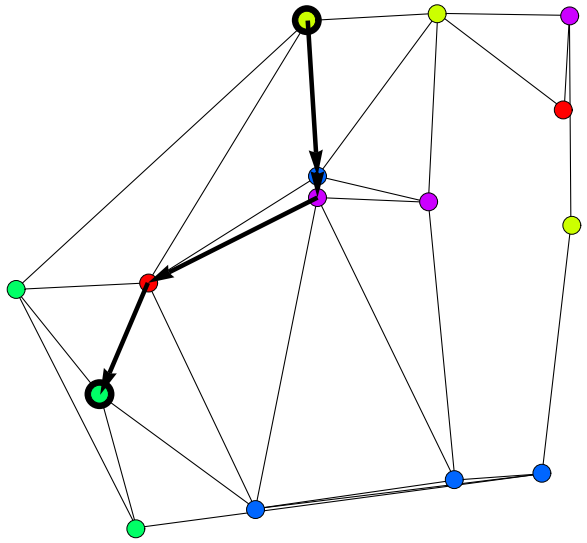


Figura 2.7 Ruta aproximada para grafo de 15 vértices y 30 aristas

Podemos generalizar el algoritmo, obteniendo la ruta aproximada a partir de cada uno de los vértices del grafo dado, en la Figura 2.8 podemos ver 5 de las gráficas generadas obteniendo la ruta aproximada.

sol=Map[shPath[e,v,clusterList[col],#]&,Range[Length[v]]]//Timing

{{{1,7,4,5,2},1.63913},{{2,5,4,7,1},1.63913},{{3,14,9,11,6,7,5,2},4.15045},{{4,7,1,6,11,6,7,5,2},3.85479},{{5,2,12,4,7,1},2.71243},{{6,9,11,6,7,5,2},3.04928},{{7,4,7,1,6,11,6,7,5,2},3.9265},{{8,5,4,7,1},1.6757},{{9,11,6,7,5,2},2.60763},{{10,13,6,11,6,7,5,2},4.39406},{{11,9,6,7,5,2},2.51664},{{12,2,5,4,7,1},2.28497},{{13,7,1,6,11,6,7,5,2},4.16488},{{14,9,11,6,7,5,2},3.31009},{{15,2,5,4,7,1},2.10127}}

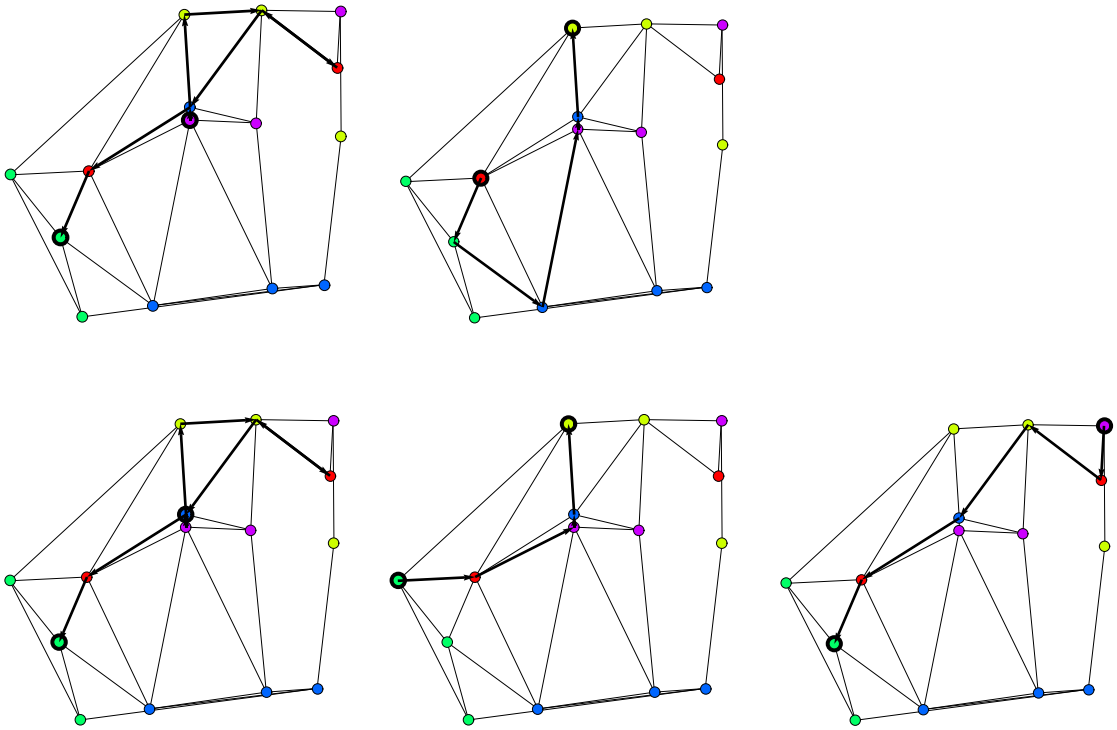


Figura 2.8 Generalización del algoritmo del vecino más cercano.

A continuación se muestra otro ejemplo, con un grafo de 250 vértices, 500 aristas y utilizando 25 categorías generado de forma aleatoria, Figura 2.9

```
{nv,ne,nc}={250,500,25};
 v=RandomReal[{-1,1},{nv,2}];
 f=Flatten[(Thread[{First[#1],Last[#1]}]&)/@DelaunayTriangulation[v],1];
 g=Union[Sort/@f];
 tg=ToUnorderedPairs[ShortestPathSpanningTree[FromUnorderedPairs[g],1]];
 k=Join[tg,Take[Complement[g,tg],ne-Length[tg]]];
 col=Table[If[i≤nc,i,RandomInteger[{1,nc}]],{i,nv}];

Graphics[{
            Line/@({v[[First[#1]]],v[[Last[#1]]]}&)/@k,PointSize[0.02`],EdgeForm[Thin],
            MapIndexed[{Hue[col[[#2[[1]]]]/nc],Disk[#1,0.0125`]}&,v]
          },PlotRange→1.1
        ]
```

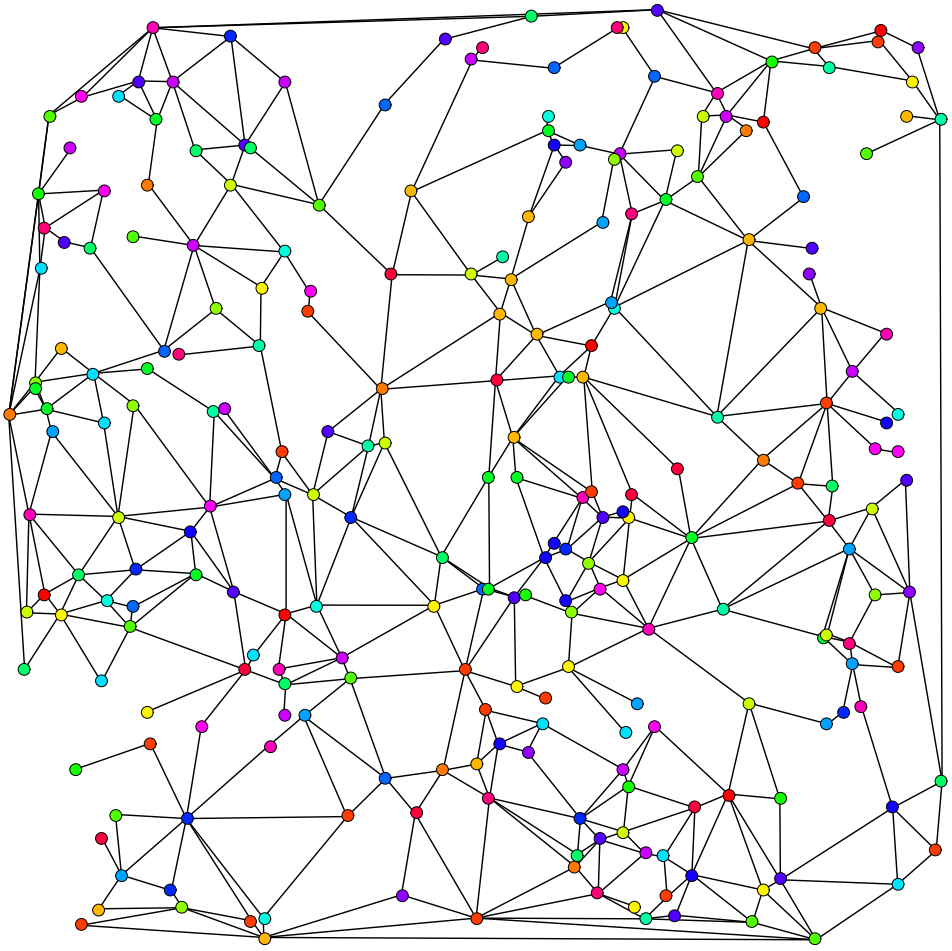


Figura 2.9 Grafo con 250 vértices, 500 aristas y 25 categorías

Se calcula en base al algoritmo del vecino más cercano la ruta óptima a partir del vértice inicial .

```
solution=shPath[e,v,clusterList[col],1]//Timing
{162.663,{{1,146,30,131,115,73,153,73,250,4,187,22,130,22,52,74,58,119,58,50,199,140,60,37,105,17,233,51,27,106,27,226,166,89,59,180,5,65,243,160,42,203,69,161,135,77},5.45965}}


path=First[Last[solution]];
Graphics[{
   Line/@({v[[First[#1]]],[[Last[#1]]]}&)/@k,
   RGBColor[0,0,0],Disk[v[[First[path]]],0.02],Disk[v[[Last[path]]],0.02],Black,

EdgeForm[Thin],MapIndexed[{Hue[col[[#2[[1]]]]/nc],Disk[#1,0.01]}&,v],Thickness[0.003],Arrowheads[0.0125],Arrow/@Partition[v[[path]],2,1]
            },PlotRange→1.1
          ]
```

Generando el camino aproximado siguiente a partir del vértice inicial 1, Figura 2.10

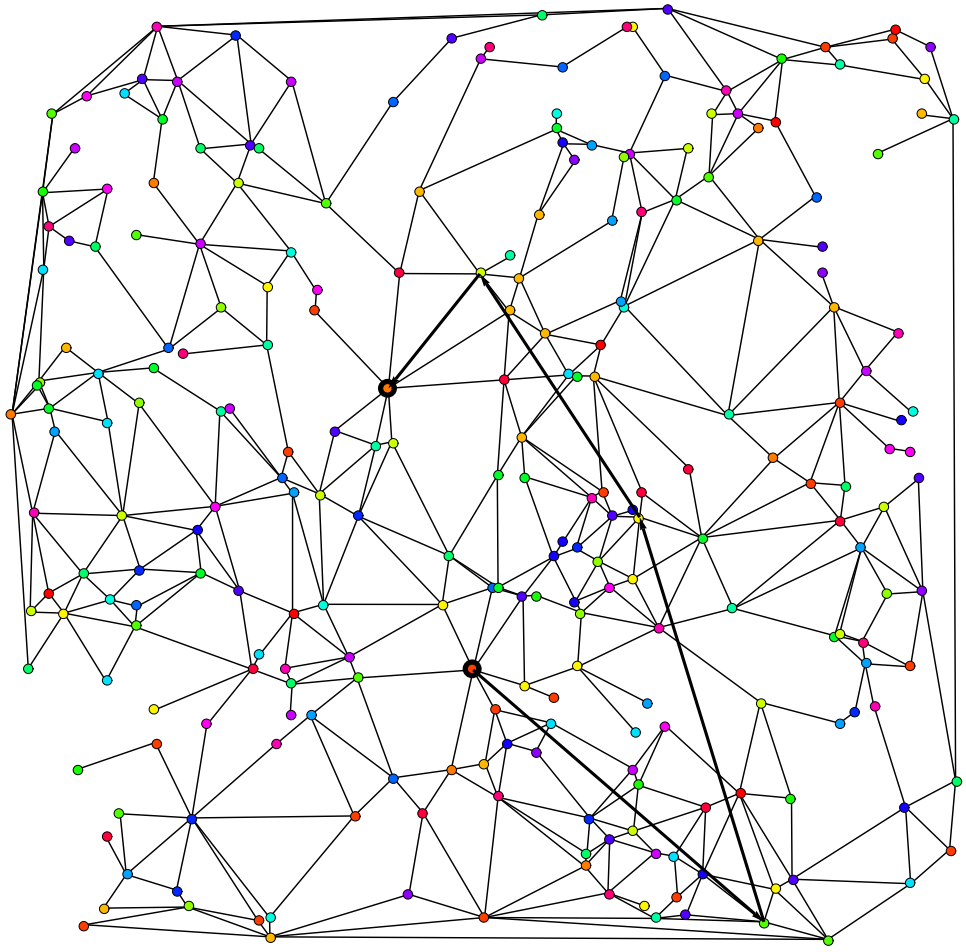


Figura 2.10 Ruta aproximada para grafo de 250 vértices y 500 aristas

Esta sección muestra el funcionamiento del algoritmo del vecino más cercano utilizando un grafo real modelado a través de una red de tráfico vehicular de un sector del centro histórico de la ciudad de Querétaro, el cual tiene sentidos y distancias reales.

El grafo está delimitado por las avenidas 5 de Febrero, Universidad, Corregidora y Constituyentes, en el capítulo IV se detalla la obtención y manejo de la red de tráfico vehicular. Para este ejemplo se utilizaron 412 vértices, 637 aristas todas con el sentido real de las avenidas y 6 categorías diferentes.

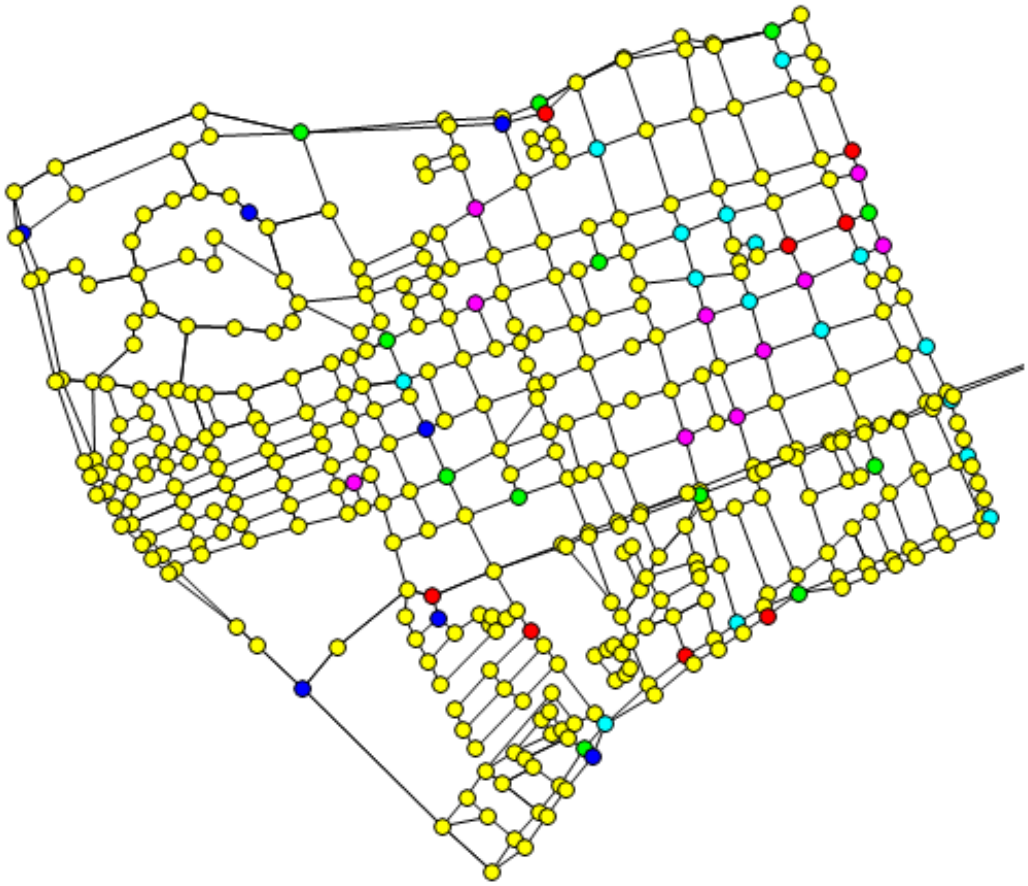
Las categorías utilizadas fueron las siguientes:

Categoría 1. Cualquier punto que no pertenezca a alguna categoría definida.
Categoría 2. Centros comerciales o mercados.
Categoría 3. Hoteles
Categoría 4. Gasolineras
Categoría 5. Iglesias
Categoría 6. Bancos

Todas las categorías fueron ubicadas dentro de alguno de los 412 vértices del grafo modelado, ubicando estas por los colores siguientes; para la categoría 1 se utilizó el color amarillo, categoría 2 color verde, categoría 3 color azul cielo, categoría 4 color azul marino, categoría 5 rosa y finalmente para la categoría 6 se utilizó el color rojo.

Una vez definidos los vértices inicial y final respectivamente, el problema consiste en encontrar el recorrido que minimice la distancia total empleada cubriendo las 6 categorías especificadas. La Figura 2.13 muestra el grafo dirigido del centro de la ciudad de Querétaro con el mapeo de vértices-categorías correspondientes.

```
Graphics
        [{Line/@({v[[First[#1]]],v[[Last[#1]]]}&) /@k,RGBColor[0,0,0],Disk[v[[First[path]]],0.05],
          Disk[v[[Last[path]]],0.05],Black,PointSize[0.02`],EdgeForm[Thin],
          MapIndexed[{Hue[col[[#2[[1]]]] /nc],Disk[#1,0.03`]}&,v],Thickness[0.007],
          Arrowheads[0.04],Arrow/@Partition[v[[path]],2,1]
         },PlotRange→1.1
        ]
```



.

Figura 2.11 Grafo del centro de Querétaro con 6 categorías definidas

Una vez armado el grafo se definieron las posiciones de las categorías de la siguiente manera.

```
clusterList[clustering_]:=Module[{m=Max[clustering]},Map[Flatten[Position[clustering,#]]&,
Range[m]]]
```

```
clusterList[col]
{{{1,4,33,40,62,72,86,112,293,333,344,363},{20,43,88,101,102,107,113,116,118
,125,133,294,323,351,352,371},{53,143,236,246,254,374,392},{17,36,77,105,115
,117,119,124,129,223},{100,110,111,239,266,321,375,397}}
```

Se calcula en base al algoritmo del vecino más cercano la ruta aproximada a partir del vértice inicial 144 y terminando en el vértice final 105 para este ejemplo, la lista *solution* indica la secuencia de vértices por los que pasa dicha ruta.

solution=shPath[e,v,clusterList[col],144,105]//Timing

```
{21.559,{{144,145,147,245,246,154,240,1,373,7,11,17,18,19,20,30,29,28,18,374
,375,9,13,19,20,91,92,93,94,99,104,105}}
```

La Figura 2.14 muestra la ruta óptima generada por el algoritmo la cuál empieza en el vértice 144, continua en la categoría azul marino (Gasolinera) seguido de la categoría verde (centro comercial), visita la categoría rosa (Iglesia) pasa a la categoría azul cielo (Hotel), llega a la categoría roja (Banco) y finalmente termina en el vértice final 105, esta ruta es generada cumpliendo con los sentidos de las avenidas respectivos.

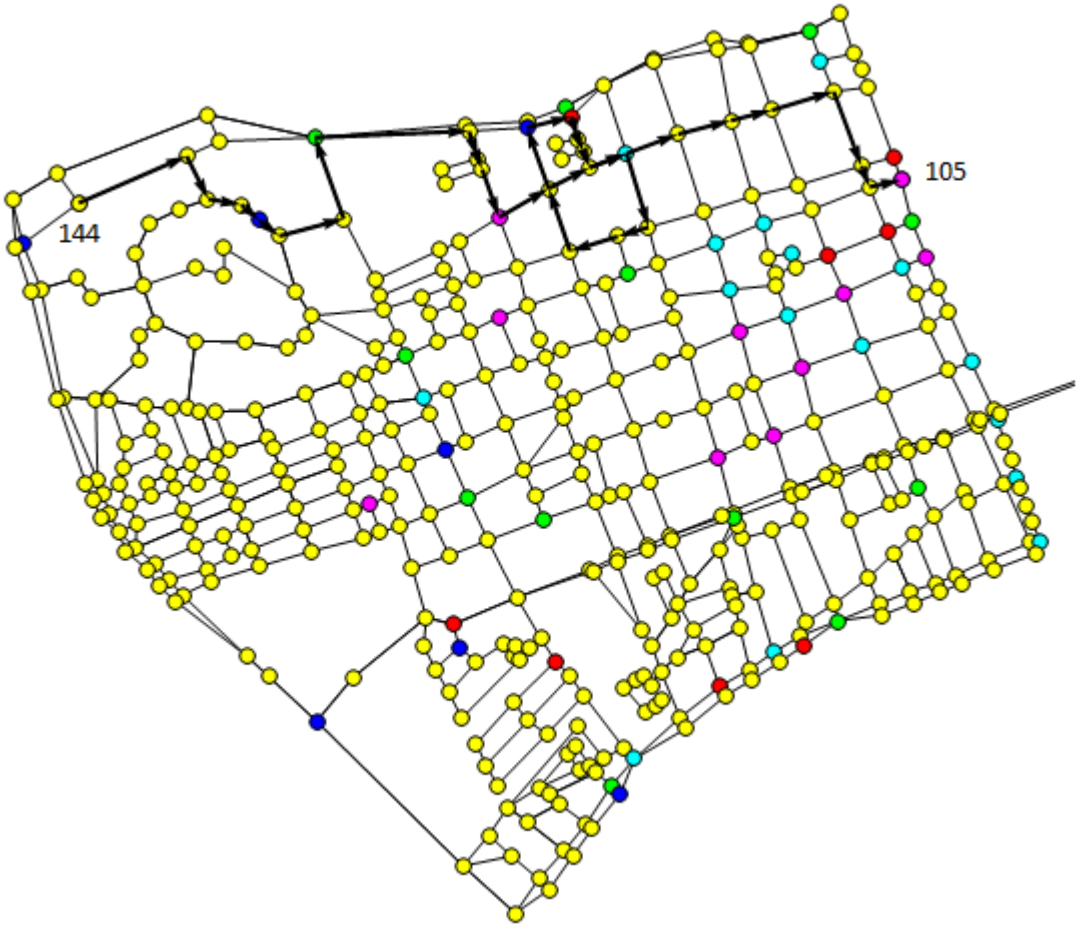


Figura 2.12 Ruta aproximada utilizando 6 categorías para el grafo muestra

II.3 Algoritmo de la Mínima distancia

El algoritmo de la mínima distancia aproxima el problema de TPQ seleccionando un conjunto de vértices $\{v_{i,}\ldots,v_k\}$ donde cada vértice pertenece a una categoría en R, un vértice inicial S y un vértice final E, tal que la suma del costo de ir del vértice inicial S al vértice v_i definida por $c(S,v_i)$, más el costo de ir del vértice v_i al vértice final E definido por $c(v_i,E)$ para todo $v_{i,}$ es el costo mínimo de los vértices que pertenecen a la categoría $R_{i.}$ Una vez que el conjunto de vértices con costo mínimo se ha seleccionado, el algoritmo genera la ruta aproximada T_k utilizando el algoritmo del vecino más cercano.

