UNIVERSIDAD AUSTRAL DE CHILE
FACULTAD DE CIENCIAS DE LA INGENIERIA
ESCUELA DE INGENIERIA CIVIL EN INFORMATICA

ANÁLISIS DE ALGORITMOS DE LOCALIZACIÓN ÓPTIMA Y SU IMPLEMENTACIÓN EN SISTEMAS DE INFORMACIÓN GEOGRÁFICA.

Tesis para optar
al título de Ingeniero Civil
en Informática

PROFESOR PATROCINANTE:
Mauricio Ruiz-Tagle M.
PROFESOR CO-PATROCINANTE:
Martín Solar M.

TANIA DENISSE LETELIER SANTIBAÑEZ

VALDIVIA - CHILE
2003

AGRADECIMIENTOS

Quiero agradecer a:

Mauricio Ruiz-Tagle, profesor patrocinante, por su gran apoyo y orientación durante el desarrollo de mi tesis.

Martin Solar, profesor co-patrocinante, por su apoyo desde mis primeras busquedas de temas para la tesis.

Marcelo Neculman y Jimena Muñoz,de Forestal Valdivia S.A., por permitir que mi trabajo de tesis sea una solución real a un problema real.

Gonzalo Paredes y Mauricio Acuña, por la orientación y el gran apoyo en material bibliográfico.

Timothy Gibson, por su ayuda en donde mis conocimientos en inglés y economía no fueron suficientes.

María Eliana de la Maza, por su disponibilidad a ayudar.

Quiero agradecer en forma especial a mi familia, mis padres y amigos, por la preocupación en que éste proyecto concluyera con éxito.

Dedico esta tesis a mi hijo Andrés.

INDICE

INDICE DE ANEXOS

RESUMEN

Para minimizar sus costos, una empresa que cuenta con una red de distribución debe asegurar que sus centros de distribución se ubiquen eficientemente. Un programa computacional bien diseñado, junto con la información geográfica correcta, puede lograr este objetivo de una manera que requiera poco esfuerzo humano.

El problema planteado consiste en tener un número determinado de centros o puntos que demandan la entrega de un bien o su recolección. En base a la ubicación de todos estos centros se requiere encontrar la mejor localización de una planta industrial que sea capaz de satisfacer los requisitos de todos los otros centros. Este tipo de problema se denomina problema de localización óptima. Para resolver el problema planteado, en primer lugar, se presenta un estudio sobre algunos tipos de problemas de localización óptima y sus correspondientes algoritmos de solución. Posteriormente, se realiza el diseño e implementación del prototipo en la herramienta para Sistemas de Información Geográfica (SIG) y en el lenguaje de programación elegidos, aplicando los algoritmos de solución a problemas de localización óptima estudiados, para finalmente realizar pruebas al prototipo evaluando su comportamiento en la ejecución. El análisis, diseño e implementación, fue aplicado a un problema de localización de canchas de acopio de madera, en parte de la superficie de plantaciones de una empresa forestal.

El estudio de esta tesis se centra en la implementación, usando SIG, de algoritmos que resuelvan problemas de localización óptima.

SUMMARY

In order to minimize its costs, a company that uses a distribution network must locate its distribution centers efficiently. A well-designed computer program, coupled with accurate geographical information, can accomplish this objective in a way that avoids labor-intensive practices.

The problem at hand involves the placement of an industrial facility within a network of centers or points involved in the delivery or collection of a product. The basis for decisions regarding the location of a new facility is the cost of transportation between it and the various points comprising the existing infrastructure. This is called an optimal location problem. In examining optimal location, I begin by presenting a study of the different types of optimal location problems and the algorithms that correspond to their solutions. Next, I describe the design and implementation of a prototype program using the appropriate Geographic Information Systems (GIS) tool and programming language, applying the algorithms for solution to the optimal location problems examined. Finally, I perform tests on the prototype, examining its execution capabilities. The analysis, design and implementation were applied to the placement of wood storage facilities within a forestry company's plantation network.

The focus of this thesis is the implementation, using GIS, of algorithms used to resolve optimal location problems.

1. Introducción

1.1. Preámbulo

Por muchos años el diseño de redes camineras y el transporte de productos han sido de gran importancia para empresas de todos los rubros, públicas y privadas.

Hoy en día el desafío es encontrar las mejores ubicaciones o localizaciones óptimas de centros de actividades en zonas donde ya se cuenta con toda la infraestructura caminera, por lo que estudios sistemáticos para la elaboración de modelos de decisión óptimos son una necesidad ante la magnitud de las inversiones en transporte.

Desde hace varias décadas se han estudiado e investigado métodos que permitan facilitar la tarea de decisión de lugares óptimos. Producto de esos estudios se crearon muchos modelos matemáticos que pueden ayudar , en mayor o menor grado, a solucionar problemas de este tipo.

El presente proyecto abarca la investigación y estudio de algunos de estos modelos matemáticos que permitan encontrar alguna solución al problema de localización óptima para centros de actividades forestales. Teniendo como objetivo dar solución a un problema frecuente en ciertas empresas forestales, se planteará un sistema que implemente algunos algoritmos que apoye la elección de los lugares geográficos más convenientes para la instalación de sus centros de actividades forestales. Pudiendo ser, por ejemplo, instalaciones industriales, canchas de acopio, canchas de trozado, recursos de control de incendios, por mencionar algunos.

Debido a los altos costos de transporte se hace cada vez más necesario incorporar técnicas modernas de modelación y análisis, auxiliados hoy, por la cada vez mayor y más eficaz innovación tecnológica, cuyo soporte fundamental cuando se trata de análisis espacial o territorial son los Sistemas de Información Geográfica (SIG), capaces de trabajar con una base de datos espacial y una base de datos temática, permitiendo determinar, las localizaciones óptimas de los servicios, de manera de alcanzar conjuntamente la eficacia y la equidad espacial. Por esta razón se estudió una herramienta que contenga los instrumentos necesarios para implementar estos modelos en un entorno gráfico que facilite la utilización de la aplicación, considerando las herramientas SIG disponibles.

Después de decidir los modelos matemáticos a utilizar y la herramienta en la cual se implementó, se comenzará el proceso de definición y desarrollo del software. La construcción del software se guió por la metodología de ciclo de vida correspondiente al modelo de cascada [Pre96].

El presente trabajo intenta entregar un prototipo en donde se puedan evaluar diferentes algoritmos de localización y, apoyándose en las ventajas que ofrecen los SIG, determinar los sitios más óptimos para la instalación de posibles nuevos centros industriales forestales dado un conjunto de puntos de oferta de un insumo.

Si bien este proyecto se centra en dar solución al problema de localización de centros de acopio del área forestal esta solución es aplicable a problemas en diferentes áreas, como por ejemplo áreas de servicio público, en actividades agropecuarias, industriales, comerciales, entre otras.

Para este problema de decisión se debe tener presente factores tales como:

- El área en que se encontrará la localización óptima debe ser limitada.
- Se debe contar con puntos de localización candidatos de los cuales se elegirá el óptimo.
- Tener preestablecidas las variables que se considerarán para tomar la decisión, por lo general se consideran variables de costo, específicamente costo de transporte, pero podría haber otras variable s involucradas.

1.2. Definición del problema.

Las empresas forestales poseen planificaciones, o plan de manejo forestal, que se refiere a actividades que durante un periodo de tiempo realizan en sus bosques o terrenos que disponen para esos fines. Algunas de las actividades que se desarrollan son actividades de caracterización, evaluación, planificación, aprovechamiento, regeneración, reposición, protección y control del bosque conducentes a asegurar la producción sostenible y la conservación de la diversidad biológica y el ambiente [URL1]. Dentro de las actividades de planificación y aprovechamiento se pueden destacar la siembra, poda, raleo y cosecha. Se pondrá especial énfasis en la actividad de raleo y cosecha, por ser las actividades que se consideran en el desarrollo de esta tesis.

El raleo es una corta de algunos árboles de un bosque en base a un criterio de selección. Esta actividad se realiza con el objetivo de obtener la mayor rentabilidad de la plantación (árboles de mayor diámetro), manejando en forma adecuada la densidad en cada etapa del desarrollo del rodal. La intervención de raleo permite también tener una plantación sana y con una mayor resistencia a la amenaza de

distintas plagas forestales. Dentro de esta actividad se realizan otras como determinación de la densidad residual, marcación de raleos, construcción de caminos (si se estima conveniente), clasificación de productos.

La cosecha es la actividad de corta del bosque sembrado, dentro de esta actividad también se observan otras como, determinación de densidad, construcción de caminos, clasificación de productos con respectivos volúmenes, entre otras.

De estas actividades se obtienen productos que son comercializables, para lo cual deben ser trasladados a un lugar para su almacenamiento y posterior envío a su lugar de destino. Este lugar intermedio entre el bosque y su lugar de comercialización se llama cancha de acopio, la ubicación de la cancha de acopio es fundamental si se quiere disminuir al máximo los costos de esta actividad.

La ubicación de la cancha de acopio se convierte en una tarea difícil cuando los predios a ralear están esparcidos en una superficie muy amplia y condicionada además a rutas camineras establecidas y en diferentes condiciones de mantención. En la actualidad la elección del lugar de ubicación de las canchas de acopio se realiza basandose en la experiencia.

Debido a la mayor competitividad de las empresas forestales, la tendencia es realizar cada tarea de la forma más eficiente y auxiliados por la tecnología. Esto es una realidad, como lo demuestran la existencia de softwares que apoyan los planes de manejo. Así, se decidió implementar una aplicación que ayude en la decisión de ubicar una cancha de acopio para la actividad de raleo y cosecha.

Específicamente, esta aplicación busca ubicar una cancha de acopio para almacenar los productos del raleo o cosecha a efectuarse en un periodo determinado

de tiempo, la capacidad de esta cancha será lo suficientemente grande como para almacenar todo el volumen que se obtenga de esta actividad. Cada predio entrega una cantidad de producto de raleo o cosecha determinada, en forma aproximada, mediante:

Producto(m3) = Superficie(m2) * 50 , para el caso del raleo,

En el caso de la cosecha el producto a cosechar es calculado previamente mediante programas simuladores de propiedad de la empresa.

Se debe tener en cuenta que se utilizarán caminos públicos para llegar hasta los predios, con excepción de predios a los que se accede por medio de caminos privados de la forestal; no se prevé la construcción de nuevos caminos. Existen distintos tipos de caminos, se diferencian tres aunque en la realidad se pueden discriminar más. Los tipos de caminos son: Ruta 5, camino de pavimento y camino de ripio-tierra, cada uno de ellos tiene distinto costo de transitar determinado por la empresa.

De acuerdo a todas las características del ámbito donde surge el problema se busca la localización óptima para una cancha de acopio, utilizando conocimiento surgido de años de estudio en materia de localización óptima. Se ha determinado resolver este problema mediante la aplicación de algoritmos de localización óptima y no directamente con programación lineal ya que los algoritmos existentes resuelven el problema planteado y como se ha podido comprobar con la investigación realizada para esta tesis, las actuales y futuras investigaciones sobre esta materia apuntan a la creación, desarrollo e implementación de nuevos algoritmos de localización óptima [URL7].

1.3. Antecedentes

Trabajos relacionados.

En nuestro país y en general en Latinoamérica no se conocen referencias de estos trabajos, si bien pueden existir ya que la tecnología y el conocimiento existen, no se han publicado trabajos en este tema, sólo en contadas excepciones como el proyecto desarrollado en Argentina en el año 2000 "Evaluación y diagnóstico de la situación hospitalaria en la Provincia del Chaco. Aplicación de Modelos de Localización-Asignación óptima mediante S.I.G., para posibles nuevos hospitales", en donde se trabajó con dos tipos de datos, raster y vectorial, y se compararon para cada uno de ellos dos métodos de localización [URL5].

Es posible encontrar trabajos relacionados con localizaciones óptimas y sistemas de información geográfica desarrollados en países de América del Norte y Europa, proyectos de diversas áreas, sectores privados y públicos. Proyectos como localizaciones de establecimientos educacionales, de salud, comerciales, entre otros.

Las investigaciones en este tema, están bastante avanzadas. En los países desarrollados ya se consideran los problemas de localización óptima resueltos mediante SIG como un tema de mucho potencial y por ello existen investigadores de importantes universidades abocados a su desarrollo. Los últimos estudios indican que la orientación de este tema es hacia la aplicación de algoritmos heurísticos, algoritmos genéticos y otros métodos no tradicionales

Realidad de la empresa.

Actualmente la empresa Forestal Valdivia S.A. cuenta con un gran patrimonio territorial que se extiende entre la IX y X regiones, por lo que la administración, control y manejo de los predios es una tarea que demanda muchos recursos.

Se cuenta con un sistema de información geográfica para facilitar la administración del patrimonio, las herramientas utilizadas son Arcinfo y Arcview con la extensión Image analysis, que permite la georreferenciación y rectificación de imágenes. Este sistema es utilizado en las diferentes áreas de la empresa, en actividades que van desde la evaluación de un terreno para su compra hasta todas las involucradas en su explotación forestal.

Los datos de los predios son extraídos mediante fotos aéreas y ortofotos, apoyados por mapas territoriales. Estos datos son utilizados en actividades de evaluación de suelos, determinación de pendientes, determinación de porcentaje utilizable de suelo, que son tareas imprescindibles de realizar previo a la compra. Otras actividades en que la empresa utiliza su sistema es para hacer catastros, planificaciones de faenas, de utilización de maquinaria de construcción o mejoramiento de caminos, actividades de control de incendio.

1.4. Objetivos

Objetivos Generales

Implementar a nivel de prototipo una aplicación sobre un SIG, que resuelva el problema de localización óptima de un centro industrial forestal.

Objetivos especificos

1. Investigar y determinar los modelos matemáticos más apropiados para encontrar una solución al problema de localización óptima.
2. Evaluar y determinar la herramienta SIG a utilizar.
3. Aplicar una metodología formal de desarrollo de software para la implementación de los algoritmos en la herramienta SIG elegida.

2. Localización óptima.

2.1. Problema de localización óptima.

Problema de localización óptima en redes

La palabra "óptima" está utilizada en un sentido matemático. Es decir, se definen objetivos cuantificables que dependen de las localizaciones de los recursos. Entonces se identificarán los algoritmos para encontrar un óptimo o por lo menos las mejores localizaciones del recurso.

Dos factores limitan la elección del óptimo de los sitios sugeridos por los modelos de optimización. Primero, en muchos casos, los objetivos no cuantificables influenciarán, en gran parte, en la decisión de la localización. A menudo, los factores cualitativos que influencian la localización de decisiones son críticamente importantes. Así, hasta el punto que los procedimientos aplicados no hagan caso de preocupaciones y de factores cualitativos. Por lo tanto, los sitios identificados por los algoritmos matemáticos son óptimos solamente en un sentido estrecho de la palabra. En segundo lugar, el funcionamiento de un sistema es afectado por muchos factores de los cuales las localizaciones son solamente una, como por ejemplo, la habilidad de un servicio de ambulancias no depende solo de la cercanía de la ambulancia al lugar del suceso sino también de la pericia del chofer de la ambulancia y de todo el equipo médico que acude en auxilio.

Los problemas de localización en redes ocurren cuando nuevos recursos son localizados en una red. Las redes de interés pueden ser redes camineras, redes de transporte aéreo, redes fluviales, etcétera.

Para un determinado problema de localización, los nuevos recursos son idealizados como puntos, y pueden ser localizados en cualquier parte en la red; pero puede ser impuesto al problema que el nuevo recurso se ubique cercano a un recurso existente. Usualmente las funciones objetivo son minimizadas, por lo general el objetivo es minimizar una suma de los costos de transporte de las distancias de viaje entre recursos existentes y los nuevos recursos más cercanos, o un máximo de las pérdidas proporcionales a tales distancias, o el número total de los nuevos recursos que se localizarán[Han79].

Clasificación de problemas de localización en redes.

Los problemas de localización pueden ser clasificados según diferentes criterios: de acuerdo a criterios de optimización o función objetivo, tipos de redes, localización de uno o múltiples recursos y al grado de generalidad en los modelos.

Los criterios de optimización están determinados por el tipo de problema que se presenta en la red. En ocasiones puede ser apropiado un criterio que minimice la función de costos relacionada tanto con tiempos de viaje como con distancias de viaje y posiblemente con otros atributos de viaje. Este tipo de criterio es denominado *minisum*, puesto que un conjunto dado de localizaciones afecta a cada nodo de la red y un conjunto óptimo reduce al mínimo el efecto negativo total del recorrido. Las localizaciones que optimizan el criterio minisum son llamadas a menudo *medianas*.

Otro tipo de criterio utilizado es el llamado *minimax* , este criterios en más apropiado en aquellos casos donde es más adecuado minimizar la máxima distancia desde un nodo en la red a la posible localización óptima. Las localizaciones que optimizan este criterio son llamadas *centros* en una red. En ocasiones puede ser necesario aplicar múltiples criterios a un problema tanto minisum como minimax.

Al clasificar los problemas de localización de acuerdo al número de recursos lo más utilizado es dividirlos en problemas de localización de *un recurso* o de *múltiples recursos*.

Otro criterio de clasificación de los problemas de localización depende de la naturaleza de la red, si es *estática* o *dinámica*. Si la red es estática, los nodos y enlaces no cambian en el tiempo, sus valores pueden ser medidos y por lo tanto son determinísticos. Sin embargo, una red es probabilística cuando los valores de los enlaces o nodos cambian en el tiempo, el estado de la red en el futuro es incierto; y se asumen limites o rangos de comportamiento.

La estructura de la red puede clasificarse en *no orientada* y *orientada*. Una red es no orientada cuando los valores de los atributos de viaje, tales como tiempo de viaje, desde un punto *x* a cualquier punto *y* es igual para el valor del atributo desde *y* a *x*. En redes orientadas estos valores no son necesariamente iguales. Así la localización óptima de un recurso depende en gran medida si los viajes se inician o terminan en tal recurso.

Las redes pueden ser *cíclicas* o *acíclicas*. Cuando una sola ruta existe entre cualquier par de nodos en una red no orientada, la red es llamada como *acíclica* o *árbol*.

Otro criterio de clasificación de los problemas de localización es de acuerdo a los puntos de demanda y puntos de localización de recursos. En cualquier problema de localización los puntos de demanda deben ser identificados. Así, dos escenarios son posibles: (1) las demandas pueden ocurrir en cualquier parte de la red o (2) las demandas pueden ocurrir en los nodos. Los posibles puntos de localización de recursos también deben ser localizados, aquí también se consideran dos escenarios: (1) los recursos pueden ser localizados en cualquier parte de la red o (2) los recursos pueden ser restringidos a algún nodo en la red.

La figura 1, indica los criterios para clasificar los problemas de localización óptima en redes.

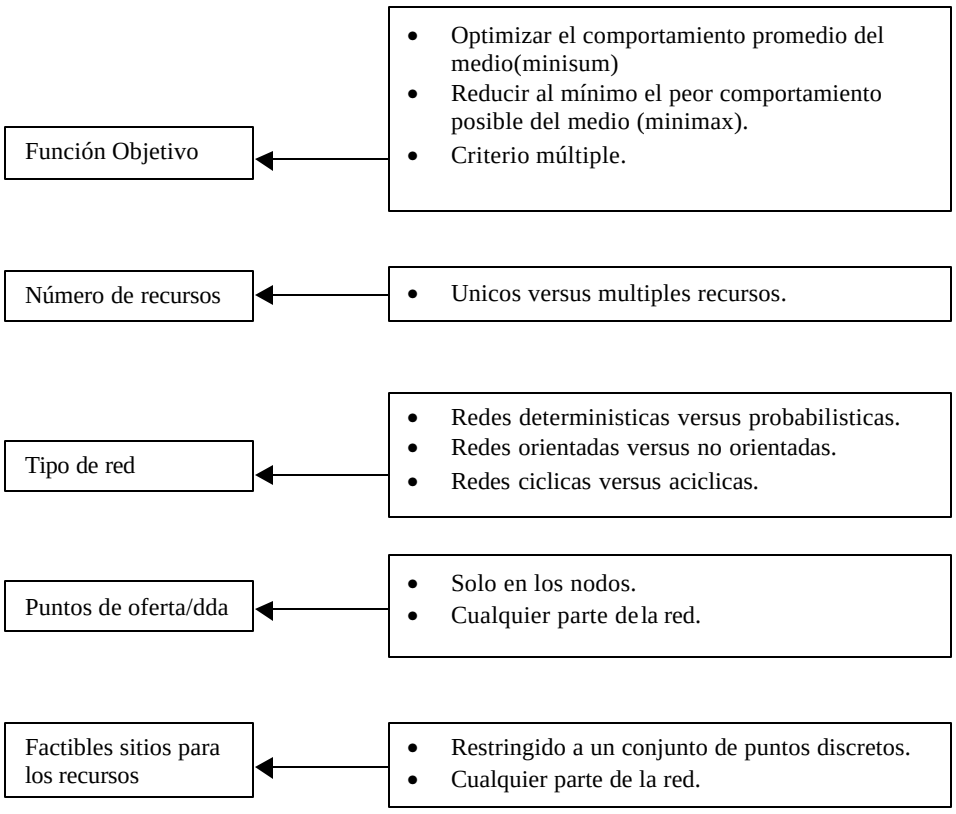

Figura 1. Criterios de clasificación de problemas de localización en redes.

Según los criterios de clasificación los problemas de localización se agrupan o clasifican como se muestra en la figura 2, donde se observa el árbol familiar para problemas de localización en redes [Tan83].

