Sistema Automatizado de Ruteo en Internet

Servicios de relevante importancia en optimización y logística para la empresa.

Adolfo Agorio Lea Kaufman Pablo Rodríguez
Universidad de la República
Facultad de Ingeniería
Ingeniería en Computación
Departamento de Investigación Operativa
Tesis de grado, Taller V
Marzo del 2001

Contenido

Contenido 2

Resumen 11

Introducción General 12

Contexto **12**
- Interés Académico 12
- Responsables 13
- Objetivos del Proyecto 13

Definición del Problema **13**

Contexto de Trabajo **14**

Formación Ofrecida al Estudiante **14**

Limitaciones de Soluciones Existentes **14**

Principal Logro de Este Documento **15**

Vista General del Resto del Documento **16**

Requerimientos de Usuario 17

Documento de Requerimientos **17**
- Introducción 17
- Problemática 17
- Soluciones Existentes 17
- Requerimientos 18
  - Funcionalidades de Ruteo 18
  - Funcionalidades geográficas 18
  - Funcionalidades extra 18
  - Interfase gráfica 18
  - Configuración del producto 18
  - Manejo de los datos de entrada y de salida 19
  - Modularizacíon 19
  - Aplicación en Internet 19
  - Herramientas 19
  - Investigación 19

Análisis de Requerimientos 20

Introducción **20**
- Propósito 20
- Alcance 20
  - Básico 20
  - Especial 21
- Definiciones, acrónimos y abreviaciones 21
- Referencias 21
- Vista general 21

Descripción general **21**

Perspectiva del Producto 21
Requerimientos de Hardware....22
Ejemplo de interfaz visual....22
Funcionalidades....22
Actores....29
Dominio del Problema 29
Restricciones generales....30
Asunciones y dependencias....30

Evaluación del Código Disponible....31

El porqué de una evaluación....31
Evaluación....31
Introducción 31
Legibilidad 31
Modularidad....32
Correctitud....32
Eficiencia....32
Genericidad....32
Conclusiones 33

Desarrollo Teórico 34

Introducción....34
Modelado 34
Complejidad 34
Problemas NP-Duros....34
Problemas de ruteo....35
Problemas clásicos....35
VRP Estocástico....36
Introducción 36
Descripción Formal....36
Definiciones previas:....36
Problema:....36
Solución planteada 37
Idea preliminar:....37
Formalización 37
Teorema [2]....37
Hipótesis: 37
Tesis:....37
Demostración....37
Teorema Central del limite. [9] 39
Suposición adicional 39
Interpretación de resultados....40
Resumen....41

Diseño e Implementación de los Algoritmos de Ruteo....43

Algoritmos de Cálculo....43
Consideraciones Generales....43
Clarke & Wright Aplicado 44
TSP y VRP....44
MTSP....45
MDVRP....45
VRPTW....45
MDVRPTW 45

VRPHF 46
DARP 46
CPP 47
SVRP 47
Diseño 47
Rutas. 48
Caminos. 48
Sitios 49
Diagrama General. 50
Especificación Técnica de la Biblioteca de Vínculos Dinámicos : Router 51
Introducción 51
Arquitectura de Comunicación: Interfase 52
Entrada de la Biblioteca 52
Generalidades. 52
Especificación 53
Principales identificadores. 53
Salida de la Biblioteca 56
Generalidades. 56
Especificación 57
Principales identificadores. 57
Funciones Exportadas. 59
Cargar Estructura 59
Descripción 59
Declaración 59
Parámetros 60
Valor de Retorno. 60
Calcular 60
Descripción 60
Declaración 60
Parámetros 60
Valor de Retorno. 60
Dijkstra 60
Descripción 60
Declaración 60
Parámetros 60
Valor de Retorno. 60
Códigos de Error 61

Diseño e Implementación del Sistema 62

Introducción. 62
Servidores. 62
Servidor Problema 63
Realidad 63
Nodo 64
Punto 64
Ventana 64
Sitio 64
Arco 64
Red 64
Par 65
Tipo_Vehiculo 65
Trayecto 65
Camino 65
Ruta 65
Problema 65
MOR 66
TDN 66
Formularios 66

- Frm_mapa....66

Módulos....67

- Declaraciones....67
- Reportes....68
- Rutear....69

Módulos de clases....69

- clsNodo....69
- clsPunto....69
- clsVentana....70
- clsSitio....70
- clsTipo_Vehículo....72
- ClsArco....73
- ClsRed....73
- ClsPar....73
- ClsTrayecto....74
- ClsCamino....75
- ClsRuta....77
- ClsProblema....78