Podemos ejemplificar el funcionamiento del algoritmo teniendo como partida un punto inicial S, un conjunto de categorías m= 4 para este ejemplo y un punto final E, Figura 2.15

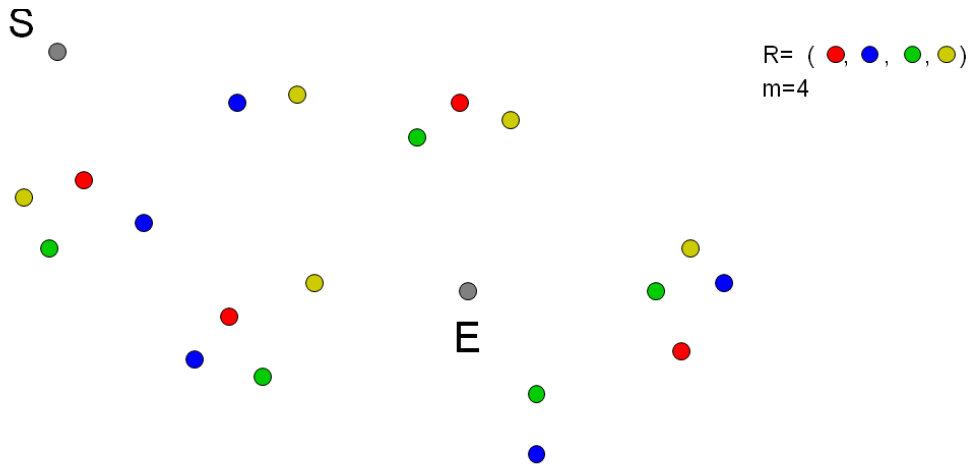


Figura 2.13 Puntos iniciales para el algoritmo de la mínima distancia

Por cada vértice v_i que pertenece al conjunto de categorías R, utilizando el vértice inicial S y el vértice final E, se obtiene el vértice que minimice la condición $c(S,v_i)+c(v_i,E)$, para el ejemplo se analizan todos los miembros de la categoría en rojo seleccionando el vértice de menor costo, Figura 2.16

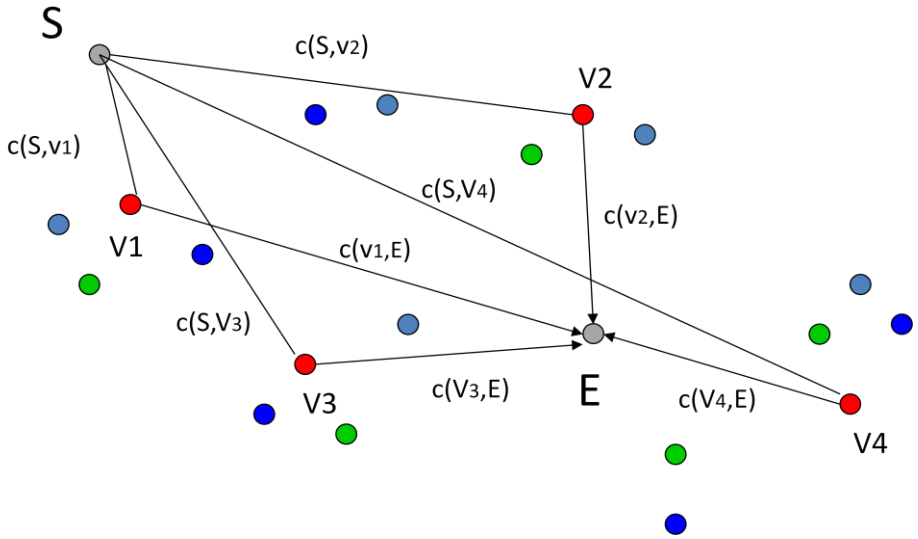


Figura 2.14 Análisis de costos por categoría

Para nuestro ejemplo, el vértice Vt_1 de la categoría del color rojo es seleccionado como el de menor costo, el vértice Vt_2 para la categoría de color azul, Vt_3 y Vt_4 para la categoría gris y verde respectivamente. Una vez que se obtiene el conjunto de vértices con costo mínimo se traza la ruta aproximada T_k, comenzando en el vértice inicial S, pasando por cada vértice seleccionado utilizando el algoritmo del vecino más cercano y finalizando al conectar el último vértice con el vértice final S, la Figura 2.17 muestra el camino aproximado final utilizando el algoritmo de la mínima distancia.

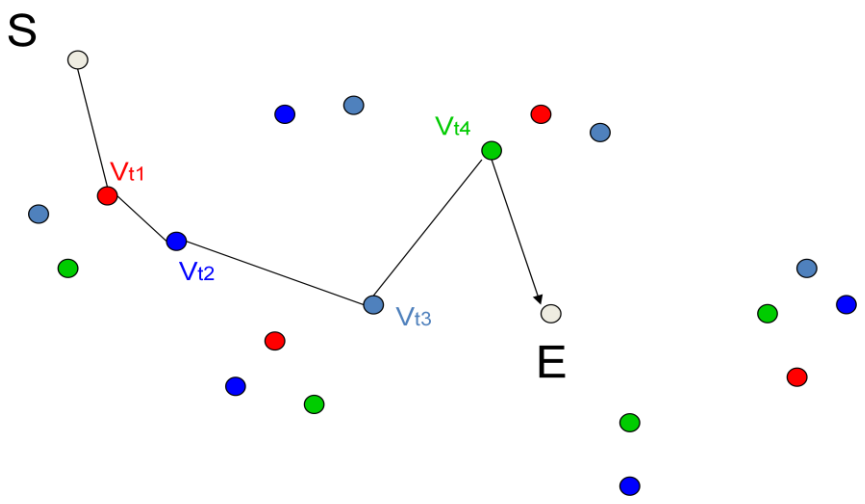


Figura 2.15 Ruta aproximada para algoritmo de la mínima distancia

El algoritmo de la mínima distancia tiene un mejor rango de aproximación en comparación al algoritmo del vecino más cercano ya que este maneja un rango de aproximación de m con respecto a la solución óptima. Podemos describir el algoritmo de la siguiente manera:

Algoritmo AMD(G,R,S,E)
U = 0
For i = 1 to m do
  U ← π(v)= R_i : c(S, v)+c(v,E) se minimiza
  v = S, T_k ← { S }
While U ≠ 0 do
  v = VMC (v,U)
  T_a ← { v }
  Quitar v de U
End while
T_k ← { E }

II.4 Implementación del algoritmo de la mínima distancia

En la siguiente sección se implementa el algoritmo de la mínima distancia, generamos un grafo aleatorio que contiene 15 vértices, 30 aristas y 5 categorías representadas por los colores azul, amarillo, morado, rojo y verde donde cada una de ellas es asignada a un vértice distinto. La lista *col* contiene las categorías a la que pertenece cada vértice.

```
col=Table[If[i≤nc,i,RandomInteger[{1,nc}]],{i,nv}];
```

La Figura 2.18 muestra el grafo generado para la utilización del algoritmo de la mínima distancia.

```
Graphics
  [{Line/@({v[[First[#1]]],v[[Last[#1]]]}&)/@k,
     EdgeForm[Thin],MapIndexed[{Hue[col[[#2[[1]]]]/nc],Disk[#1,0.035]}&,v]},
    }PlotRange→1.1
  ]
```

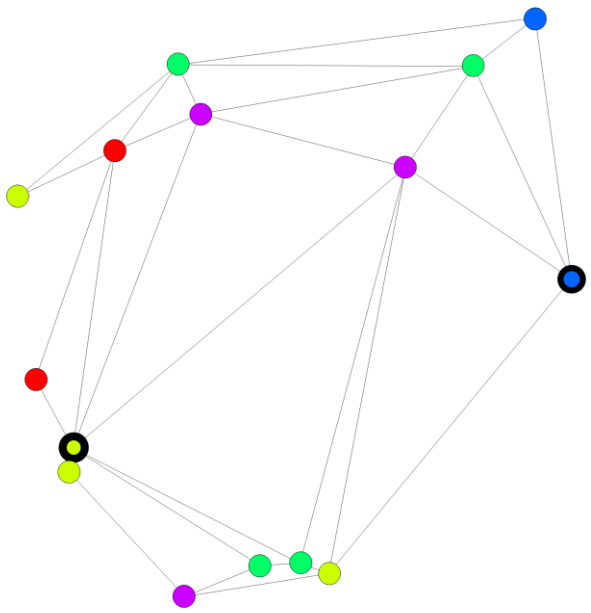


Figura 2.16 Grafo con 15 vértices, 30 aristas y 5 categorías para el algoritmo de la mínima distancia

Se agrupan los vértices por categoría, para el ejemplo los vértices 1,10,13 y 14 pertenecen a la categoría 1, los vértices 2,8, 9 y 15 a la categoría dos, los vértices 3 y 6 a la categoría 3, los vértices 4,7 y 11 a la categoría 4 y finalmente los vértices 5 y 12 a la categoría 5, dichos subconjuntos se almacenan en la lista *clusterList.*

```
clusterList[clustering_]:=
Module[{m=Max[clustering]},
        Map[Flatten[Position[clustering,#]]&,Range[m]]
    ];
```

```
col
{1,2,3,4,5,5,5,2,4,4,4,3,5,3,3}
```

```
clusterList[col]
{{1,10,13,14},{2,8,9,15},{3,6},{4,7,11},{5,12}}
```

La función myShortestClusterM selecciona el vértice por categoría que minimiza la condición $c(S+v_i)+c(v_i+E)$.

```
myShortestClusterM[g_,sv_,ev_,cluster_]:=Module[{p,p1,p2,p3,c,c1,c2,pos},
    p1=Map[ShortestPath[g,sv,#]&,cluster];
    p2=Map[ShortestPath[g,#,ev]&,cluster];
    p=Join[p1,p2,2];
    c1=Map[N[CostOfPath[Graph[e,v],#]]&,p1];
    c2=Map[N[CostOfPath[Graph[e,v],#]]&,p2];
    c=c1+c2;
    pos=Flatten[Position[c,Min[c]]];
    {Extract[cluster,First[pos]]}]
```

La función shPathM genera el conjunto de vértices seleccionados por la función myShortestClusterM.

```
shPathM[edgSet_,vertSet_, clusterList_,sv_,ev_]:=

Module[{path={sv},auxClusterList=clusterList,k,svertex=sv,evertex=ev,g=Graph[edgSet,vertSet]
},auxClusterList=DeleteCases[auxClusterList,{___,sv,___}];k=Map[myShortestClusterM[Graph[e
dgSet,vertSet],svertex,evertex,#]&,auxClusterList];k ]
```

En la lista *e* se generan las aristas y el peso de cada una de ellas.

```
e=Map[{#,EdgeWeight→Norm[v→First[#]]-v→Last[#]]]}&,k]
```

Una vez obtenidos los vértices que minimizan la condición utilizando la función shPathM, se genera la lista final que contiene el camino aproximado para la solución del problema utilizando el algoritmo del vecino más cercano usando la función shPath.

En la lista *solution* se generan los vértices de la solución aproximada del algoritmo de la mínima distancia

```
solution=shPath[e,v,solutionM=shPathM[e,v,clusterList[col],1,15],1]
{0.062,{{1,12,1,13,4,15,9,14,3},2.76936}}
```

Con los vértices solución obtenidos, se grafica el grafo que muestra la ruta que representa la solución al problema Figura 2.19.

```
Graphics
    [{Line/@({v[[First[#1]]],v[[Last[#1]]]}&) /@k,RGBColor[0,0,0],Disk[v[[First[path]]],0.05],
      Disk[v[[Last[path]]],0.05],Black,PointSize[0.02`],EdgeForm[Thin],
      MapIndexed[{Hue[col[[#2[[1]]]] /nc],Disk[#1,0.03`]}&,v],Thickness[0.007],
      Arrowheads[0.04],Arrow/@Partition[v[[path]],2,1]
     },PlotRange→1.1
    ]
```

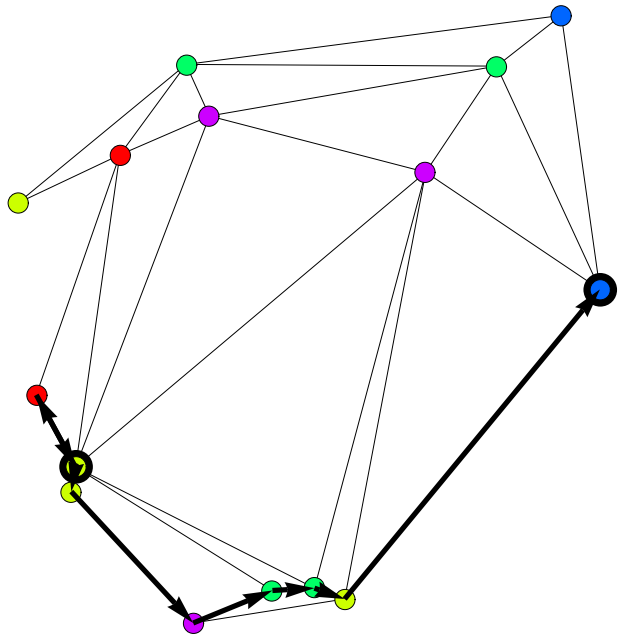


Figura 2.17 Ruta aproximada para el algoritmo de la mínima distancia

Esta sección muestra el funcionamiento del algoritmo de la mínima distancia utilizando un grafo real modelado a través de una red de tráfico vehicular de un sector del centro histórico de la ciudad de Querétaro, el cual describe sentidos y distancias reales. La figura 2.20 muestra el grafo generado utilizando 412 vértices, 736 aristas y 6 categorías definidas con los colores (rojo para bancos, rosa para las iglesias, azul cielo para los hoteles, azul marino para las gasolineras y finalmente verde para los centros comerciales).

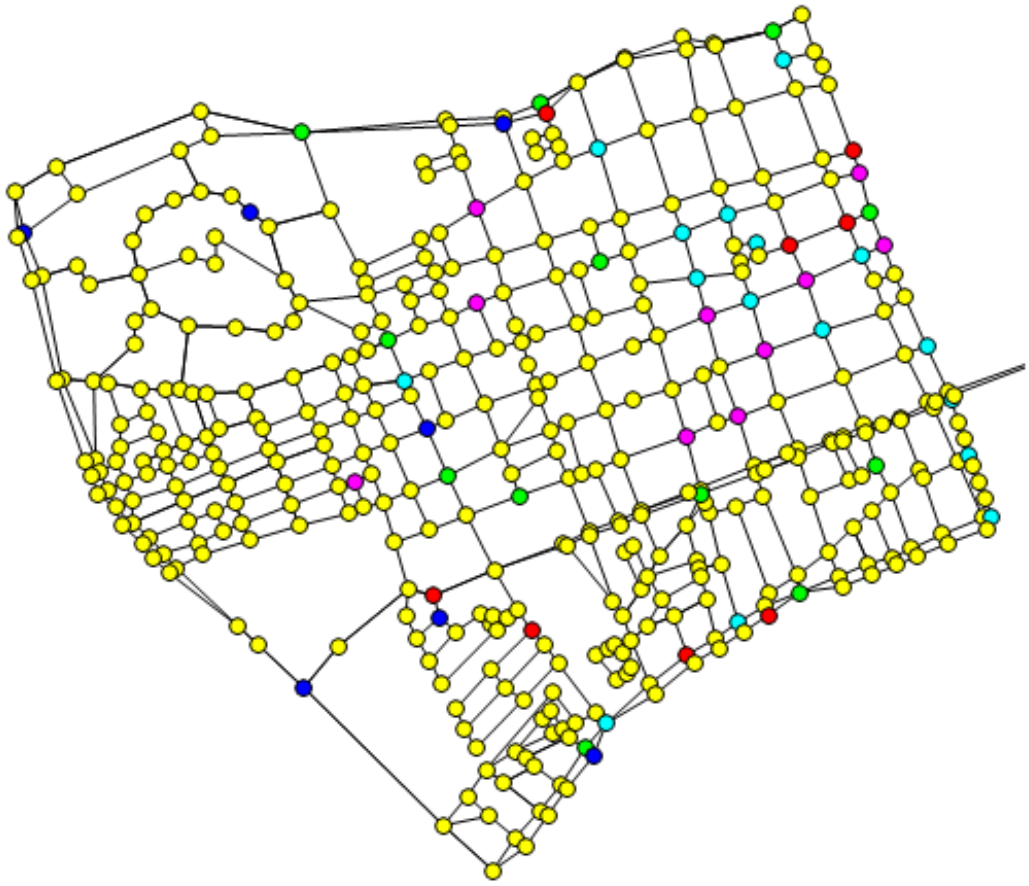


Figura 2.18 Grafo del centro de Querétaro con 6 categorías definidas

Una vez definido el grafo, se calcula en base al algoritmo de la mínima distancia la ruta aproximada a partir del vértice inicial 164 y terminando en el vértice final 105 para este ejemplo, la lista *solution* indica la secuencia de vértices por los que pasa dicha ruta.

solution=shPath[e,v,shPathM[e,v,clusterList[col],164,105],164,105]//Timing

```
{2.281,{{164,165,166,167,168,169,170,171,172,411,33,34,35,36,37,38,39,40,41,
101,102,103,104,105,100,99,98,97,96,30,29,28,18,374,375,9,13,19,20,91,92,93,
94,99,104,105},4.6854}}
```

La Figura 2.21 muestra la ruta aproximada generada por el algoritmo la cuál empieza en el vértice 164, continua en la categoría verde (centro comercial), visita la categoría azul cielo (Hotel), pasa a la categoría rosa (Iglesia), llega a la categoría roja (Banco), sigue a la categoría azul marino (Gasolinera) y finalmente termina en el vértice final 105, esta ruta es generada cumpliendo con los sentidos de las avenidas respectivas.

Figura 2.19 Ruta aproximada utilizando 6 categorías para el grafo muestra del centro de Querétaro.

En la siguiente sección se muestra una evaluación de desempeño para los algoritmos del vecino más cercano y mínima distancia, utilizando grafos aleatorios y manejando diferentes números de vértices, aristas y categorías.

Se generaron 10 grafos aleatorios manejando 100, 150, 250, 350, 450, 550, 650, 750, 800, 900 y 1000 vértices con 25 categorías, se obtuvo el tiempo (segundos) de ejecución de los algoritmos para cada uno de los grafos generados y se compararon resultados.

La Figura 2.22 muestra el tiempo de ejecución de los algoritmos manejando los grafos aleatorios mencionados anteriormente, observando que el algoritmo de la mínima distancia realiza mejores tiempos de ejecución en comparación con el algoritmo del vecino más cercano.

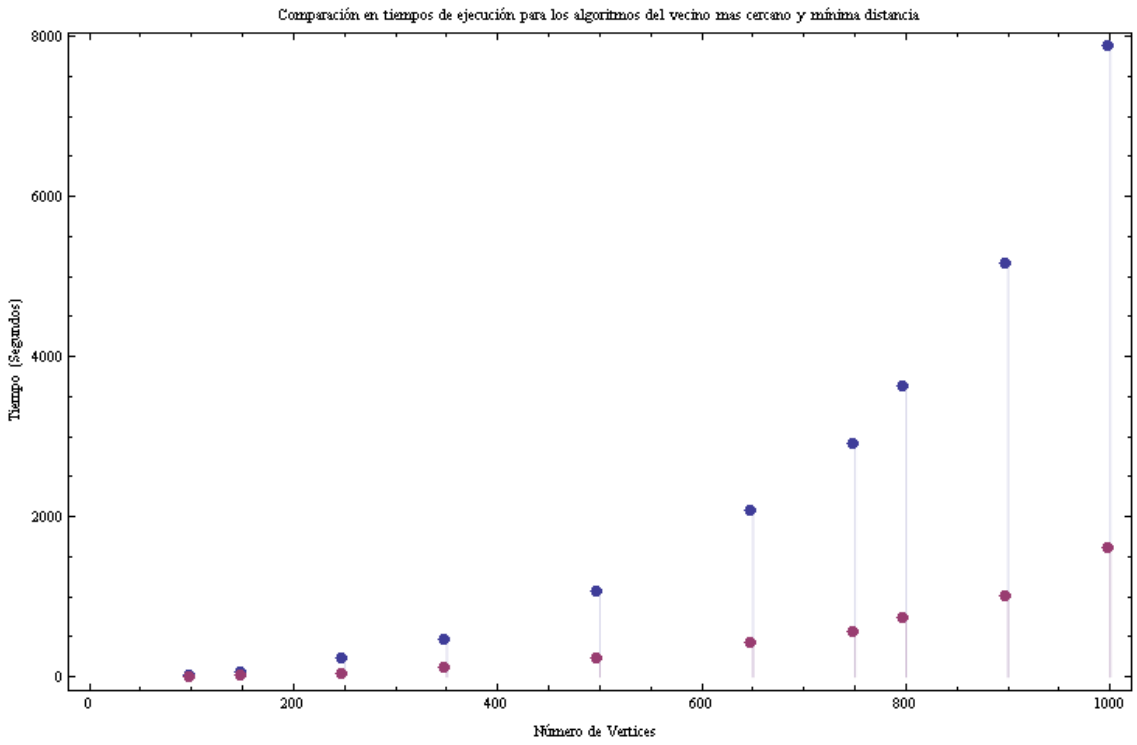


Figura 2.20 Evaluación del los algoritmos voraces utilizando diferentes números de vértices

Para la siguiente evaluación se generaron 10 grafos aleatorios con 400 vértices y 800 aristas cada uno, pero ahora utilizando diferentes números de categorías, se obtuvo el costo de ejecución del algoritmo midiendo el tiempo en segundos.