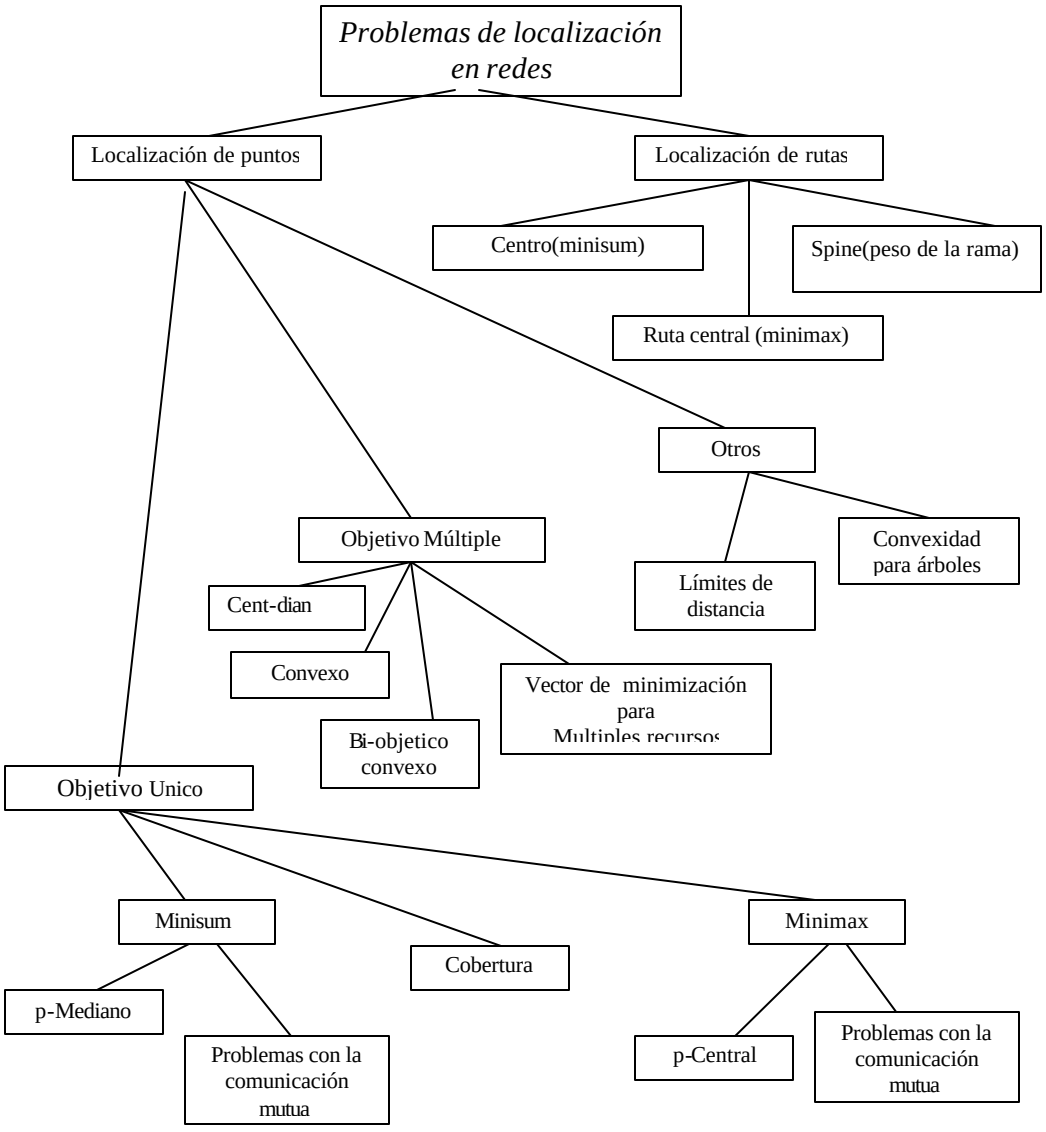

Figura 2. Arbol familiar para problemas de localización en redes.

En los apartados siguientes se explicarán aquellos tipos de problemas que representan el problema que se quiere resolver en esta tesis, descritos en el capítulo 1, estos problemas son los derivados de la rama de localización de puntos con objetivo único, como se indica en la figura 2.

Problemas P-Central

Este modelo minimiza la distancia de cobertura para que cada nodo de demanda sea cubierto por uno de los recursos dentro de la distancia determinada endógenamente. Este modelo se conoce también como Minimax ya que se minimiza la distancia máxima entre una demanda y el recurso más cercano a la demanda. Se distinguen dos tipos de problemas p-central, uno donde los recursos pueden ser localizados en cualquier parte de la red (o sea, en los nodos o en las aristas de la red) llamados *problema central absoluto* y problemas en donde los recursos pueden ser localizados solo en los nodos de la red, llamados *problema central vértice* [Das95].

Su formulación es la siguiente:

Entradas:

d_{ij} = distancia desde el nodo i al sitio candidato a recurso j.

h_i = demanda del nodo i.

P = número de recursos a localizar.

Variables de decisión:

X_j = 1 si el recurso se localiza en el sitio j

0 si no se localiza

Y_{ij} = fracción de demanda del nodo i que es abastecida por el recurso j.

W = máxima distancia entre un nodo y el recurso más cercano.

Minimizar	W		(1 a)
Sujeto a :	$\sum_j Y_{ij} = 1$	$\forall$ i	(1 b)
	$\sum_j X_j = P$		(1 c)
	$Y_{ij} \leq X_j$	$\forall$ i, j	(1 d)
	$W \geq \sum_j d_{ij} Y_{ij}$	$\forall$ i	(1 e)
	$X_j = 0,1$	$\forall$ i	(1 f)
	$Y_{ij} \geq 0$	$\forall$ i, j	(1 g)

La función objetivo (1 a) minimiza la distancia máxima entre un nodo y el recurso más cercano al nodo. La ecuación (1 b) indica que todos los nodos de demanda deben ser asignados a un recurso *j*. La ecuación (1 c) estipula que *P* recursos deben ser localizados. La ecuación (1 d) establece que los nodos de demanda *i* no pueden ser asignados a un recurso en el nodo *j* a menos que ese recurso esté localizado en el nodo *j*. La ecuación (1 e) establece que la distancia máxima entre un nodo de demanda y el recurso más cercano al nodo (*W*) debe ser más grande que la distancia entre cualquier nodo de demanda *i* y un recurso *j* al que es asignado. Las ecuaciones (1 f) y (1 g) establecen la binariedad y no negatividad, respectivamente.

Problemas P-Mediano

El problema p-mediano consiste en encontrar la localización de P recursos en una red de modo que el costo total esté minimizado. El costo de suplir las demandas del nodo *i* es determinado por el producto de la demanda del nodo *i* y la distancia entre el nodo *i* y el recurso más cercano a ese nodo [Das95]. Su formulación es la siguiente:

Entradas:

d_{ij} = distancia desde el nodo i al sitio candidato j.

h_i = demanda del nodo i.

P = número de recursos a localizar.

Variables de decisión:

X_j = 1 si el recurso se localiza en el sitio j

0 si no se localiza

Y_{ij} = 1 si los nodos de demanda i son satisfechos por recursos ubicados en nodo j.

0 si no.

Minimizar $\sum_i \sum_j h_i d_{ij} Y_{ij}$ (2 a)

Sujeto a:

$\sum_j Yij = 1 \quad \forall i$ (2 b)

$$\sum_j X_j = P \qquad (2\ c)$$

$$Y_{ij} - X_j \leq 0 \quad \forall\ i, j \qquad (2\ d)$$

$$X_j = 0, 1 \quad \forall\ j \qquad (2\ e)$$

$$Y_{ij} = 0, 1 \quad \forall\ i, j \qquad (2\ f)$$

La función objetivo (2 a) minimiza la demanda-peso-distancia total entre cada nodo de demanda y los recursos más cercanos. La ecuación (2 b) requiere a cada nodo de demanda *i* ser asignado a exactamente un recurso *j*. La ecuación (2 c) establece que exactamente *P* recursos sean localizados. La ecuación (2 d) indica que los nodos de demanda *i* pueden ser solo asignados a un recurso de localización *j* si un recurso es localizado en el nodo *j*. La ecuación (2 e) y (2 f) condicionan a las variables a una binariedad.

2.2. Algoritmos de localización óptima.

En el problema de localización planteado, se tienen los siguientes elementos definidos: se busca sólo una localización óptima; teniendo en cuenta que el número de nodos candidatos a ser mejor localización es igual al número de nodos de toda la red. Para ello se estudiaron los algoritmos comentados en los siguientes apartados.

Algoritmo para Problemas P-Medianos.

Mientras los problemas p-medianos pueden ser resueltos en tiempo polinomial cuando la red es un árbol, el problema es un NP-completo en un grafo general. Así, un número significativo de algoritmos heurísticos se están proponiendo para la solución de los problemas p-medianos.

Uno de esos algoritmos es el algoritmo Myopic. Kuehn y Hamburger (1963) fueron los primeros en sugerir este algoritmo, como forma de resolver problemas generales de localización de recursos, posteriormente se comenzaron a estudiar resultados computacionales con este mismo algoritmo, los primeros fueron Cornuejols, Fisher, y Nemhauser (1977).

Este algoritmo agrega recursivamente medianas a la solución, inicia con una mediana y secuencialmente va agregando medianas hasta que *p* se han seleccionado. El primer punto medio a ser seleccionado (mediana) es la mediana del grafo. Después, la siguiente mediana a ser ingresado a la solución es cualquier nodo no seleccionado que mejore el valor de la función objetivo.

Si se localiza solamente un recurso en la red, se podría encontrar fácilmente la localización óptima enumerando todas las localizaciones posibles y eligiendo la mejor. Específicamente, puesto que se sabe que por lo menos una solución óptima a cualquier problema p-mediano consiste en localizar el recurso únicamente en los nodos, se podría evaluar la función objetivo 1-mediano, que consiste en simular la localización en cada nodo. Entonces se elige la localización con el valor más pequeño de $Z_j = \sum_i (h_i * d_{ij})$, solo si se quiere localizar un recurso, está claro que este

método daría una solución óptima, puesto que se prueban todas las localizaciones posibles.

A continuación se describe el algoritmo Myopic para problemas p-medianos:

Paso 1:

Inicializar k=0 (k cuenta el número de recursos que se localizarán), y $X_k=\phi$ (X_k da la localización de la planta *k* que se localiza en cada etapa del algoritmo).

Paso 2:

Incrementar *k*, el contador en el número de recursos localizados.

Paso 3:

Calcular $Z_j^k=\Sigma_i h_i\, d(i,j \cup X_{k-1})$ para cada nodo j que no está en X_{k-1}. Notar que Z_j^k determina el valor de la función objetivo si se localiza el k-ésimo recurso en j. Notar que los primeros *k-1* recursos localizados $\in$ a X_{k-1}(y j no pertenece a este).

Paso 4:

Buscar el nodo j*(k) que minimice Z_j^k, $j^*(k)= \text{argmin}_j \{ Z_j^k \}$. Notar que j*(k) determina la mejor localización para el k-ésimo recurso. Determinar la localización de los primeros *k-1* recursos. Agregar el nodo j*(k) al conjunto X_{k-1} para obtener el conjunto X_k ; que es $X_k = X_{k-1} \cup j^*(k)$.

Paso 5:

Si $k=P$ (*P* es el número de recursos que se quiere localizar), parar; el conjunto X_p es la solución para el algoritmo. Si $k<P$, ir a paso 2.

Algoritmo para Problema P-Central Vértice

Este algoritmo resuelve el problema p-central vértice. Utiliza un método basado en coberturas, fue desarrollado en 1987 por Soon-dal Park profesor, en ese entonces de la Universidad Nacional de Seul. Este algoritmo comienza con un número de coberturas igual o inferior al número de nodos de la red, y en cada iteración va disminuyendo en uno el número de coberturas, hasta igualar al número de recursos a localizar. Antes de describir el algoritmo es necesario conocer la notación:

Nalc, que corresponde al número de recursos a localizar; *Nnodes*, número de nodos de la red; *Ncan*, número de nodos candidatos a ser recurso.

El algoritmo es como sigue:

Paso 1:

Crear arreglo de coberturas *clust(i)=j* para cada nodo *i* , donde *j* es el nodo más cercano.

Paso2:

Si *Nalc= Ncan* ir a paso 7.

Paso 3:

Buscar la menor distancia entre nodos que dan cobertura, almacenar este par de nodos.

Paso 4:

Seleccionar las dos coberturas del par de nodos seleccionados en el paso 3.

Para cada nodo de esas coberturas buscar los nodos más lejanos

(perteneciente a esas coberturas), y entre esos pares buscar el que esté más cercano. Almacenar la distancia entre el par *(i,j)* y el nodo *j*.

Paso 5:

Crear una nueva cobertura con todos los nodos de las coberturas seleccionadas en el paso 4 , el nodo *j* almacenado en el paso anterior les da cobertura. Los nodos seleccionados en el paso 3 dejan de dar cobertura y pasan a formar parte de la nueva cobertura.

Paso 6:

Realizar una reasignación de nodos a las coberturas, en toda la red.

Paso 7:

Si el número de coberturas o el número de iteraciones es igual a *Nalc*, parar, los nodos que dan cobertura son las localizaciones óptimas. Si no, ir a paso 3.

Algoritmo para Problema 1 -Mediano en árboles.

El método teórico-gráfico es exitoso cuando la red presenta una estructura de árbol no orientada. En particular, Goldman (1971) ha presentado un algoritmo que resuelve el problema 1-mediano en un árbol eficientemente. Para entender la base de este algoritmo se necesita entender la siguiente notación [Han79].

Se tiene que $g(V_j)$ es la demanda total de los nodos en el subconjunto V_j del conjunto *V*. Así $g(V_j) = \Sigma_{v_i \in V_j} g(i)$. Retirar cualquier nodo v_i y su correspondiente arista, y desconectar el árbol *T* en dos o más componentes y particiones del conjunto de nodos. Por lo tanto para cualquier nodo v_I podemos definir conjuntos mutuamente

excluyentes V_i, V_i, y $\{v_i\}$, tal que $V_i \cup V_i \cup \{v_i\} = V$. Cuando v_i no es un nodo final, cualquier ruta desde $v_j \in V_i$ a $v_k \in V_i$ debe pasar a través de v_i , y cuando v_i es un nodo final, entonces V_i o $\underline{V_i}$ debe ser vacío.

Se ha observando que si la mitad o más de la demanda total está en un nodo, entonces una solución óptima consiste en localizar el recurso en ese nodo. Para ver que esto es verdad, se considera cualquier solución en la cual el recurso esté situado en un nodo extremo de la red con excepción del nodo en el cual se ubica por lo menos la mitad de la demanda total. Considerar que el nodo con por lo menos la mitad de la demanda total sea el nodo v_a con la demanda $g(v_a)$. Ahora, considerar mover el recurso desde su posición actual una distancia d hacia el nodo v_a. Los cambios en la función objetivo pueden ser determinados por d tiempos de diferencia entre el número de demandas que son cercanas al recurso en su nueva localización y los números de demandas que son lejanas al nuevo recurso. Pero dado que se acerca al nodo v_a, se acerca también a por lo menos la mitad de la demanda total, $g(V)$. La función objetivo mostrará una caída de $d(2g(v_a)-g(V))>=0$ como mínimo. Así, moviendo el recurso hacia el nodo v_a una distancia d no le restará nada a la función objetivo. Por lo tanto, se ha demostrado que al menos una solución óptima consiste en ubicar el recurso en el nodo que contiene la mitad o más de la demanda. Notar que esta propiedad es independiente si la red es un árbol o no. En otras palabras, si se localiza un único recurso en la red y si un solo nodo tiene por lo menos la mitad o más de la demanda total, es óptimo colocarlo en ese nodo. Pero cuando no existe ningún nodo que contenga por lo menos la mitad o más de la demanda total puede demostrarse que una solución óptima para el problema 1-mediano en un árbol puede

ser encontrada acoplando la demanda de un nodo del extremo sobre el nodo incidente y eliminando del árbol al nodo extremo y su arista. Este proceso es repetido hasta que un nodo del nuevo árbol contenga por lo menos la mitad o más de la demanda total del árbol. Un nodo extremo es un nodo que es incidente sólo en otro nodo. Notar que no es necesario conocer el largo de las aristas del árbol, sólo se debe conocer la demanda de cada uno de los nodos.

El siguiente algoritmo, escrito por Goldman, determina el mediano absoluto de un árbol:

Paso 1:

Si T consiste de un único nodo parar; el nodo es el absoluto mediano.

Paso 2:

Buscar un nodo final v_i y su arista asociada (i,j). Si $g(i) >= ½\ g(V)$ ir a paso 4, sino ir a paso 3.

Paso 3:

Modificar T, borrando v_i y (i,j), y incrementar $g(j)$ por $g(i)$. Retornar al paso 1.

Paso 4:

Parar, v_i es el absoluto mediano.

Cálculo de rutas mínimas

En los algoritmos para redes recién explicados se utiliza una matriz de rutas mínimas, la cual debe ser calculada previamente a la aplicación de tal algoritmo. Se

ha decidido resolver este problema utilizando el algoritmo de Dijkstra, algoritmo iterativo que debe ejecutarse para todos y cada uno de los nodos de la red. Este algoritmo funciona asignando un indicador de la distancia del nodo origen al nodo en cuestión. Así, se hace una partición de la red: se crea un conjunto de nodos con indicador permanente (conjunto *P*) y un conjunto de nodos con indicador tentativo (conjunto *T*), es decir, un conjunto de nodos cuyo indicador puede cambiar en las siguientes iteraciones [Hut82] [Ahu93].

Los pasos que se siguen son:

Se considera :

- d_{ij} distancia del enlace entre dos nodos *i* y *j* contiguos. Si no existe enlace entre dichos nodos, valdrá infinito. Si *i* es igual a *j* (se trata del mismo nodo), esta distancia será nula.

- Para un nodo *s*, calcula el camino más corto con origen *s* y destino el resto de los nodos de la red *(*árbol de divergencia*)*.

- D_i es la distancia del camino que va de nodo *s* al nodo *i*. $D_{ij} = D[\ i,j]$ es la distancia entre dos nodos no contiguos.

- *P* es el conjunto de los nodos que tienen una etiqueta de distancia (D_i) que es permanente. $P[\ i,j] = (K_1 = i, ..., K_n = j)$ es el camino tal que

  $d[P[i,j]] = \sum_{n=1}^{m-1} d_{k_n k_{n+1}} = D[i,j]$

- *T* es el conjunto de los nodos que no están en *P*.

Paso 1. Inicialización:

$D_s = 0$

$D_i = d_{s,j}$ " $j <> s$

P = {s}

$T = \{resto\_nodos\}$

Paso 2:

Encontrar i Ï $P | D_i = min_{j \check{I} P}\{D_j\}$. Puede existir empate, es decir, dos caminos igual de cortos, eligiéndose uno de ellos arbitrariamente o de acuerdo con un criterio marcado (no se soporta balanceo de carga). Hacer $P = P$ $\grave{E}\{i\}$. Si P contiene a todos los nodos, se para. Si no, se continua con el paso 3.

Paso 3:

Actualización : $\forall j \notin P$, $D_j = min\{Dj , d_{i,j}+D_i\}$. Ir a paso2.

3. Herramientas de Desarrollo

3.1. Herramienta SIG

¿Qué es un SIG?

S.I.G. o G.I.S. : Acrónimo de Sistema de Información Geográfica o Geographic Information System [URL6].

Algunas definiciones:

- Según el *National Center for Geographic Information Analysis* (NCGIA) un SIG es:"Sistema de hardware, software y procedimientos elaborados para la obtención, gestión, manipulación, análisis, modelado ,representación y salida de datos espacialmente referenciados para resolver problemas concretos de planificación y gestión" [URL2]
- "Conjunto de programas y aplicaciones informáticas que permiten la gestión de datos organizados en bases de datos, referenciados espacialmente y que pueden ser visualizados mediante mapas" (Ozemoy, Smith y Sicherman, en su libro "Evaluating computerized geographic information system using analysis", 1981)[Mag91]
- "Un poderoso conjunto de herramientas para coleccionar, almacenar, extraer, transformar y desplegar datos espaciales del mundo real" [Bur98]

Su aplicación principal es el manejo de datos geográficos a través de un banco de datos, cuyo objetivo es la creación de modelos digitales y de mapas temáticos.

Estas cualidades distinguen a los SIG de otros sistemas de información y lo hacen valioso para una amplia gama de las empresas públicas y privadas para explicar los acontecimientos, predecir resultados y planear estrategias [URL3].

La vida de un sistema de información geográfica está formada por los siguientes procesos:

- Adquisición de datos: Es el proceso de identificación y recopilación de los datos requeridos para la aplicación
- Proceso previo: consiste en la manipulación de los datos de diferentes formas para que puedan ser ingresados al SIG.
- Administración de datos: Consiste en la creación de funciones para tareas de acceso, actualización, ingreso y borrado de datos en la base de datos.
- Análisis y manipulación: son el foco de atención para el usuario del sistema. En esta parte del sistema trabajan operadores analíticos que derivan nueva información.
- Generación del producto: en esta fase son creadas las salidas finales del SIG. Estas pueden incluir reportes estáticos, mapas y gráficos [Sta90].

Un sistema SIG consta de cinco componentes:

- Hardware: Computador en el cual el software SIG se ejecuta. El software de SIG se ejecuta en una amplia gama de hardware, en red o aislados. Además

son necesarios dispositivos de ingreso y despliege de datos como impresoras, plotters, digitalizadores, entre otros.

- Software: El software SIG proporciona las funciones y herramientas para guardar, analizar y mostrar la información geográfica
- Datos: El componente más importante, posiblemente, son los datos. Los datos geográficos y los datos alfanuméricos se pueden relacionar sin ningún problema. Se divide en cuatro grupos: datos de entrada, datos de almacenamiento y administración de base de datos, datos de salida y representación, datos de trasformación.
- Personas: La tecnología SIG es un valor limitado sin las personas que se ocupan de él y desarrollan soluciones para resolver problemas en la vida real.
- Métodos: Un SIG funciona correctamente cuando se tiene un plan de trabajo bien diseñado y unos objetivos de la organización bien definidos.

En la figura 3 se observan los componentes de tipo hardware mas impoprtantes de un sistema de información geográfico.

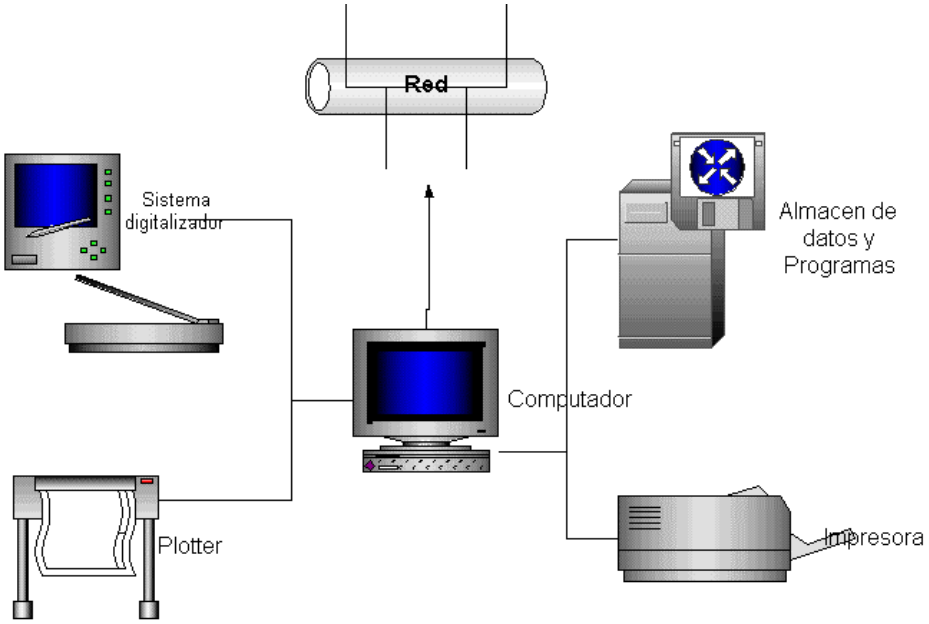

Figura 3. Los componentes hardware mas importantes de un sistema de información geográfico.

En la figura 4 se observan los componentes de tipo software que forman parte de un SIG.