Servidor de Cálculo....79

Realidad....79

Nodo....80

TDN....80

Módulos....80

- Declaraciones....80

Módulos de clases....81

- clsNodo....81
- clsRaiz....81

Servidor de Localización....81

Manejo de los datos....81

Comunicación entre Servidores....82

Servidor Web....82

Realidad....82

Usuario....82

Problema....82

Elemento....82

Implementación....83

General....83

Index.asp....83

Problema....83

Problema.asp....83

Mantenimiento....83

Mantenimiento.asp....83

Contenido.htm....83

La_lista.asp....83

Nada_registro.asp....84

Los_elegidos.asp....84

Mantenimiento BD....84

MantenimientoBD.asp....84

ContenidoBD.htm....84

La_listaBD.asp....84

Nada_registroBD.asp....84

Los_elegidosBD.asp....84

Escenarios....84

Definir.asp....85

Recuperar_Escenario....85

Cargar_Escenario.asp....85

localización....85

Localizar.htm 85
Contenido.htm 85
el_input.htm 85
Nombre_de_calle.htm 85
Nombre_de_esquina.htm 85
Mapa.htm 86
Reportes 86
Includes 86

Testeo 87
Plan de Verificación y Validación 87
Objetivos 87
Alcance 87
Referencias 87
Ítems de Testeo 87
Software 88
Documentos 88
Características que No Serán Testeadas 88
Enfoque 88
Criterios de Aceptación y No Aceptación de los Ítems de Testeo 89
Criterios de Suspensión y Requerimientos de Reasunción 89
Criterios de Suspensión 89
Introducción 89
Software 89
Documentos 89
Requerimientos de Reasunción 90
Software 90
Documentos 90
Plan de prueba del sistema 90
Router 90
Interfase 90
Entorno de prueba del sistema 91
Router 91
Interfase 91
Especificación de caso de testeo 91
Prueba mínima en implementación 91
VRP (Vehicle Routing Problem) 92
MDVRP (Multi-Depot Vehicle Routing Problem) 92
VRPTW (Vehicle Routing Problem with Time-Windows). 92
MDVRPTW (Multi-Depot Vehicle Routing Problem with Time-Windows). 93
VRPHF (VRP Flota Heterogénea). 93
TSP (Travelling Salesman Problem). 93
MTSP (M - Travelling Salesman Problem) 93
DARP (Dial-a-ride Problem). 94
CPP (Chinese Postman Problem). 94
SVRP (Stochastic Vehicle Routing Problem). 94
DIJKSTRA. 94
Prueba exhaustiva en integración 95
Informes del testeo 95
Router 95

Conclusiones 96
Resumen de la Contribución del Documento 96
Resultado Principal 96
Aplicaciones del Resultado 96
Dirección de Investigaciones Futuras y Cuestiones Abiertas 97

Bibliografía 99

Apéndice A: Control de Reuniones 100

Primera Reunión Administrativa 100
Fecha 100
Participantes 100
Objetivo 100
Temas Tratados 100
Funcionalidades de Ruteo 100
Funcionalidades Extra 100
Configuración del producto 101
Manejo de los mapas 101
Aplicación en Internet 101
Herramientas 101
Tareas Mediatas Requeridas 101
Segunda Reunión Administrativa 101
Fecha 101
Participantes 101
Objetivo 101
Temas Tratados 102
Nuevos Requerimientos 102
Modularizacíon 102
SVRP 102
Consideraciones Generales 102
Bibliografía recomendada 102
Herramientas 102
Tareas Mediatas Requeridas 102
Geocodificación (AddressMatching) 103
Tercera Reunión Administrativa 103
Fecha 103
Participantes 103
Objetivo 103
Temas Tratados 103
Consideraciones Generales 103
Tareas Mediatas Requeridas 103
Cuarta Reunión Administrativa 104
Fecha 104
Participantes 104
Objetivo 104
Temas Tratados 104
Consideraciones Generales 104
Herramientas 104
Tareas Mediatas Requeridas 104

Apéndice B: Plan de Trabajo y Avances del Proyecto 105

Identificación de tareas 105
Diagramas de precedencia (Gantt) 107
Especificación Global 107
Especificación por Sub-Tareas 108