La Figura 2.23 muestra el tiempo de ejecución de los algoritmos utilizando grafos sintéticos con 25, 50, 75, 100, 125, 150, 175, 200, 225 y 250 categorías respectivamente, observando que el tiempo de ejecución crece sustancialmente al aumentar el número de categorías, pero se obtiene un mejor rendimiento con el algoritmo de la mínima distancia.

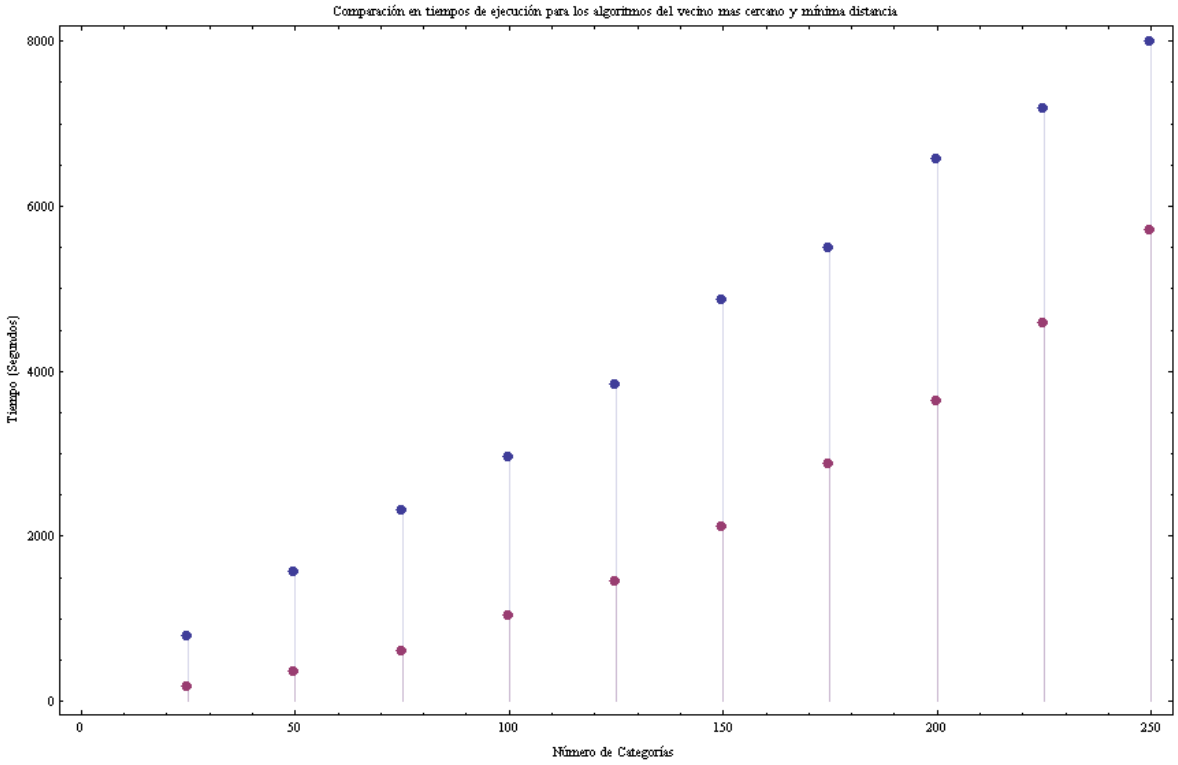


Figura 2.21 Evaluación de los algoritmos voraces utilizando diferentes números de categorías

III. MODELADO DE UNA RED DE TRÁFICO VEHICULAR: UN CASO DE ESTUDIO

Para el modelado de la red de tráfico se construyó un grafo dirigido basado en información geográfica, con distancias y sentidos reales de las avenidas del centro histórico de la ciudad de Querétaro.

A través del IMT (Instituto Mexicano del Transporte) se obtuvo la información geográfica mediante un sistema GIS basado en coordenadas geográficas UTM correspondientes el estado de Querétaro, con el cual se pueden representar las distancias en metros.

El Sistema de Coordenadas UTM (Lichten S.M. 1991) por sus siglas en ingles Universal Transversal de *Mercator,* es un sistema de coordenadas basado en la proyección geográfica transversa de *Mercator*, que se construye como la proyección de *Mercator* normal, pero en vez de hacerla tangente al Ecuador, la hace tangente a un meridiano. A diferencia del sistema de coordenadas tradicional, expresadas en longitud y latitud las magnitudes en el sistema UTM se expresan en metros únicamente al nivel del mar que es la base de la proyección del elipsoide de referencia. *Arcview* fue el sistema de información geográfica utilizado para analizar y procesar la información, con este sistema se obtuvieron las coordenadas de los puntos dentro del cuadro seleccionado, así como las distancias en metros entre cada uno de ellos.

El cuadro “muestra” seleccionado dentro del estado de Querétaro para el modelado de la red de tráfico es limitado por las Avenidas de 5 de febrero, Universidad, Constituyentes y Corregidora.

La Figura 3.1 muestra el cuadro del centro de la ciudad de Querétaro seleccionado, así como los puntos o vértices definidos para el caso de estudio.

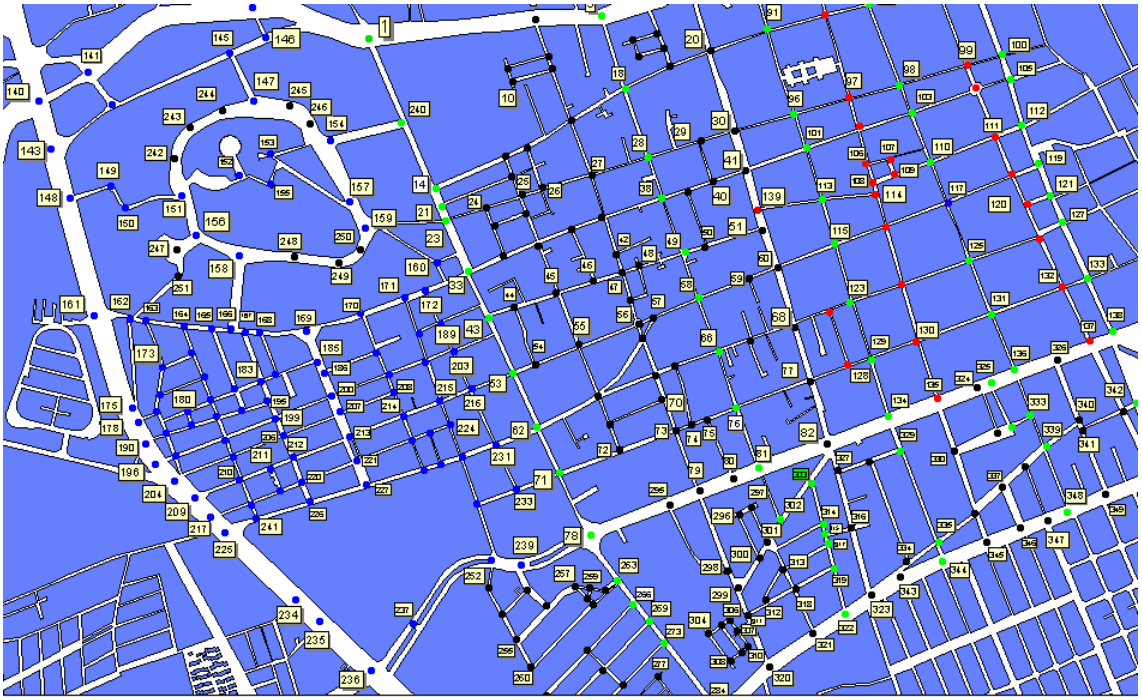


Figura 3.1 Cuadro muestra del centro de la ciudad de Querétaro

Cada punto seleccionado representa las coordenadas UTM dentro del mapa, con dichas coordenadas se calculan las distancias entre cada vértice.

Utilizando el lenguaje de programación funcional de alto nivel Mathematica se pueden representar los puntos del mapa, generar las aristas entre cada punto y a su vez calcular las distancias entre los puntos de interés.

III.1 Red de tráfico urbano del centro histórico de Querétaro

La modelación de la red de tráfico urbano se realizó utilizando las coordenadas obtenidas del grafo modelado con *Arcview* y utilizando el lenguaje de programación funcional Mathematica, para el análisis y generación de resultados. El objetivo principal de este estudio es caracterizar la red urbana de tránsito modelada mediante un grafo con el fin de determinar la mejor solución en términos de dirección de aristas del grafo.

El caso de estudio consiste en una red urbana de tránsito dentro de la ciudad de Querétaro representada por el grafo G conformado por un conjunto de 411 vértices y 736 aristas. Un ejemplo de la definición de vértices dentro de Mathematica como puntos coordenados obtenidos del mapa se puede ver a continuación:

```
v1 = {2165365.420053, 564605.583622}; v2 = {2165764.318402, 564639.699928}; v3 = {2165924.402608, 564647.572922};
v4 = {2166024.127195, 564684.313559}; v5 = {2166126.476114, 564744.673178}; v6 = {2165698.71012, 564521.605022};
v7 = {2165795.810376, 564550.472665}; v8 = {2166000.508214, 564589.837634}; v9 = {2166058.243501, 564602.95929};
v10 = {2165709.207445, 564487.488716}; v11 = {2165806.307701, 564521.605022}; v12 = {2166013.62987, 564550.472665};
```

El conjunto *vértices* definido a continuación, consiste en todos los vértices del grafo asociado a nuestro caso de estudio. El conjunto contiene 411 vértices.

```
vertices=ToExpression["v"<>ToString[#]]&/@Range[411];
```

La Figura 3.2 muestra los puntos de intersección, así como el ajuste lineal de avenidas que se utilizan para el caso de estudio, cabe señalar que cada punto corresponde a un vértice definido anteriormente.

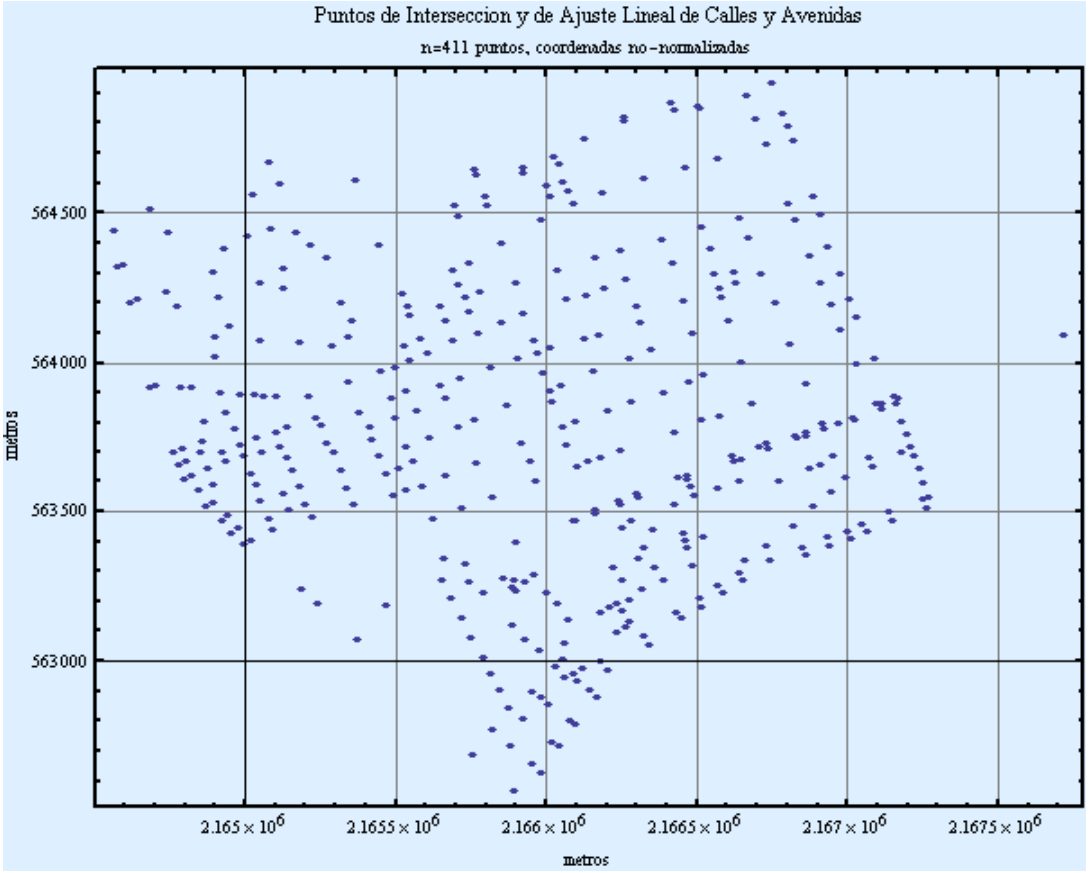


Figura 3.2 Puntos de intersección utilizados para el caso de estudio

El conjunto *vertices* muestra las coordenadas de los puntos o vértices en una forma no normalizada o nativa directamente de UTM, estas obtenidas del archivo en *Arcview*. Ya que los puntos coordenados de los vértices son cantidades difíciles de manejar, se procedió a normalizarlos con respecto a los valores mínimos de las coordenadas.

```
{x1,x2,y1,y2}={Min[First/@vertices],Max[First/@vertices],
               Min[Last/@vertices],Max[Last/@vertices]}

vertices=Floor[#-{x1,y1}]&/@vertices;
```

Las variables x1, y1 almacenan el valor mínimo de las ordenadas, para normalizar los pares ordenados restamos dicho valor a cada par, en la variable final *vértices* se obtiene las coordenadas de los puntos normalizados. La Figura 3.3 muestra los puntos de intersección así como el ajuste lineal de Avenidas con las ordenadas normalizadas.

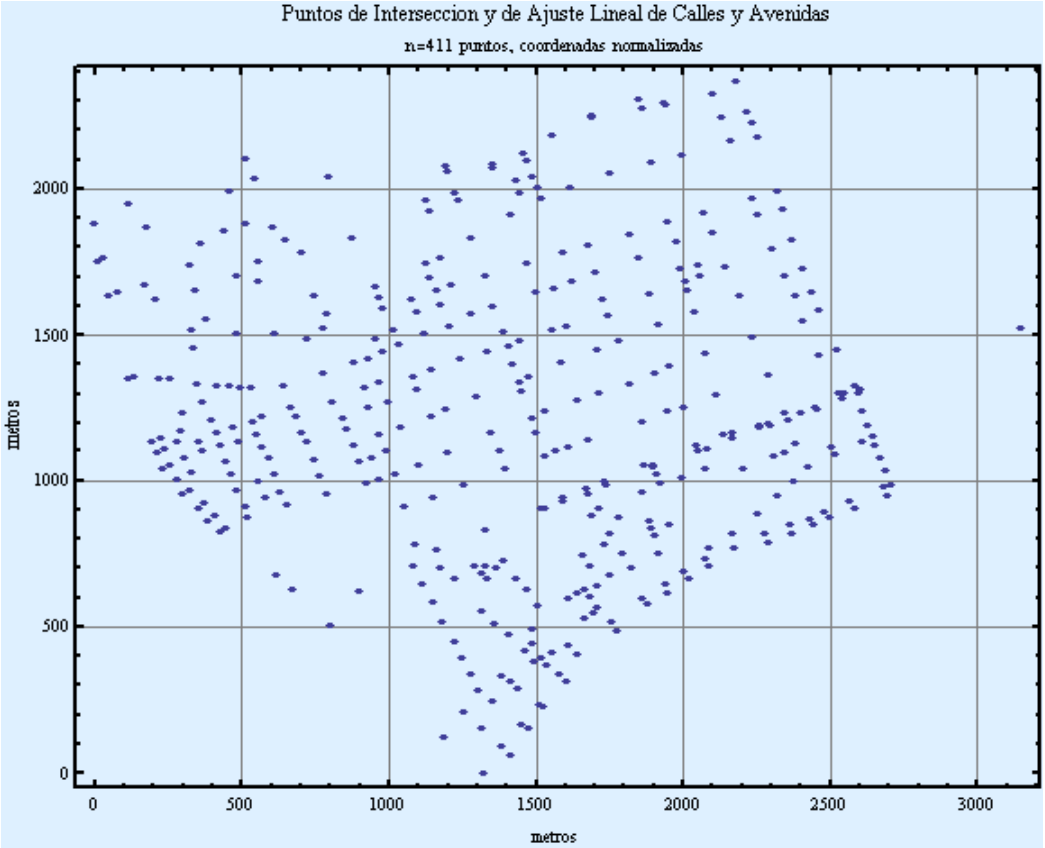


Figura 3.3 Puntos de intersección con coordenadas normalizadas

En seguida se etiqueta cada par ordenado con un número de vértice correspondiente.

```
verticesLabeled=Table[{First[vertices[[i]]],Last[vertices[[i]]],i},{i,1,Length[vertices]}];
```

IV.2 Definición de la conectividad entre puntos de intersección de avenidas

A continuación se definen las aristas como pares ordenados, esto es, las aristas que corresponden al grafo dirigido. Cada componente corresponde a los puntos coordenados de los vértices que define la arista. En total se tienen 736 aristas. Cabe mencionar que las etiquetas no son consecutivas, las cuales están en el rango de enteros entre 1 y 736. Un ejemplo de cómo se definen las aristas se muestra a continuación:

a1={v2,v1};a2={v14,v240};a3={v3,v2};a4={v373,v7};a5={v4,v3};a6={v18,v374};a7={v5,v4};a8={v375,v9};
a9={v5,v20};a10={v7,v6};a11={v6,v10};a12={v7,v11};a13={v9,v8};a14={v8,v12};a15={v9,v13};a16={v10,v11};
a17={v11,v17};a18={v12,v13};a19={v13,v19};a20={v14,v15};a21={v21,v14};a22={v15,v16};a23={v15,v22};
a24={v16,v17};a25={v16,v26};a26={v17,v18};a27={v17,v27};a28={v18,v19};a29={v28,v18};a30={v19,v20};
a31={v20,v30};a32={v22,v21};a33={v23,v21};a34={v25,v22};a35={v24,v23};a36={v33,v410};a37={v25,v24};
a38={v24,v31};a39={v26,v25};a40={v32,v25};a41={v27,v26};a42={v28,v27};a43={v27,v37};a44={v29,v28};
a45={v38,v28};a46={v30,v29};a47={v29,v40};a48={v30,v41};a49={v31,v32};a50={v31,v34};a51={v35,v32};

La lista *aristas* que se presenta a continuación muestra un ejemplo de la definición de las aristas del grafo correspondiente a la red de tráfico urbano.

aristas = {a1,a2,a3,a4,a5,a6,a7,a8,a9,a10,a11,a12,a13,a14,a15,a16,a17,a18, a19, a20, a21, a22, a23, a24,a25,a26, a27,
a28, a29, a30, a31, a32, a33, a34, a35, a36, a37, a38, a39, a40,a41,a42,a43,a44,a45,a46,a47,a48, a49, a50, a51, a52, a53,
a54, a55, a56, a57, a58, a59, a60, a61, a62, a63, a64, a65, a66, a67, a68, a69, a70, a71, a72, a73, a74,,a75, a76, a77, a79,
a80, a81,a82,a83,a84,a85,a86,a87,a88,a89,a90,a91,a92,a93,a94,a95,a96,a97,a98,a99, a100,a101, a102, a103, a104, a105,
a106, a107, a108, a109, a110, a111, a112, a113, a114, a115, a116, a117, a118, a119, a120, a121,a122,a123,a124,a126,
,a127,a128,a129,a130,a131,a132,a133,a134,a135,a136,a137,a138,a139,a140,a141,a142,a143,a144, a145,a146,a147
,a148,a149,a150,a151,a152,a153,a154,a155,a156,a157,a158,a160};

IV.3 Definición de los sentidos vehiculares

Las direcciones de las avenidas del centro de la ciudad de Querétaro, que están representadas por 736 aristas están basadas en información proporcionada por el INEGI, a través de un mapa que tiene registrados los sentidos y nombres de las principales avenidas del centro de la ciudad.