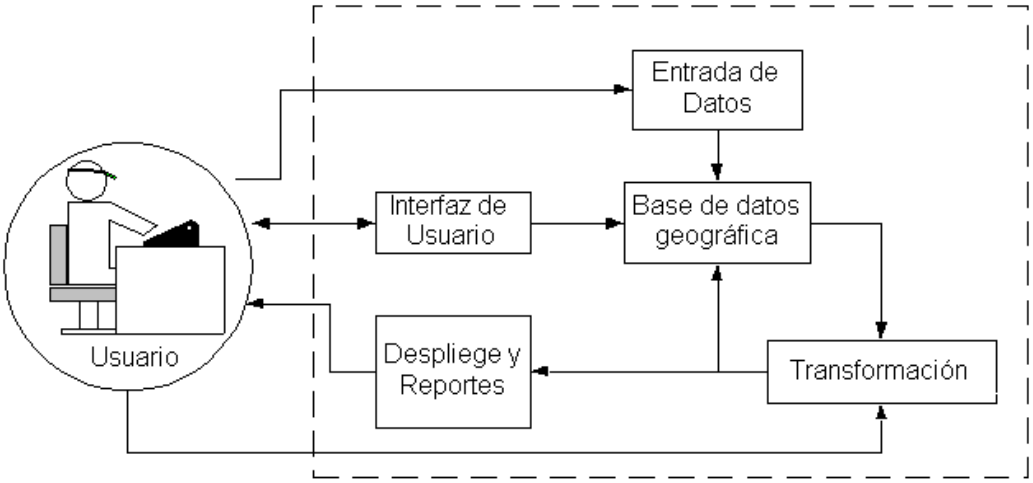

Figura 4. Los principales componentes de software de un sistema de información geográfica.

Un SIG almacena la información como una colección de capas temáticas que pueden ser encontradas juntas en la geografía, ver figura 5.

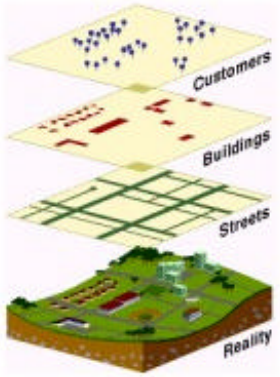

Figura 5. Capas temáticas.

Un aspecto fundamental dentro de los sistemas SIG es la forma de almacenar la información. Si bien en el inicio de estos sistemas era habitual que la gestión de esta información se realizara mediante programas propios, la tendencia actual es la

de desligar el producto SIG del gestor de la base de datos utilizado, de forma que sea posible utilizar cualquiera de los productos que para este fin existen en el mercado.

La exactitud y el nivel de resolución son elementos importantes en el desarrollo de una base de datos de un SIG, y vienen determinados por el uso al que vaya destinado el sistema. Así, un SIG diseñado para aplicaciones de ingeniería requerirá, en general, un alto nivel de exactitud y una gran resolución. Sin embargo, sistemas pensados para planificaciones o análisis parcelarios no requieren ese alto nivel de exactitud y detalle, sobre todo teniendo en cuenta que el precio de una base de datos gráfica aumenta exponencialmente cuando se incrementa el nivel de resolución. Ambos aspectos, coste y nivel de detalle, deben ser analizados cuidadosamente con objeto de optimizar el diseño de una base de datos para un Sistema de Información Geográfica [URL4].

La generación de la base de datos inicial incluye la captura e integración de datos que generalmente proceden de fuentes diversas. Estas fuentes a menudo presentan diferentes escalas y formatos que deben ser unificados. Una base de datos completamente integrada requiere entidades de control y referencia a las que se deben ajustar otras entidades que se incorporan en las distintas capas de la base de datos. Cada una de las capas y entidades tienen una serie de características que influirán en el desarrollo de la base de datos inicial, en los procesos de mantenimiento y en las aplicaciones en las que vayan a ser utilizadas.

El SIG trabaja fundamentalmente con dos modelos geográficos: vectoriales y raster. En la figura 6 se observa las diferencias entre estos dos formatos.

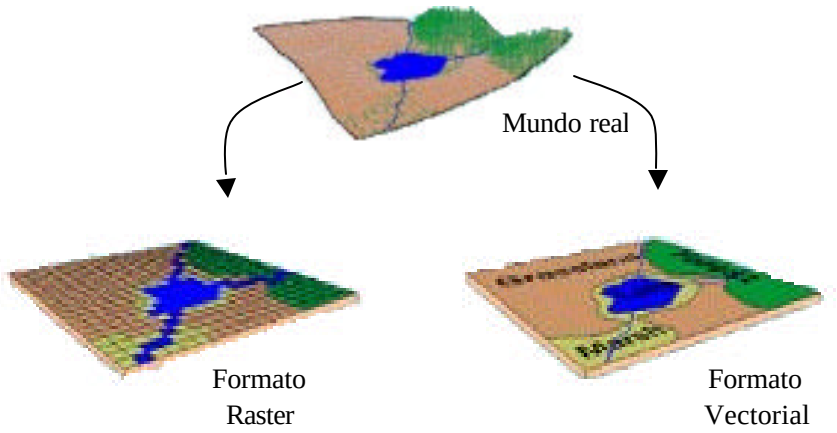

Figura 6. Representaciópn del mundo real en formato Raster y Vectorial.

En el modelo raster el espacio es discretizado en pequeños rectangulos o cuadrados, de forma que el tamaño que tienen estos elementos es fundamental y determina la resolución. Utiliza una única primitiva muy similar al punto, el pixel. Una malla de puntos de forma cuadrada o rectangular que contiene valores numéricos representa las entidades cartográficas y sus atributos a la vez. Los modelos lógicos menos complejos son los basados en el modelo conceptual raster, en buena medida porque la georreferenciación y la topología son implícitas a la posición - columna y fila - del pixel en la malla. Cada atributo temático es almacenado en una capa propia. La separación entre datos cartográficos y datos temáticos no existe, pues cada capa representa un único tema y cada celda contiene un único dato numérico. La precisión de la georreferenciación en el modelo raster está sesgada conceptualmente por la porción del territorio que representa el pixel, la cual es la unidad de medida lineal y superficial mínima del sistema. El modelo raster tiene

serias limitaciones conceptuales en la precisión de la referenciación, con un margen de error equivalente a la mitad de la base y de la altura del pixel.

Para emular la precisión del sistema vectorial, el sistema raster necesita mucho más espacio de almacenamiento de datos. El almacenamiento interno de la información asociada a cada capa es un aspecto de la mayor relevancia.

El modelo raster tiene una organización muy simple de los datos, lo cual permite realizar con gran facilidad ciertos procesos de análisis, como por ejemplo la superposición de planos, muy fácil de programar mediante operaciones con matrices, esta operación computacionalmente muy costosa cuando los temas están en formato vectorial, se realiza muy rápidamente y automáticamente si los temas son raster, pero el resultado estará afectado de un error debido a la discretización. Sus gráficos, aunque deficientes, se pueden realizar con dispositivos baratos, como por ejemplo una impresora matricial. Sus inconvenientes son el gran volumen de almacenamiento que requiere, la baja calidad de las representaciones gráficas y la dificultad de realizar análisis complejos sobre los gráficos así almacenados. Por último, el modelo raster no reconoce explícitamente la existencia de objetos geográficos, y por tanto, en las aplicaciones en que sea esencial su empleo, este modelo no podrá ser utilizado.

El modelo vectorial se basa en tres primitivas básicas:

- el nodo: es la unidad básica para representar entidades con posición pero sin dimensión (al menos a la escala escogida)
- la línea o el arco: representa entidades de una dimensión y está restringido a línea recta en algunas implementaciones
- el polígono o área: se utiliza para representar las entidades bidimensionales.

Algunos autores añaden una cuarta, el volumen. Entre ellas existen una serie de relaciones tales como que una línea se define por dos o más puntos (nodos), o un área está limitada por una serie de líneas, lo cual constituye una mínima definición topológica.

La posición de los datos puede ser georreferenciada directamente, por medio de un sistema de coordenadas, o indirectamente, utilizando por ejemplo la dirección postal, en ambos casos la solución es muy eficaz. Los atributos no espaciales son almacenados en una base de datos alfanuméricos interrelacionada con la base de datos cartográficos, ofreciendo con ello posibilidades muy distintas de las del modelo raster. Las interrelaciones topológicas se explicitan hasta el último detalle y con gran sofisticación. En el caso vectorial, no hay ninguna limitación conceptual en la precisión de la georreferenciación, hay únicamente una limitación matemática y física de dígitos del hardware.

Éste modelo es mucho más parecido a la percepción humana del espacio que la que ofrecen los modelos raster y en parte por ello tiene más variantes y más dificultades añadidas.

Está más de acuerdo con la cartografía tradicional y, por ello, resulta más intuitiva. Pero la principal ventaja de este modelo respecto del modelo raster es su capacidad para expresar las relaciones espaciales existentes entre las entidades, esto es, la información topológica, que es la que dota al modelo de la semántica necesaria para representar el conocimiento territorial.

Históricamente se han diferenciado SIG vectoriales y raster. Actualmente los principales sistemas SIG combinan ambos tipos de estructuras.

Selección de la herrramienta SIG

Sobre lo que a herramientas SIG se trata , en nuestro país son utilizadas principalmente Mapinfo, Idrisi de Clark Labs, Pamap de PCI Geomatics Group y ArcInfo de Esri.

A continuación se hará una pequeña referencia sobre cada una de estas herramientas:

MAPINFO :

Algunas caracteristicas:

- Permite escoger entre diferentes formas interactivas de visualizar datos.
- Permite editar y crear nuevos mapas y datos de tablas.
- Produce mapas que muestran claramente lo que pasa con lo datos y se puede conectar de manera directa a un servidor de bases de datos como Oracle, Informix 5, Sybase.
- Es un paquete de software comercial.

Algunas características desventajosas son : no contiene originalmente herramientas especificas o personalizadas de edición y corrección de cartografía. Su lenguaje propio de creación de aplicaciones y personalización no es orientado a objetos. No es posible generar aplicaciones con funcionalidades extra que sean independientes del proyecto. [URL5]

IDRISI :

Desarrollado por Clark Labs, es una tecnología de modelación geográfica que apoya la toma de decisiones ambiental para el mundo real.

Es una herramienta creada por una institución educativa y de investigación, por lo tanto está orientada a esos fines.

Costo asequible o posibilidades de conseguir en forma gratuita para todos sus usuarios potenciales.

Funciona en computadores personales y no se requieren tarjetas gráficas caras ni dispositivos periféricos para poder hacer uso completo del poder analítico del sistema.

Es un programa de tipo raster, pero permite la utilización de información de tipo vectorial en algunas de sus funciones.

PAMAP :

Herramienta para el análisis de información espacial tanto como no-espacial.

Pamap cuenta con diferentes módulos con los cuales es posible abarcar diferentes áreas del análisis espacial.

Entre las funcionalidades de éstos módulos están:

- Pamap Mapper: Generación de reportes estadísticos, análisis de proximidades para puntos, segmentos, polígonos y pixeles, superficies de coberturas para ser usado en análisis de distancias, superposición de múltiples capas de información, entre otras.

- Pamap topographer: Modelos Digitales de Elevación (MDE) desde coberturas de líneas en 3D o desde cotas, generación de mapas de pendientes.
- Pamap Modeller : Puede ser aplicado en diferentes campos como el análisis de imágenes satelitales y análisis de monitoreo ambiental.
- Pamap Networker : Generar topología de redes desde elementos puntuales o desde lineas, generación de rutas óptimas. Calcular la ruta de menor costo o la más rápida con base en la red, entre otras [URL3].

ARCINFO :

Es una herramieta que permite administrar un SIG, uno de sus módulos es ArcView, el que está encargado de la visualización y proceso de los datos. Algunas carasteristicas de ArcView son:

- Trabaja con información de tipo vectorial.
- Ofrece opciones de análisis espacial y tratamiento de datos geográficos.
- Permite la representación de datos por georeferenciación sobre una cartografía, analizar las características y patrones de distribución de esos datos, y generar informes finales con los resultados.
- Permite integración de aplicaciones a traves de Comandos a nivel de sistema, Comunicación entre aplicaciones (IAC) y mediante llamadas a librerias de enlace dinámico (DLL).

La herramienta seleccionada fue Arcview, su elección fue debido a las funcionalidades que presta, a su uso más masificado y a que en Forestal Valdivia cuentan con este software.

ArcView

El GIS ArcView, es una herramienta poderosa y fácil de usar, que permite trabajar con información geográfica. ArcView permite visualizar, explorar , buscar y analizar datos espaciales [Env94].

Una sesión de Arcview se compone de: vistas, tablas, gráficos, layouts, scripts. Estos componentes se guardan en un archivo llamado proyecto. A continuación se explica brevemente cada uno de estos componentes:

- Vistas :El GIS ArcView trabaja con datos de mapas geográficos interactivos llamadas vistas. Cada vista en ArcView contiene una tabla con los datos geográficos, lo que le hace fácil de entender y de analizar. En las vistas se pueden editar, añadir, eliminar datos de un cierto archivo.
- Tablas : ArcView trabaja con datos tabulares en lo que se conoce como tablas, lo que permite obtener un rango lleno de características para conseguir estadísticas sumarias, ordenar los datos, realizar búsquedas, etc.
- Gráficos : Los gráficos de ArcView ofrecen una amplia variedad de formatos, tanto en barras, pie, lineas, etc., que ofrecen una poderosa capacidad de visualización de los datos que se integra totalmente en el ambiente geográfico

del ArcView. ArcView permite trabajar simultáneamente con datos tanto geográficos y tabulares.

- Layouts : Los layouts o esquemas de ArcView permiten crear una calidad alta de mapas a color. Se dispone de una amplia facilidad para elegir los dispositivos de impresión. Los esquemas son inteligentes porque tienen un eslabón vivo a los datos que representan. Cuando imprime un esquema, cualquier cambio a los datos se incluye automáticamente, así sabe que todo en su mapa será actualizado.
- Scripts : Los scripts en ArcView son macroinstrucciones escritas en Avenue que es el lenguaje de programación de ArcView y ambiente de desarrollo. Con Avenue se puede personalizar casi cada aspecto de ArcView, de agregar un botón nuevo, correr un script que se desarrolle, o crear una aplicación [Env96].

ArcView usa datos que describen cualquier parte de la superficie de la Tierra o las características de los datos geográficos. Este no solamente incluye datos de cartografía y datos científicos, sino también datos de ventas, archivo de la tierra, fotografías, banco de datos del cliente, etc.

Los *datos espaciales* son el corazón de cada aplicación en ArcView. Los datos espaciales almacenan la localidad de la característica geométrica de un atributo. Los datos que se pueden usar son los siguientes:

ArcView (shapefiles), archivos propios del ArcView.

ARC/ INFO (coverages), archivos creados en el SIG Arcinfo.

ARC/ INFO (grids) Capas de los datos en Arcinfo,

Los *datos de imágenes* incluyen imágenes del satélite, fotografías aéreas. Los datos que puede usar incluyen:

TIFF imáge nes de tipo tiff.

TIFF/ LZW datos de la imagen tiff comprimidos

ERDAS , *ERDAS* IMAGINA archivos de otros SIG.

ARC/ INFO datos de tipo grid.

JPEG imágenes de tipo JPG

Los *datos tabulares* pueden incluir casi cualquiera de los tipos de datos que se disponen si contiene o no datos geográficos. Se pueden desplegar tablas en una vista directamente, otros proveen atributos adicionales que se puede unir a datos espaciales ya existentes. ArcView soporta formatos como:

dBASE III archivos, pueden ser creados en Excel

dBASE IV archivos , pueden ser creados en Excel

INFO tablas de tipo info, creadas por el Arc/Info.

Archivo de texto separado con tabs o comas [URL6].

3.2. Lenguaje de Programación

Se determinó que el lenguaje de programación a utilizar sería Visual Basic por su compatibilidad con Arcview y debido a que las últimas versiones de esta herramienta permiten la creación de procesos escritos en este lenguaje.

Visual Basic es el lenguaje de programación que permite crear aplicaciones para Microsoft Windows.Las aplicaciones escritas en este lenguaje están basadas en objetos y son manejadas por eventos.

La palabra "Visual" hace referencia al método que se utiliza para crear la interfaz gráfica de usuario (GUI). En lugar de escribir numerosas líneas de código para describir la apariencia y la ubicación de los elementos de la interfaz, simplemente se pueden agregar objetos prefabricados en su lugar dentro de la pantalla. La palabra "Basic" hace referencia al lenguaje BASIC (Beginners All-Purpose Symbolic Instruction Code), un lenguaje utilizado por más programadores que ningún otro lenguaje en la historia de la informática o computación. Visual Basic ha evolucionado a partir del lenguaje BASIC original y ahora contiene centenares de instrucciones, funciones y palabras clave, muchas de las cuales están directamente relacionadas con la interfaz gráfica de Windows. Los principiantes pueden crear aplicaciones útiles con sólo aprender unas pocas palabras clave, pero, al mismo tiempo, la eficacia del lenguaje permite a los profesionales alcanzar cualquier objetivo que pueda lograrse mediante cualquier otro lenguaje de programación de Windows.

El programa creado en Visual Basic es conocido como proyecto (*.vbp), donde el área de diseño es un formulario (*.frm). Un formulario es en realidad una ventana de Windows, de esta manera el programador dará forma al programa situando los diferentes botones, cajas de texto, etc., en toda el área del formulario.

La programación en Visual Basic utiliza principalmente objetos gráficos como botones, ventanas, cajas de texto, gráficos, etiquetas, etc. conocidos como ***controles***. Todo control cuenta con una variedad de características modificables llamadas ***propiedades***. La forma o diseño de un control depende de los valores

indicados en sus propiedade s como son: color, tamaño, ubicación, títulos, tipos de letra, bordes, visibilidad, alineación, etc.

Otro de los aspectos manejados en VB son los ***eventos***, los cuales identifican acciones como click, dobleclick, movimiento del puntero, entre otras, aplicables sobre el control. A cada evento le corresponde una función la cual contiene código, es decir, las instrucciones de la tarea a realizar bajo dicho evento [URL0].

4. Descripción funcional del Sistema.

Las empresas forestales hoy se enfrentan frecuentemente a problemas que demandan decisiones estratégicas de localización. Para las diferentes tareas que involucra la actividad forestal es necesario contar con lugares donde almacenar las materias resultantes de estas acciones. De las actividades de cosecha y raleo, por ejemplo, se generan materias primas que deben ser almacenadas antes de su traslado al punto final de comercialización o utilización; en otras tareas forestales también es necesario tomar decisiones de localización como ubicación de cuadrillas de control de incendio o torres de vigilancia o en la ubicación de servicios de salud de emergencia.

Actualmente, en las empresas nacionales estas decisiones se realizan en conjunto entre quien dirige la actividad, los supervisores de dicha actividad y los encargados de los predios involucrados basados en la experiencia y conocimiento del lugar.

Esta tesis pretende analizar algoritmos de localización óptima e implementarlos en un Sistemas de información geográfica, para poder brindar una herramienta que ayude en las decisiones de localización de establecimientos forestales.

4.1. Ingreso de información.

La información necesaria para poder aplicar un algoritmo de localización óptima está siempre ligada a costos o tiempos, ya que siempre el objetivo es reducir

estas variables. Para el caso descrito es necesario contar con la siguiente información:

- De predios: es necesario conocer que predios son los que estarán implicados en la actividad forestal a desarrollar, los cuales deben contar con identificadores únicos; y su oferta de materia prima. La oferta de materia prima se refiere al volumen de materia que se extraerá del lugar, ésta puede ser calculada aproximadamente considerando el área o superficie del terreno.
- De rutas: es necesario conocer las rutas involucradas en la actividad, o sea debe tener una identificación que la haga única; también debe contarse con el costo asociado a transitar por la ruta y el punto de inicio y término de esa ruta, los cuales coinciden con los predios. Al referirse al costo de transitar éste podría ser reemplazado por el tiempo en transitar cuando el caso lo estime necesario. Este costo de transitar depende del estado del camino, se diferencian tres tipos de caminos , ruta 5 , camino pavimentado y camino de ripio-tierra, cada uno de éstos establece un costo diferente estimado por al empresa.

Para ingresar la información previamente se deben conocer los predios y rutas que participarán en la actividad. Así la información a ingresar corresponde a las rutas seleccionadas, incluidos los predios que ellas unen y su costo asociado, y los predios seleccionados y las ofertas de material de cada uno de ellos. En resumen el ingreso de información corresponde a ingresar la red en donde se presenta el problema de localización.

4.2. Obtención de la localización.

Después de ser ingresada la información se procede a aplicar un algoritmo de localización óptima.

En primer lugar se hará diferencia de acuerdo a la estructura de la red, la que puede ser de tipo árbol y no árbol, ambas con una componente conexa. Para ambos tipos de redes existen algoritmos de localización óptima, que darán como resultado el predio que será el mejor lugar para instalar un establecimiento. Se debe tener claro que los predios que participarán de la actividad forestal serán los mismos candidatos a ser localización óptima.

En segundo lugar, se podrá determinar la localización óptima siguiendo el criterio de minimizar la suma de costos o minimizar el máximo costo.

Cualquiera de los algoritmos aplicados entregará un resultado, este resultado podrá ser visual y mediante archivos que detallen la información.

4.3. Resultado e Informe de salida.

Después de la obtención de la localización óptima se generarán las salidas que permitirán conocer el resultado del algoritmo aplicado, estas salidas serán:

- Salida visual
- Archivo detallado

Salida visual

Se pretende entregar una salida visual a través de Arcview, creando una cobertura que permita visualizar el predio elegido complementado con alguna información adicional, como: sector geográfico cubierto, fecha de obtención de la localización y actividad para la cual se utilizará esa localización.

Al entregar una salida por medio de Arcview se estará contribuyendo al sistema de información geográfica de la empresa y asegurando el almacenamiento del resultado en un sistema utilizado habitualmente.

Archivo detallado

De la generación de la solución al problema planteado pueden surgir diferentes datos anexos a la solución buscada. Se pretende entregar un archivo de salida que contenga el nodo o predio elegido y los costos de llegar desde ese predio a los otros considerados en el problema. También puede surgir información anexa dependiendo del algoritmo que se aplique, como matrices de rutas óptimas, matrices de distancia –demanda entre otras.

El archivo detallado se entregará en un formato de uso frecuente dentro de la empresa, para lo que se evalúan diferentes alternativas.

5. Diseño e implementación del Sistema

5.1. Diseño

Diseño de Datos

El diseño de datos es la primera actividad de diseño que se realiza durante la ingeniería de software. El impacto de la estructura de datos sobre la estructura de programa y la complejidad procedimental, hace que el diseño de datos tenga una gran influencia en la calidad del software [Pre96].

El diseño de datos aplicado generó el modelo entidad-relación que se ve en la figura 7.