Apéndice C: Normas y Estándares de Documentación 110

Apéndice D: Normas y Estándares de Implementación 110

Alcance 110
Contenido 110
Normas de Programación 110
Convenciones de Nomenclatura 111
Diagramado del Código 111
Estructura de la Clase 111
Estructura de las Rutinas 112
Técnicas de Programación 112

Apéndice E: Definición de Problemas 113

Problemas de Ruteo de Vehículos 113
TSP 113
Formulación Matemática. 114
MTSP 115
Formulación Matemática. 115
VRP Clásico 115
MDVRP 117
Formulación Matemática. 117
VRPHF: Problemas con Flota Heterogénea 117
VRPTW: Problemas con Ventanas de Tiempo 119
CPP 119
DARP 120
SVRP 120
Formulación Matemática. 120
Ruteo con compartimentos 121

Apéndice F: Reporte de Incidencias 122

Problemas de ruteo: 122
Modularización: 122
Sitio Web: 122

Apéndice G: Geocodificación (address matching) 123

introducción 123
Definición del Problema 123
Solución Planteada 123
Variantes 124
Consideraciones 125
Conclusiones 125
Referencias 125

Apéndice H: Implementación y Esquema de Clases 126

Caminos 126

Rutas **128**
Sitios **131**
Vehículos **133**
Varios útiles **134**

Apéndice I: Sistemas de Información Geográfica

..... **136**
Definición de un GIS **136**
Forma de Trabajo de un GIS **136**
Relacionando Información de Distintos Orígenes 136
Captura de Datos 137
Integración de los Datos 137
Estructuras de Datos 137
Modelado de Datos 137
Función de un GIS **137**
Devolución de Información 138
Modelo Topológico 138
Redes 138
Salida 138
Aplicaciones de GIS **138**
GIS a Través de la Historia 138
Creación de Mapas 138
Selección de Lugares Estratégicos 139
Planificación de Sistemas de Emergencia 139
Simulación de Efectos Climáticos y Desastres Ambientales 139
El Futuro de los GIS 139
Historia y Cambio Global del Clima 139
Incorporando el Tiempo 139
Conclusiones **140**

Apéndice J: Estudio de Soluciones Existentes 141

Soluciones Existentes **141**
DynaRoute 141
GeoRoute 5 141
GeoRoute Postal 141
RouteSmart 142
SHIPCONS II 142
TransCAD 142
Trapeze-Taxi 142
Direct Route 142
GeoRoute-Municipal 143
LoadExpress Plus 143
RiMMS 143
RoutePro Dispatcher 143
Routronics 2000 143
STARS 143
Optrak 144
Truckstops for Windows 144
ArcLogistics Route 144
Resumen **144**

Apéndice K: Biblioteca de Vínculos Dinámicos : Router 145

Introducción **145**

Características del Usuario **146**
Herramienta de Desarrollo **146**
Arquitectura de Comunicación: Interfase **146**
Entrada de la Biblioteca **147**
Generalidades 147
Especificación 147
Principales identificadores 147
Salida de la Biblioteca **151**
Generalidades 151
Especificación 151
Principales identificadores 152
Funciones Exportadas **154**
Cargar Estructura 154
Descripción 154
Declaración 154
Parámetros 155
Valor de Retorno 155
Calcular 155
Descripción 155
Declaración 155
Parámetros 155
Valor de Retorno 155
Dijkstra 155
Descripción 155
Declaración 155
Parámetros 156
Valor de Retorno 156
Códigos de Error **156**
Error Genérico 156
Error al leer un Sitio 156
Errores correspondientes a CargarEstructura 157
Errores correspondientes a calcular el Dikjstra 157
Errores correspondientes a los algoritmos de ruteo (función Calcular) 158
Ejemplos **161**
CargarEstructura 161
Dijkstra 161
Calcular 162
TSP 162
MDVRP 163
VRPTW 165
MDVRPTW 166
MTSP 168
DARP 171
CPP 174
VRPHF 174
SVRP 179
Caminito.ini 180
Entrada para Cargar de la Red **183**
Especificación 183
Interface para el Calculo del Dijkstra **184**
Entrada 184
Especificación 184
Salida 185
Especificación 185

Resumen

El presente documento se enmarca dentro de los proyectos de Taller V 2000 del Departamento de Investigación Operativa del Instituto de Computación de la Facultad de Ingeniería.
Su función es describir la problemática planteada para el proyecto específico, las soluciones propuestas y un seguimiento de su concreción. Se presentan también conclusiones sobre los resultados obtenidos y posibles caminos por los cuales seguir desarrollando el trabajo realizado.