El conjunto ***a*** definido a continuación representa algunas de las aristas dirigidas del grafo correspondiente al caso de estudio.

a = {{3, 2}, {373, 7}, {4, 3}, {18, 374}, {5, 4}, {375, 9}, {5, 20}, {7, 6}, {6, 10}, {7, 11}, {9, 8}, {8, 12}, {9, 13}, {10, 11}, {11, 17}, {12, 13}, {13, 19}, {14, 15}, {21, 14}, {15, 16}, {15, 22}, {16, 17}, {16, 26}, {17, 18}, {17, 27}, {18, 19}, {28, 18}, {19, 20}, {20, 30}, {22, 21}, {23, 21}, {25, 22}, {24, 23}, {33, 410}, {25, 24}, {24, 31}, {26, 25}, {32, 25}, {27, 26}, {28, 27}, {27, 37}, {29, 28}, {38, 28}, {30, 29}, {29, 40}, {30, 41}, {31, 32}, {31, 34}, {35, 32}, {33, 34}, {43, 33}, {34, 35}, {35, 36}, {45, 35}, {36, 37}, {36, 46}, {37, 38}, {37, 42}, {38, 39}, {49, 38}, {39, 40}}}

La diferencia entre los conjuntos ***a*** y ***aristas*** es que el primero define las aristas con las etiquetas de sus vértices, mientras que el segundo define las aristas con los puntos coordenados de sus vértices. La Figura 3.4 presenta las aristas direccionadas dentro del grafo, el cual representa los sentidos de las Avenidas dentro del límite propuesto para el caso de estudio.

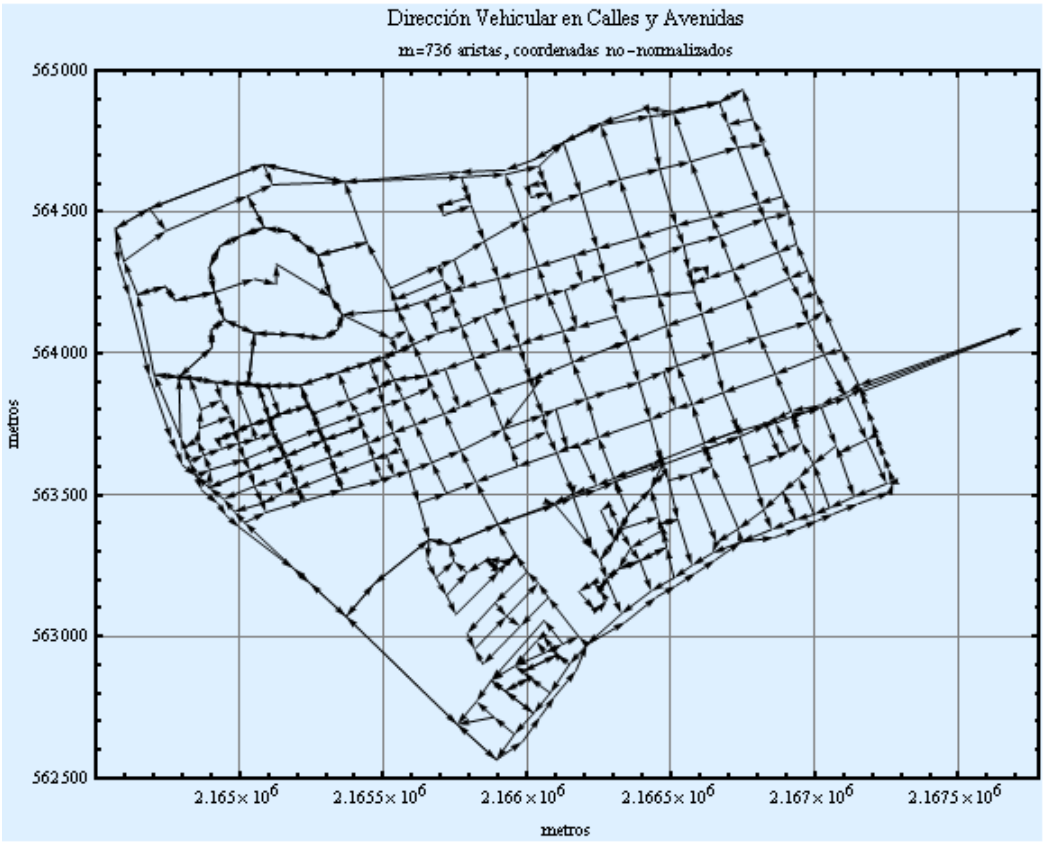


Figura 3.4 Aristas direccionadas, representan los sentidos de las avenidas

Debido a que no hubo un orden cuando se generaron las etiquetas de los vértices, se tuvo la necesidad de retiquetar los vértices, dándoles un orden descendente de izquierda a derecha con respecto al grafo. La Figura 3.5 muestra la localización de las etiquetas de los vértices, esto es el grado de una localidad de un vértice con respecto a otro.

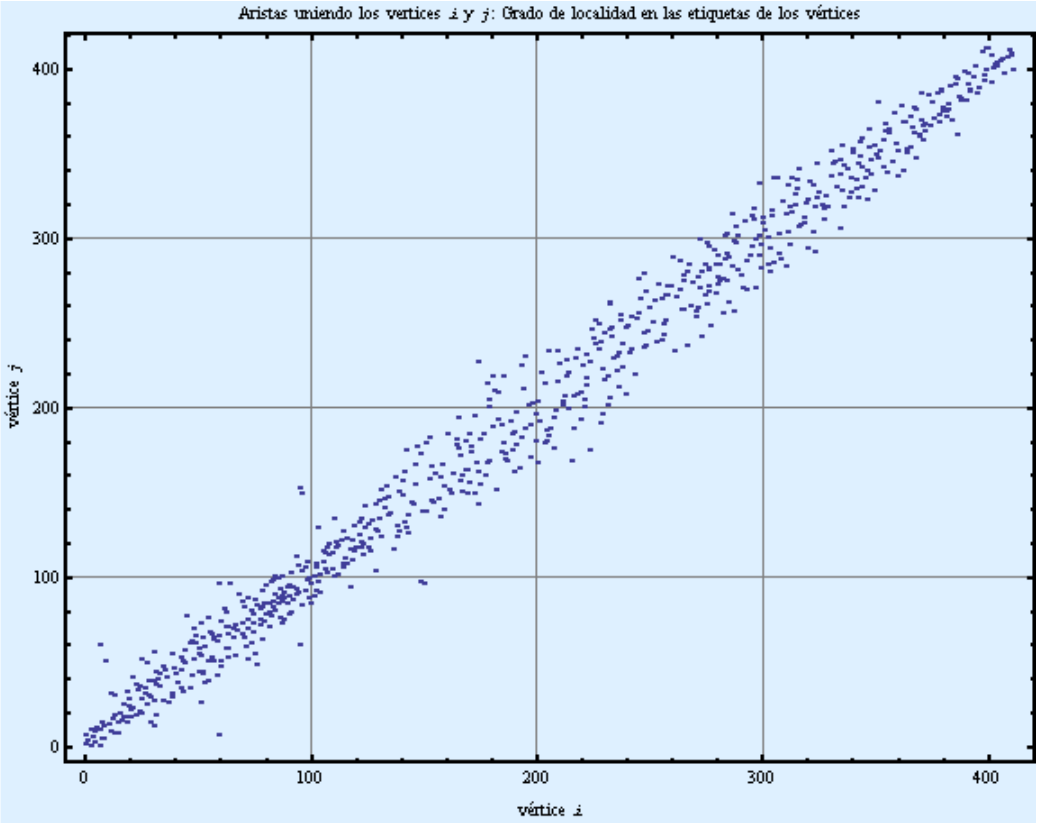


Figura 3.5 Localización de etiquetas de los vértices

La lista *verticesLabeled* contiene los vértices presentados de una manera ordenada.

```
verticesLabeled=Table[{First[verticesSorted[[i]]],Last[verticesSorted[[i]]],i},
                {i,1,Length[verticesSorted]}];
```

IV. MODELADO DE LA RED DE TRÁFICO VEHICULAR

IV.1 Cálculo de la longitud de las aristas del grafo

Una vez definidos los vértices y direcciones de las aristas dentro del grafo se calcula la longitud de las aristas que unen dos vértices respectivamente.

Se calcula la longitud de cada arista dirigida, y se almacena dicha información en la lista ***edgesWeigthed***, la cual contiene los puntos o vértices adyacentes y la longitud total de la arista que los une. Así la arista que une el vértice i al j con longitud total $w_{i,j}$ se representa como $\{\{i,j\}\rightarrow w_{i,j}\}$, se define de la siguiente manera:

```
edgesWeighted=Map[{First[#],Last[#],Floor[Norm[vertices[[First[#]]]-vertices[[Last[#]]]]]}&,a];
```

La lista *edgesWeighted* contiene todas las longitudes de las aristas en metros por consecuente de las avenidas del centro de la ciudad de Querétaro. Las longitudes mínimas y máximas obtenidas para la red de tráfico tienen los siguientes valores en metros:

8 metros para la longitud mínima y 597 metros para la longitud máxima

```
totalLengthEdge=Last/@edgesWeighted2;
{Min[totalLengthEdge],Max[totalLengthEdge]
{8,597}
```

Esto indica que la distancia mínima y máxima que se recorre para ir de un punto a otro en la red de tráfico es 8 y 597 metros respectivamente.

La Figura 4.1 muestra la relación de las longitudes de cada una de las aristas que se tienen en el grafo, teniendo un total de 736 aristas con una longitud máxima de 597 metros.

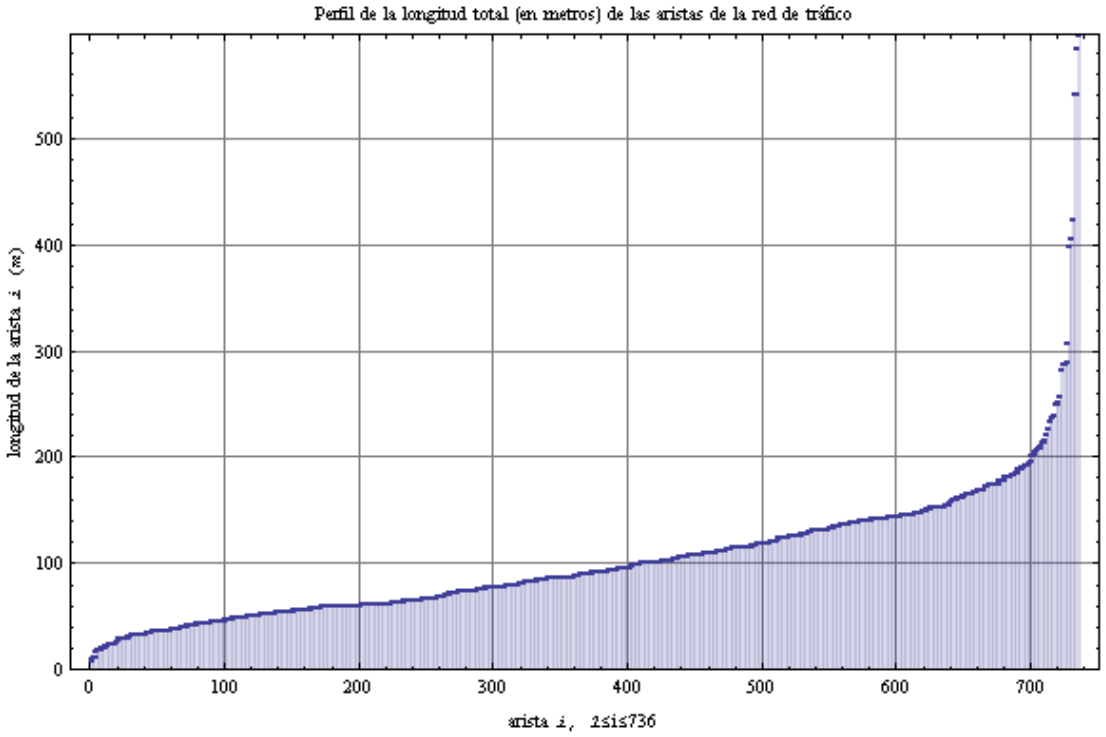


Figura 4.1 Relación de las longitudes de cada una de las aristas

IV.2 Generación de archivos de datos: Vértices, Aristas y Distancias de la red de tráfico

Una vez definidos los vértices, etiquetas de vértices y distancias de las aristas, se almacenan los datos en archivos para facilitar el manejo de la información. La lista *edgesWeigthed* consiste en la colección de aristas y su longitud total en metros del grafo de la red de tráfico, los primeros dos elementos de la lista indican la dirección de la arista y el tercero la distancia en metros:

Short[edgesWeighted,5]

```
{{1,142,288},{1,373,406},{2,1,400},{3,2,160},{4,3,105},{4,5,119},{5,4,119},{
5,20,190},{5,83,147},{5,376,144},{6,10,35},{7,6,101},{7,11,31},{8,12,41},{9,
8,59},{9,13,37},{10,11,102},{11,17,133,{394,395,80},{395,396,83},{396,397,81
},{397,344,108},{398,399,86},{399,400,75},{400,401,60},{401,402,93},{402,403
,116},{403,352,37},{404,372,584},{405,134,99},{406,81,143},{407,369,55},{408
,365,97},{409,43,44},{410,23,82},{411,33,62},{411,172,52}}
```

.La lista *edgesWeighted* es guardada en un archivo .dat con el mismo nombre como se muestra a continuación:

```
Export["C:\\maestria\\matematica\\nuevos\\edgesWeighted.dat",edgesWeighted];
```

Los vértices del grafo también se guardan en dos archivos: vertices.dat y verticesLabeled.dat. La diferencia radica en que verticesLabeled.dat incluye la información del primero más la etiqueta del vértice. Las listas correspondientes llamadas vértices y *verticesLabeled* lucen de la siguiente manera:

```
Short[vertices,5]
{{796,2041},{1195,2075},{1355,2083},{1454,2120},{1557,2180},{1129,1957},{122
6,1986},{1431,2025},{1489,2038},{1140,1923},{1237,1957},{1444,1986},{1504,20
04},{956,1666},{1124,1744},{1176,1765},{1281,1831},{1412,1907},{1520,1962},{
1617,1999},{1601,314},{1773,486},{1880,574},{1949,615},{2019,660},{2086,706}
,{2291,785},{2372,816},{2442,844},{2497,868},{2583,904},{2692,945},{2600,131
6},{2139,1152},{1866,1047},{2399,1231},{2079,1107},{1096,1314},{1015,1516},{
978,1440}}

Short[verticesLabeled,5]
{{796,2041,1},{1195,2075,2},{1355,2083,3},{1454,2120,4},{1557,2180,5},{1129,
1957,6},{1226,1986,7},{1431,2025,8},{1489,2038,9},{1140,1923,10},{1237,1957,
11},{1444,1986,12},{1504,2004,13},{956,1666,14},{1124,1744,15},{2086,706,397
},{2291,785,398},{2372,816,399},{2442,844,400},{2497,868,401},{2583,904,402}
,{2692,945,403},{2600,1316,404},{2139,1152,405},{1866,1047,406},{2399,1231,4
07},{2079,1107,408},{1096,1314,409},{1015,1516,410},{978,1440,411}}
```

A continuación los archivos tipo .dat correspondientes a las listas que contienen la información de los vértices son generadas:

```
Export["C:\\maestria\\matematica\\nuevos\\vertices.dat",vertices];
Export["C:\\maestria\\matematica\\nuevos\\verticesLabeled.dat",verticesLabeled];
```

Una vez definidos los vértices, dirección de las aristas y las distancias, se modelaron los diferentes escenarios los cuales se presentan a continuación.

IV.3 Importación de los datos del grafo

Los datos correspondientes de los vértices del grafo dirigido que modela la red de tráfico son asignados a las listas *vértices y verticesLabel* de la siguiente manera:

```
vertices=Import["C:\\maestria\\matematica\\nuevos\\vertices.dat"];
verticesLabeled=Import["C:\\maestria\\matematica\\nuevos\\verticesLabeled.dat","Data"];
```

A continuación se importan los datos de las aristas a la lista *edgesWeighted*:

```
edgesWeighted=Import["C:\\maestria\\matematica\\nuevos\\edgesWeighted.dat"];
```

IV.4 Datos básicos del grafo

El grafo de la red de tráfico consiste del siguiente numero de aristas y vértices m,n respectivamente:

```
{m,n}=Map[Length,{edgesWeighted,vertices}]
```
{736,412}

A las aristas almacenadas en la lista *edgesWeigthed* se les asigna un formato de la forma {{v_1, v_2}, *distancia*}, donde v_1 y v_2 nos indican la dirección de la arista y el tercer elemento la distancia en metros entre los vértices, esta lista es asignada a la variable *edgesWeigthed1*.

```
edgesWeighted1=Map[{{#[[1]],#[[2]]}, #[[3]]}&,edgesWeighted];
```

```
Short[edgesWeighted1,5]
```
{{{3,2},160},{{373,7},75},{{4,3},105},{{18,374},169},{{5,4},119},{{375,9},60},{{5,20},190},{{7,6},101},{{6,10},35},{{7,11},31},{{9,8},59},{{8,12},41},{{9,13},37},{{10,11},102},{{11,17},133},{{12,13},62},{{156,151},102},{{158,156},115},{{248,158},132},{{250,249},60},{{159,250},53},{{157,159},74},{{154,157},153},{{150,151},139},{{149,150},60},{{148,149},101},{{247,156},57},{{251,247},62},{{162,251},154},{{166,158},179},{{287,276},61},{{249,248},109}}

La lista *edgesWeigthed2* contiene las aristas ordenadas con respecto a la distancia, en forma ascendente.

```
edgesWeighted2=Sort[edgesWeighted1,Last[#1]<Last[#2]&];
edgesWeighted2=Map[#[[1]] → #[[2]]&,edgesWeighted2];

Short[edgesWeighted2,5]
{{295,359}→8,{82,363}→10,{362,81}→12,{79,360}→12,{258,261}→19,{368,136}→20,{
309,310}→20,{364,134}→21,{137,370}→22,{135,365}→22,{315,314}→23,{306,305}→23
,{352,350}→24,{363,353}→25,{371,138}→25,{317,315}→25,{286,282}→25,{146,1}→25
1,{328,334}→256,{277,276}→282,{142,1}→288,{1,142}→288,{161,148}→289,{380,381
}→290,{144,145}→309,{2,1}→400,{1,373}→406,{141,142}→425,{142,141}→425,{236,2
74}→542,{274,236}→542,{404,372}→584,{371,372}→596,{372,138}→597}
```

En seguida se construye la matriz de adyacencias ***matDis*** modificada a partir de la información obtenida de las aristas. La matriz ***matDis*** se define como una matriz porosa permitiendo solo almacenar la información de las aristas existentes. Así la entrada i,j consiste en la longitud de la arista (i,j).

```
matDis=SparseArray[edgesWeighted2];
```

Cada posición i,j de la matriz contendrá un 0 si no existe una arista que una el vértice i con el vértice j y un 1 en caso contrario. La matriz generada tiene un tamaño de 411x411.

IV.5 Algoritmo de Dijkstra

El algoritmo de Dijkstra también conocido como el algoritmo de caminos mínimos es un algoritmo para la determinación del camino más corto dado un vértice origen al resto de vértices en un grafo dirigido y con pesos o distancias en cada arista.

La idea principal de este algoritmo consiste en ir explorando todos los caminos más cortos que parten del vértice origen y que llevan a todos los demás vértices; cuando se obtiene el camino más corto desde el vértice origen al resto de vértices que componen el grafo el algoritmo termina. El algoritmo es una especialización de la búsqueda de costo uniforme, y como tal, no funciona en grafos con aristas de costos negativos. Con el algoritmo de Dijkstra podemos calcular dado un punto inicial la mínima distancia a cada uno de los puntos restantes.

El orden de complejidad del algoritmo es $O(|V|^2+|E|) = O(|V|^2)$.

Aplicando el algoritmo para nuestros datos obtuvimos las distancias mínimas de un punto inicial dado a todos los puntos marcados en el mapa, a continuación algunos ejemplos de las distancias obtenidas.

Dijkstra[Graph[e,v1,EdgeDirection→True],1]//Timing

```
{0.541,{{1,1,2,3,4,7,2,9,4,6,7,8,9,21,14,15,11,17,13,19,23,21,24,25,26,27,17
,38,30,20,24,25,43,31,32,35,27,37,38,39,30,47,44,45,35,36,48,49,50,39,41,48,
54,44,45,52,58,59,60,51,56,231,55,63,64,65,66,60,63,64,72,69,70,65,76,77,68,
295,73,79,82,77,5,83,84,85,86,86,88,95,20,91,92,93,94,101,92,103,94,105,41,9
7,102,99,104,102,106,106,107,111,104,119,114,108,123,114,125,111,118,118,120
,68,122,123,124,120,126,122,130,124,136,126,138,82,130,135,132,137,113,141,1
42,1,140,141,144,142,145,143,150,151,242,151,155,153,152,151,155,156,250,159
,148,161,162,163,164,165,166,167,168,171,188,33,176,164,161,179,174,175,191,
181,177,181,167,168,169,185,170,189,43,178,190,191,181,193,183,190,196,193,1
95,207,187,215,189,196,198,199,213,201,204,205,212,206,214,215,216,53,209,21
0,211,212,213,214,224,230,217,220,221,222}
```

IV.6 Longitud de rutas optimas entre dos puntos

Dada la matriz *matDis* que contienen la distancia entre vértices, se pueden calcular las distancias o rutas óptimas, desde un vértice i a un vértice j utilizando la función *GraphDistance* de Mathematica, la cual utiliza el algoritmo de Dijkstra para obtener dichas rutas.

```
allDis=Table[Floor[GraphDistance[matDis,i,j,Weighted→True]],{i,n},{j,n}];]
```

La lista *allDis* contiene todas las rutas óptimas posibles que se obtienen de ir de un vértice i a un vértice j.

La Figura 4.2 muestra la relación de la longitud total que une el vértice i con el vértice j, la zona con más intensidad del color indica menor longitud.