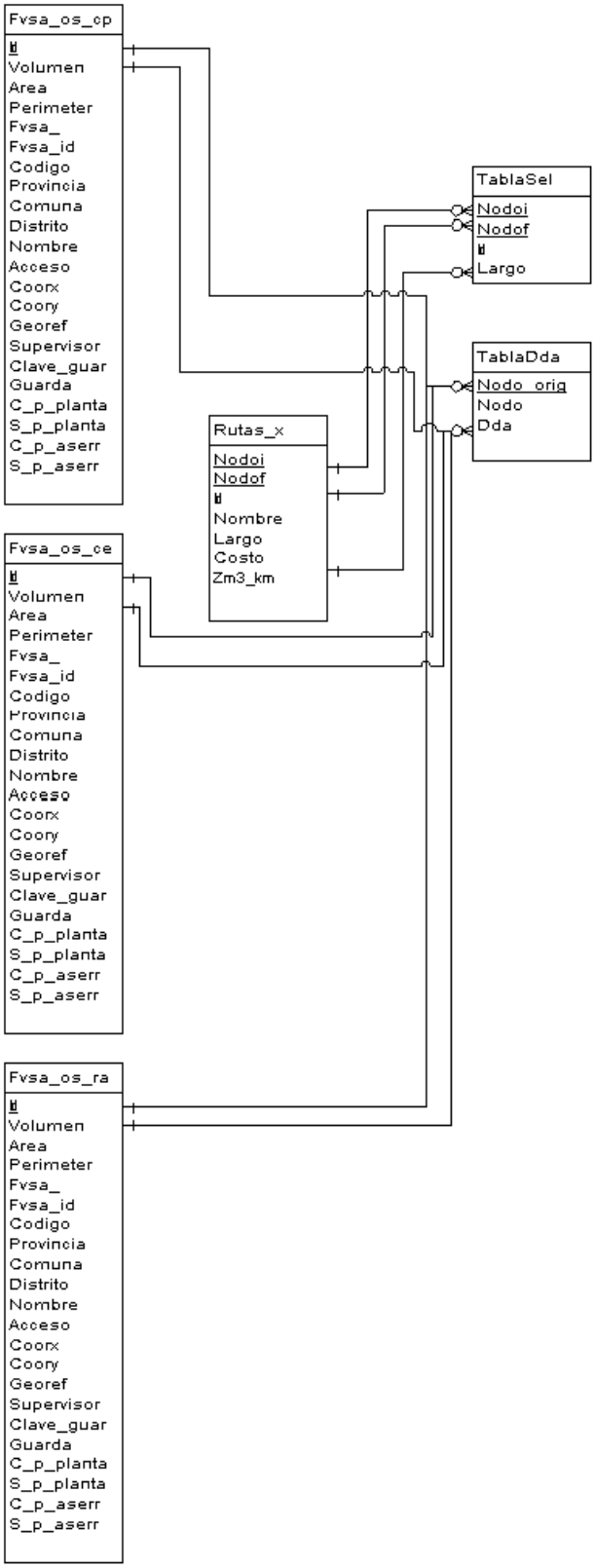

Figura 7. Modelo Entidad Relación.

A continuación se detallan las tablas participantes:

Fvsa_os_cp : Contiene los datos de los predios que se encuentran en un determinado sector geográfico(OS), a los cuales se les aplicará una determinada actividad forestal(C) y tienen una determinada especie forestal(P).

Fvsa_os_ce : Contiene los datos de los predios que se encuentran en un determinado sector geográfico(OS), a los cuales se les aplicará una determinada actividad forestal(C) y tienen una determinada especie forestal(e).

Fvsa_os_ra : Contiene los datos de los predios que se encuentran en un determinado sector geográfico(OS), a los cuales se les aplicará una determinada actividad forestal(RA).

Rutas: Contienen los datos de las rutas.

Tabla_dda: Contiene los datos de los predios seleccionados.

Tabla_sel: Contiene los datos de las rutas seleccionadas.

Existen cuatro tablas más que son creadas para almacenar datos de salida de procesos, por lo que no se consideran en el modelo entidad relación, estas tablas son las siguientes:

Tabla_rut: Contiene la matriz de rutas mínimas.

Tabla_dd: Contiene la matriz de distancias-demandas.

Tabla_out: Contiene datos del predio elegido.

Tabla_Det: Contiene los costos de ir del un predio al predio elegido.

A continuación se detalla la estructura de las tablas participàntes en el sistema:

Tablas Fvsa_os_cp, Fvsa_os_ce y Fvsa_os_ra

Estas tablas presentan la misma estructura y corresponden a coberturas dentro de Arcview.

Campo	Descripción
AREA	Area o superficie del terreno del predio.
PERIMETER	Perímetro del terreno del predio.
FVSA_ID	Código de identificación del predio.
REGION	Región de ubicación del predio.
PROVINCIA	Provincia de ubicación del predio.
COMUNA	Comuna de ubicación del predio.
DISTRITO	Distrito de ubicación del predio.
NOMBRE	Nombre del predio.
ACCESO	Descripción de la forma de acceder al predio.
COORX	Coordenada x de ubicación del predio.
COORY	Coordenada y de ubicación del predio.
GEOREF	Código de georeferenciación.
SUPERVISOR	Nombre del supervisor del predio.

CLAVE_GUAR	Clave de guarda del predio.
GUARDA	Nombre del guarda del predio.
ID	Identificador corregido del predio.
VOLUMEN	Volumen a extraer del predio.

Tabla TablaSel

Campo	Descripción
ID	Número de la arista seleccionada.
LARGO	Largo de la arista, este campo puede corresponder también a tiempo o costo de transitar.
NODOI	Número de nodo donde se inicia la arista.
NODOF	Número de nodo donde finaliza la arista.

Tabla TablaDda

Campo	Descripción
NODO_ORIG	Número original del nodo seleccionado.
NODO	Número reasignado del nodo seleccionado.
DDA	Demanda del nodo seleccionado.

Tabla TablaRut

Esta tabla corresponde a una matriz.

Campo	Descripción
N0	Número de nodo.
N1	Distancia de un nodo a nodo 1.
....	
Nx	Distancia de un nodo a nodo x.

Tabla TablaOut

Campo	Descripción
NODO	Número reasignado del nodo seleccionado.
NODO_ORIG	Número original del nodo seleccionado.
Nombre	Descripción del nodo elegido.

Tabla TablaDet

Campo	Descripción
NODO	Número reasignado del nodo seleccionado.
NODO_ORIG	Número original del nodo seleccionado.
COSTO	Costo de ir del nodo elegido al nodo especificado en el campo NODO.

Tabla TablaRes

Campo	Descripción
ID	Número identificador del nodo elegido..
NODO	Número reasignado del nodo seleccionado.
NODO_ORIG	Número original del nodo seleccionado.

Diseño Arquitectónico

El objetivo principal del diseño arquitectónico es desarrollar una estructura de programa modular y representar las relaciones de control entre módulos. Además, el diseño arquitectónico mezcla la estructura de programa y la estructura de datos y define las interfaces que facilitan el flujo de los datos a lo largo del programa [Pre96].

En esta etapa del diseño se comenzó con una representación sencilla del sistema, a través de un DFD en nivel 0 se esquematizó muy en resumen el problema planteado, ver figura 8.

Figura 8. DFD de nivel 0

En la figura 8 se puede observar la fuente de datos del sistema, que corresponde a coberturas de Arcview. El proceso principal corresponde a la búsqueda de la localización óptima que entrega como resultado información acerca de esa localización.

El proceso central, “Resolver problema de localización óptima”, puede ser expandido como se ve en la figura 9.

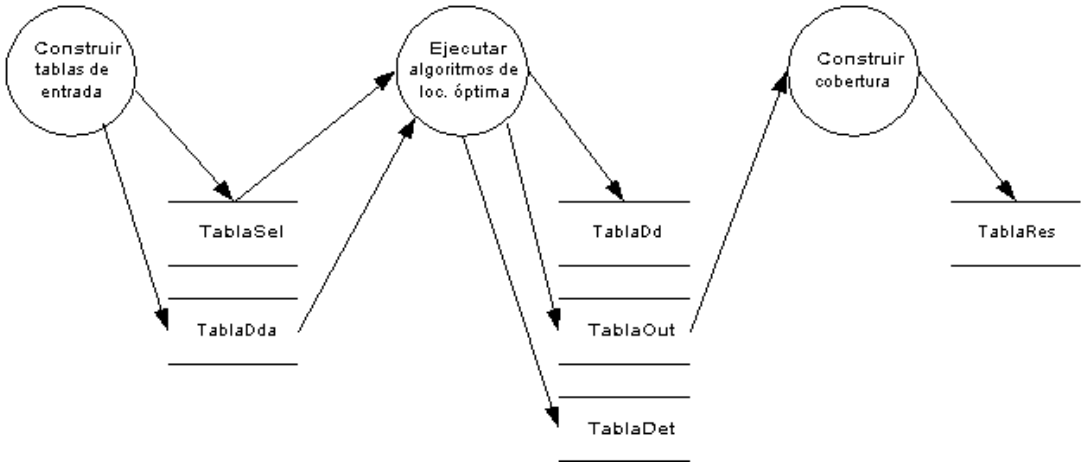

Figura 9. DFD nivel 1, proceso “Resolver problema de localización óptima”.

El siguiente DFD, ver figura 10, refina el proceso “Ejecutar algoritmo de localización óptima”:

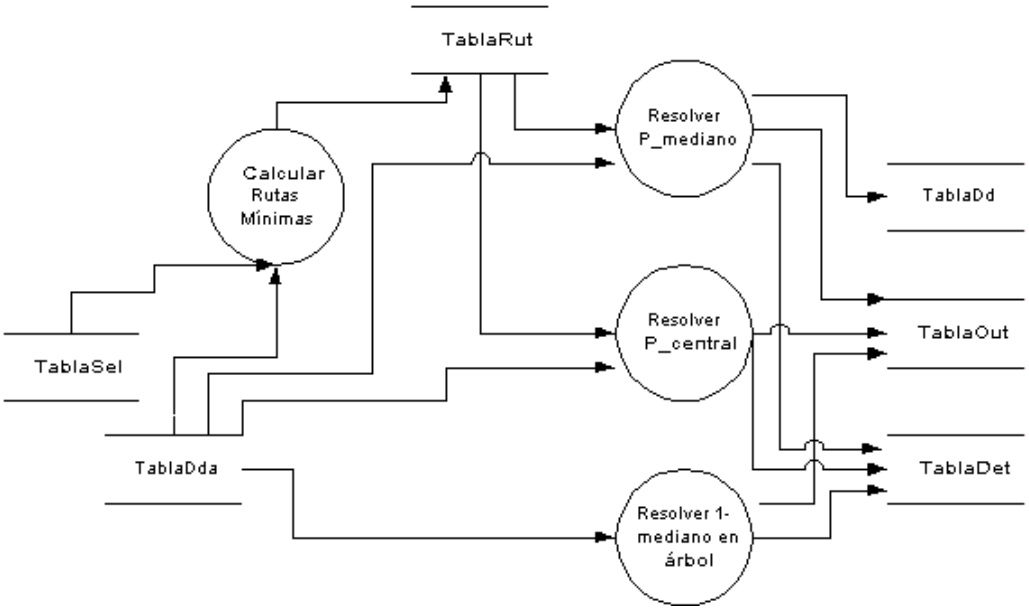

Figura 10. DFD nivel 2, proceso “Ejecutar algoritmo de localización óptima”.

Diseño Procedimental

El diseño procedimental transforma los elementos estructurales en una descripción procedimental del software. La representación gráfica que se usará para describir los procesos más importantes de este sistema es el diagrama de flujo.

Los procesos que se detallaran a continuación corresponden a aquellos procesos claves del sistema.

El primer proceso a especificar es “Calcular rutas mínimas”, figura 11. Este proceso entrega como resultado una matriz con las rutas mínimas entre nodos, en él se aplica el algoritmo de Dijkstra para cada par de nodos.

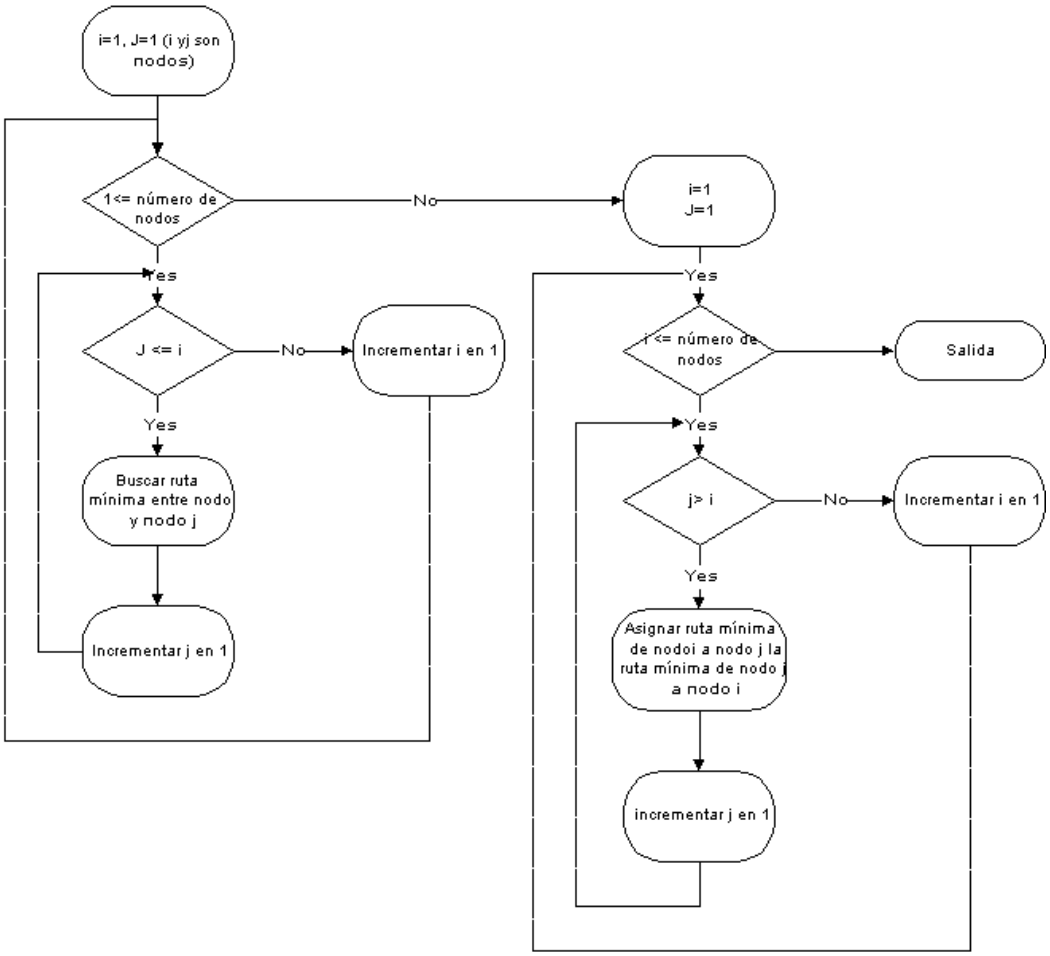

Figura 11. Algoritmo para cálculo de matriz de rutas mínimas.

El siguiente proceso, figura 12, corresponde a “Resolver p_mediano”. En este proceso es implementado el algoritmo Myopic, detallado anteriormente.

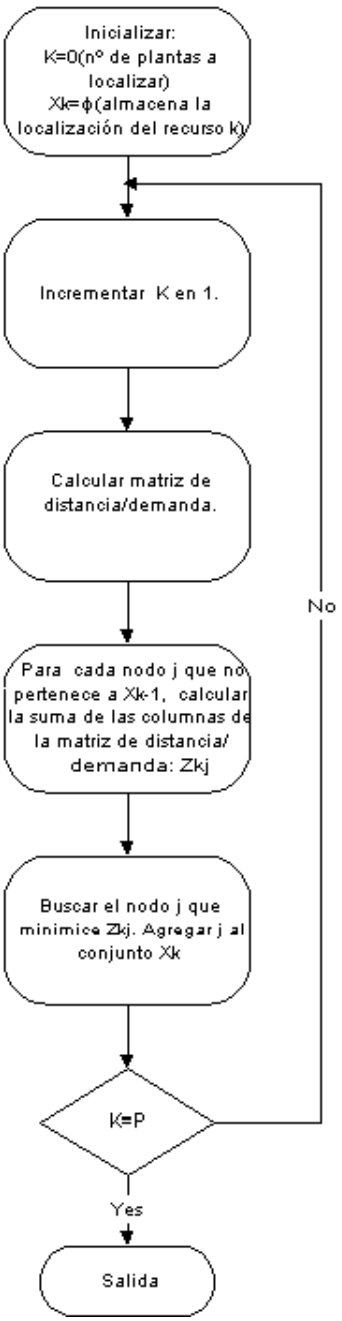

Figura 12. Algortimo Myopic para resolución de problemas p-mediano.

El siguiente proceso, figura 13, corresponde a “Resolver p_central”, mediante la implementación del algoritmo detallado anteriormente.

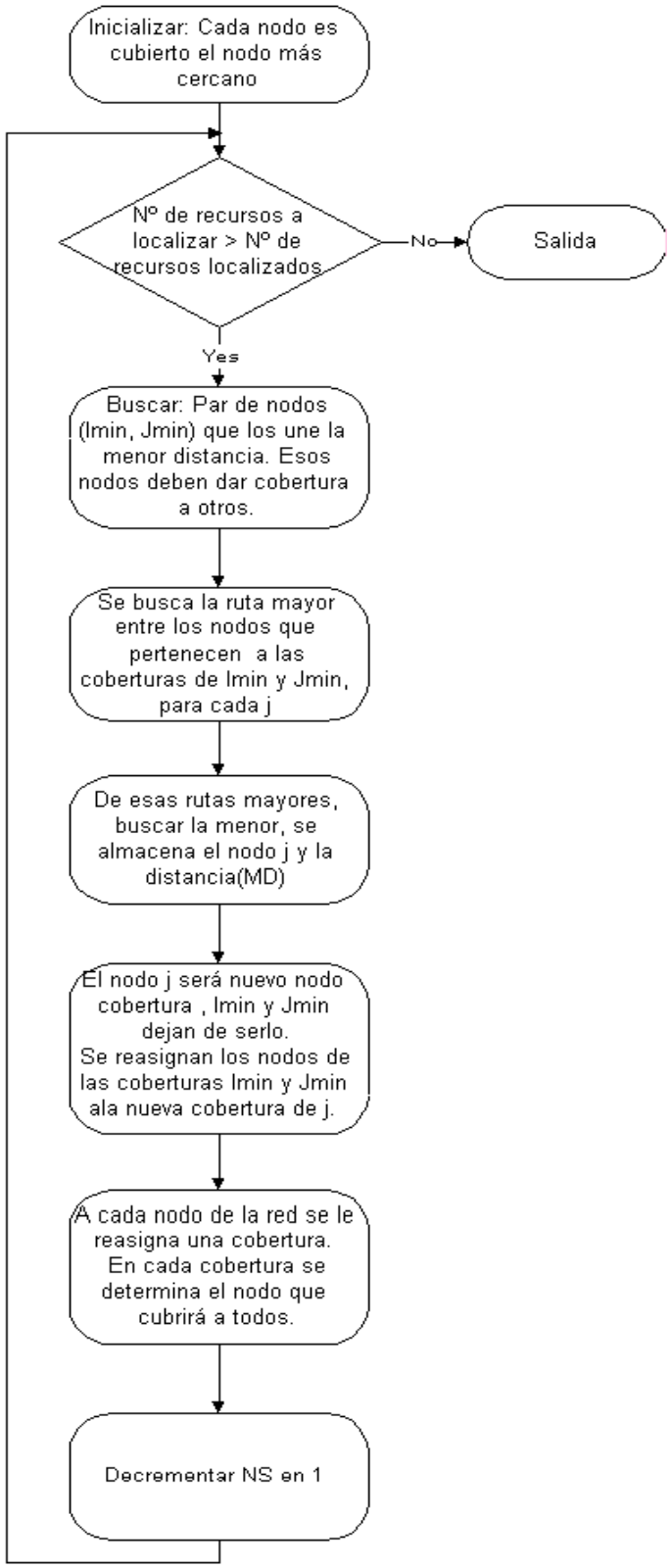

Figura 13. Algortimo para resolución de problema p-central.

El siguiente proceso, figura 14, corresponde a “Resolver 1-mediano para un árbol”, este proceso, bastante más sencillo que los anteriores, es resuelto mediante la metodología de minimizar el máximo utilizando el algoritmo que fue descrito en capítulo 2.

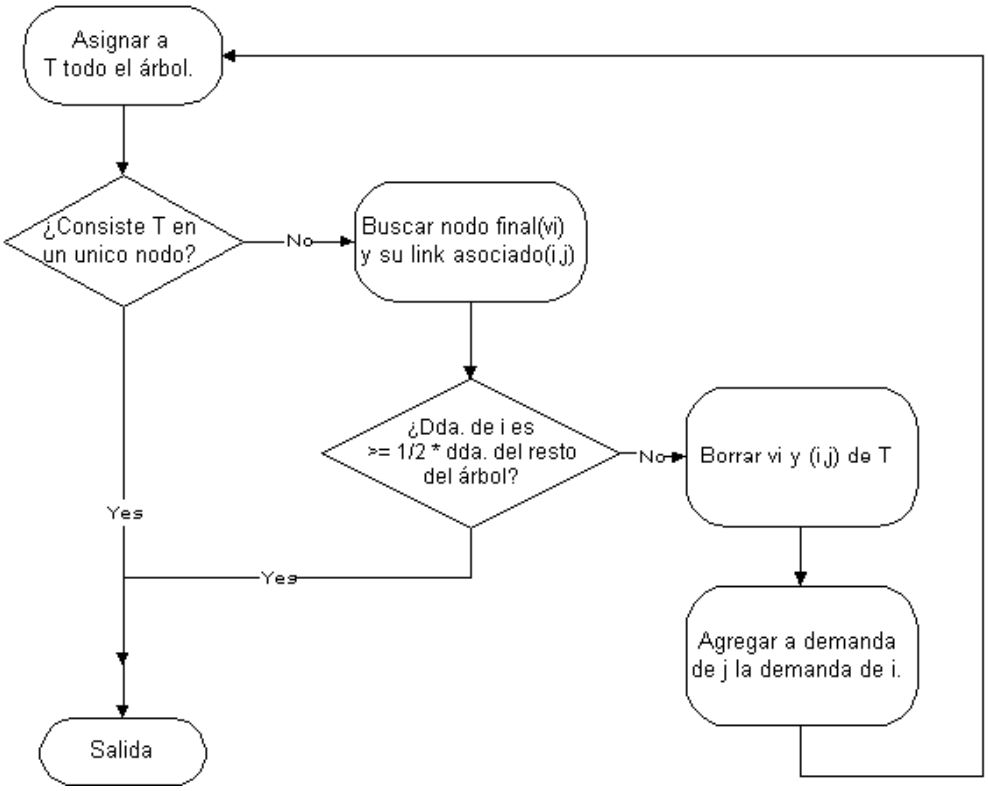

Figura 14. Algoritmo para solución de problemas 1-mediano en árboles.

Diseño de Interfaz

El diseño de interfaz establece la disposición y los mecanismos para la interacción hombre máquina.[Pre96]

Para el sistema que se implementará se ha decidido realizar un diseño de interfaz que permita utilizar fácilmente las funcionalidades de la aplicación, permita al usuario cierta libertad al momento de ejecutarse uno u otro proceso, pero que a su vez abstraiga lo menos posible al usuario de su ambiente tradicional de trabajo en Arcview.