En este caso el proyecto se trata de continuar y extender uno anterior. El tema propuesto es la resolución a través de Internet de diferentes Problemas de Ruteo de Vehículos (VRP), un área en creciente desarrollo tanto por su interés económico (para las empresas que se dedican a la recolección/distribución de bienes y servicios), como académico (dada la dificultad que presenta el problema); y su integración con un Sistema de Información Geográfica (SIG), un área también en pleno desarrollo..
La propuesta concreta para el trabajo es realizada por la Empresa Ingenieros Consultores Asociados (ICA), y consiste en realizar un prototipo que resuelva la mayor cantidad de problemas de ruteo posibles, integrados con un SIG. El interés por este tipo de producto surge pues en el mercado no existe ningún producto que resuelva diversos tipos de problemas de ruteo. Dentro de esos problemas están incluidos algunos que requieren de la presentación de soluciones novedosas por no existir soluciones públicas.

El proyecto exige a su vez una parte de investigación tanto sobre VRP y SIG específicamente como sobre problemas que aparecen si se quisiera realizar un producto completo como ser Address Matching. Los resultados de estas investigaciones también se describen aquí.

Capítulo 1

Introducción General

Introducción General

El problema clásico de Ruteo de Vehículos (VRP por sus siglas en Ingles) definido como: "determinar un conjunto de rutas de costo mínimo para una flota homogénea de vehículos que sirve a un conjunto de clientes dispersos geográficamente" ha sido un área de creciente interés tanto privado como académico durante las dos ultimas décadas.
El interés a nivel privado se justifica por los enormes costos que deben afrontar las empresas dedicadas a la distribución y/o recolección por concepto de fletes.
A nivel académico, el interés se justifica por la enorme complejidad asociada a la resolución de este tipo de problemas (el VRP pertenece a la clase de problemas denominados NP-duros).

Durante las dos ultimas décadas se han realizado grandes avances en el desarrollo de métodos de solución del VRP tanto a nivel de los llamados Métodos Exactos (que obtienen una solución optima pero en tiempo de calculo inaceptable) como a nivel de Métodos Heurísticos (que obtienen una solución "aceptable" rápidamente).

Otro aspecto que se ha venido desarrollando en la ultima década es la integración de los Sistemas de Información Geográfica (SIG) y los algoritmos de solución para VRP. Esta integración forma una poderosa herramienta de apoyo a la toma de decisiones. La idea del Taller V planteado es rediseñar ampliando los resultados obtenidos en el Taller V "Ruteo de Vehículos" de 1999 realizado por Carlos Pedezert, Javier Barreiro y Nicolás Sosa integrando los algoritmos desarrollados en dicho Taller con un SIG, además de resolver nuevos problemas de ruteo y desarrollar un sitio Web que ofrezca los servicios desarrollados.

Contexto

Este proyecto es desarrollado por un grupo de Taller V de la carrera de computación de la Facultad de Ingeniería a pedido del área de optimización de la Empresa Ingenieros Consultores Asociados (ICA), cediendo a la misma todos los derechos sobre el producto resultante.
El grupo de Taller V es integrado por: Adolfo Agorio, Lea Kaufman y Pablo Rodríguez.

Interés Académico

El Taller se enmarca en el área de interés del Departamento de Investigación Operativa (Área Optimización Combinatoria).

Responsables

ICA: Ing. Leonardo Loureiro, Gerente de Marketing
InCo: Ing. Omar Viera, Responsable Académico.

Objetivos del Proyecto

Se pretende desarrollar un software genérico S.A.R. (Sistema Automatizado de Ruteo) a través de Internet que resuelva la mayor cantidad posible de problemas de ruteo, logrando de esta forma un paquete genérico, que permita además una buena interacción con el usuario final.
Para lograr completamente estos objetivos se desea además integrar la resolución de estos problemas a un SIG (Sistema de Información Geográfica)

Definición del Problema

Implementar un prototipo que integre algoritmos para VRP con un SIG en el ambiente de Intente. Este prototipo no debe depender del mapa a usar, debe tener una interfase que maneje varios mapas y debe poseer todas las funcionalidades básicas del mapeo y manejar el concepto de proyectos.