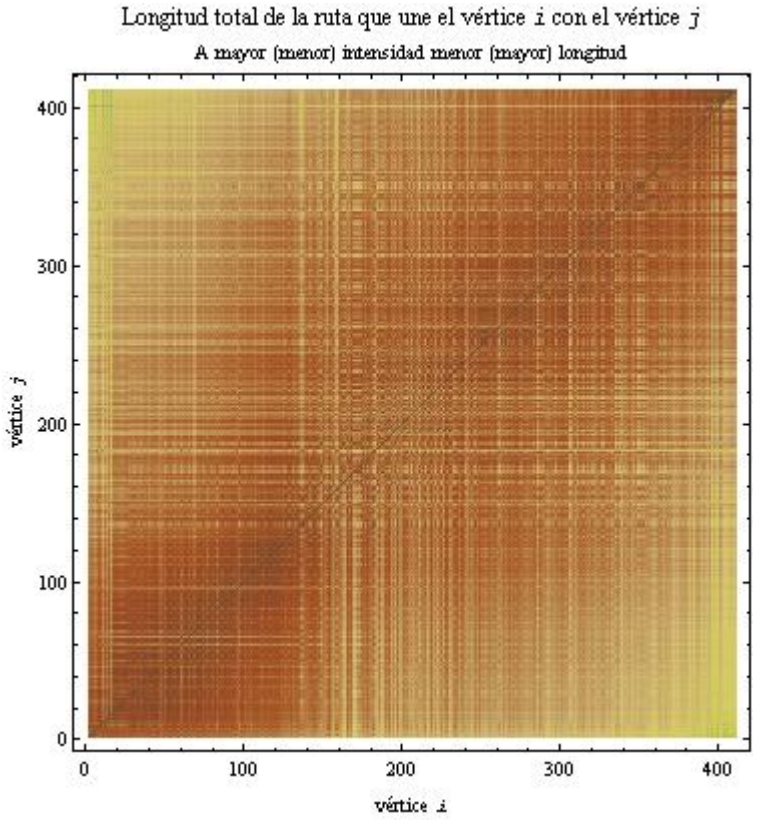


Figura 4.2 Relación de la longitud total que une el vértice i con el vértice j

Una vez calculadas todas las posibles rutas óptimas y almacenadas en la lista *allDist* se observa que se tiene un total de 169,332 rutas óptimas posibles dentro de la red de tráfico analizada, la variable *totalLengthPath* almacena dichas rutas.

totalLengthPath=Sort[Select[Flatten[allDis],#!=0&]];

La longitud promedio de rutas óptimas uniendo un par vértices de la red de tráfico es de 1645.79 metros y se puede definir de la siguiente manera:

averageLength=Plus@@(Plus@@#&)@allDis/((n^2-n)//N
1645.79

Select[Flatten[allDis],#→averageLength&]//Length/n^2//N
0.478644

IV.7 Representaciones del mapa

Una vez definido los vértices así como las direcciones de las aristas con las respectivas distancias y las rutas óptimas, se puede representar el grafo de la siguiente manera:

Primeramente se obtiene las posiciones del conjunto de vértices definido

positions=Table[i→vertices[[i]],{i,1,n}];

```
{1→{796,2041},2→{1195,2075},3→{1355,2083},4→{1454,2120},5→{1557,2180},6→{1
129,1957},7→{1226,1986},8→{1431,2025},9→{1489,2038},10→{1140,1923},11→{1237
,1957},12→{1444,1986},13→{1504,2004},14→{956,1666},15→{401→{2497,868},402→
{2583,904},403→{2692,945},404→{2600,1316},405→{2139,1152},406→{1866,1047},4
07→{2399,1231},408→{2079,1107},409→{1096,1314},410→{1015,1516},411→{978,144
0}}}
```

Con la siguiente función se puede graficar los vértices con los respectivos sentidos o direcciones de las calles.

GraphPlot[matDis,DirectedEdges→True,VertexCoordinateRules→positions]

El siguiente grafo representa los 411 vértices y las 736 aristas o avenidas dirigidas de la red de tráfico analizada para el centro de la ciudad de Querétaro, Figura 4.3

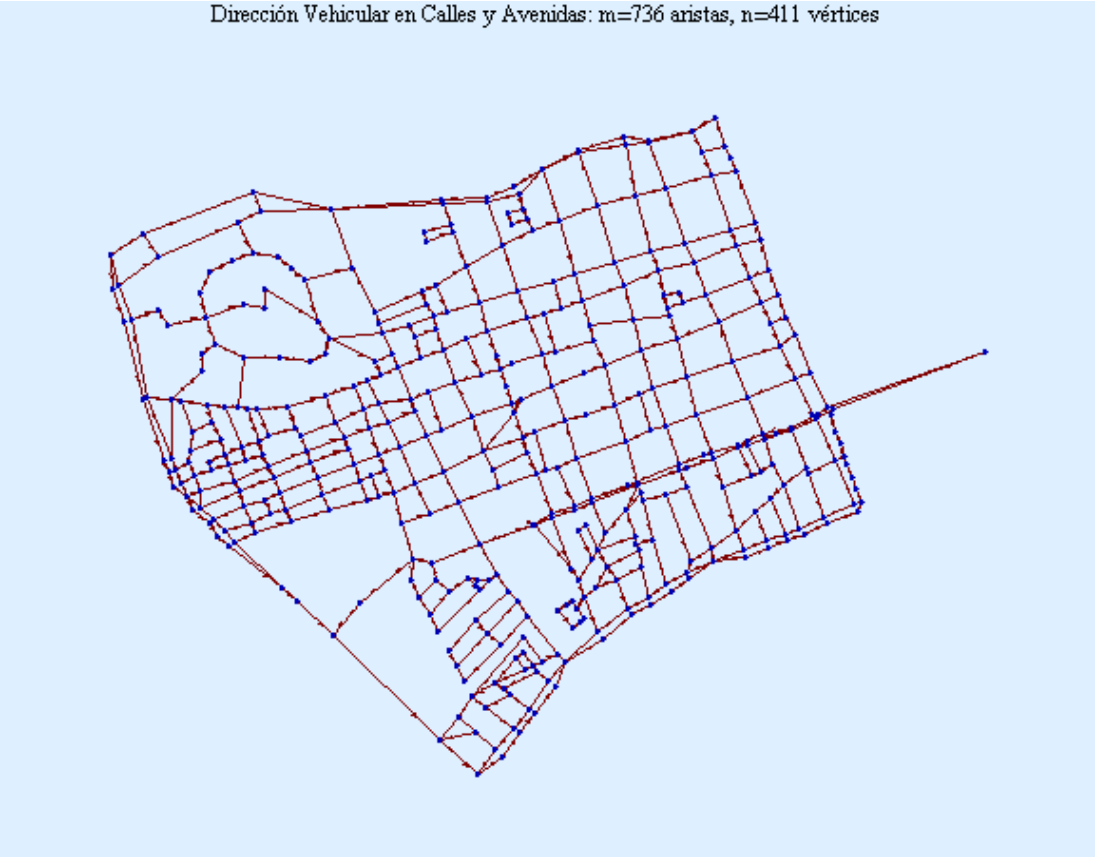


Figura 4.3 Todos los vértices y direcciones de las aristas

IV.8 Rutas óptimas más largas

La ruta más larga corresponde a la que conecta el vértice i al vértice j. Dicha ruta óptima {i, j} cuya longitud total corresponde a la entrada {i, j} en la matriz *allDis* posee el valor máximo entre todos los elementos de *matDis*. A continuación la longitud total máxima y los vértices {i,j}:

max=Max[allDis]
Position[allDis,max]

Longitud máxima 5464m

```
{401,16}
```

La ruta óptima de longitud máxima es de 5464 metros, la cual inicia en el vértice 401 y termina en el vértice 16 de la red de tráfico.

Esta ruta óptima más larga consiste en la siguiente secuencia de vértices:

path=ShortestPath[g=Graph[edgesWeighted3,vertices1,EdgeDirection→True],401,16]

```
{401,399,393,388,383,379,373,367,365,353,351,349,346,335,306,286,262,233,206
,183,151,96,60,7,1,2,4,6,11,13,16}
```

Se puede expresar la misma ruta de longitud máxima, pero en una secuencia de aristas como vemos a continuación:

edges=Flatten[Drop[Table[{{path[[i]],RotateLeft[path][[i]]}},{i,1,Length[path]}],-1],1]

En la lista *edges* se define la ruta óptima máxima de vértice a vértice:

```
{{401,399},{399,393},{393,388},{388,383},{383,379},{379,373},{373,367},{367,
365},{365,353},{353,351},{351,349},{349,346},{346,335},{335,306},{306,286},{
286,262},{262,233},{233,206},{206,183},{183,151},{151,96},{96,60},{60,7},{7,
1},{1,2},{2,4},{4,6},{6,11},{11,13},{13,16}}
```

Dado que el conjunto de vértices en *edges* es un subconjunto del conjunto *edgesWeighted*, se puede encontrar dicha ruta óptima *edges* y plasmarla en el grafo. El conjunto de aristas del grafo expresado en otro formato se encuentra en la lista *edgesWeightedNew1* como sigue:

```
edgesWeightedNew1=Map[{{#[[1]],#[[2]]},EdgeWeight→ Last[#]}&,edgesWeighted];
```

Se busca modificar los colores de las aristas correspondientes a la ruta óptima seleccionada dada por el conjunto de aristas en la lista *edges*. Así que se construye el patrón a buscar en *edgeWeigthedNew1*.

```
Map[{First[#],Last[#],__}&,edges]
{400,399,__},{399,393,__},{393,388,__},{388,383,__},{383,379,__},{379,373,__
},{373,367,__},{367,365,__},{365,353,__},{353,351,__},{351,349,__},{349,346,
__},{346,335,__},{335,306,__},{306,286,__},{286,262,__},{262,233,__},{233,20
6,__},{206,183,__},{183,151,__},{151,96,__},{96,60,__},{60,7,__},{7,1,__},{1
,2,__},{2,4,__},{4,6,__},{6,11,__},{11,13,__},{13,16,__}}
```

Se calcula las posiciones que guardan las aristas de la ruta óptima dada en *edges*, en la lista *edgesWeightedNew1*:

```
Flatten[Map[Position[edgesWeightedNew1,#]&,Map[{{First[#],Last[#]},__}&,edges]]]
```

```
{683,671,216,206,195,704,181,170,161,143,152,142,139,137,135,133,131,123,5,3
,1,695,225,223,221,222,228,235,254,279,284}
```

Tales posiciones son usadas para modificar las directivas de las aristas en el grafo, particularmente el color de la arista. Así que los cambios quedan de la siguiente manera:

```
changes=Map[
            edgesWeightedNew1[[#]]→Union[edgesWeightedNew1[[#]],
            {EdgeColor→Red,VertexColor→Red}]&,
            Flatten[Map[Position[edgesWeightedNew1,#]&,
            Map[{{First[#],Last[#]},__}&,edges]]]
          ];
```

Finalmente se genera el conjunto de aristas correspondientes a la ruta que nos interesa colorear distintivamente de las demás:

```
edgesWeightedNew1=edgesWeightedNew1/.changes;
g2=Graph[edgesWeightedNew1,vertices1];
```

Los datos son generados y almacenados en el grafo *g2*, la ruta óptima se representa en color rojo dentro del grafo como se puede ver a continuación, Figura 4.4:

ShowGraph[g2,VertexStyle→Disk[Small],EdgeStyle→Thickness[.001]]

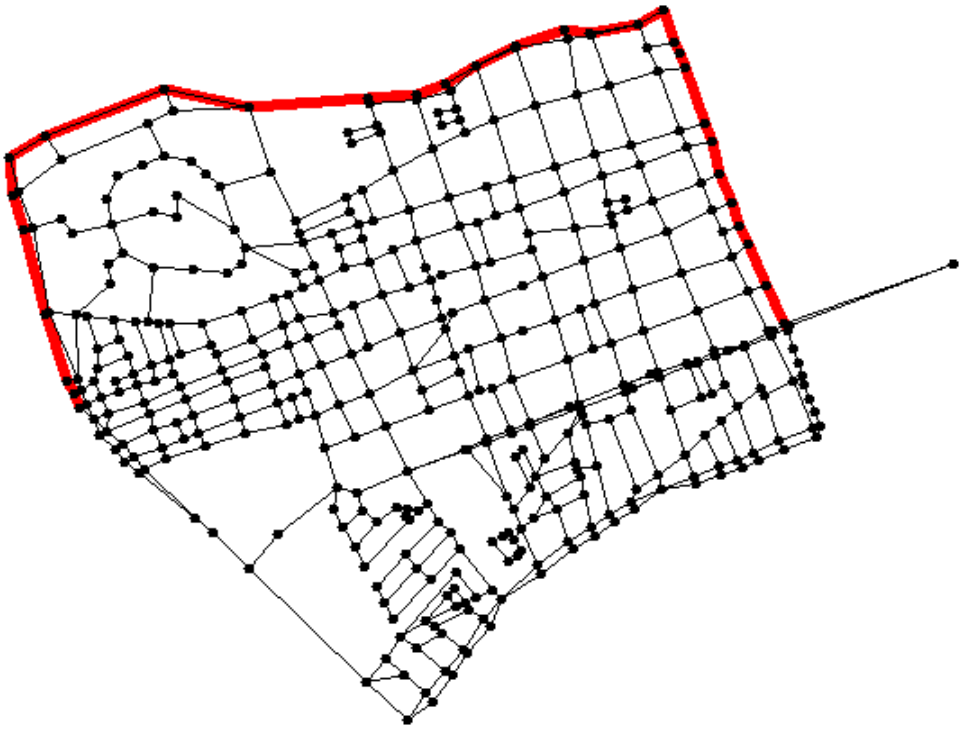


Figura 4.4 Ruta óptima de longitud máxima (5464 m)

Se puede analizar la misma ruta óptima pero ahora desde el punto j al punto i (inversa al caso anterior), la ruta va del vértice 16 al 401:

path1=ShortestPath[g=Graph[edgesWeighted3,vertices,EdgeDirection→True],16,401]

```
{16,20,24,33,40,46,77,83,97,104,129,143,175,227,238,255,273,299,332,344,357,
362,378,386,395,401}
```

edges=Flatten[Drop[Table[{{path1[[i]],RotateLeft[path1][[i]]}},{i,1,Length[path1]}],-1],1]

```
{{16,20},{20,24},{24,33},{33,40},{40,46},{46,77},{77,83},{83,97},{97,104},{1
04,129},{129,143},{143,175},{175,227},{227,238},{238,255},{255,273},{273,299
},{299,332},{332,344},{344,357},{357,362},{362,378},{378,386},{386,395},{395
,401}}
```

edgesWeightedNew2=Map[{{#[[1]],#[[2]]},EdgeWeight→Last[#]}&,edgesWeighted];

Calculamos las posiciones de las aristas dentro de *edgesWeightedNew2*

Flatten[Map[Position[edgesWeightedNew2,#]&,Map[{{First[#],Last[#]},__}&,edges]]]

```
{707,372,708,709,710,711,456,458,460,463,465,467,135,668,670,671,673,676,731
,680,682,683,730,687,689}
```

Las posiciones son usadas para modificar las directivas de las aristas en el grafo, particularmente el color de la arista. Así que los cambios se visualizan de la siguiente manera:

```
changes=Map[
        edgesWeightedNew2[[#]]→Union[edgesWeightedNew2[[#]],
        {EdgeColor→Red,VertexColor→Red}]&,
        Flatten[Map[Position[edgesWeightedNew2,#]&,
        Map[{{First[#],Last[#]},__}&,edges]]]
    ];
```

Finalmente el conjunto de aristas correspondientes a la ruta que interesa colorear distintivamente de las demás, se genera:

```
edgesWeightedNew2=edgesWeightedNew2/.changes;
```

Los datos son generados y almacenados en el grafo *g3*, la ruta óptima se representa en color rojo dentro del grafo como se puede ver a continuación, Figura 4.5:

```
g3=Graph[edgesWeightedNew2,vertices1];
ShowGraph[g3,VertexStyle→Disk[Small],EdgeStyle→Thickness[.001]];
```

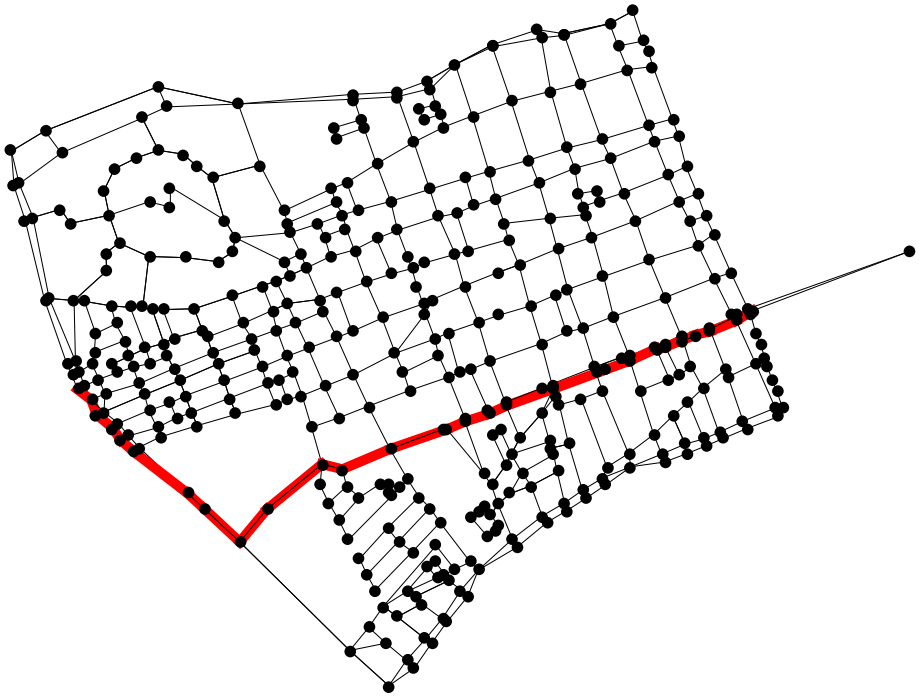


Figura 4.5 Ruta óptima del vértice 16 al vértice 401

IV.9 Frecuencia de uso de aristas por rutas óptimas entre dos vértices

La entrada {i,j} de la matriz producida por la función GraphDistanceMatrix corresponde a la longitud total de la ruta del vértice i al j.

GraphDistanceMatrix[matDis]//MatrixForm

La función GraphDistanceMatrix calcula todas las distancias entre cada uno de los vértices a partir de la matriz porosa **matDis.** Asociada a la matriz anterior se obtiene la secuencia de vértices que forman la ruta óptima que va del vértice i al j:

p=Table[GraphPath[matDis,i,j],{i,n},{j,n}]

```
{598.731,{{{1},{1,2},{1,7,10,50,61,47,35,27,31,12,9,5,3},≪407≫,{1,7,60,96,
153,182,210,215,221,225,246,276,297,311,314,315,320,321,323,331,337,343,344,
348,354,358,345,359,374,384,390,398,410,411},{1,7,60,96,153,182,210,215,221,
225,246,276,297,317,341,350,355,363,369,376,380,387,394,395,401,412}},≪410
≫,{≪1≫}}}
```

Enseguida se recuperan la aristas que son usadas en todas las rutas que van desde i a j, para todo i,j:

q=Flatten[Map[Partition[#,2,1]&,p,{2}],2];

```
{{1,2},{1,7},{7,10},{10,50},{50,61},{61,47},{47,35},{35,27},{27,31},{31,12},
{12,9},{9,5},{5,3},{1,2},{2,4},{1,7},{7,10},{10,50},{50,61},{61,47},{47,35},
{35,27},{27,31},{31,12},{12,9},{9,5},{1,2},{2,4},{4,6},{1,7},{1,2},{2,4},{4,
6},{6,11},{11,13},{13,16},{16,20},{20,18},{18,17},{17,15},{15,8},{1,7},{7,10
},{10,50},{50,61},{61,47},<<2787141>>,{384,390},{390,398},{398,410},{410,411
},{411,409},{412,399},{399,393},{393,387},{387,394},{394,395},{395,402},{402
,392},{392,391},{391,381},{381,375},{375,366},{366,354},{354,358},{358,345},
{345,359},{359,374},{374,384},{384,390},{390,398},{398,410},{412,399},{399,3
93},{393,387},{387,394},{394,395},{395,402},{402,392},{392,391},{391,381},{3
81,375},{375,366},{366,354},{354,358},{358,345},{345,359},{359,374},{374,384
},{384,390},{390,398},{398,410},{410,411}}
```

A continuación se calcula la frecuencia con que cada arista es parte de una ruta óptima uniendo cualquier par de puntos:

freq=Tally[q];

Short[freq,10]

```
{{{1,2},2934},{{1,7},443},{{7,10},568},{{10,50},1085},{{50,61},998},{{61,47}
,1192},{{47,35},967},{{35,27},757},{{27,31},1091},{{31,12},1732},{{12,9},138
1},{{9,5},1043},{{5,3},3245},{{2,4},2934},{{4,6},2934},{{6,11},2934},{{11,13
},2934},{{13,16},2934},{{16,20},2934},{{20,18},2582},{{18,17},2212},{{17,15}
,2260},{{15,8},806},{{15,14},1454},{{20,28},1727},<<686>>,{{311,283},5854},{
{343,324},399},{{267,280},185},{{275,266},871},{{292,271},2520},{{285,275},4
22},{{286,289},607},{{306,335},46},{{350,328},2985},{{392,391},2521},{{411,3
98},892},{{327,320},1402},{{340,336},2856},{{361,352},3469},{{381,377},3577}
,{{373,367},2670},{{361,337},6204},{{346,335},1296},{{339,324},2335},{{348,3
64},140},{{368,362},6},{{371,386},450},{{366,354},3030},{{393,388},1980},{{3
72,358},1813}}
```

Se ordena dicha frecuencia de aristas en forma ascendente

freqSort=Sort[freq,Last[#1]≤Last[#2]&];