Antes de llamar a la aplicación que realiza la búsqueda de la localización óptima es necesario preparar los datos, todas las actividades que estén relacionadas con preparación de datos o creación de tablas se harán en Arcview mediante la inserción de botones en el menú de botones de Vistas, estos botones serán dispuestos en la parte izquierda del menú, como se observa en la figura 15. Se ha determinado un número de botones a insertar según las tareas que se realizan. Por lo que se consideraron tres, uno para creación de tablas, un segundo para revisión y preparación de la red y un último botón para realizar la llamada a la aplicación que realiza la búsqueda de la localización óptima.

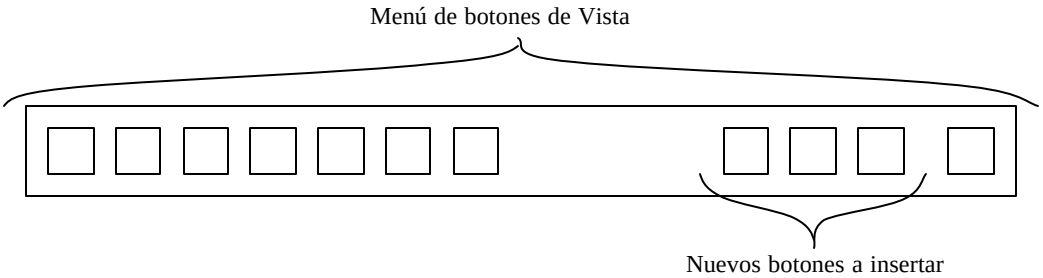

Figura 15. Ubicación de nuevos botones en la interfaz de Arcview.

El diseño de la interfaz de la aplicación pretende ser lo más simple posible para entregar en una sola pantalla la información más relevante. Se dispondrán en esta pantalla las tablas más importantes, una tabla de nodos seleccionados, otra de posibles distancias entre esos nodos y una última que correspondería a una matriz de rutas mínimas. También se ubicarán en esta pantalla botones para la ejecución de los algoritmos de localización óptima, otro botón para la ejecución del algoritmo de rutas mínimas y otro para salir de la aplicación. El layout de esta pantalla será aproximadamente como se muestra en la figura 16.

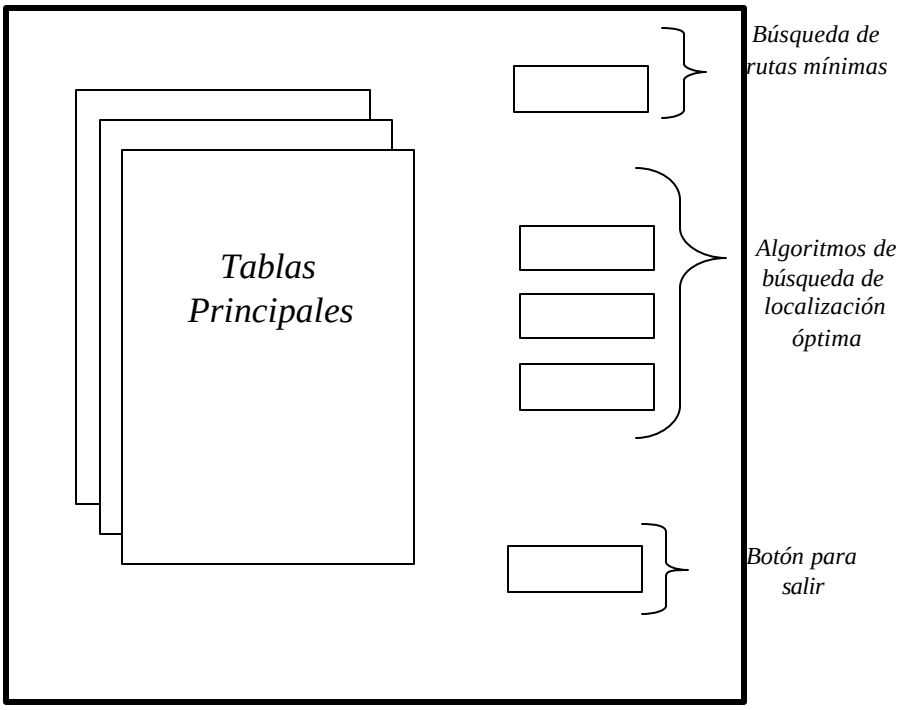

Figura 16. Diseño de la interfaz de la aplicación.

5.2. Implementación

La implementación del sistema es realizada tanto en Arview como en Visual Basic.

Implementación en Arcview

La implementación hecha en Arcview es por medio de scripts escritos en el lenguaje propio, Avenue. A continuación se describen estos scripts:

Crear_TablaDda: Este script crea la tabla que contiene datos de los nodos que han sido seleccionados.

Crear_TablaSel: Este script crea una tabla con datos de las aristas seleccionadas.

Crear_TablaRut: Este script crea una tabla para ser llenada al calcular rutas mínimas de la red.

Crear_TablaDd: Este script crea una tabla para ser llenada al calcular la distancia/demanda para cada nodo.

Crear_TablaOut: Este script crea una tabla para ser llenada con los datos del nodo elegido como localización óptima.

Crear_TablaDet: Este script crea una tabla para ser llenada con los detalles resultantes de la búsqueda de la localización óptima.

Cambia: este script llama a un archivo ejecutable encargado de la preparación de los datos, la cual es transparente para el usuario.

Localiza: Este script llama al archivo ejecutable encargado de la ejecución del sistema propiamente tal. Y llama al script Nodo_elegido.

Nodo_elegido: Crea una nueva cobertura con el(los) nodo(s) seleccionado(s) como localización óptima.

Implementación en Visual Basic

La implementación realizada en Visual Basic se encarga de los procesos principales del sistema. Esta implementación puede dividirse en dos partes, una encargada de la preparación de los datos y otra encargada de la ejecución de los procesos que resuelven el problema de localización óptima.

Específicamente la primera parte tiene como función revisar la red para detectar y eliminar aquellas aristas que no unen dos nodos pertenecientes a la red, y detectar y eliminar nodos que no estén unidos a la red por medio de una arista perteneciente a ésta. Otra función es reasignar números a los nodos para que estos presenten una numeración correlativa necesaria para los procesos posteriores.

La segunda parte tiene como objetivo la ejecución del sistema propiamente tal, en esta parte se implementan los algoritmos necesarios para la búsqueda de la localización óptima, que son los siguientes: algoritmo myopic, algoritmo para problemas p-central y algortimo para problema 1-central en árbol. Además de otros procesos previos como determinar si la red es árbol y calcular matriz de rutas mínimas.

6. Descripción de Funcionamiento del Sistema.

El sistema comienza su ejecución en un proyecto Arcview, antes de comenzar la ejecución de la aplicación se debe en primer lugar tener abiertas las coberturas que representarán los nodos y la cobertura que representará las aristas, en ese orden. Posteriormente deben seleccionarse los nodos y aristas que formarán parte de la red a la cual se le ubicará una localización óptima.

Para iniciar la ejecución en primer lugar debe presionarse el botón *1* del menú de Vistas, que se observa en la Figura 17

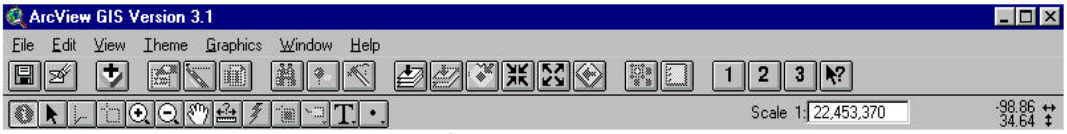

Figura 17. Menú de proyecto en Arcview.

En esta etapa se crean las tablas que se utilizarán a lo largo de la aplicación.

Para continuar, se debe presionar el botón *2* del menú, de esta forma se ejecutará el script Cambia con los resultados comentados anteriormente. Al término de la ejecución se desplegará un mensaje, como se observa en la figura 18, para informar al usuario que ya puede continuar con la ejecución del sistema.

Figura 18. Mensaje “Termino de proceso”

Para ello debe presionar el botón *3*, con esto se llama al script Localiza y se despliega la pantalla de la aplicación, esta pantalla es la que muestra la figura 19.

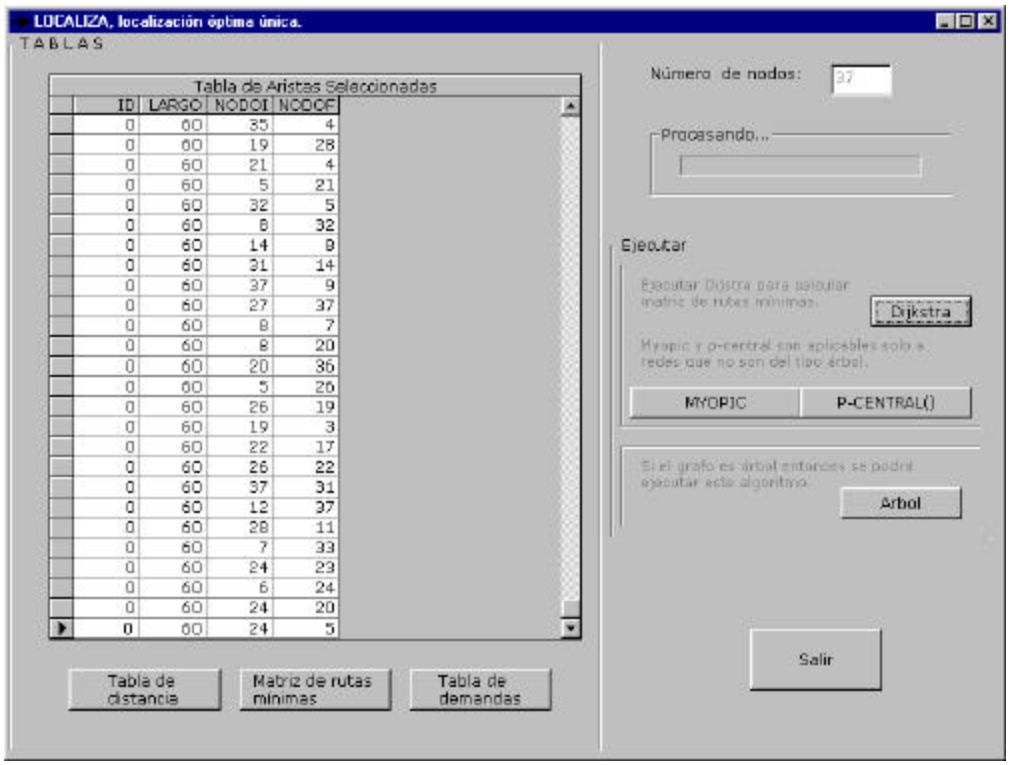

Figura 19. Interfaz de la aplicación.

Como se puede ver por un lado se ubican las tablas principales del sistema, como *Tabla de nodos seleccionados*, *Tabla de aristas seleccionadas* y *Matriz de rutas mínimas*; y por el otro lado los botones para su ejecución.

Cuando la red es un árbol solo el botón *Arbol* , visualizado en la figura 20, es habilitado ya que en este caso solo el algoritmo implementado tras ese botón sirve para resolver el problema. En el caso que la red no sea un árbol todos los botones son

habilitados con excepción del botón *Arbol*, estos botones pueden observarse en la figura 21.

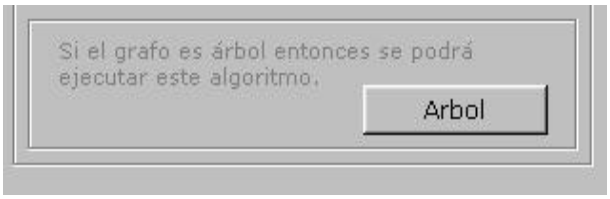

Figura 20. Botón "Arbol", de la interfaz de la aplicación.

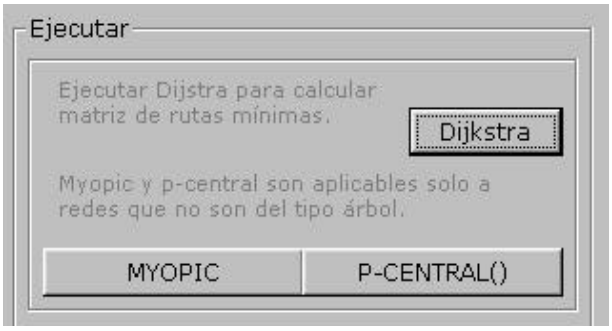

Figura 21. Botones "Dijkstra", "Myopic" y "P -central" de la interfaz de la aplicación.

Cuando la búsqueda termina se despliega en pantalla un mensaje que informa el nodo elegido como localización óptima. En la figura 22 se observa el mensaje desplegado en una búsqueda por medio del algoritmo Myopic.

Figura 22. Mensaje que entrega el resultado del algoritmo aplicado.

Se consideró necesario condicionar la ejecución de los procesos principales a diferentes botones para que el usuario tuviera el control en las etapas que componen la búsqueda.

Al terminar la ejecución del algoritmo de localización óptima se debe regresar al proyecto en Arcview para poder ver la nueva cobertura hecha con el nodo elegido.

Al regresar al proyecto en Arcview el usuario se encontrará con un mensaje, como el observado en la figura 23, al responder y presionar el botón *Yes* del mensaje se creará la nueva cobertura que contendrá el nodo elegido.

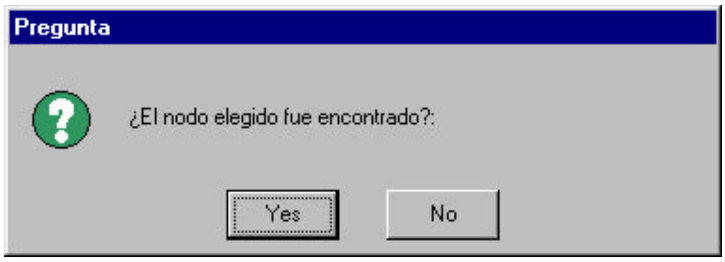

Figura 23. Mensaje desplegado en Arcview.

En la figura 24 se observa la nueva cobertura creada en el proyecto de Arcview y el nodo elegido en la red seleccionada.

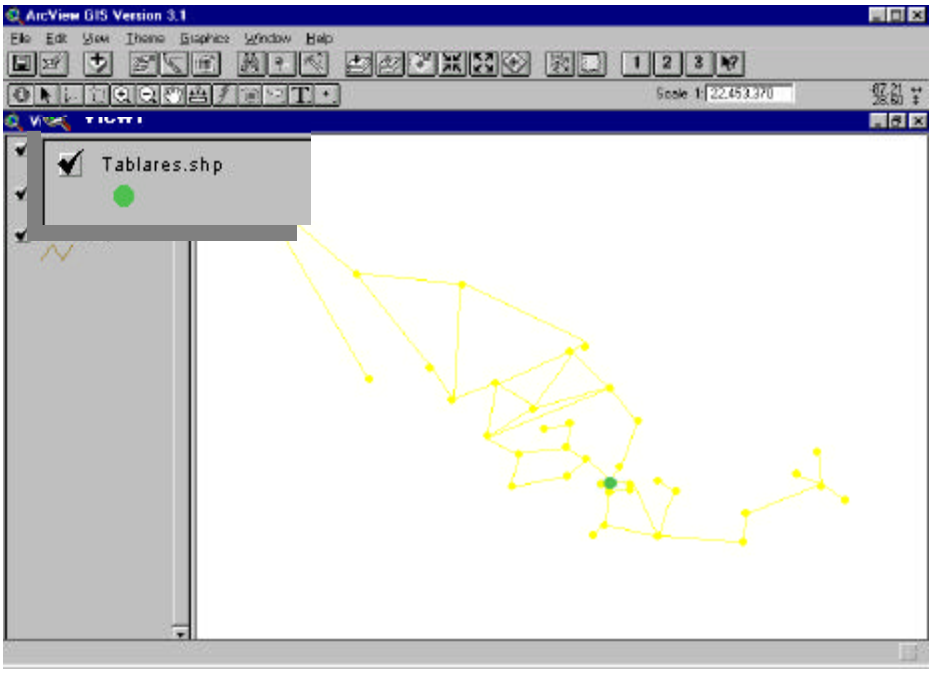

Figura 24. Nueva cobertura en Arcview.

7. Pruebas

La etapa de prueba consiste en probar diferentes ejemplos reales y observar como funciona el sistema.

Las pruebas se realizaron en las oficinas de la empresa Forestal Valdivia S.A., utilizandose datos reales de algunas zonas patrimoniales, empleandose recursos de la empresa que fueron necesarios para tales pruebas.

Se utilizó un computador personal de las siguientes características:

- Pentium 233 Mhz
- 64 M RAM
- 4.3 G de disco duro

Las pruebas tienen como objetivo:

Analizar los resultados obtenidos con quienes tienen como misión resolver este tipo de problemas dentro de la empresa.

Analizar los tiempos de ejecución de los diferentes algoritmos.

Evaluar los tiempos de ejecución de los principales procesos del sistema.

Al evaluar los tiempos de ejecución se pretende conocer la eficiencia de los algoritmos implementados, ésto se realizó observando los tiempos de ejecución y respaldando estos resultados mediante el análisis de los algoritmos. Para medir la eficiencia en tiempo de un algoritmo se utilizan funciones matemáticas sin unidades concretas, o sea, al decir que la complejidad de un algoritmo es T(n)= $cf(n)$ se hace referencia a que el tiempo de ejecución del algoritmo es proporcional a cierta función $f(n)$, donde la constante de proporcionalidad c depende del computador. Para

expresar la complejidad en tiempo se utilizará la notación O(). De manera formal, se dice que una función f(n) es de orden O(g(n)) si y solo si se cumple que existen unas constante s positivas *c* y n_0, ambas independientes de n, tales que:

$f(n) \leq cg(n), \forall\ n \geq n_0$

Decir que *f(n)* es *O(g(n))* supone que *cg(n)* es una cota superior del tiempo de ejecución del algoritmo.[Gal93]

En la tabla 1 se observan los tiempos de ejecución que presentó cada uno de los procesos más importantes.

	Tiempo(minutos) de ejecución para Procesos		
Nºde nodos	Dijkstra	Myopic	P-central
214	6	9	10
225	9	15	20
368	15	41	400

Tabla 1. Tiempo de ejecución para procesos principales.

Para el proceso que resuelve el problema en un árbol se realizaron ejemplos con menor número de nodos. Los tiempos de ejecución observados fueron bastante menores que para redes no árbol de similar número de nodos, en las cuales se aplicaron los otros algoritmos, pero los tiempos que se tarda en resolver un problema en una red del tipo árbol y otra que no lo es no son comparables ya que la estructura de la red es diferente y por lo tanto el problema es otro. Sólo como información

complementaria se detallan tiempos de ejecución para el algoritmo que resuelve el problema de localización en un árbol, ver tabla 2:

Nº de Nodos	Tiempos de ejecución (Expresados en segundos)
30	12
50	24
100	60

Tabla 2. Tiempo de ejecución para algoritmo 1-mediano en árboles.

En base a los resultados obtenidos se puede concluir que el sistema resuelve el problema de localización óptima de un recurso en una red, sea ésta de tipo árbol o no.

Sobre los tiempos de ejecución, todos los procesos importantes presentan un aumento considerable al aumentar el número de nodos.

En general, los procesos se comportan bastante bien cuando el número de nodos no sobrepasa los 150, pero al aumentar este número el tiempo se convierte en una característica a considerar. Como se puede observar en la tabla 1, bajo los 250 nodos los tiempos de ejecución de los diferentes algoritmos son considerables pero bastante aceptables, sobre este número como lo muestra el ejemplo con 368 nodos el tiempo de ejecución es muy elevado sobre todo en el algoritmo que resuelve problemas p-central, en este caso podría ser necesario considerar otras formas de resolver este problema o intentar su ejecución en una máquina más poderosa.

Refiriéndose en forma individual a cada algoritmo de localización óptima y al algoritmo de cálculo de rutas mínimas se puede concluir lo siguiente:

El algoritmo para calcular rutas mínimas consigue su objetivo en un tiempo que aumenta si el número de nodos que contiene la red aumenta. En este algoritmo se debe calcular la ruta mínima entre todos los pares de nodos, por lo tanto, mientras más aristas hayan más tiempo se necesita para encontrar la mejor ruta, este tiempo se incrementa ya que esta operación debe realizarse nxn veces. Por lo que el tiempo de ejecución aumenta regido por una función de orden $O(n^2)$, condicionada por el número de aristas que presenta la red.

Con respecto al algoritmo Myopic, éste presenta, al igual que el algoritmo de cálculo de rutas mínimas, tiempos de ejecución aceptables cuando el número de nodos no es muy elevado, a medida que se aumenta el número de nodos de la red el tiempo de ejecución aumenta ya que el número de operaciones de este algoritmo obedece a una función de orden $O(n^2)$, condicionada unicamente por el número de nodos. Como lo indica la literatura [Das95], el número de operaciones de este algoritmo es en realidad del orden $O(Pn^2)$, donde P es el número de recursos a localizar y se aproxima a $n/2$, por lo que para los casos donde $P > 1$, la función resultante es del orden $O(n^3)$. Se puede concluir que este algoritmo es bastante más eficiente cuando las redes no presentan gran número de nodos pero se vuelve muy ineficiente cuando este número aumenta y se quiere localizar mas de un recurso en la red.

El algoritmo p-central es el algoritmo que más aumenta su tiempo de ejecución a medida que aumenta el número de nodos en la red, debido a que itera n veces (suponiendo n número de nodos en la red) procesos que aumentan el número de operaciones proporcionalmente al aumento de nodos y procesos que obedecen

funciones cuadráticas y cúbicas en cuanto al número de operaciones, por lo que el exagerado aumento en el tiempo de ejecución se debe a que el número de operaciones se rige por una función de orden $O(n^4)$.

Con respecto al algoritmo que resuelve problemas 1-mediano en un árbol se puede concluir que presenta los menores tiempos de ejecución entre todos los algoritmos de localización probados, debido a lo sencillo, presentando un número de operaciones directamente proporcional al número de nodos, o sea el tiempo de ejecución del algoritmo se rige por una función de orden $O(n)$.

En resumen, de las pruebas realizadas se puede concluir que los resultados obtenidos satisfacen las expectativas de los usuarios, encontrándose incluso coincidencias con decisiones antiguas de localización tomadas por la empresa. Los tiempos de ejecución son aceptables considerando el problema que cada uno de ellos resuelve. Sobre esto último, no es de extrañar que un algoritmo que resuelve únicamente problemas 1-mediano ocupe poco tiempo en su ejecución, así como que los otros dos algoritmos sean más lentos debido a que no solo resuelven el problema de una localización si no también lo hacen para múltiples localizaciones, y hay que destacar que los problemas p-central son complejos y por lo tanto requieren de un algoritmo con un gran número de operaciones.

8. Conclusión

Por medio de la investigación y estudio de los problemas de localización óptima se logró conocer los esfuerzos y logros alcanzados durante años en esta materia, en donde se ha incursionado cada vez en métodos más complejos destinados a resolver problemas más complicados en una realidad cada vez más competitiva y compleja. Sobre este tema se concluye que a pesar de acumularse al día de hoy décadas de estudio, esta rama de la investigación operativa sigue creciendo y generando nuevas ramas que abarcan nuevos problemas a los que se intenta dar solución por medio de métodos y algoritmos cada vez más inteligentes.