Con respecto a los problemas de ruteo a resolver, a continuación se detalla una lista de los problemas clásicos contemplados en este proyecto:

- **VRP** (Vehicle Routing Problem). En este problema, dado un depósito y uno o varios clientes, se determina la cantidad de vehículos (todos con la misma capacidad de carga) y la ruta tales el recorrido de los vehículos sea mínimo.
- **MDVRP** (Multi-Depot Vehicle Routing Problem). Este problema es similar al anterior, pero la cantidad de depósitos es mayor a 1.
- **VRPTW** (Vehicle Routing Problem with Time-Windows). En este problema, cada cliente tiene una ventana de tiempo (o time-window), es decir, un rango de tiempo en el que el cliente puede ser atendido. Este problema considera un solo depósito, que tiene asociado una ventana de tiempo que indica el horario en que presta servicios.
- **MDVRPTW** (Multi-Depot Vehicle Routing Problem with Time-Windows). Ídem anterior, pero se consideran varios depósitos.
- **VRPHF** (VRP Flota Heterogénea). Ídem VRP, pero en este caso los vehículos tienen diferente capacidad. Por lo tanto ya no son itercambiables entre si. Se debe generar una ruta para cada vehiculo.
- **TSP** (Travelling Salesman Problem). Este problema es un caso particular del VRP, donde se tiene un sólo depósito y un solo vehículo, y donde la demanda de los clientes es cero.
- **MTSP**). Este problema es un caso particular del VRP, donde se tiene un sólo depósito y varios vehículos, y donde la demanda de los clientes es cero.
- **DARP** (Dial-a-ride Problem). Este problema resuelve el caso en que cada cliente tiene un punto de carga y un punto de entrega, todo dentro de una ventana de tiempo. El problema es, como en los casos anteriores, minimizar el costo para servir a todos los clientes, partiendo desde un depósito dado.
- **CPP** (Chinese Postman Problem). El problema del CPP es hallar un recorrido tal que pase por todos los clientes y su costo sea mínimo, sin embargo la variante del CPP resuelta por este software recibe las demandas de los clientes, no de los arcos.
- **SVRP** (Stochastic Vehicle Routing Problem). Es una generalización del problema clásico VRP, donde los clientes tienen demanda no determinista.

Por parte de la empresa existe conocimiento previo sobre la solución a dichos problemas, y en particular una implementación primaria realizada por el Taller de 1999, sobre la cual se deben agregan funcionalidades a definir.

No se debe perder de vista en el diseño de la aplicación una aplicación en Internet, que será desarrollada por el grupo de Taller. El sistema debe además retroalimentarse, analizando la información actualizando los resultados que brinda.

Contexto de Trabajo

Windows 2000 Server[MR], MapObject 2.0, MapObject IMS, Visual C/C++, Visual Basic, Visual InterDev, SQL 7.0 Server.

Formación Ofrecida al Estudiante

Formación en el área de VRP y Sistemas de Información Geográfica (SIG), así como en todos los conocimientos básicos necesarios para el desarrollo de un producto final, como ser la documentación, planificación y interfase con las exigencias de una empresa de software.

Limitaciones de Soluciones Existentes

En primer lugar vale destacar que hasta el momento no existen a nivel mundial soluciones para una gama tan amplia de problemas de ruteo en Internet.
En Montevideo en particular se presenta un sitio que resuelve ruteo punto a punto, y recientemente otro que soluciona uno de los problemas más simples contemplados en este proyecto.

En consecuencia las soluciones existentes que pueden ser analizadas no son para Internet.
Al considerar estas soluciones al problema planteado, se observa que tienen grandes limitantes.
En la siguiente tabla se detallan algunos paquetes de software difundidos comercialmente y los problemas que resuelve cada uno de ellos (por mas detalles ver el apéndice J: Estudio de Soluciones Existentes)

	TSP	TSPTW	TSPHF	VRP	VRPTW	VRPHF	MDVRP	CPP	DARP
DynaRoute	-	-	-	-	-	-	-	-	Si
GeoRoute 5	Si	Si	Si	Si	Si	Si	Si	-	-
GeoPostal	-	-	-	-	-	-	-	Si	-
RouteSmart	Si	Si	Si	Si	Si	Si	Si	-	-
SHIPCONS II	Si	Si	Si	Si	Si	Si	Si	-	-
TransCad	Si	Si	Si	Si	Si	Si	Si	Si	-
Trapeze-Taxi	-	-	-	-	-	-	-	-	Si
Direct Route	Si	Si	Si	Si	Si	Si	S/i	-	-
GeoRoute-Municipal	-	-	-	-	-	-	-	Si	-
LoadExpress Plus	Si	Si	Si	Si	Si	Si	Si	-	-
RiMMS	Si	s/i	s/i	Si	s/i	s/i	S/i	-	-
RoutePro Dispatcher	Si	Si	Si	Si	Si	Si	Si	-	-
Routronics 2000	-	-	-	-	-	-	-	-	Si
STARS	Si	Si	Si	Si	Si	Si	Si	-	-
Optrak	Si	s/i	s/i	Si	s/i	s/i	Si	-	-
TruckStops for Windows	Si	Si	Si	Si	Si	Si	Si	-	-
ArcLogistics Route	Si	Si	Si	Si	Si	Si	Si	-	-