Short[freqSort,10]

```
{{{243,254},2},{{170,163},3},{{368,362},6},{{184,193},14},{{400,401},28},{{3
82,375},32},{{172,174},42},{{306,335},43},{{372,366},44},{{21,17},48},{{135,
137},64},{{115,127},69},{{173,149},99},{{63,71},109},{{300,282},118},{{180,2
05},119},{{142,162},122},{{348,364},139},{{34,43},158},{{180,169},158},{{131
,145},164},{{296,313},168},{{335,346},182},{{267,280},186},<<688>>,{{125,129
},12580},{{117,123},12641},{{227,238},12850},{{239,238},13044},{{244,220},13
139},{{198,190},13139},{{143,175},13226},{{122,125},13361},{{154,165},13419}
,{{251,259},13739},{{319,309},14070},{{190,175},14241},{{209,198},14373},{{3
09,288},14391},{{266,278},14558},{{139,124},14900},{{124,113},14952},{{259,2
66},16343},{{152,139},17995},{{278,248},18719},{{238,255},18803},{{255,273},
18803},{{175,227},19002},{{162,152},19597}}
```

Se puede representar la frecuencia de aristas en forma no decreciente:

freqSortMod=Map[Last[#]&,freqSort];

Short[freqSortMod,10]

```
{2,3,6,14,32,36,42,43,44,48,64,69,99,109,118,119,122,139,158,158,164,168,182
,186,190,228,234,253,256,258,259,260,274,297,303,313,313,316,321,325,334,338
,344,360,370,371,371,373,376,378,382,382,389,398,410,411,411,414,414,424,437
,439,440,441,444,450,452,456,460,460,467,467,481,488,495,496,502,505,507,512
,518,518,527,527,<<568>>,7762,7764,7790,7815,7906,7941,7999,8091,8128,8234,8
292,8294,8363,8524,8563,8563,8580,8584,8801,8955,9020,9145,9275,9372,9374,94
60,9519,9646,9694,9838,10126,10126,10251,10277,10378,10435,10944,10954,11002
,11010,11063,11241,11311,11395,11441,11571,11700,11743,11748,11903,11903,119
27,11984,12184,12252,12278,12394,12487,12487,12565,12580,12641,12850,13044,1
3139,13139,13226,13361,13419,13739,14070,14241,14373,14391,14558,14900,14952
,16343,17995,18719,18803,18803,19002,19597}
```

La Figura 4.6 muestra el perfil de las aristas en términos de la frecuencia de uso de cada arista. Así que el punto {x,y} representa que la arista etiquetada con el numero de seguimiento *x* se usa en *y* numero de rutas optimas uniendo dos puntos en el grafo.

```
ListPlot[
        freqSortMod,GridLines→Automatic,Frame→True,PlotRange→
       {0,Last[freqSortMod]},Filling→Axis

]
```

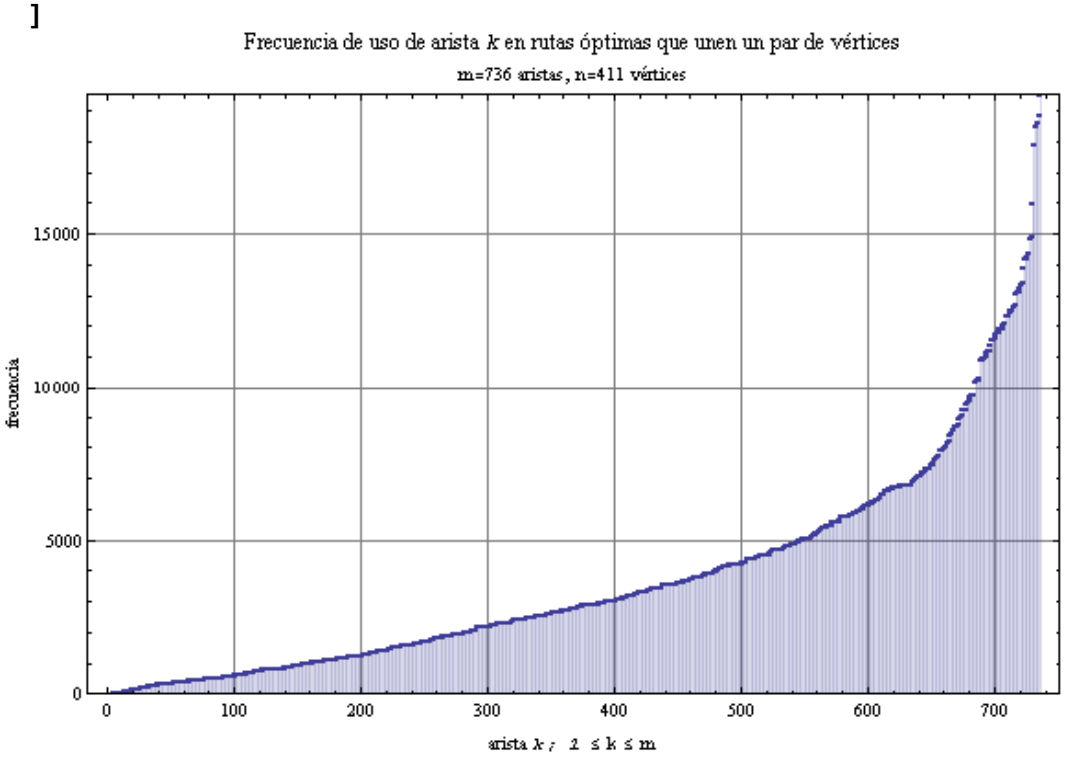


Figura 4.6 Perfil de las aristas en términos de la frecuencia de uso

Se observa que la arista más usada consiste en aquella que es parte del siguiente número de rutas óptimas entre dos vértices del grafo:

```
Last[freqSortMod]
19597
freqSortMod//Length
736
```

El número de rutas óptimas para la red de tráfico vehicular está dada por:

```
Binomial[n,2]
84666
```

Ahora, ¿cuál es el número de aristas que son usadas un número en particular de veces como parte de rutas optimas uniendo dos puntos?

```
freqSortMod1=Tally[freqSortMod];
```

La siguiente lista contiene la frecuencia de uso de las aristas, el primer elemento de la lista nos indica el número de veces que es utilizada una arista y el segundo indica el número de aristas, por ejemplo, existe una arista que es usada 2 veces, una que es usada 3 veces y una que es usada 19597 veces como se muestra a continuación:

```
Short[freqSortMod1,10]
```


```
{{2,1},{3,1},{6,1},{14,1},{32,1},{36,1},{42,1},{43,1},{44,1},{48,1},{64,1},{
69,1},{99,1},{109,1},{118,1},{119,1},{122,1},{139,1},{158,2},{164,1},{168,1}
,{182,1},{186,1},{190,1},{228,1},{234,1},{253,1},{256,1},{258,1},{259,1},{26
0,1},{274,1},{297,1},{303,1},{313,2},{316,1},{321,1},{325,1},{334,1},{338,1}
,{344,1},{360,1},{370,1},{371,2},{373,1},□545□,{10435,1},{10944,1},{10954,1}
,{11002,1},{11010,1},{11063,1},{11241,1},{11311,1},{11395,1},{11441,1},{1157
1,1},{11700,1},{11743,1},{11748,1},{11903,2},{11927,1},{11984,1},{12184,1},{
12252,1},{12278,1},{12394,1},{12487,2},{12565,1},{12580,1},{12641,1},{12850,
1},{13044,1},{13139,2},{13226,1},{13361,1},{13419,1},{13739,1},{14070,1},{14
241,1},{14373,1},{14391,1},{14558,1},{14900,1},{14952,1},{16343,1},{17995,1}
,{18719,1},{18803,2},{19002,1},{19597,1}}
```


El punto {x,y} de la gráfica 4.7 corresponde a que la(s) arista(s) que es(son) usada(s) en ***x*** número de rutas óptimas uniendo dos puntos aparecen ***y*** veces:

```
ListPlot[
        freqSortMod1,GridLines→Automatic,Frame→True,PlotRange→
        {0,Max[Last/@freqSortMod1]},Filling→Axis
       ]
```

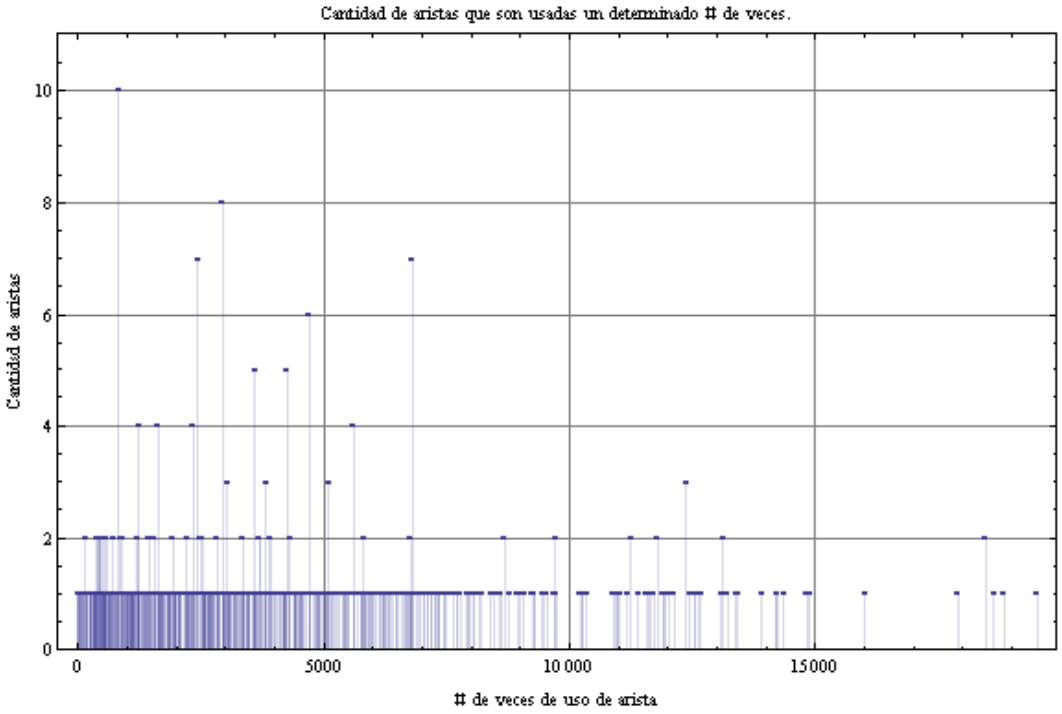


Figura 4.7 Número de veces que es usada una Arista en las rutas óptimas

De interés es el hecho de que solo hay una arista que se utiliza un número máximo de veces, 19597 es usada esta arista:

First[Last[freqSortMod1]]
19597

Siendo esta arista la número 736
arista=Flatten[Position[freqSort,{__,First[Last[freqSortMod1]]}]]{736}

La cuál va del vértice 162 al 152
freqSort[[First[arista]]]
{{162,152},19597}

Se puede saber que el tamaño total de la lista *freqSortMod1*

Length[freqSortMod1]
637

En seguida se obtiene el 10% de las aristas más usadas de nuestra red de tráfico de la manera siguiente:

altasFreq=First/@Take[freqSortMod1,-.1Length[freqSortMod1]//Ceiling]

```
{8580,8584,8801,8955,9020,9145,9275,9372,9374,9460,9519,9646,9694,9838,10126
,10273,10277,10378,10435,10944,10954,11002,11010,11063,11241,11311,11395,114
41,11571,11700,11743,11748,11903,11927,11984,12184,12252,12278,12394,12487,1
2565,12580,12641,12850,13044,13139,13226,13361,13419,13739,14070,14241,14373
,14391,14558,14900,14952,16343,17995,18719,18803,19002,19597}
```

En el arreglo *altasFeq* se obtiene el promedio de las aristas más usadas a partir de freqSortMod1. El número de aristas promedio más usadas es de 63 que es el 10% del total.

Las posiciones de las aristas promedio más usadas son:

position=Map[Flatten[Position[freqSort,{__,#}]]&,altasFreq]

```
{{669},{670},{671},{672},{673},{674},{675},{676},{677},{678},{679},{680},{68
1},{682},{683,684},{685},{686},{687},{688},{689},{690},{691},{692},{693},{69
4},{695},{696},{697},{698},{699},{700},{701},{702,703},{704},{705},{706},{70
7},{708},{709},{710,711},{712},{713},{714},{715},{716},{717,718},{719},{720}
,{721},{722},{723},{724},{725},{726},{727},{728},{729},{730},{731},{732},{73
3,734},{735},{736}}
```

Las aristas promedio más usadas son almacenadas en la lista *aristasMasUsadas*:

aristasMasUsadas=First/@Map[freqSort[[First[#]]]&,position]

```
{{324,319},{115,108},{290,281},{248,244},{118,116},{288,298},{248,236},{243,
235},{257,243},{116,111},{118,94},{235,229},{127,123},{288,257},{299,332},{3
29,325},{139,130},{337,319},{229,191},{123,121},{273,271},{123,134},{201,204
},{113,102},{175,162},{129,143},{229,239},{107,114},{121,118},{95,107},{214,
218},{130,127},{181,189},{220,209},{114,122},{218,229},{204,214},{273,299},{
134,154},{192,197},{165,181},{125,129},{117,123},{227,238},{239,238},{244,22
0},{143,175},{122,125},{154,165},{251,259},{319,309},{190,175},{209,198},{30
9,288},{266,278},{139,124},{124,113},{259,266},{152,139},{278,248},{238,255}
,{175,227},{162,152}}
```

Se calculan las posiciones que guardan las aristas promedio más usadas dadas, en la lista a*ristasMasUsadas* sobre la lista *edgesWeightedNew3*:

Flatten[Map[Position[edgesWeightedNew3,#]&,Map[{{First[#],Last[#]},__}&,
aristasMasUsadas]]];

Las posiciones son usadas para modificar las directivas del promedio de las aristas más usadas en el grafo, particularmente el color de la arista.

```
changes=Map[ edgesWeightedNew3[[#]]→Union[edgesWeightedNew3[[#]],
                    {EdgeColor→Red,EdgeStyle→Thickness[0.01],
                    VertexColor→Red}]&,
                    Flatten[Map[Position[edgesWeightedNew3,#]&,
                    Map[{{First[#],Last[#]},__}&,aristasMasUsadas]]]
                ];
```

Finalmente el conjunto de aristas promedio más usadas, se genera:

```
edgesWeightedNew3=edgesWeightedNew3/.changes;
```

Los datos son generados y almacenados en el grafo g3 en color rojo dentro del grafo como se puede ver en la Figura 5.8:

```
g3=Graph[edgesWeightedNew3,vertices1];
ShowGraph[g3,VertexStyle→Disk[Small],EdgeStyle→Thickness[.001]]
```

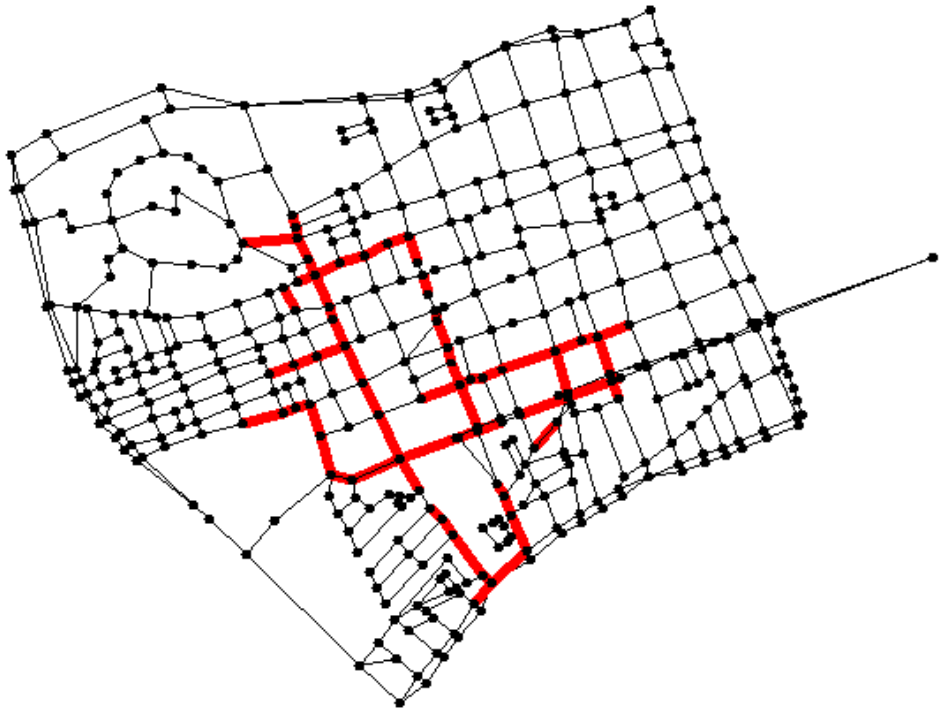


Figura 4.8 El número de aristas promedio más usadas

V. EVALUACIÓN EXPERIMENTAL

V.1 Frecuencia de uso de aristas por rutas óptimas cambiando los sentidos de las Avenidas del centro de Querétaro.

En la siguiente sección se analizan las gráficas de la frecuencia de aristas más usadas, haciendo el cambio del sentido de algunas de las avenidas principales del centro de la ciudad y aplicando los mismos procedimientos del análisis anterior para el cálculo de las aristas más usadas.

Cambio del sentido de la Avenida Tecnológico

Cambiado el sentido de la Avenida Tecnológico actual de (Sur a Norte) a (Norte Sur) y generando la frecuencia de aristas más usadas, podemos observar que se libera la utilización de las aristas que comprenden la Avenida Ignacio Pérez distribuyendo la frecuencia de uso con la Avenida Régules Figura 5.1

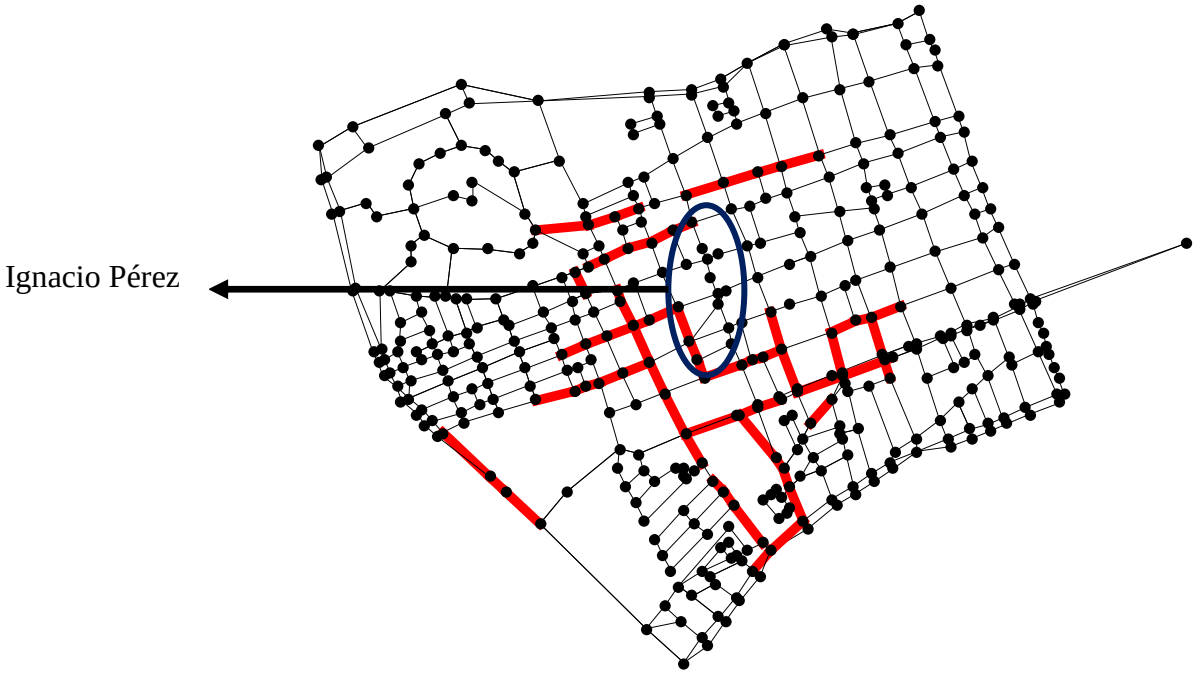


Figura 5.1 Aristas más usadas cambiado el sentido de la Avenida Tecnológico

Cambio a doble sentido de la Avenida Tecnológico

Si se utiliza la Avenida de Tecnológico en doble sentido y se efectúa nuevamente el procedimiento para la generación de la frecuencia de las aristas más usadas, observamos que las Avenidas de Ignacio Pérez y Régules son liberadas (ya no aparecen marcadas como aristas más usadas), esto significa que se mejora el promedio de uso de dichas avenidas, Figura 5.2