Al investigar sobre las herramientas SIG utilizadas mundialmente, se pudo concluir que existe un considerable número de ellas dirigidas a diferentes grupos de usuarios, en general, todas estas herramientas no sólo permiten manejar un sistema de información geográfica, sino que también presentan otras funcionalidades que permiten al usuario obtener información, no sólo datos, de su sistema. En nuestro país el uso de esta herramienta es bastante limitado y recién a fines de la década pasada se comenzó a introducir en forma más fuerte en el área privada y pública, esta tardía incorporación de las herramientas SIG podría deberse, principalmente, a los costos iniciales en los que se debe incurrir al implantar un sistema de información geográfica, más aún si se considera que por lo general la infraestructura tecnológica de las empresas no es la más actual. Se determinó utilizar la herramienta SIG elegida por ser la más empleada en nuestro país y por la compatibilidad con otras herramientas de desarrollo.

El software desarrollado permite resolver el problema planteado mediante tres algoritmos diferentes, además de entregar información adicional. Para su desarrollado, en primer lugar, se consideraron los requisitos que el problema imponía, de acuerdo a eso se realizó el diseño desde el cual se pudiera iniciar la etapa de implementación en el lenguaje seleccionado, obteniendose el prototipo deseado, por lo que el objetivo general de esta tesis se considera logrado, ya que se implementó sobre una herramienta SIG un software que resuelva el problema de localización óptima. De tal forma, que un usuario convencional de Arcview puede acceder a esta aplicación siendo absolutamente transparente para él la complejidad de los algoritmos, esencialmente por lo amistosa que resulta la interfaz.

Para finalizar, en la investigación realizada para la elaboración de este trabajo se observó que los estudios de localización óptima van encaminados hacia buscar una solución integrando algoritmos de localización avanzados y sistemas de información geográfica, centrandose estos estudios en importantes instituciones educacionales del primer mundo.

9. Bibliografía

Libros:

[Ahu93] Ahuja R K, Magnanti T L, Orlin J B. (1993). *Network Flows, Theory, Algorithms and Applications*. Prentice Hall.

[Bur98] Burrough P y McDonnell R. (1998). *Principles of Geographical Information Systems*. Oxford University Press.

[Das95] Daskin, Mark S. 1995. *Network and Discrete Location, Model, algorithms and applications*. John Wiley & Sons, Inc.

[Env94] Environmental Systems Research Institute, Inc. 1994. *Understanding GIS: the ARC/INF Method.* Environmental Systems Research Institute, Inc.

[Env96] Environmental Systems Research Institute, Inc. 1996. *Using Avenue*. Environmental Systems Research Institute, Inc.

[Gal93] Galvez J, González J C, Sánchez A, Velázquez J A. 1993. *Algorítmica. Diseño y análisis de algoritmos funcionales e imperativos.* Addison-Wesley Iberoamericana, S.A. y RA-MA Editorial.

[Han79] Handler G, Mirchandani P, 1979. *Location on Network: Theory and Algorithms*. The MIT Press.

[Hut82] Hu, T.C., 1982. *Combinatorial Algorithms*. Addison-Wesley Publishing Company.

[Mag91] Maguire D J, Goodchild M, Rhind D. 1991. *Geographical Information Systems: Principles and Applications*. Longman Scientific &Technical.

[Pre96] Pressman, Roger S. 1996. *Ingeniería de Software. Un enfoque práctico*(3ª Edición). McGraw-Hill/ Interamericana de España, S.A.

[Sta90] Star J y Estes J. (1990). *Geographic Information Systems: An Introduction*. Prentice Hall, Inc.

Revistas

[Tan83] Tansel B C., Francis R L, Loew T J. Location on Networks: A survey. Part I: The p-center and p- median problems. *Management Science*, 29(4): 482-511. April 1983.

Páginas Web

[URL0] Cyberdido(2002). Programación en visual Basic. http://www.geocities.com/cyberdido/provb.htm

[URL1] Environmental Law Alliance Worldwide (2002).Ley forestal y de fauna silvestre. http://www.elaw.org/resources/text.asp?ID=556

[URL2] NCGIA, National Center Geographic Information and analysis(2002). What is GIS?. http://www.ncgia.ucsb.edu/

[URL 3] PCI Geomatic (2000). Using PAMAP GIS Technology. http://www.pcigeomatics.com/product_ind/prpamap.html

[URL4] Quim Bahí(2001). GIS. [http://www.audifilm.com/dosier/dosier3/GIS.htm]

[URL5] Ramírez, Mirta Liliana(2000). Evaluación y diagnóstico de la situación hospitalaria en la Provincia del Chaco (Argentina). Aplicación de Modelos de Localización-Asignación óptima mediante S.I.G., para posibles nuevos hospitales. http://mailweb.udla p.mx/~tesis/lis/ramirez_l_j/capitulo2.html

[URL6] SOPDE, Sociedad de Planificación y Desarrollo(2002). Tecnología de los sistemas de información geográfica. http://www.sopde.es/sopde/cursos/gis/GIS2.HTM

[URL7] Young-Hoon Kim(2002). SCGISA,University of Sheffield. http://www.shef.ac.uk/yhkim/

ANEXO 1. Código fuente de proyectos en Visual Basic.

LOCALIZA.VBP

FrmMain.frm

```
Option Explicit

Private Type NodePoint
   X As Long
   Y As Long
End Type
Dim nNodes As Long

Private Type TreeNode
    CurrNode As Long 'número del nodo donde se está actualmente
    NextNode(0 To 5) As Long 'puntero a un nodo vecino del nodo actual
    Dist(0 To 5) As Double 'distancia al nodo vecino
    VisitNumber As Long 'número de veces visitado
    Distance As Double 'distancia acumulada al nodo actual
    TmpVar As Double 'variable temporal para almecenar distancia
End Type

Dim TreeNodeList() As TreeNode

Dim CurrDestNode As Long
Dim CurrSrcNode As Long

Dim Es_Arbol As Boolean

Private Declare Function GetTickCount Lib "kernel32" () As Long
Private Declare Function QueryPerformanceCounter Lib "kernel32"
(lpPerformanceCount As Currency) As Long
Private Declare Function QueryPerformanceFrequency Lib "kernel32" (lpFrequency
As Currency) As Long
Dim MatRut() As Double


Private Function DijkstraPathFinding(NodeSrc As Long, NodeDest As Long) As
Boolean
'//0.
    Dim i As Long
```

```
    Dim bRunning As Boolean
    Dim CurrentVisitNumber As Long
    Dim CurrNode As Long
    Dim LowestNodeFound As Long
    Dim LowestValFound As Double

    If NodeSrc = NodeDest Then
       DijkstraPathFinding = True
       TreeNodeList(NodeDest).Distance = 0
       Exit Function
    End If

'//1. Inicialización de variables
    For i = 0 To nNodes - 1
       TreeNodeList(i).VisitNumber = -1 '-1 no ha sido visitado
       TreeNodeList(i).Distance = -1 'no se conoce la distancia
       TreeNodeList(i).TmpVar = 999999 ' guarda distancia
    Next i

    '//1º variables
    TreeNodeList(NodeSrc).VisitNumber = 1
       CurrentVisitNumber = 1
       CurrNode = NodeSrc
    TreeNodeList(NodeSrc).Distance = 0
    TreeNodeList(NodeSrc).TmpVar = 0

'//2. inicio de busqueda
    Do While bRunning = False
       '//2a. Ir a cada nodo vecino
             'variable temporal= distancia al recurso + peso del arco
            If Not (TreeNodeList(CurrNode).NextNode(0) = -1) Then
TreeNodeList(TreeNodeList(CurrNode).NextNode(0)).TmpVar =
MIN(TreeNodeL ist(CurrNode).Dist(0) + TreeNodeList(CurrNode).Distance,
TreeNodeList(TreeNodeList(CurrNode).NextNode(0)).TmpVar)
            If Not (TreeNodeList(CurrNode).NextNode(1) = -1) Then
TreeNodeList(TreeNodeList(CurrNode).NextNode(1)).TmpVar =
MIN(TreeNodeList(CurrNode).Dist(1) + TreeNodeList(CurrNode).Distance,
TreeNodeList(TreeNodeList(CurrNode).NextNode(1)).TmpVar)
            If Not (TreeNodeList(CurrNode).NextNode(2) = -1) Then
TreeNodeList(TreeNodeList(CurrNode).NextNode(2)).TmpVar =
MIN(TreeNodeList(CurrNode).Dist(2) + TreeNodeList(CurrNode).Distance,
TreeNodeList(TreeNodeList(CurrNode).NextNode(2)).TmpVar)
            If Not (TreeNodeList(CurrNode).NextNode(3) = -1) Then
TreeNodeList(TreeNodeList(CurrNode).NextNode(3)).TmpVar =
MIN(TreeNodeList(CurrNode).Dist(3) + TreeNodeList(CurrNode).Distance,
TreeNodeList(TreeNodeList(CurrNode).NextNode(3)).TmpVar)
```

```
            If Not (TreeNodeList(CurrNode).NextNode(4) = -1) Then
TreeNodeList(TreeNodeList(CurrNode).NextNode(4)).TmpVar =
MIN(TreeNodeList(CurrNode).Dist(4) + TreeNodeList(CurrNode).Distance,
TreeNodeList(TreeNodeList(CurrNode).NextNode(4)).TmpVar)
            If Not (TreeNodeList(CurrNode).NextNode(5) = -1) Then
TreeNodeList(TreeNodeList(CurrNode).NextNode(5)).TmpVar =
MIN(TreeNodeList(CurrNode).Dist(5) + TreeNodeList(CurrNode).Distance,
TreeNodeList(TreeNodeList(CurrNode).NextNode(5)).TmpVar)

        '//2b. Decide que variable temporal es mas baja
            LowestValFound = 999999
            For i = 0 To nNodes - 1 ' dar una vuelta si son muchos nodos.
              If (TreeNodeList(i).TmpVar <= LowestValFound) And
(TreeNodeList(i).TmpVar >= 0) And (TreeNodeList(i).VisitNumber < 0) Then
'se tiene un nuevo valor bajo.
                LowestValFound = TreeNodeList(i).TmpVar
                LowestNodeFound = i
              End If
            Next i
            '**NB: Si hay multiples valores mas bajos => se elegirá el ultimo

        '//2c. incrementa el nº de visitas de este nodo y asigna tmpvar -> distance
            CurrentVisitNumber = CurrentVisitNumber + 1
            TreeNodeList(LowestNodeFound).VisitNumber = CurrentVisitNumber
            TreeNodeList(LowestNodeFound).Distance =
TreeNodeList(LowestNodeFound).TmpVar
            CurrNode = LowestNodeFound '//copia la variable para la proxima

        '//2d. Si CurrNode no es el nodo destino => sigue iterando
            If CurrNode = NodeDest Then
              bRunning = True
            Else
              bRunning = False
            End If
    Loop
DijkstraPathFinding = True
End Function

Public Function MIN(A As Double, B As Double) As Double
    'retirna el minimos entre dos numeros
    If A < B Then MIN = A
    If A > B Then MIN = B
    If A = B Then MIN = A
End Function
```

```
Private Function GetDist2D(X As Long, Y As Long, X1 As Long, Y1 As Long) As
Long
    GetDist2D = Sqr(((X - X1) ^ 2) + ((Y - Y1) ^ 2))
End Function
Private Sub Dijkstra()
Dim Result As Boolean
Dim nodoi As Long
Dim nodof   As Long
Dim i As Long
Dim j As Long
Dim barra As Double

 Datarut.Refresh
 'limpia datarut antes de actualizarla

 While Not Datarut.Recordset.EOF
   Datarut.Recordset.Delete
   Datarut.Recordset.MoveFirst
 Wend

 Datarut.Refresh
 For i = 1 To nNodes
    frmMain.ProgressBar1.Value = i * 80 / nNodes
   Datarut.Recordset.AddNew
   Datarut.UpdateRecord
   Datarut.Recordset.MoveLast
   Datarut.Recordset.Edit
   Datarut.Recordset.Fields(0).Value = i
   For j = 1 To i
    If i = j Then
     Datarut.Recordset.Fields(j).Value = 0
     MatRut(i, j) = 0
    Else
     Result = DijkstraPathFinding(i, j)
     Datarut.Recordset.Fields(j).Value = TreeNodeList(j).Distance
     MatRut(i, j) = TreeNodeList(j).Distance
    End If
   Next j
    Datarut.UpdateRecord
 Next i

 Datarut.Recordset.MoveFirst
 barra = frmMain.ProgressBar1.Value
 For i = 1 To nNodes
   frmMain.ProgressBar1.Value = barra + (i * 20 / nNodes)
   Datarut.Recordset.Edit
```

```
   For j = i To nNodes
    Datarut.Recordset.Fields(j).Value = MatRut(j, i)
   Next j
   Datarut.UpdateRecord
   Datarut.Recordset.MoveNext
 Next i


End Sub

Private Sub Command1_Click()
 Dijkstra
End Sub

Private Sub Command2_Click()
 frmmyopic.Show

 Unload frmmyopic
End Sub

Private Sub Command3_Click()
 FrmCentral.Show

 Unload FrmCentral
End Sub

Private Sub Command4_Click()
FrmTree.Show

Unload FrmTree

End Sub
Private Sub vecino()
Dim i As Long

'***** Se asigna a cada nodo sus nodos vecinos *****
Datasel.Refresh
For i = 1 To nNodes  'nºnodos
' Busca el registro donde nodoi=I
  Datasel.Recordset.MoveFirst
  While Not (Datasel.Recordset.EOF)
    While Not (Datasel.Recordset.Fields("nodoi").Value = i)
       Datasel.Recordset.MoveNext
       If Datasel.Recordset.EOF Then GoTo F_3
    Wend
```

```
   If TreeNodeList(i).NextNode(0) = -1 Then TreeNodeList(i).NextNode(0) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(1) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(1) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(2) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(2) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(3) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(2) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(3) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(4) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(2) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(3) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(4) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(5) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(2) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(3) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(4) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(5) =
Datasel.Recordset.Fields("nodof").Value
   If Not (Datasel.Recordset.EOF) Then Datasel.Recordset.MoveNext
  Wend
F_3:
Next i
End Sub

Private Sub Command5_Click()
End

End Sub

Private Function Arbol() As Boolean
 Dim v, W, P, nodo As Integer
 Dim nAristas, cuenta, i, j, hasta As Integer
 Dim response As String
 Dim Existe As Boolean
 Dim X As Long
 Dim valor As Long
```

```
' numero de aristas
cuenta = 0
frmMain.Datasel.Refresh
frmMain.Datasel.Recordset.MoveFirst
While Not frmMain.Datasel.Recordset.EOF
 cuenta = cuenta + 1
 frmMain.Datasel.Recordset.MoveNext
Wend
frmMain.Datasel.Recordset.MoveFirst
nAristas = cuenta / 2

frmMain.Datadda.Recordset.MoveFirst

Vs.Rows = nNodes + 1
B2.Rows = nNodes + 1
A2.Rows = nNodes + 1
For i = 1 To nNodes
 Vs.Row = i
 Vs.Text = frmMain.Datadda.Recordset.Fields("nodo").Value
 B2.Row = i
 B2.Text = frmMain.Datadda.Recordset.Fields("nodo").Value
 A2.Row = i
 A2.Text = frmMain.Datadda.Recordset.Fields("nodo").Value
 frmMain.Datadda.Recordset.MoveNext
Next i

P = 0
B2.Row = 1
While B1_dif_Vs
 X = B2.Text
 A1.Rows = A1.Rows + 1
 A1.Row = A1.Rows - 1
 A1.Text = X
 For i = 1 To A2.Rows - 1
  A2.Row = i
  If A2.Text = X Then ' entonces se elimina de la lista A2
   While A2.Row <> A2.Rows - 1
    A2.Row = A2.Row + 1
    valor = A2.Text
    A2.Row = A2.Row - 1
    A2.Text = valor
    A2.Row = A2.Row + 1
   Wend
   A2.Rows = A2.Rows - 1
   Exit For
  End If
```

```
 Next

 While Cociclo
  For v = 1 To A1.Rows - 1
   A1.Row = v
   If A1.Text <> "" Then
    nodo = A1.Text
    frmMain.Datasel.Recordset.MoveFirst
F_3:   While (Not frmMain.Datasel.Recordset.EOF) And (Not
(frmMain.Datasel.Recordset.Fields("nodoi").Value = Val(nodo)))
     frmMain.Datasel.Recordset.MoveNext
     If frmMain.Datasel.Recordset.EOF Then GoTo F_1
    Wend
    For W = 1 To A2.Rows - 1
     A2.Row = W
     If frmMain.Datasel.Recordset.Fields("nodof").Value = A2.Text Then
      A1.Rows = A1.Rows + 1
      A1.Row = A1.Rows - 1
      A1.Text = A2.Text
      While A2.Row <> A2.Rows - 1
       A2.Row = A2.Row + 1
       valor = A2.Text
       A2.Row = A2.Row - 1
       A2.Text = valor
       A2.Row = A2.Row + 1
      Wend
      A2.Rows = A2.Rows - 1
      Exit For
     End If
    Next
    frmMain.Datasel.Recordset.MoveNext
    If (Not frmMain.Datasel.Recordset.EOF) Then GoTo F_3 Else GoTo F_1
   End If
F_1: Next
 Wend
 'B1 = B1 + A1
  For i = 1 To A1.Rows - 1
   Existe = False
   A1.Row = i
   For j = 1 To B1.Rows - 1
    B1.Row = j
    If B1.Text = A1.Text Then Existe = True
   Next
   If Not Existe Then
    B1.Rows = B1.Rows + 1 ' agrega una fila mas.
    B1.Row = B1.Rows - 1  ' posiciona en la ultima fila
```

```
      B1.Text = A1.Text
    End If
   Next
  'B2 = B2 - B1 eliminar elementos de b2 que estan en b1
   For i = 1 To B1.Rows - 1
    B1.Row = i
    For j = 1 To B2.Rows - 1
     B2.Row = j
     If B1.Text = B2.Text Then
       While B2.Row <> B2.Rows - 1
        B2.Row = B2.Row + 1
        valor = B2.Text
        B2.Row = B2.Row - 1
        B2.Text = valor
        B2.Row = B2.Row + 1
       Wend
       B2.Rows = B2.Rows - 1
       Exit For
     End If
    Next
   Next
   P = P + 1
 Wend
 If nAristas = nNodes - 1 And P = 1 Then
   Arbol = True
  Else
   Arbol = False
 End If
End Function
Function B1_dif_Vs() As Boolean
Dim i, j, conta As Integer
Dim igual As Boolean

 igual = False
 conta = 0
 If B1.Rows = Vs.Rows Then
  For i = 1 To B1.Rows - 1
   B1.Row = i
   For j = 1 To Vs.Rows - 1
    Vs.Row = j
    If B1.Text = Vs.Text Then igual = True: conta = conta + 1
   Next
  Next
 End If
 If igual And (conta = (Vs.Rows - 1)) Then B1_dif_Vs = False Else B1_dif_Vs =
True
```

```
End Function
Function Cociclo() As Boolean
Dim nodo, i, j As Integer

    frmMain.Datasel.Refresh
    frmMain.Datasel.Recordset.MoveFirst

    Cociclo = False
    For i = 1 To A1.Rows - 1
     A1.Row = i
     nodo = Val(A1.Text)
     frmMain.Datasel.Recordset.MoveFirst
F_3:  While (Not frmMain.Datasel.Recordset.EOF) And (Not
(frmMain.Datasel.Recordset.Fields("nodoi").Value = nodo))
      frmMain.Datasel.Recordset.MoveNext
      If frmMain.Datasel.Recordset.EOF Then GoTo F_1
     Wend
     For j = 1 To A2.Rows - 1
      A2.Row = j
      If frmMain.Datasel.Recordset.Fields("nodof").Value = A2.Text Then Cociclo =
True: GoTo F_2
     Next
     frmMain.Datasel.Recordset.MoveNext
     If (Not frmMain.Datasel.Recordset.EOF) Then GoTo F_3 Else GoTo F_1
F_1: Next
F_2:
End Function


Private Sub Command6_Click()
DBGrid1.Visible = True
DBGrid2.Visible = False
DBGrid3.Visible = False

End Sub

Private Sub Command7_Click()
DBGrid2.Visible = True
DBGrid1.Visible = False
DBGrid3.Visible = False
End Sub

Private Sub Command8_Click()
DBGrid3.Visible = True
DBGrid1.Visible = False
DBGrid2.Visible = False
```

```
End Sub

Private Sub Form_Load()
Dim i, cuenta As Long

'Determinar número de nodos
nNodes = 0
Datadda.Refresh
Datadda.Recordset.MoveFirst
While Not Datadda.Recordset.EOF
  nNodes = nNodes + 1
  Datadda.Recordset.MoveNext
Wend
Text1.Text = nNodes

ReDim TreeNodeList(nNodes) As TreeNode
ReDim MatRut(1 To nNodes, 1 To nNodes) As Double


Es_Arbol = Arbol

If Not Es_Arbol Then
   Command4.Enabled = False
  Else
   Command3.Enabled = False
End If


'***** Se asigna el nº de nodo a CurrNode *****
For i = 1 To nNodes
  TreeNodeList(i).CurrNode = i
  TreeNodeList(i).NextNode(0) = -1
  TreeNodeList(i).NextNode(1) = -1
  TreeNodeList(i).NextNode(2) = -1
  TreeNodeList(i).NextNode(3) = -1
  TreeNodeList(i).NextNode(4) = -1
  TreeNodeList(i).NextNode(5) = -1
Next i


'***** Se asigna a cada nodo sus nodos vecinos *****
Datasel.Refresh
For i = 1 To nNodes  'nºnodos
' Busca el registro donde nodoi=I
  Datasel.Recordset.MoveFirst
  While Not (Datasel.Recordset.EOF)
```

```
   While (Not Datasel.Recordset.EOF) And (Not
(Datasel.Recordset.Fields("nodoi").Value = i))
     Datasel.Recordset.MoveNext
     If Datasel.Recordset.EOF Then GoTo F_1
   Wend
   If TreeNodeList(i).NextNode(0) = -1 Then TreeNodeList(i).NextNode(0) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(1) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(1) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(2) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(2) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(3) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(2)  <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(3) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(4) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(2) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(3) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(4) =
Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(5) = -1 And TreeNodeList(i).NextNode(0) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(1) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(2) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(3) <>
Datasel.Recordset.Fields("nodof").Value And TreeNodeList(i).NextNode(4) <>
Datasel.Recordset.Fields("nodof").Value Then TreeNodeList(i).NextNode(5) =
Datasel.Recordset.Fields("nodof").Value
   If Not Datasel.Recordset.EOF Then Datasel.Recordset.MoveNext
  Wend
F_1:
Next i


'**** Se asigna la distncia entre cada nodo y todos sus vecinos ****
Datasel.Refresh
For i = 1 To nNodes  'nºnodos
' Busca el registro donde nodoi=I
  Datasel.Recordset.MoveFirst
  While Not (Datasel.Recordset.EOF)
    While (Not Datasel.Recordset.EOF) And (Not
(Datasel.Recordset.Fields("nodoi").Value = i))
```

```
    Datasel.Recordset.MoveNext
    If Datasel.Recordset.EOF Then GoTo F_2
   Wend
  If Not (TreeNodeList(i).NextNode(0) = -1) And
Datasel.Recordset.Fields("nodof").Value = TreeNodeList(i).NextNode(0) Then
TreeNodeList(i).Dist(0) = Datasel.Recordset.Fields("largo").Value
  If Not (TreeNodeList(i).NextNode(1) = -1) And
Datasel.Recordset.Fields("nodof").Value = TreeNodeList(i).NextNode(1) Then
TreeNodeList(i).Dist(1) = Datasel.Recordset.Fields("largo").Value
  If Not (TreeNodeList(i).NextNode(2) = -1) And
Datasel.Recordset.Fields("nodof").Value = TreeNodeList(i).NextNode(2) Then
TreeNodeList(i).Dist(2) = Datasel.Recordset.Fields("largo").Value
  If Not (TreeNodeList(i).NextNode(3) = -1) And
Datasel.Recordset.Fields("nodof").Value = TreeNodeList(i).NextNode(3) Then
TreeNodeList(i).Dist(3) = Datasel.Recordset.Fields("largo").Value
  If Not (TreeNodeList(i).NextNode(4) = -1) And
Datasel.Recordset.Fields("nodof").Value = TreeNodeList(i).NextNode(4) Then
TreeNodeList(i).Dist(4) = Datasel.Recordset.Fields("largo").Value
  If Not (TreeNodeList(i).NextNode(5) = -1) And
Datasel.Recordset.Fields("nodof").Value = TreeNodeList(i).NextNode(5) Then
TreeNodeList(i).Dist(5) = Datasel.Recordset.Fields("largo").Value
  If Not Datasel.Recordset.EOF Then Datasel.Recordset.MoveNext
 Wend
F_2:
Next i

End Sub
```

FrmMyopic.frm

```
Dim nNodes As Long
Dim MatDd() As Double
Private Type Campos
 nodoO As Integer
 nodoN As Integer
End Type
Dim tablaD() As Campos

Private Sub Myopic()
Dim i, A, N, P As Long
Dim SumDD() As Double
Dim K As Long
Dim barra As Double
Dim response As String
```

```
ReDim SumDD(1 To nNodes) As Double

 frmMain.Datadda.Refresh
 frmMain.Datadda.Recordset.MoveFirst
 For i = 1 To nNodes
   tablaD(i).nodoO = frmMain.Datadda.Recordset.Fields(0).Value
   tablaD(i).nodoN = frmMain.Datadda.Recordset.Fields(1).Value
   frmMain.Datadda.Recordset.MoveNext
 Next i

  'limpia la tabla de salida
  While Not frmMain.Dataout.Recordset.EOF
   frmMain.Dataout.Recordset.Delete
   frmMain.Dataout.Recordset.MoveFirst
  Wend

  'limpia la tabla de salida
  While Not frmMain.Datadet.Recordset.EOF
   frmMain.Datadet.Recordset.Delete
   frmMain.Datadet.Recordset.MoveFirst
  Wend

barra = frmMain.ProgressBar1.Value

'***** Paso 1 *****
K = 0
P = 1
List1.CLEAR

While K < P ' nº de iteraciones < nº de plantas a instalar.
'***** Paso 2 *****
 K = K + 1
 N = 0
'***** Paso 3 *****
Datasel.Refresh
 For i = 1 To nNodes
    frmMain.ProgressBar1.Value = barra + (i * 45) / (nNodes * P)
    SumDD(i) = 0
    Datadd.Recordset.MoveFirst
     For A = 1 To nNodes
      SumDD(i) = SumDD(i) + Datadd.Recordset.Fields(i).Value
      Datadd.Recordset.MoveNext
    Next A
 Next i

'***** Paso 4 *****
```

```
N = Nodo_Elegido(SumDD)

'la tabla debe estar vacía
 frmMain.Dataout.Refresh
 frmMain.Dataout.Recordset.AddNew
 frmMain.Dataout.UpdateRecord
 frmMain.Dataout.Recordset.MoveLast
 frmMain.Dataout.Recordset.Edit
 frmMain.Dataout.Recordset.Fields(0).Value = N
 frmMain.Dataout.Recordset.Fields(1).Value = tablaD(N).nodoO
 frmMain.Dataout.Recordset.Fields(2).Value = "Con Myopic, localización nº " +
CStr(K)
 frmMain.Dataout.UpdateRecord
 frmMain.Datarut.Recordset.MoveFirst
 For i = 1 To nNodes
  frmMain.Datadet.Refresh
  frmMain.Datadet.Recordset.AddNew
  frmMain.Datadet.UpdateRecord
  frmMain.Datadet.Recordset.MoveLast
  frmMain.Datadet.Recordset.Edit
  frmMain.Datadet.Recordset.Fields(0).Value = i
  frmMain.Datadet.Recordset.Fields(1).Value = tablaD(i).nodoO
  frmMain.Datadet.Recordset.Fields(2).Value =
frmMain.Datarut.Recordset.Fields(N).Value
  frmMain.Datadet.UpdateRecord
  frmMain.Datarut.Recordset.MoveNext
 Next i

 frmMain.Dataout.Recordset.MoveLast

 List1.AddItem N

'***** Paso 5 *****
 If K <> P Then
   Datadd.Recordset.MoveFirst
   While Not Datadd.Recordset.EOF
     Datadd.Recordset.Edit
     For i = 1 To nNodes
       If Val(Datadd.Recordset.Fields(i).Value) >
Val(Datadd.Recordset.Fields(N).Value) Then
        Datadd.Recordset.Fields(i).Value = Datadd.Recordset.Fields(N).Value
       End If
     Next
     Datasel.Recordset.Update
     Datasel.Recordset.MoveNext
   Wend
```

```
 End If

 Wend 'While k<num_fac

frmMain.ProgressBar1.Value = 100

response = MsgBox("El nodo elegido es: " & N, 0, "MYOPIC")

End Sub

Private Sub Form_Load()
  nNodes = Val(frmMain.Text1.Text)
  ReDim MatDd(1 To nNodes, 1 To nNodes) As Double
  ReDim tablaD(1 To nNodes) As Campos

  Construye_Matriz_distdda
  Myopic
End Sub

Private Sub Construye_Matriz_distdda()
Dim dda As Double
Dim i, A As Long

Datadd.Refresh
frmMain.Datarut.Recordset.MoveFirst

'limpia datadd
While Not Datadd.Recordset.EOF
   Datadd.Recordset.Delete
   Datadd.Recordset.MoveFirst
Wend

For i = 1 To nNodes
    frmMain.ProgressBar1.Value = i * 50 / nNodes
   Datadd.Recordset.AddNew
   Datadd.UpdateRecord
   Datadd.Recordset.MoveLast
   frmMain.Datadda.Refresh
   While Not (frmMain.Datadda.Recordset.Fields("nodo") = i)
     frmMain.Datadda.Recordset.MoveNext
   Wend
   dda = frmMain.Datadda.Recordset.Fields("dda").Value
   ' Si el grafo es arbol entonces las demandas cero se hace igual a 1
   'If Es_Arbol And dda = 0 Then dda = 1
   Datadd.Recordset.Edit
    Datadd.Recordset.Fields(0).Value = i
```

```
  Datadd.Recordset.Fields(i).Value = 0
  For A = 1 To nNodes
    Datadd.Recordset.Fields(A).Value = frmMain.Datarut.Recordset.Fields(A).Value
* dda
  Next A
  frmMain.Datarut.Recordset.MoveNext
  Datadd.UpdateRecord
  Datadd.Recordset.MoveNext
 Next i
  Datasel.UpdateRecord

End Sub
Private Function Es_Planta(X As Long) As Boolean
Dim i As Long

 Es_Planta = False
 For i = 0 To List1.ListCount - 1
  If List1.List(i) = X Then
    Es_Planta = True
  End If
 Next
End Function
Private Function Nodo_Elegido(lista() As Double) As Long
Dim X As Long
Dim menor As Double

  menor = 9999999999#
  For X = 1 To nNodes
   If Not Es_Planta(X) Then
     If menor > lista(X) Then
       menor = lista(X)
       Nodo_Elegido = X
     End If
   End If
  Next X
End Function
```

Central.frm

```
Option Explicit
Dim Y(), CENTER(), CCT(), CLUST() As Long
Dim TRVT(), RD(), D(), W() As Long
Dim nNodes, NCAN, NALC, WSUM, IERROR, NS As Long
Dim CT, IMINCL, JMINCL, N_co As Long
Dim MD As Double
```

```
Private Type Campos
   nodoO As Integer
   nodoN As Integer
   oferta As Integer
End Type
Dim tablaD() As Campos

Private Sub Form_Load()
Dim j As Long

N_co = 0
nNodes = Val(frmMain.Text1.Text)

ReDim Y(1 To nNodes), CENTER(1 To nNodes), CCT(1 To nNodes), CLUST(1 To
nNodes) As Long
ReDim TRVT(1 To nNodes, 1 To nNodes), RD(1 To nNodes, 1 To nNodes), D(1 To
nNodes, 1 To nNodes), W(1 To nNodes) As Long
ReDim tablaD(1 To nNodes) As Campos

FILE_MODE  ' Obtiene datos de tablas, y calcula las ppals variables a utilizar en los
procesos
INITAZ ' Inicializa la variable CLUST
If NALC = NS Then OUTPUT
While NS > NALC Or N_co <> NALC
  frmMain.ProgressBar1.Value = (nNodes - NS + 1) * 75 / nNodes
  MINPAIR ' busca el par de nodos en el que se hace minima D()
  CLUSTER ' Se obtiene MD(menor  D() de los mayores D(i))
  CENDOWN (MD) 'Asigna a CLUST nodos.
Wend
OUTPUT ' Proceso que entrega resultados

End Sub

Private Sub FILE_MODE()
Dim Sw As String
Dim i, j As Long
Dim X As Byte
Dim SSUM As Double


NCAN = nNodes
NALC = 1

frmMain.Datarut.Refresh
frmMain.Datarut.Recordset.MoveFirst
For i = 1 To nNodes
```

```
   For j = 1 To NCAN
    TRVT(i, j) = Val(frmMain.Datarut.Recordset.Fields(j).Value)
   Next j
   frmMain.Datarut.Recordset.MoveNext
Next i
WSUM = 0

frmMain.Datadda.Refresh
frmMain.Datadda.Recordset.MoveFirst
For i = 1 To nNodes
  W(i) = frmMain.Datadda.Recordset.Fields("dda").Value
  frmMain.Datadda.Recordset.MoveNext
Next i

For i = 1 To nNodes
   CCT(i) = 0
  WSUM = WSUM + W(i)
Next i

  For i = 1 To nNodes
     SSUM = 0
     For j = 1 To NCAN
        D(i, j) = TRVT(i, j) * W(i) / WSUM
        SSUM = SSUM + D(i, j) 'suma de la dist/dda ponderada
     Next j
 Next i
 For j = 1 To NCAN
    CENTER(j) = j
    CCT(CENTER(j)) = j
 Next j
End Sub
Private Sub INITAZ()
Dim i, j As Long
Dim MIND As Double

NS = NCAN
For i = 1 To nNodes
    MIND = 999999
    For j = 1 To NCAN
       Y(j) = 1
       'Se le asigna a MIND la dist/dda ponderada menor para c/i
       'A CLUST se le asigna el j donde D() es menor para cada i.
       If D(i, j) <= MIND Then
         MIND = D(i, j)
         CLUST(i) = j
       End If
```

```
    Next j
Next i
End Sub
Private Sub MINPAIR()
Dim i, j As Long
Dim MIND As Double

'Se busca el menor valor de D() para todos lo i y j, y se asigna
' D()->MIND, i->IMINCL y j->JMINCL
 MIND = 999999
 Print #2,
 For i = 1 To nNodes
   If CCT(i) <> 0 And Y(CCT(i)) <> 0 Then
     For j = 1 To NCAN
        If Y(j) <> 0 And i <> CENTER(j) And D(i, j) < MIND Then
         MIND = D(i, j)
         IMINCL = CCT(i)
         JMINCL = j
        End If
     Next j
   End If
 Next i
End Sub

Private Sub CLUSTER()
Dim i, j, JB As Long
Dim FIRMAX As Double

For i = 1 To nNodes
   For j = 1 To nNodes
      RD(i, j) = 999999
     'Si clust(i) es parte del par mínimo y clust(j) igual y j no es clust =>
     ' RD(i,j) almacena la distancia entre i y j
     If CLUST(i) = IMINCL And CLUST(j) = IMINCL And CCT(j) <> 0 Then RD(i,
CCT(j)) = D(i, CCT(j))
     If CLUST(i) = JMINCL And CLUST(j) = JMINCL And CCT(j) <> 0 Then
RD(i, CCT(j)) = D(i, CCT(j))
     If CLUST(i) = IMINCL And CLUST(j) = JMINCL And CCT(j) <> 0 Then RD(i,
CCT(j)) = D(i, CCT(j))
     If CLUST(i) = JMINCL And CLUST(j) = IMINCL And CCT(j) <> 0 Then RD(i,
CCT(j)) = D(i, CCT(j))
     'RD(i,j) tiene valor solo si i,j pertenecen
      'a los cluster de IMINCL o JMINCL

   Next j
Next i
```

```
JB = 1
CT = 1

'1º se busca la arista mayor para cada nodo, siempre y cuando
'   ambos nodos pertenescan al cluster de IMINCL oJMINCL
MD = -999999
For i = 1 To nNodes
    If RD(i, JB) <> 999999 And RD(i, JB) > MD Then MD = RD(i, JB)
Next i
If MD = -999999 Then MD = 999999
JB = JB + 1
While Not (JB > NCAN)
  FIRMAX = -999999
  For i = 1 To nNodes
    If RD(i, JB) <> 999999 And RD(i, JB) > FIRMAX Then
       FIRMAX = RD(i, JB)
    End If
    'Firmax contiene el RD() mayor para c/ j
  Next i
  ' 2º De aquellas aristas mayores se busca la menor, y
  '    se almacena el nº del nodo(CT) que se convertirá
  '    (en cendown) en la nueva cabeza de cluster.
  If FIRMAX = -999999 Then FIRMAX = 999999
  If FIRMAX < MD Then MD = FIRMAX: CT = JB
  JB = JB + 1
Wend
'Se obtiene MD, que es el RD() menor de todos Firmax; y el nodo j donde se
encuantra el RD menor
End Sub

Private Sub CENDOWN(DIGGLE As Double)
Dim i, j, ISOURCE, JSOURCE, RCT As Long
Dim MNFIR, K, JB As Long
Dim DWIB, MIND, FIRMAX As Double

 DWIB = DIGGLE * 2
 RCT = CT
 MIND = DIGGLE ' MD

'Se asigna el cluster
 For i = 1 To nNodes
    If CLUST(i) = IMINCL Or CLUST(i) = JMINCL Then
     CLUST(i) = RCT
    End If
 Next i
```

```
 Y(IMINCL) = 0 ' IMINCL ya no es cabeza de cluster
 Y(JMINCL) = 0 ' JMINCL ya no es cabeza de cluster
 Y(RCT) = 1  ' RCT ahora es cabeza de cluster
 NS = NS - 1
 ' ***** Se generó 1 nueva cabeza y se eliminaron 2, entonces ahora
 ' ***** se deben reasignar los nodos a los cluster.
'A CLUST(i) se le asigna j si:
'j es cabeza de cluster y la distancia entre i,j es la menor
 For i = 1 To nNodes
   MNFIR = 999999
   For j = 1 To NCAN
     If Y(j) <> 0 And D(i, j) <= MNFIR Then
      MNFIR = D(i, j)
      CLUST(i) = j
     End If
    Next j
 Next i

 '**** Para cada cluster se busca el nodo que esté mas lejos
 '**** y ese es la nueva cabeza de ese cluster.
 For j = 1 To NCAN
   If Y(j) <> 0 Then ' Si j es cabeza de cluster
     'Se forma la matriz RD(), esta será
     ' = 999999 si i,k no pertenecen al mismo cluster
     ' = D(i,k), si i,k pertenecen al mismo cluster j
     For i = 1 To nNodes
       For K = 1 To NCAN
         RD(i, K) = 999999
         If (CLUST(i) = j And CLUST(K) = j) Then RD(i, K) = D(i, K)
       Next K
     Next i
     JB = 1: CT = 1: MD = -999999
     For i = 1 To nNodes
        If RD(i, JB) <> 999999 And RD(i, JB) > MD Then MD = RD(i, JB)
        ' ahora MD contiene la mayor distancia entre dos nodos
        ' que pertenecen al mismo cluster, para JB=1
     Next i
     If MD = -999999 Then MD = 999999
     JB = JB + 1
     While Not JB > NCAN
       FIRMAX = -999999
       For i = 1 To nNodes
         If RD(i, JB) <> 999999 And RD(i, JB) > FIRMAX Then FIRMAX = RD(i,
JB)
         'MD contiene la mayor distancia entre dos nodos
         'que pertenecen al mismo cluster
```

```
      Next i
      If FIRMAX = -999999 Then FIRMAX = 999999
      If FIRMAX < MD Then MD = FIRMAX: CT = JB
      ' MD el menor firmax(=mayores distancias entre nodos del mismo cluster).
      JB = JB + 1
    Wend
    Y(j) = 0: Y(CT) = 1
  End If
 Next j

' **** Como las antiguas cabezas fueron reemplazadas por otras
' **** se debe hacer una reasignacion de nodos a los cluster
 For i = 1 To nNodes
   MNFIR = 999999
   For j = 1 To NCAN
      If Y(j) <> 0 And D(i, j) <= MNFIR Then
        MNFIR = D(i, j)
        CLUST(i) = j
      End If
   Next j

  'determinar número de coberturas
  N_co = 0
  For j = 1 To NCAN
    If Y(j) = 1 Then N_co = N_co + 1
  Next j
Next i

End Sub
Private Sub OUTPUT()
Dim i, j, Nloc As Long
Dim TMAX, SMALT, MEANT, ISUM As Double
Dim barra As Double
Dim response As String

 Nloc = 0
 frmMain.Datadda.Refresh
 frmMain.Datadda.Recordset.MoveFirst
 barra = frmMain.ProgressBar1.Value
 For i = 1 To nNodes
   frmMain.ProgressBar1.Value = barra + i * 10 / nNodes
   tablaD(i).nodoO = frmMain.Datadda.Recordset.Fields(0).Value
   tablaD(i).nodoN = frmMain.Datadda.Recordset.Fields(1).Value
   frmMain.Datadda.Recordset.MoveNext
 Next i
```

```
frmMain.Dataout.Refresh
barra = frmMain.ProgressBar1.Value
For j = 1 To NCAN
   frmMain.ProgressBar1.Value = barra + (j * 15) / nNodes
   If Y(j) <> 0 Then
    Nloc = Nloc + 1
    'limpia la tabla de salida
    While Not frmMain.Dataout.Recordset.EOF
      frmMain.Dataout.Recordset.Delete
      frmMain.Dataout.Recordset.MoveFirst
    Wend
    'limpia la tabla de salida
    While Not frmMain.Datadet.Recordset.EOF
      frmMain.Datadet.Recordset.Delete
      frmMain.Datadet.Recordset.MoveFirst
    Wend
    'la tabla debe estar vacía

    frmMa in.Dataout.Recordset.AddNew
    frmMain.Dataout.UpdateRecord
    frmMain.Dataout.Recordset.MoveFirst
    frmMain.Dataout.Recordset.Edit
    frmMain.Dataout.Recordset.Fields(0).Value = CENTER(j)
    frmMain.Dataout.Recordset.Fields(1).Value = tablaD(CENTER(j)).nodoO
    frmMain.Dataout.Recordset.Fields(2).Value = "Solución p-central nº " +
CStr(Nloc)
    frmMain.Dataout.UpdateRecord
    TMAX = 0
    SMALT = 0
    ISUM = 0
    For i = 1 To nNodes
      If CLUST(i) = j Then
        frmMain.Datadet.Refresh
        frmMain.Datadet.Recordset.AddNew
        frmMain.Datadet.UpdateRecord
        frmMain.Datadet.Recordset.MoveLast
        frmMain.Datadet.Recordset.Edit
        frmMain.Datadet.Recordset.Fields(0).Value = i
        frmMain.Datadet.Recordset.Fields(1).Value = tablaD(i).nodoO
        frmMain.Datadet.Recordset.Fields(2).Value = TRVT(i, j)
        frmMain.Datadet.UpdateRecord
        SMALT = SMALT + TRVT(i, j)
        ISUM = ISUM + 1
        If TRVT(i, j) > TMAX Then TMAX = TRVT(i, j)
      End If
```

```
    Next i
    MEANT = SMALT / ISUM
  End If
Next j

frmMain.Dataout.Recordset.MoveFirst
response = MsgBox("El nodo elegido es: " &
frmMain.Dataout.Recordset.Fields(0).Value, 0, "P-Central")

End Sub

Private Sub n_nodos_Change()
 n_candidatos.Text = n_nodos.Text
End Sub
```