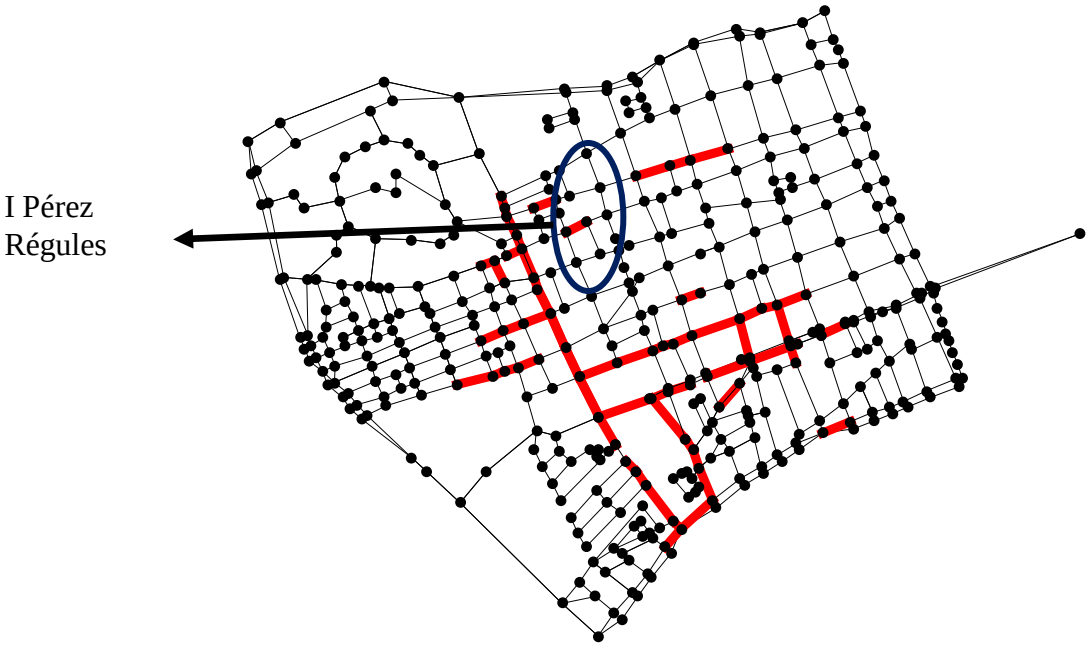


Figura 5.2 Aristas promedio más usadas considerando la AvenidaTecnológico en doble sentido

Cambio del sentido de la Avenida Ezequiel Montes

Cambiado el sentido de la Avenida Ezequiel Montes actual de (Sur a Norte) a (Norte Sur) y generando la frecuencia de aristas más usadas, se puede observar que se libera la utilización de las aristas que comprenden las Avenidas de Arteaga y Régules, Figura 5.3

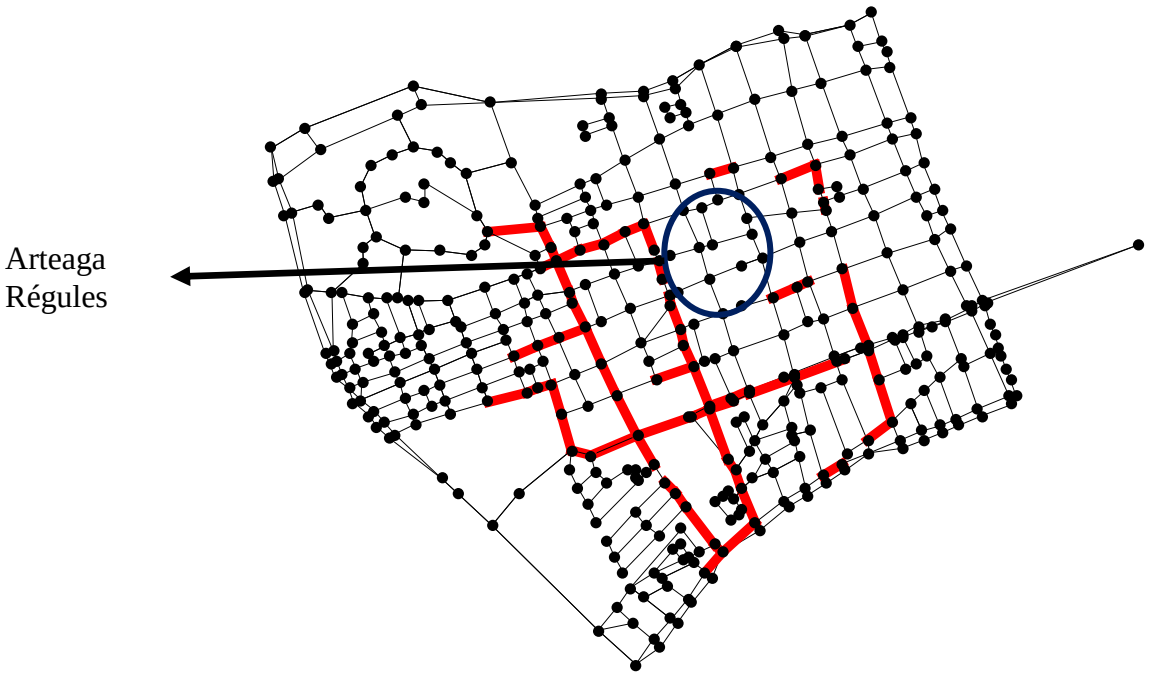


Figura 5.3 Aristas más usadas cambiado el sentido de la Avenida Ezequiel Montes

Cambio a doble sentido la calle Ezequiel Montes

Utilizando la calle de Ezequiel Montes en doble sentido y generando nuevamente el procedimiento para la generación de la frecuencia de las aristas más usadas, observamos que la calle de Arteaga es liberada en cuestión a la utilización de aristas, Figura 5.4

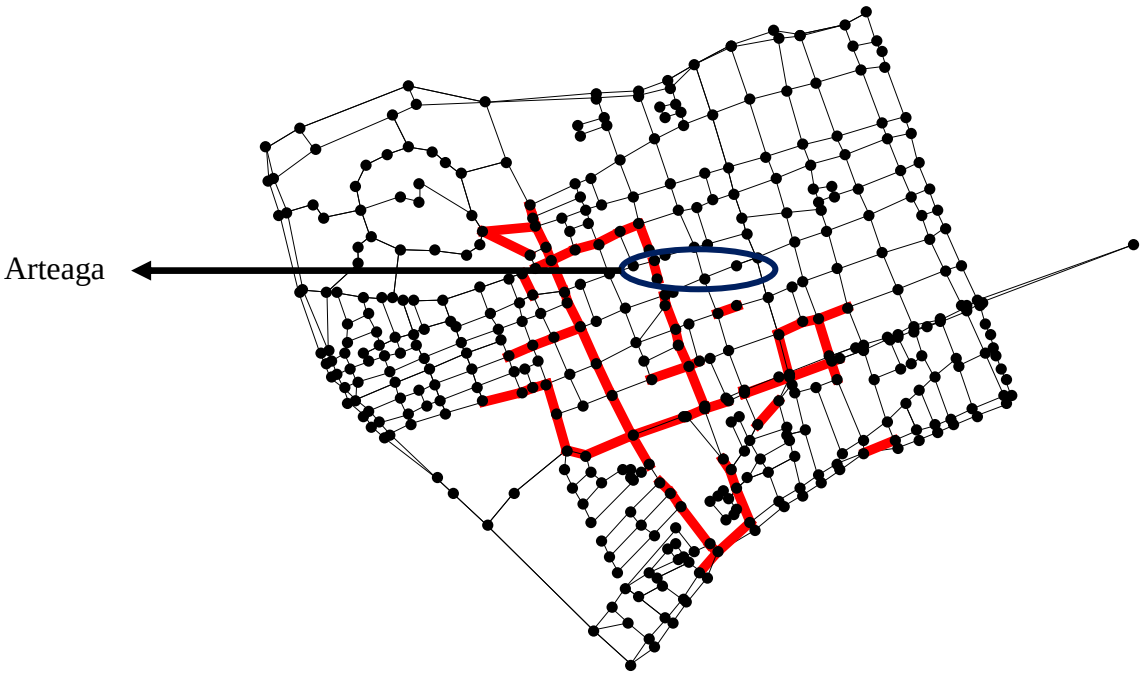


Figura 5.4 Aristas promedio más usadas considerando la Avenida Ezequiel Montes en doble sentido

V.2 Búsqueda Local: Cambiando los sentidos de avenidas menos usadas

La técnica de Búsqueda local utiliza estrategias generales para resolver problemas de optimización sobre espacios de búsqueda exponenciales. La idea básica de los algoritmos de Búsqueda local es la siguiente:

- Comenzar con una solución inicial generada aleatoriamente o hallada con algún otro algoritmo.
- Aplicar a la solución actual una transformación de algún conjunto dado de transformaciones para mejorar la solución. La solución mejorada se convierte en la solución actual.
- Repetir lo anterior hasta que ninguna transformación del conjunto mejore a la solución actual.

Basándose en el algoritmo de Búsqueda local, se realizaron cambios en los sentidos de las aristas menos usadas para tratar de mejorar el flujo de utilización de aristas dentro del cuadro principal del centro de la ciudad de Querétaro, el objetivo principal es mejorar el tamaño de la ruta máxima, así como el promedio total de rutas utilizadas.

Estructuras de Datos de la Red de Tráfico Vehicular

La matriz de adyacencias *matDis* modificada se construye a continuación a partir de la información de las aristas. La matriz *matDis* se define como una matriz porosa permitiendo solo almacenar la información de las aristas existentes.

Así la entrada i,j consiste en la longitud de la arista (i,j).

```
edgesWeighted1=Map[{{#[[1]],#[[2]]}, #[[3]]}&,edgesWeighted];
edgesWeighted2=Sort[edgesWeighted1,Last[#1]<Last[#2]&];
edgesWeighted2=Map[#[[1]] →#[[2]]&,edgesWeighted2];
edgesWeighted3=Map[{{#[[1]],#[[2]]},EdgeWeight→#[[3]]}&,edgesWeighted];
matDis=SparseArray[edgesWeighted2];
```

La dimension de la matriz es de 412x412

```
Dimensions[matDis]
{412,412}
```

La entrada {i,j} de la matriz producida por la función *GraphDistanceMatrix* corresponde a la longitud total de la ruta del vértice i al j.

Todas las Rutas óptimas uniendo dos puntos de la red

```
Timing[allDis=GraphDistanceMatrix[matDis]]
```

Asociada a la matriz anterior se tiene la secuencia de vértices que forman la ruta que va del vértice i al j:

```
Timing[p=Table[GraphPath[matDis,i,j],{i,n},{j,n}]]
```

Generación de la solución inicial

Se generan los parámetros iniciales que se van a manejar los cuales corresponden al tamaño de la ruta máxima, el promedio total de rutas y el porcentaje de rutas arriba del promedio para la solución inicial 0:

```
i=0;
Print["ITERATION #",i];
diferencias=Abs[Transpose[allDis]-allDis];
objectiveFunctionValue=Apply[Plus,Flatten[diferencias,1]];
averageLength=Plus@@(Plus@@#&)@allDis/(n^2-n);
aboveAverageLengthPath=(Select[Flatten[allDis],# ≥ averageLength&]//Length)/(n^2-n);
Print["PathMaximumTotalLength:",Max[allDis],
"ObjectiveFunctionValue:",objectiveFunctionValue//N];
Print["PathAverageTotalLength:",averageLength,
 "%ofPathsAbovetheAverage:",aboveAverageLengthPath//N];
```

Transformaciones para mejorar la solución

Las transformaciones que se realizan para mejorar la solución son el cambio de direcciones de las aristas menos usadas, invirtiendo su sentido. Una vez invertido el sentido de las aristas se vuelven a calcular todos los parámetros para ver si existe una mejora, si es así, esta solución se toma como la solución actual.

```
i++;
q=Flatten[Map[Partition[#,2,1]&,p,{2}],2];
freqSort=Sort[Tally[q],Last[#1]≤Last[#2]&];
freqSortMod=Map[Last[#]&,freqSort];
freqSortMod1=Tally[freqSortMod];
bajasFreq=First/@Take[freqSortMod1,0.03 Length[freqSortMod1]//Ceiling];
position=Map[Flatten[Position[freqSort,{{__},#}]]&,bajasFreq];
aristasMenosUsadas=First/@Map[freqSort[[First[#]]]&,position];
aristasUniDireccionales=Map[First,Select[Map[{#,¬(matDis[[First[#]]][[Last[#]]]→0\[And]
matDis[[Last[#]]][[First[#]]]→0)}&,aristasMenosUsadas],Last[#]→True&]];

changes=Map[
                edgesWeighted3[[#]]→
                Union[edgesWeighted3[[#]],{EdgeColor→Red,EdgeStyle→Thickness[0.01],
                VertexColor→Red}]&,Flatten[Map[Position[edgesWeighted3,#]&,
                Map[{{First[#],Last[#]},__}&,aristasUniDireccionales]]]
            ];

edgesWeighted3=edgesWeighted3/.changes;
vertices3=Map[{#}&,vertices];
g3=Graph[edgesWeighted3,vertices3];
ShowGraph[g3,VertexStyle□Disk[Small],EdgeStyle→Thickness[.001]]
matDisNormal=Normal[matDis];
Map[matDisNormal[[First[#],Last[#]]]&,aristasUniDireccionales];
Map[matDisNormal[[First[#],Last[#]]]&,Map[Reverse,aristasUniDireccionales]];
Map[(matDisNormal[[Last[#],First[#]]]=
matDisNormal[[First[#],Last[#]]])&,aristasUniDireccionales];
Map[(matDisNormal[[First[#],Last[#]]]=0)&,aristasUniDireccionales];
Map[matDisNormal[[First[#],Last[#]]]&,aristasUniDireccionales];
Map[matDisNormal[[First[#],Last[#]]]&,Map[Reverse,aristasUniDireccionales]];
matDis=SparseArray[matDisNormal];
allDis=GraphDistanceMatrix[matDis];diferencias=Abs[Transpose[allDis]-allDis];
objectiveFunctionValue=Apply[Plus,Flatten[diferencias,1]];
averageLength=Plus@@(Plus@@#&)@allDis/(n^2-n);
aboveAverageLengthPath=(Select[Flatten[allDis],# ≥ averageLength&]//Length)/(n^2-n);
```

Para la red de tráfico vehicular modelada anteriormente, se tiene una solución inicial (iteración 0) con un ruta máxima de 5464 metros, una ruta promedio total de 1645.79 metros y un porcentaje de rutas arriba del promedio de 0.479809.

ITERATION # 0

Path Maximum Total Length: 5464. Objective Function Value: 6.27844×10^7

Path Average Total Length: 1645.79 % of Paths Above the Average: 0.479809

Se genera la lista que contiene las aristas menos usadas, que se necesita invertir para tratar de mejorar la solución inicial, en la Figura 5.5 se puede observar en color rojo dichas aristas.

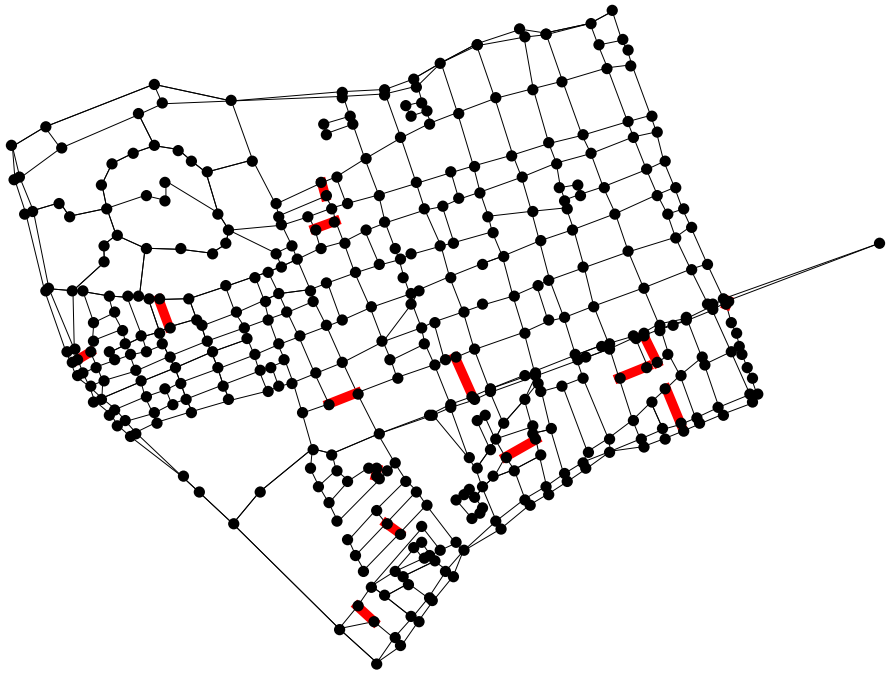


Figura 5.5 Aristas menos usadas Búsqueda Local

Se generan las siguientes 5 iteraciones donde se observa si los cambios en las aristas mejoran la solución inicial presentada.

En la iteración número 1 invertimos 14 aristas, las mostradas en la Figura anterior y se generan nuevamente los promedios, en este caso se obtuvo una ruta máxima de 5111 metros, una ruta promedio total de 1624.43 metros y un porcentaje de rutas arriba del promedio de 0.480795 con lo cual se puede concluir que con estos cambios existe una mejora en el promedio de uso de las aristas.

ITERATION # 1

Path Maximum Total Length: 5111. Objective Function Value: 5.80652×10^{7}

Path Average Total Length: 1624.43 % of Paths Above the Average: 0.480795

14 Aristas Invertidas

```
{{243,254},{170,163},{368,362},{184,193},{400,401},{382,375},{172,174},{21,1
7},{135,137},{63,71},{300,282},{142,162},{348,364},{131,145}}
```

Para la iteración 2 tomamos como solución actual la iteración 1, ya que mejoramos el promedio de uso para las aristas. Se generan nuevamente las aristas menos usadas para este caso 4 y se invierten nuevamente, los parámetros obtenidos después de la modificación son los siguientes:

ITERATION # 2

Path Maximum Total Length: 5111. Objective Function Value: 5.80374×10^{7}

Path Average Total Length: 1624.63 % of Paths Above the Average: 0.481138

4 Aristas Invertidas: `{{163,170},{137,116},{71,63},{364,348}}`

De las iteraciones número 3 a la 5 se invierten 4 aristas por cada una respectivamente teniendo como resultados los siguientes valores por iteración:

ITERATION # 3

Path Maximum Total Length: 5111. Objective Function Value: 5.80652×10^{7}

Path Average Total Length: 1624.43 % of Paths Above the Average: 0.480795

4 Aristas Invertidas: `{{170,163},{63,71},{116,137},{348,364}}`

ITERATION # 4

Path Maximum Total Length: 5111. Objective Function Value: 5.80374×10^{7}

Path Average Total Length: 1624.63 % of Paths Above the Average: 0.481138

4 Aristas Invertidas: `{{163,170},{137,116},{71,63},{364,348}}`

ITERATION # 5

Path Maximum Total Length: 5111. Objective Function Value: 5.80652×10^{7}

Path Average Total Length: 1624.43 % of Paths Above the Average: 0.480795

4 Aristas Invertidas: `{{170,163},{63,71},{116,137},{348,364}}`

Basándose en los resultado de las iteraciones anteriores se puede concluir, que no existe una mejora significativa con respecto a la iteración número 1, ya que los valores generados por las siguientes iteraciones no muestran una mejora en los parámetros con respecto a esta, por lo tanto ya no se sigue invirtiendo aristas y dejamos como solución los valores obtenidos en la iteración 1.

ITERATION # 1
Path Maximum Total Length: 5111. Objective Function Value: 5.80652×10^{7}
Path Average Total Length: 1624.43 % of Paths Above the Average: 0.480795
Con 14 Aristas Invertidas

V.3 Búsqueda Local: Cambiando sentido de la Avenida Ezequiel Montes

Usando la misma técnica de búsqueda local, se pueden cambiar los sentidos de Avenidas completas dentro de la red de tráfico vehicular verificando si se puede mejorar el flujo de aristas que comprenden dichas avenidas, esto es manejando un promedio más balanceado de utilización de las mismas.

En el siguiente ejemplo se cambia el sentido de la calle de Ezequiel montes y se analizan los resultados. Los siguientes puntos son los vértices pertenecientes a la Av. de Ezequiel Montes

EzequielMontesAvVertices={233,246,256,264,270,274,280,284,288,298,299,302,305,313,318};
Enseguida se generan las aristas que representan la Avenida

EzequielMontesAvEdges=Partition[EzequielMontesAvVertices,2,1]
{{233,246},{246,256},{256,264},{264,270},{270,274},{274,280},{280,284},{284,288},{288,298},{298,299},{299,302},{302,305},{305,313},{313,318}}

aristasUniDireccionales=Map[First,Select[Map[{#,¬(matDis[[First[#]]][[Last[#]]]→0 and matDis[[Last[#]]][[First[#]]]→0)}&,EzequielMontesAvEdges],Last[#]→True&]];

aristasUniDireccionales
{{233,246},{246,256},{256,264},{264,270},{270,274},{274,280},{280,284},{284,288},{288,298},{298,299},{299,302},{302,305},{305,313},{313,318}}

Se recuperan la aristas que son usadas en todas las rutas que van desde i a j, para todo i,j, se calcula el tamaño de la ruta máxima, el promedio total de rutas y el porcentaje de rutas arriba del promedio para la solución inicial 0. En las siguientes iteraciones consecuentes se invierte la dirección de la Avenida Ezequiel Montes, se vuelven a calcular los promedios para ver si mejoramos la solución inicial 0, como se muestra a continuación:

Para la solución inicial (iteración 0) se tiene un ruta máxima de 5464 metros, una ruta promedio total de 1645.79 metros y un porcentaje de rutas arriba del promedio de 0.479809.

ITERATION # 0
Path Maximum Total Length: 5464. Objective Function Value: 6.27844×10^{7}
Path Average Total Length: 1645.79 % of Paths Above the Average: 0.479809

Se genera el grafo que contiene las aristas de la calle Ezequiel Montes, Figura 5.6

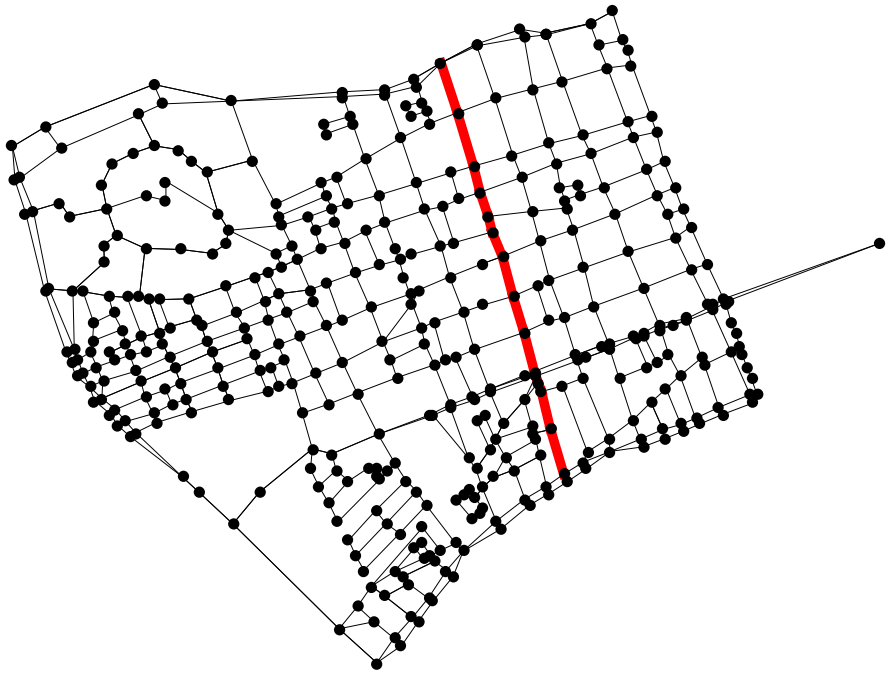


Figura 5.6 Aristas que abarcan la Avenida Ezequiel Montes

En la primera iteración se invierten las 14 aristas que abarcan la calle de Ezequiel montes y se vuelven a calcular los parámetros de la solución. Se puede observar que el máximo total de la ruta mejora al cambiar el sentido de la calle, de 5464 a 5459 metros, pero el promedio total de las rutas incrementa 20 metros aproximadamente de 1645.79 a 1665.14 metros así como también aumenta el número de rutas arriba del promedio con respecto a la solución inicial.

ITERATION # 1

Path Maximum Total Length: 5459. Objective Function Value: 6.71103×10^{7}

Path Average Total Length: 1665.14 % of Paths Above the Average: 0.480506

14 Aristas Invertidas:

{{233,246},{246,256},{256,264},{264,270},{270,274},{274,280},{280,284},{284, 288},{288,298},{298,299},{299,302},{302,305},{305,313},{313,318}}

Para las siguientes iteraciones se invierten las aristas menos usadas, hasta llegar a la iteración número 5, en la cual se invirtieron 5 aristas obteniendo una ruta máxima de 5106 y un promedio total de rutas de 1644.66 mejorando con esto la solución inicial (iteración 0).

ITERATION # 2

Path Maximum Total Length: 5106. Objective Function Value: 6.26666×10^{7}

Path Average Total Length: 1645.41 % of Paths Above the Average: 0.48184

9 Aristas Invertidas:

{{170,163},{368,362},{243,254},{184,193},{400,401},{382,375},{172,174}, {21,17},{135,137}}

ITERATION # 3

Path Maximum Total Length: 5106. Objective Function Value: 6.27164×10^{7}

Path Average Total Length: 1644.15 % of Paths Above the Average: 0.481226

5 Aristas Invertidas: **{{163,170},{137,116},{142,162},{348,364},{63,71}}**

ITERATION # 4

Path Maximum Total Length: 5106. Objective Function Value: 6.2601×10^{7}

Path Average Total Length: 1643.84 % of Paths Above the Average: 0.481557

6 Aristas Invertidas:

```
{{170,163},{71,63},{116,137},{276,261},{316,334},{300,282}}
```

ITERATION # 5

Path Maximum Total Length: 5106. Objective Function Value: 6.25757×10^{7}

Path Average Total Length: 1643.66 % of Paths Above the Average: 0.481315

5 Aristas Invertidas: `{{163,170},{137,116},{282,285},{282,275},{63,71}}`

Se pueden tomar los valores de la iteración número 2 como la solución final, ya que las siguientes iteraciones no mejoraron los promedios con respecto a esta solución.

V.4 Búsqueda Local: Cambiando el sentido de la Avenida Tecnológico

En el siguiente ejemplo cambiamos el sentido de la Avenida Tecnológico y analizamos los resultados.

Los siguientes puntos son los vértices pertenecientes a la avenida

TecnoligicoAvVertices=
```
{248,244,220,209,198,190,175,162,152,139,130,127,123,121,
118,116,111,101,96};
```

Enseguida se generan las aristas que representan la Avenida

TecnologicoAvEdges=Partition[TecnologicoAvVertices,2,1]
```
{{248,244},{244,220},{220,209},{209,198},{198,190},{190,175},{175,162},{162,
152},{152,139},{139,130},{130,127},{127,123},{123,121},{121,118},{118,116},{
116,111},{111,101},{101,96}}
```

aristasUniDireccionales=Map[First,Select[Map[{#,¬(matDis[[First[#]]][[Last[#]]]→0 \[And] matDis[[Last[#]]][[First[#]]]→0)}&,TecnologicoAvEdges],Last[#]→True&]];

aristasUniDireccionales

```
{{248,244},{244,220},{220,209},{209,198},{198,190},{190,175},{175,162},{162,
152},{152,139},{139,130},{130,127},{127,123},{123,121},{121,118},{118,116},{
116,111},{111,101},{101,96}}
```

Se calcula el tamaño de la ruta máxima, el promedio total de rutas y el porcentaje de rutas arriba del promedio para la solución inicial 0. En las siguientes iteraciones consecuentes se invierte la dirección de la Avenida Tecnológico, se vuelven a calcular los promedios para ver si mejoramos la solución inicial 0, como se muestra a continuación:

Para la solución inicial (iteración 0) tenemos un ruta máxima de 5464 metros, una ruta promedio total de 1645.79 metros y un porcentaje de rutas arriba del promedio de 0.479809.

ITERATION # 0

Path Maximum Total Length: 5464. Objective Function Value: 6.27844×10^{7}

Path Average Total Length: 1645.79 % of Paths Above the Average: 0.479809

Generamos el grafo que contiene las aristas de la Avenida Tecnológico, Figura 5.7

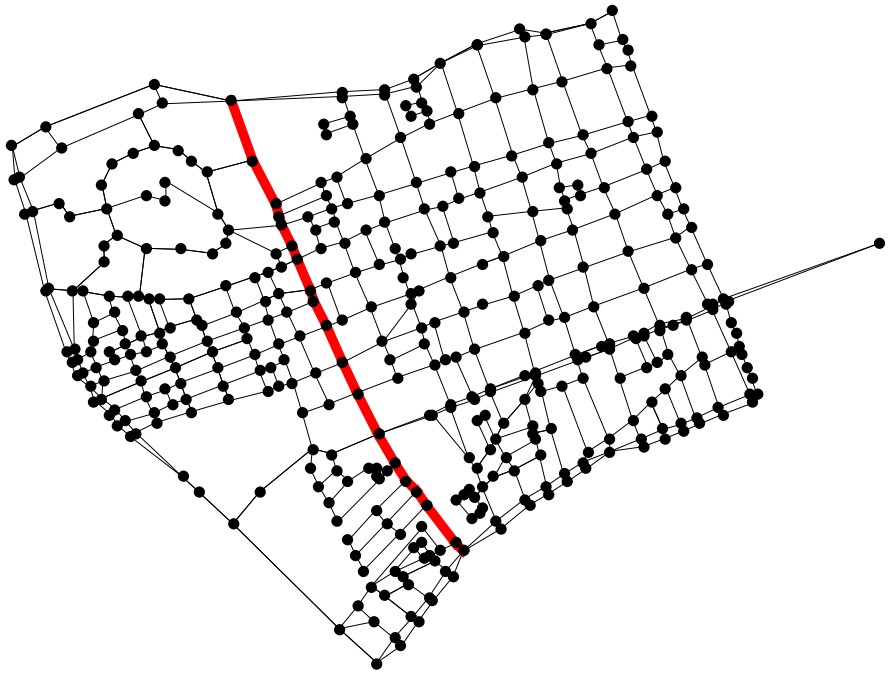


Figura 5.7 Aristas que abarcan la Avenida Tecnológico

En la primera iteración invertimos las 18 aristas que abarcan la calle de Tecnológico. Se puede observar que el máximo total de la ruta permanece igual en 5464 metros, pero el promedio total de las rutas incrementa 21 metros aproximadamente de 1645.79 a 1666.5 metros, así como también aumenta el número de rutas arriba del promedio con respecto a la solución inicial 0.

ITERATION # 1

Path Maximum Total Length: 5464. Objective Function Value: 7.39703×10^{7}

Path Average Total Length: 1666.5 % of Paths Above the Average: 0.481846

18 Aristas Invertidas

```
{{248,244},{244,220},{220,209},{209,198},{198,190},{190,175},{175,162},{162,
152},{152,139},{139,130},{130,127},{127,123},{123,121},{121,118},{118,116},{
116,111},{111,101},{101,96}}
```

Para la iteración número 2, en la cual invertimos 3 aristas se obtuvo una ruta máxima de 5111 y un promedio total de rutas de 1640.89 mejorando con esto la solución de la iteración 1, se siguen observando las siguientes iteraciones, pero ya no existe una cambio significante en la solución, por lo tanto el algoritmo termina en este punto (iteración 5), teniendo como resultados finales los valores de la iteración número 2.

ITERATION # 2

Path Maximum Total Length: 5111. Objective Function Value: 6.96423×10^{7}

Path Average Total Length: 1649.5 % of Paths Above the Average: 0.483299

11 Aristas Invertidas:

```
{{243,254},{170,163},{368,362},{184,193},{400,401},{21,17},{60,65},
{382,375},{63,71},{112,121},{300,282}}
```

ITERATION # 3

Path Maximum Total Length: 5111. Objective Function Value: 6.9377×10^{7}

Path Average Total Length: 1648.13 % of Paths Above the Average: 0.482862

5 Aristas Invertidas: `{{163,170},{348,364},{71,63},{282,275},{277,295}}`

ITERATION # 4

Path Maximum Total Length: 5111. Objective Function Value: 6.75801×10^7

Path Average Total Length: 1640.97 % of Paths Above the Average: 0.483045

5 Aristas Invertidas: `{{170,163},{63,71},{364,348},{135,137},{131,145}}`

ITERATION # 5

Path Maximum Total Length: 5111. Objective Function Value: 6.76108×10^7

Path Average Total Length: 1640.89 % of Paths Above the Average: 0.482838

3 Aristas Invertidas: `{{163,170},{348,364},{71,63}}`

VI. CONCLUSIONES

Una vez llevado a cabo el análisis y la evaluación experimental de los algoritmos de aproximación más relevantes para la planeación de rutas y utilizando grafos aleatorios como base de información, así como la modelación de una red de tráfico urbano con datos reales, se generó la información necesaria para el análisis de los algoritmos realizando una evaluación experimental de cada uno de ellos, llegando a las conclusiones siguientes:

- El problema TPQ se puede resolver con los algoritmos *greedy* del vecino más cercano y el algoritmo de la Mínima distancia, teniendo como rango de aproximación $\mathbf{2^{m+1} -1}$ **y m,** respectivamente, con respecto a la solución óptima. Se implementaron los algoritmos con Mathematica, utilizando grafos aleatorios con diferentes números de aristas, vértices y categorías.

- Se generaron 10 grafos aleatorios manejando 100, 150, 250, 350, 450, 550, 650, 750, 800, 900 y 1000 vértices con 25 categorías, se obtuvo el tiempo (segundos), observando que el algoritmo de la mínima distancia realiza mejores tiempos de ejecución en comparación con el algoritmo del vecino más cercano.

- Se generaron 10 grafos aleatorios con 400 vértices y 800 aristas cada uno, pero ahora utilizando diferentes números de categorías, se obtuvo el costo de ejecución del algoritmo midiendo el tiempo en segundos, observando que el tiempo de ejecución crece sustancialmente al aumentar el número de categorías, pero se obtiene un mejor rendimiento con el algoritmo de la mínima distancia.

- Para la Red de tráfico Vehicular del centro de la ciudad de Querétaro, se generaron todas las posibles rutas óptimas; obteniendo un total de 169,332 rutas óptimas. La ruta más larga óptima que se obtuvo tiene una distancia de 5464 metros, la cual empieza entre el cruce de la Avenida Zaragoza y la Avenida Corregidora, terminando en el cruce de la Avenida 5 de Febrero y Universidad.

- Las aristas más usadas abarcan la Avenida Tecnológico y parte de las Avenidas Zaragoza e Ignacio Pérez. Se realizaron cambios en los sentidos de algunas avenidas para ver si se podía balancear el uso de estas aristas. Por ejemplo, se cambió el sentido de la Avenida Tecnológico actual y se generó la frecuencia de aristas más usadas, observando que se libera la utilización de las aristas que comprenden la Avenida Ignacio Pérez, distribuyendo la frecuencia con la Avenida Régules.

- Se experimento utilizando la Avenida Tecnológico en doble sentido realizando nuevamente el procedimiento para la generación de la frecuencia de las aristas más usadas, con este cambio se observó que las avenidas de Ignacio Pérez y Régules dejaron de ser parte de las aristas más usadas, lo que significa que se mejoró el promedio de uso de dichas avenidas.

- Se cambió el sentido de la Avenida Ezequiel Montes actual y se generó la frecuencia de aristas más usadas, observandose que se libera la utilización de las aristas que comprenden las avenidas Arteaga y Régules.

- Se utilizó la Avenida Ezequiel Montes en doble sentido, observando que la Avenida Arteaga se liberó en cuestión a la utilización de aristas.

- Para la red de tráfico vehicular original se pudieron obtener las siguientes estadísticas: Se obtuvo una ruta óptima máxima de 5464 metros, una ruta promedio de 1645.79 metros y un porcentaje de rutas arriba del promedio de 0.479809. Con la utilización de técnicas de Búsqueda local, y realizando cambios en los sentidos de las aristas menos usadas, se pudo mejorar el promedio de utilización.

- Usando el algoritmo de Búsqueda local, se cambio la dirección de las 14 aristas menos usadas de la red de tráfico analizada, pudiéndose mejorar las estadísticas iníciales de la siguiente manera: La ruta optima máxima generada fue de 5111 metros, una ruta promedio total de 1624.43 metros y un porcentaje de rutas arriba del promedio de 0.480795, con lo cual podemos concluir que con dichos cambios podemos tener una mejora en la utilización de las calles y avenidas del centro de la ciudad de Querétaro.

- Utilizando la técnica de Búsqueda local y cambiando los sentidos de algunas avenidas completas dentro del cuadro principal, verificamos si podíamos mejorar la utilización de dichas avenidas manejando un promedio más balanceado de utilización de las mismas.

Por ejemplo se modificó el sentido de la Avenida Ezequiel montes obteniendo los resultados siguientes:

Una ruta óptima máxima de 5106 metros y un promedio total de rutas óptimas de 1644.66 metros, mejorando con esto los promedios iniciales. Finalmente se cambio el sentido de la Avenida Tecnológico donde se obtuvieron los resultados siguientes:

Una ruta óptima máxima de 5111 metros y un promedio total de rutas de 1640.89 metros, mejorando sustancialmente los promedios iniciales.

Se puede concluir en forma general que se pueden utilizar algoritmos de aproximación *greedy* para la optimización de rutas en redes en particular para el problema de TPQ. Basándose en la técnica de búsqueda local y cambios aleatorios de los sentidos de algunas de las principales avenidas del centro de la ciudad de Querétaro, se puede mejorar el promedio de uso de las avenidas, optimizando con esto las rutas óptimas en la red de tráfico vehicular.

VII. REFERENCIAS BIBLIOGRÁFICAS

Arora. S. (1998) Approximation schemes for NP-hard geometric optimization problems: A survey. Math Progr.

Arora. S. (1998) Polynomial-Time Approximation Schemes for Euclidean TSP andother Geometric Problems. Journal of the ACM (5).

Berchtold C. y Bohm 1997 A Cost Model for Nearest Neighbor Search in High-Dimensional Data Space.

Beckmann N. H. Kriegel, and R. Schneider. (1990) The R*-tree: An E_cient andRobust Access Method for Points and Rectangles.In Proc. of SIGMOD.

Cormen T, C. Leiserson, y R. Rivest Stein. (1997) Introduction to Algorithms. The MIT Press.

Deetz, Charles H. (1944). Elementos de proyección de mapas y su aplicación a la construcción de mapas y cartas.

Dumitrescu A. y J. S. B. Mitchell. (2001). Approximation algorithms for tsp with neighborhoods in the plane. In SODA '01: Proceedings of the twelfth annual ACM-SIAM symposium on Discrete algorithms, páginas 38-42.

Feifei , Dihan Cheng, Marios Hadjieleftheriou, George Kollios, and Shang-HuaTeng (2006). On Trip Planning Queries in Spatial Databases Computer Science dept.,Boston University, páginas 273-290.

Garey M. and Johnson D. (1979). Computers and Intractability: A Guide to NP-Completeness, ISBN-10: 0716710455.

Hartmut Guting R., M. H. Bohlen, M. Erwig, C. S. Jensen, N. A. Lorentzos, M. Schneider, y M. Vazirgiannis. (2002). A foundation for representing and querying moving objects. ACM Transactions on Database Systems.

Lichten S.M. (1991) “High Precision Applicationes of the Global PositioningSystem”. Jet Propulsion Laboratory. California Institute of Technology. Pasadena, California. National Aeronautics and Space Administration, Washington, Technology 2000, Volumen 2, páginas 276-284.

Mathematica 6.0 (2007). Wolfram Research, Inc.

Max J. Egenhofer. (1993) What's special about spatial?. database requirements for vehicle navigation in geographic space.

Papadias, Q. Shen, Y. Tao, and K. Mouratidis (2004). Group Nearest NeighborQueries. In Proc. of ICDE

Roussaopoulos N., S. Kelly, y F. Vincent. (2001).Nearest Neighbor Queries. In Proc. of SIGMOD.

Shahabi, M. R. Kolahdouzan, y M. Sharifzadeh. (2002). A road network embedding technique for k-nearest neighbor search in moving object databases. In GIS: Proc. of the ACM international symposium on Advances in geographic information systems, páginas 94-100.

Spielman D. and S.-H. Teng. (2001). Smoothed analysis of algorithms: Why the simplex algorithm usually takes polynomial time. In Proc. of STOC.

Skiena S. (2007). Computational Discrete Mathematics: Combinatorics and Graph Theory with Mathematica. Publisher: Cambridge University Press

Tao y D. Papadias. (2002). Time-Parameterized Queries in Spatio TemporalDatabases. In Proc. of SIGMOD.

Tao, D. Papadias, y Q. Shen. (2002) Continuous Nearest Neighbor Search. In Proc. of 28th VLDB.

Tao, J. Zhang, D. Papadias, and N. Mamoulis. (2004). An E_cient Cost Model for Optimization of Nearest NeighborSearch in Low and Medium Dimensional Spaces. IEEE Transactions on Knowledge and Data Engineering, páginas 1169-1184.

Theodoridis, E. Stefanakis, and T. Sellis. (2000). E_cient Cost Models for Spatial Queries Using R-trees. IEEE Transactions on Knowledge and Data Engieering, páginas 19-32.

Vazirgiannis and O. Wolfson. (2001). A spatiotemporal model and language for moving objects on road networks.In SSTD: Proc. of the International Symposium on Advances in Spatial and Temporal Databases, páginas 20-35.

Xiong, M. F. Mokbel, and W. G. Aref. (2005). Sea-cnn: Scalable processing of continuous k-nearest neighbor queriesin spatio-temporal databases. In ICDE.

Xiong, M. F. Mokbel, W. G. Aref, S. E. Hambrusch, and S. Prabhakar.(2004). Scalable spatio-temporal continuousquery processing for location-aware services.In SSDBM, pages 317-327.

Yiu and N. Mamoulis.(2004). Clustering Objects on a Spatial Network. In Proc. of SIGMOD.

VIII. GLOSARIOS DE TÉRMINOS

Algoritmos de aproximación: Son algoritmos usados para encontrar soluciones aproximadas a problemas de optimización.

Algoritmos *greedy*: Utilizados para resolver un determinado problema. Siguen una metaheurística, consistente en elegir la opción óptima en cada paso local con la esperanza de llegar a una solución general óptima.

Árboles-R: Son estructuras de datos de árbol que se encuentran comúnmente en las implementaciones de bases de datos y sistemas de archivos que utilizan para métodos de acceso espacial, es decir, para indexar información multidimensional.

Aristas: Uniones entre nodos o vértices.

Base de Datos Espacial: Sistema administrador de bases de datos que maneja datos existentes en un espacio o datos espaciales.

Coordenadas UTM: Sistema de coordenadas basado en la proyección geográfica transversa de Mercator, que se construye como la proyección de *Mercator* normal, pero en vez de hacerla tangente al Ecuador, la hace tangente a un meridiano.

Grafo completo: Grafo donde todas sus aristas conectan cada par de vértices.

Grafo dirigido: Grafo en el cual todas las aristas tienen un único sentido.

Indexación: Registrar ordenadamente información para elaborar su índice.

IMT: Instituto Mexicano del Transporte.

Normalización: Modificación de coordenadas con respecto a los valores mínimos y máximos.

Problemas NP completos: Conjunto de problemas de decisión de complejidad relacionada en los cuales no existe una solución en tiempo polinomial.

LTPQ: Consulta de planeación de rutas cíclica.

Programación lineal: Consiste en optimizar (minimizar o maximizar) una función lineal, que se denomina función objetivo, de tal forma que las variables de dicha función estén sujetas a una serie de restricciones que expresamos mediante un sistema de inecuaciones lineales.

Red Vehicular: Conjunto de calles o avenidas las cuales tiene un flujo dirigido.

Ruta óptima: El mejor conjunto de puntos que conectan dos puntos extremos.

Sistema GIS: Integración organizada de hardware, software y datos geográficos diseñado para capturar, almacenar, manipular, analizar y desplegar en todas sus formas la información geográficamente referenciada.

TPQ: Consulta de planeación de rutas

TSP: Problema del agente viajero.

Vértices: Punto donde concurren las dos semirrectas que conforman un ángulo.