Arbol.frm

```
Dim nNodes As Long
Private Type TreeNode
   CurrNode As Long
   dda As Long
   NextNode(0 To 5) As Long
   Dist(0 To 5) As Double
   VisitNumber As Long
   Distance As Double
   TmpVar As Double
End Type
Dim TreeNodeList() As TreeNode
Private Type Campos
  nodoO As Integer
  nodoN As Integer
End Type
Dim tablaD() As Campos

Public Function Nodo_Final(nodo As Long) As Boolean
   If nodo = 0 Then
      Nodo_Final = False
   Else
     If TreeNodeList(nodo).NextNode(1) = -1 Then
     'Esto implica que solo TreeNodeList(I).NextNode(2),..(3)...
     'tb. son -1 y por lo tanto solo hay un vecino
        Nodo_Final = True
       Else
        Nodo_Final = False
```

```
      End If
    End If
End Function

Private Sub Form_Load()
Dim i, Solucion, Gv, nodo, nodos_T, nodo_vecino As Long
Dim response As String

nNodes = Val(frmMain.Text1.Text)

ReDim TreeNodeList(1 To nNodes) As TreeNode
ReDim tablaD(1 To nNodes) As Campos

frmMain.Datadda.Refresh
For nodo = 1 To nNodes
   TreeNodeList(nodo).dda = frmMain.Datadda.Recordset.Fields(2)
   tablaD(nodo).nodoO = frmMain.Datadda.Recordset.Fields(0).Value
   tablaD(nodo).nodoN = frmMain.Datadda.Recordset.Fields(1).Value
   frmMain.Datadda.Recordset.MoveNext
Next nodo

'***** Se asigna el nº de nodo a CurrNode *****
For i = 1 To nNodes
  TreeNodeList(i).CurrNode = i
  TreeNodeList(i).NextNode(0) = -1
  TreeNodeList(i).NextNode(1) = -1
  TreeNodeList(i).NextNode(2) = -1
  TreeNodeList(i).NextNode(3) = -1
  TreeNodeList(i).NextNode(4) = -1
  TreeNodeList(i).NextNode(5) = -1
Next i

'***** Se asigna a cada nodo sus nodos vecinos *****
frmMain.Datasel.Refresh
For i = 1 To nNodes - 1 'nºnodos
' Busca el registro donde nodoi=I
  frmMain.Datasel.Recordset.MoveFirst
  While Not (frmMain.Datasel.Recordset.EOF)
    While (Not frmMain.Datasel.Recordset.EOF) And (Not
(frmMain.Datasel.Recordset.Fields("nodoi").Value = i))
      frmMain.Datasel.Recordset.MoveNext
      If frmMain.Datasel.Recordset.EOF Then GoTo F_1
    Wend
    If TreeNodeList(i).NextNode(0) = -1 Then TreeNodeList(i).NextNode(0) =
frmMain.Datasel.Recordset.Fields("nodof").Value
```

```
   If TreeNodeList(i).NextNode(1) = -1 And TreeNodeList(i).NextNode(0) <>
frmMain.Datasel.Recordset.Fields("nodof").Value Then
TreeNodeList(i).NextNode(1) = frmMain.Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(2) = -1 And TreeNodeList(i).NextNode(0) <>
frmMain.Datasel.Recordset.Fields("nodof").Value And
TreeNodeList(i).NextNode(1) <> frmMain.Datasel.Recordset.Fields("nodof").Value
Then TreeNodeList(i).NextNode(2) =
frmMain.Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(3) = -1 And TreeNodeList(i).NextNode(0) <>
frmMain.Datasel.Recordset.Fields("nodof").Value And
TreeNodeList(i).NextNode(1) <> frmMain.Datasel.Recordset.Fields("nodof").Value
And TreeNodeList(i).NextNode(2) <>
frmMain.Datasel.Recordset.Fields("nodof").Value Then
TreeNodeList(i).NextNode(3) = frmMain.Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode (4) = -1 And TreeNodeList(i).NextNode(0) <>
frmMain.Datasel.Recordset.Fields("nodof").Value And
TreeNodeList(i).NextNode(1) <> frmMain.Datasel.Recordset.Fields("nodof").Value
And TreeNodeList(i).NextNode(2) <>
frmMain.Datasel.Recordset.Fields("nodof").Value And
TreeNodeList(i).NextNode(3) <> frmMain.Datasel.Recordset.Fields("nodof").Value
Then TreeNodeList(i).NextNode(4) =
frmMain.Datasel.Recordset.Fields("nodof").Value
   If TreeNodeList(i).NextNode(5) = -1 And TreeNodeList(i).NextNode(0) <>
frmMain.Datasel.Recordset.Fields("nodof").Value And
TreeNodeList(i).NextNode(1) <> frmMain.Datasel.Recordset.Fields("nodof").Value
And TreeNodeList(i).NextNode(2) <>
frmMain.Datasel.Recordset.Fields("nodof").Value And
TreeNodeList(i).NextNode(3) <> frmMain.Datasel.Recordset.Fields("nodof").Value
And TreeNodeList(i).NextNode(4) <>
frmMain.Datasel.Recordset.Fields("nodof").Value Then
TreeNodeList(i).NextNode(5) = frmMain.Datasel.Recordset.Fields("nodof").Value
   frmMain.Datasel.Recordset.MoveNext
 Wend
F_1:
Next i
'*****************************************************
Solucion = 0
nodos_T = nNodes
While Solucion = 0
  ' Paso1 si el arbol T consiste en un unico nodo parar.
  If nNodes = 1 Then
    nodo = 1
    While TreeNodeList(nodo).CurrNode = 0
      nodo = nodo + 1
    Wend
    Solucion = nodo
```

```
  End If
  'Paso 2 Buscar un nodo final vi y su link asociado(i,j).

  i = 1
  While Not (Nodo_Final(TreeNodeList(i).CurrNode))
    i = i + 1
  Wend
  Gv = 0
  For nodo = 1 To nNodes
   Gv = TreeNodeList(nodo) .dda + Gv
  Next nodo
   Gv = Gv  - TreeNodeList(i).dda
  If TreeNodeList(i).dda >= Gv / 2 Then
    'Paso4 , parar
    Solucion = TreeNodeList(i).CurrNode
  Else
     'Paso 3, modificar TreeNodeList (T)borrando TreeNodeList(I).Currnode y su link.
Ing¡crementar g(j)por g(i), ir a paso 1
     nodo_vecino = TreeNodeList(i).NextNode(0)
     TreeNodeList(nodo_vecino).dda = TreeNodeList(nodo_vecino).dda +
TreeNodeList(i).dda
     'Eliminar nodo vecino de nodo_vecino
     If TreeNodeList(nodo_vecino).NextNode (0) = i Then
       TreeNodeList(nodo_vecino).NextNode(0) =
TreeNodeList(nodo_vecino).NextNode(1)
       TreeNodeList(nodo_vecino).NextNode(1) =
TreeNodeList(nodo_vecino).NextNode(2)
       TreeNodeList(nodo_vecino).NextNode(2) =
TreeNodeList(nodo_vecino).NextNode(3)
       TreeNodeList(nodo_vecino).NextNode(3) =
TreeNodeList(nodo_vecino).NextNode(4)
       TreeNodeList(nodo_vecino).NextNode(4) =
TreeNodeList(nodo_vecino).NextNode(5)
       TreeNodeList(nodo_vecino).NextNode(5) = -1
     End If
     If TreeNodeList(nodo_vecino).NextNode(1) = i Then
       TreeNodeList(nodo_vecino).NextNode(1) =
TreeNodeList(nodo_vecino).NextNode(2)
       TreeNodeList(nodo_vecino).NextNode(2) =
TreeNodeList(nodo_vecino).NextNode(3)
       TreeNodeList(nodo_vecino).NextNode(3) =
TreeNodeList(nodo_vecino).NextNode(4)
       TreeNodeList(nodo_vecino).NextNode(4) =
TreeNodeList(nodo_vecino).NextNode(5)
       TreeNodeList(nodo_vecino).NextNode(5) = -1
     End If
```

```
    If TreeNodeList(nodo_vecino).NextNode(2) = i Then
      TreeNodeList(nodo_vecino).NextNode(2) =
TreeNodeList(nodo_vecino).NextNode(3)
      TreeNodeList(nodo_vecino).NextNode(3) =
TreeNodeList(nodo_vecino).NextNode(4)
      TreeNodeList(nodo_vecino).NextNode(4) =
TreeNodeList(nodo_vecino).NextNode(5)
      TreeNodeList(nodo_vecino).NextNode(5) = -1
    End If
    If TreeNodeList(nodo_vecino).NextNode(3) = i Then
      TreeNodeList(nodo_vecino).NextNode(3) =
TreeNodeList(nodo_vecino).NextNode(4)
      TreeNodeList(nodo_vecino).NextNode(4) =
TreeNodeList(nodo_vecino).NextNode(5)
      TreeNodeList(nodo_vecino).NextNode(5) = -1
    End If
    If TreeNodeList(nodo_vecino).NextNode(4) = i Then
      TreeNodeList(nodo_vecino).NextNode(4) =
TreeNodeList(nodo_vecino).NextNode(5)
      TreeNodeList(nodo_vecino).NextNode(5) = -1
    End If
    If TreeNodeList(nodo_vecino).NextNode(5) = i Then
      TreeNodeList(nodo_vecino).NextNode(5) = -1
    End If

    TreeNodeList(i).CurrNode = 0
    TreeNodeList(i).dda = 0
    TreeNodeList(i).NextNode(0) = -1
    TreeNodeList(i).Dist(0) = 0

    nodos_T = nodos_T - 1
  End If
 Wend

 'limpia la tabla de salida
 While Not frmMain.Dataout.Recordset.EOF
  frmMain.Dataout.Recordset.Delete
  frmMain.Dataout.Recordset.MoveFirst
 Wend

frmMain.Dataout.Refresh
frmMain.Dataout.Recordset.AddNew
frmMain.Dataout.UpdateRecord
frmMain.Dataout.Recordset.MoveLast
frmMain.Dataout.Recordset.Edit
frmMain.Dataout.Recordset.Fields(0).Value = Solucion
```

```
frmMain.Dataout.Recordset.Fields(1).Value = tablaD(Solucion).nodoO
frmMain.Dataout.Recordset.Fields(2).Value = "Solución 1-mediano"
frmMain.Dataout.UpdateRecord

response = MsgBox("El nodo elegido es: " & Solucion, 0, "Arbol")

End Sub
```

CAMBIA.VBP

Cambia.frm

```
Private Type Campos
   nodoA As Integer
   nodoN As Integer
   vol As Double
End Type
Dim tablaD() As Campos
Dim Nnodos As Long


Private Sub Eliminar_aristas()
Dim estai As Boolean
Dim estaf As Boolean

'Se eliminan las aristas que no pertenecen a la red
estai = False
estaf = False
Datasel.Refresh
Datadda.Refresh
Datasel.Recordset.MoveFirst
Datadda.Recordset.MoveFirst
While Not Datasel.Recordset.EOF
.Recordset.MoveFirst
  estai = False
  estaf = False
  While Not Datadda.Recordset.EOF
    If Datasel.Recordset.Fields(2).Value = Datadda.Recordset.Fields(0).Value Then
estai = True
    If Datasel.Recordset.Fields(3).Value = Datadda.Recordset.Fields(0).Value Then
estaf = True
    Datadda.Recordset.MoveNext
  Wend
  If Not (estai And estaf) Then Datasel.Recordset.Delete
  Datasel.Recordset.MoveNext
Wend

End Sub
Private Sub Eliminar_cruces()
Dim cruce As Boolean

Datasel.Refresh
Datasel.Recordset.MoveFirst
Datadda.Refresh
```

```
Datadda.Recordset.MoveFirst
While Not Datadda.Recordset.EOF
 If Datadda.Recordset.Fields("nodo_orig") > 2000 Then
  cruce = False
  Datasel.Recordset.MoveFirst
  While Not Datasel.Recordset.EOF
  'Si el cruce esta unido a una arista de la red...
   If Datasel.Recordset.Fields("nodoi").Value =
Datadda.Recordset.Fields("nodo_orig") Or Datasel.Recordset.Fields("nodof").Value
= Datadda.Recordset.Fields("nodo_orig") Then
     cruce = True
   End If
   Datasel.Recordset.MoveNext
  Wend
  '...se conserva si no se elimina
  If Not cruce Then Datadda.Recordset.Delete
 End If
 Datadda.Recordset.MoveNext
Wend

End Sub

Private Sub Renombrar_nodos()
Dim i As Integer
Dim Num_nodos As Integer
ReDim tablaD(1 to Nnodos) As Campos

'A cada nodo se le asigna una nueva enumeración
Datasel.Refresh
Datadda.Refresh
Datasel.Recordset.MoveFirst
Datadda.Recordset.MoveFirst

i = 1
While Not Datadda.Recordset.EOF
 tablaD(i).nodoA = Datadda.Recordset.Fields("nodo_orig").Value
 tablaD(i).nodoN = i
 tablaD(i).vol = Datadda.Recordset.Fields("dda").Value
 i = i + 1
 Datadda.Recordset.MoveNext
Wend

'se renombran los nodos de tablasel o datasel
Datasel.Recordset.MoveFirst
While Not Datasel.Recordset.EOF
  Datasel.Recordset.Edit
```

```
 i = 1
 While Datasel.Recordset.Fields("nodoi").Value <> tablaD(i).nodoA
  i = i + 1
 Wend
 Datasel.Recordset.Fields("nodoi").Value = tablaD(i).nodoN

 i = 1
 While Datasel.Recordset.Fields("nodof").Value <> tablaD(i).nodoA
  i = i + 1
 Wend
 Datasel.Recordset.Fields("nodof").Value = tablaD(i).nodoN
 Datasel.Recordset.Update
 If Not Datasel.Recordset.EOF Then Datasel.Recordset.MoveNext
Wend

'se renombran los nodos de tabladda o datadda
Datadda.Recordset.MoveFirst
i = 1
While Not Datadda.Recordset.EOF
 Datadda.Recordset.Edit
 Datadda.Recordset.Fields("nodo") = i
 i = i + 1
 Datadda.Recordset.Update
 If Not Datadda.Recordset.EOF Then Datadda.Recordset.MoveNext
Wend

End Sub

Private Function Numero_nodos() As Long
Dim i As Integer

i = 0
Datadda.Refresh
'Se cuenta el nº de nodos
While Not Datadda.Recordset.EOF
 i = i + 1
 If Not Datadda.Recordset.EOF Then Datadda.Recordset.MoveNext
Wend
Numero_nodos = i
End Function
Private Sub Agregar_registros()
ReDim nodoi(1To 450) As Integer
Dim nodof(1 To 450) As Integer
Dim largo(1 To 450) As Double
Dim i, j As Long
```

```
'Se agregan registros, se repite la información de los antiguos registros,
'donde nodoi será nodof y nodof será nodoi
Datasel.Refresh
i = 0

While Not Datasel.Recordset.EOF
 i = i + 1
 nodoi(i) = Datasel.Recordset.Fields(2).Value
 nodof(i) = Datasel.Recordset.Fields(3).Value
 largo(i) = Datasel.Recordset.Fields(1).Value
 If Not Datasel.Recordset.EOF Then Datasel.Recordset.MoveNext
Wend

For j = 1 To i
  Datasel.Recordset.MoveLast
  Datasel.Recordset.AddNew
  Datasel.Recordset.Fields("ID") = 0
  Datasel.Recordset.Fields("Largo") = largo(j)
  Datasel.Recordset.Fields("nodoi") = nodof(j)
  Datasel.Recordset.Fields("nodof") = nodoi(j)
  Datasel.Recordset.Update
Next j
End Sub

Private Sub Form_Load()
Dim i As Long
Dim mensaje As String

Nnodos = Numero_nodos

Eliminar_aristas
Eliminar_cruces
Renombrar_nodos
Agregar_registros

mensaje = MsgBox("Proceso Cambia terminado.", vbInformation, "Termino de
proceso")

Unload Form1
End Sub
```

Anexo 2. Scripts en Avenue del proyecto hecho en ArcView.

Script: Cambia

```
if (_nueva) then
   System.Execute("c:\Lcz\cambia.exe")
   _nueva=false
 end
```

Script: Crear Tablas

```
'Si la red es nueva, no se ha ejecutado este script para esa red,
'entonces _nueva es true. Si por el contrario ya se eljecuto este sript
'entonces _nueva es false.

 av.Run("Crear_Tabladda",{})
 av.Run("Crear_Tablasel",{})
 av.Run("Crear_Tablarut",{})
 av.Run("Crear_Tablaout",{})
 av.Run("Crear_TablaDet",{})

MsgBox.Info("Tablas creadas","Aviso")
```

Script: Crea_TablaDd

```
'Crear la tabla con los campos correspondientes
TheVtab = VTab.MakeNew("c:\Lcz\Tabladd.dbf".AsFileName,dBase )
myTable = Table.Make(theVTab)
for each rec in 0..400
 field1= Field.Make("N"+rec.AsString, #FIELD_DOUBLE,17,2)
 TheVtab.AddFields({field1})
end
```

Script: Crea_TablaDda

```
' Este script captura los nodos seleccionados y los lleva a otra tabla.
myView = av.GetActiveDoc
myTheme = myView.GetThemes.Get(0)

if (myTheme.CanSelect) then
  mySet = myTheme.GetFTab.GetNumSelRecords
```

```
 ' Pregunta al usuario quiere borrar las ciudades seleccioadas y entrega el nº de
ciudades seleccioandas

'  if (clearSet) then
 '   myTheme.ClearSelection
  'end
end

' Entrega el nombre de cada nodo seleccionado
myFTab  = myTheme.GetFTab
'Crear la tabla con los campos correspondientes
TheVtab = VTab.MakeNew("c:\Lcz\Tabladda.dbf".AsFileName,dBase )
myTable = Table.Make(theVTab)
field1= Field.Make("Nodo_orig", #FIELD_SHORT,10,0)
field2= Field.Make("Nodo", #FIELD_SHORT,10,0)
field3= Field.Make("Dda", #FIELD_SHORT,10,0)
TheVtab.AddFields({field1,field2,field3})

for each rec in myFTab.GetSelection
  'Llenar la tabla con los registros de los nodos seleccionados
  numrec=theVTab.AddRecord
  theVTab.SetValue
(field1,numrec,myFTab.ReturnValueNumber(myFTab.FindField("ID"),rec))
  theVTab.SetValue
(field2,numrec,myFTab.ReturnValueNumber(myFTab.FindField("ID"),rec))
  theVTab.SetValue
(field3,numrec,myFTab.ReturnValueNumber(myFTab.FindField("Volumen"),rec))

end

_nueva=true
```

Script: Crea_TablaSel

```
' Este script captura los nodos seleccionados y los lleva a otra tabla.
myView = av.GetActiveDoc
myTheme = myView.GetThemes.Get(1)

if (myTheme.CanSelect) then
  mySet = myTheme.GetFTab.GetNumSelRecords
  ' Pregunta al usuario quiere borrar las ciudades seleccioadas y entrega el nº de
ciudades seleccioandas
  'clearSet = MsgBox.YesNo("El número de ciudades seleccionadas
es"++mySet.AsString+NL+"Clear selected set?","FTheme Example", FALSE)
  'if (clearSet) then
```

```
  ' myTheme.ClearSelection
 'end
end

' Entrega el nombre de cada nodo seleccionado
myFTab  = myTheme.GetFTab
'Crear la tabla con los campos correspondientes
TheVtab = VTab.MakeNew("c:\Lcz\TablaSel.dbf".AsFileName,dBase )
myTable = Table.Make(theVTab)
field1= Field.Make("ID", #FIELD_SHORT,10,0)
field2= Field.Make("Largo", #FIELD_SHORT,30,0)
field3= Field.Make("nodoi", #FIELD_SHORT,10,0)
field4= Field.Make("nodof", #FIELD_SHORT,10,0)
TheVtab.AddFields({field1,field2,field3,field4})

for each rec in myFTab.GetSelection
 'Llenar la tabla con los registros de los nodos seleccionados
 numrec=theVTab.AddRecord
 theVTab.SetValue
(field1,numrec,myFTab.ReturnValueNumber(myFTab.FindField("ID"),rec))
 theVTab.SetValue
(field2,numrec,myFTab.ReturnValueNumber(myFTab.FindField("Costo"),rec))
 theVTab.SetValue
(field3,numrec,myFTab.ReturnValueNumber(myFTab.FindField("nodoi"),rec))
 theVTab.SetValue
(field4,numrec,myFTab.ReturnValueNumber(myFTab.FindField("Nodof"),rec))

end

_nueva=true
```

Script: Crea_TablaDet

```
'Crea un tablaout, para que esté limpia.
TheVtab = VTab.MakeNew("c:\Lcz\TablaDet.dbf".AsFileName,dBase )
field1= Field.Make("NODO", #FIELD_SHORT,10,0)
field2= Field.Make("NODO_ORIG", #FIELD_SHORT,10,0)
field3= Field.Make("Costo", #FIELD_SHORT,10,0)
TheVtab.AddFields({field1,field2,field3})
```

Script: Crea_TablaRut

```
'Crear la tabla con los campos correspondientes
TheVtab = VTab.MakeNew("c:\Lcz\Tablarut.dbf".AsFileName,dBase )
```

```
myTable = Table.Make(theVTab)
for each rec in 0..400
 field1= Field.Make("N"+rec.AsString, #FIELD_DOUBLE,17,2)
 TheVtab.AddFields({field1})
end
```

Script: Crea_TablaOut

```
'Crea un tablaout, para que esté limpia.
TheVtab = VTab.MakeNew("c:\Lcz\Tablaout.dbf".AsFileName,dBase )
field1= Field.Make("NODO", #FIELD_SHORT,10,0)
field2= Field.Make("NODO_ORIG", #FIELD_SHORT,10,0)
field3= Field.Make("Nombre", #FIELD_VCHAR,30,0)
TheVtab.AddFields({field1,field2,field3})
```

Script: Nodo_Elegido

```
'Este script habre tablaout y lee el nodo elegido
Proyecto = av.GetProject
TheVtabOut = VTab.make("c:\Lcz\Tablaout.dbf".AsFileName,false,false)
TablaOut = Table.Make(theVTabOut)
Tab_out=TablaOut.GetVTab
field_nodo = Tab_out.FindField("Nodo")
field_desc = Tab_out.FindField("Nombre")

TheVtab = VTab.make("c:\Lcz\Tabladda.dbf".AsFileName,false,false)
Tabla = Table.Make(TheVtab)
Tab_dda=Tabla.GetVTab
f_nodo = Tab_dda.FindField("Nodo")
f_nodo_orig = Tab_dda.FindField("Nodo_orig")

LaView = av.GetActiveDoc
Theme_predio=LaView.GetThemes.Get(0)
Ftab_predio=Theme_predio.GetFtab
ElField_id= FTab_predio.FindField("ID")
Ftab_predio.GetSelection

anFTab=FTab.MakeNew("TablaRes".AsFileName, Point)
IdField=Field.Make("ID", #FIELD_DECIMAL,8,0)
NodoField=Field.Make("Nodo", #FIELD_DECIMAL,8,0)
NodoOrigField=Field.Make("Nodo_Orig", #FIELD_DECIMAL,8,0)
DescripField=Field.Make("Descripcion", #FIELD_VCHAR,30,0)
anFTab.AddFields({idField,NodoField,NodoOrigField,DescripField})
```

```
ShapeField=anFTab.FindField("Shape")

For Each Nodo_loc in Tab_out
  nodo_elegido=Tab_out.ReturnValueNumber(field_nodo,Nodo_loc)
  descripcion=Tab_out.ReturnValueString(field_desc,Nodo_loc)
  Nodo_orig=0
  For Each Registro in Tab_dda
    nodo_dda=Tab_dda.ReturnValueNumber(f_nodo,Registro)
    If (nodo_elegido=nodo_dda) Then
Nodo_orig=Tab_dda.ReturnValueNumber(f_nodo_orig,Registro)
   End
 End

 punto=FTab_predio.GetLabelPoint(0)
 For Each reg in FTab_predio
  campo=FTab_predio.ReturnValueNumber(ElField_id,reg)
  If (nodo_orig=campo) Then
    punto=FTab_predio.GetLabelPoint(reg)
 End
End

 TheGShape = GraphicShape.Make(punto)
 newRecNum=anFTab.AddRecord
 anFTab.SetValue(shapeField, newRecNum, punto)
 anFTab.SetValue(IdField,newRecNum,newRecNum)
 anFTab.SetValue(NodoField,newRecNum,nodo_elegido)
 anFTab.SetValue(NodoOrigField,newRecNum,nodo_orig)
 anFTab.SetValue(DescripField,newRecNum,descripcion)
end

 LaView.AddTheme(FTheme.Make(anFTab))
 newLegend = Legend.Make( #SYMBOL_MARKER )
 newLegend.GetSymbols.Get( 0 ).SetSize(12)
 aTheme=LaView.GetThemes.Get(0)
 aTheme.SetLegend( newLegend )
 aTheme.UpdateLegend  ' Redraw
 aTheme.SetVisible(TRUE)
```

Script: Localizar

```
System.Execute("c:\Lcz\Localiza.exe")

 if (MsgBox.YesNo("¿El nodo elegido fue encontrado?","Pregunta", TRUE)) then
  av.Run("Nodo_elegido",{})
 end
```