[1]Tabla Comparativa

Notar que todos los paquetes comerciales presentados no resuelven todos los casos presentados y resueltos en este informe.

Se observa que existen tres grupos de paquetes comerciales:

- Los que resuelven el problema del ruteo de vehículos con muchos depósitos, flotas heterogéneas y time-windows (donde el TSP, TSPTW, TSPHF, VRP, VRPTW y VRPHF son casos particulares)
- Los que resuelven el problema del CPP
- Los que resuelven el problema DARP

Entonces hay 3 tipos de paquetes claramente diferenciados y que no son comunes los paquetes que resuelvan determinados problemas de un tipo y problemas de otros tipos. Se puede ver que ninguno de los que resuelve DARP resuelve otro tipo de problemas, ninguno de los que resuelve CPP resuelve otro tipo de problemas y ninguno de los que resuelve VRP y sus extensiones resuelven DARP o CPP.

Principal Logro de Este Documento

Como principal logro de este documento se tiene el desarrollo completo e integral de una solución para Internet del problema planteado. Dicha solución abarca prácticamente todos los requerimientos presentados por el usuario. Además la arquitectura de la misma esta pensada para su fácil adaptabilidad a un software monousuario de oficina.

[1] Estudio realizado en 1999.

Es importante destacar que prácticamente no existen este tipo de servicios a través de Internet.
Esto convierte a nuestro producto en un desarrollo sumamente novedoso a nivel mundial.

Por otra parte en este documento se desarrollo una solución completa al problema SVRP (Stochastic Vehicle Routing Problem) y la implementación de la misma.

Vista General del Resto del Documento

A continuación se explicará brevemente la organización del Informe.
Este primer capítulo es una introducción al Problema de Ruteo de Vehículos, situando los objetivos del documento, el contexto y los principales logros del mismo.
El segundo y tercer capitulo se especifican y analizan los requerimientos del usuario.
En el capítulo 4 se trata la implementación desarrollada por la tesis de 1999 para resolver parte de los problemas planteados y su adaptabilidad a los nuevos requerimientos.
En el capítulo 5 se presenta un desarrollo teórico sobre la problemática que encierran los algoritmos de ruteo (también se incluye el apéndice Definición de Problemas, donde se entra en mas detalle a los aspectos formales), en este capitulo también se incluye un estudio completo y riguroso del problema SVRP.
Los capítulos 6 y 7describen el diseño e implementación del proyecto, comenzando por la librería de ruteo (capitulo 6) y posteriormente el sistema en general (capitulo 7).
En el capítulo 8 se presenta un plan de testeo genérico para un software desarrollado con los recursos con que se cuenta para un Proyecto Final, y luego, más en particular, un plan de testeo para la estructura desarrollada, resultados del mismo y tiempos de corridas.
El capítulo 9 presenta las conclusiones y principales logros del actual trabajo, así como líneas por las que podría continuarse con el mismo, ya sea para ampliarlo o para mejorarlo.

Capítulo 2

Requerimientos de Usuario

Requerimientos de Usuario

Documento de Requerimientos

Introducción

El objetivo del presente taller es responder en forma eficiente a la solicitud del área de optimización de la empresa ICA -Ingenieros Consultores Asociados- respecto a soluciones para el problema de ruteo. Por tanto la propiedad intelectual del mismo pertenece exclusivamente a dicha empresa.
La comunicación con dicha empresa es a través de su representante el Ingeniero Leonardo Loureiro, gerente de Marketing, quién planteo los requerimientos del taller, y con nuestro usuario el Ingeniero Luis Alberto Calderón.

El Documento de Requerimientos refleja qué problemática específica se debe tratar y de qué modo. Este documento es elaborado a partir de las reuniones con el usuario (ver apéndice de Control de Reuniones).

La conclusión y síntesis de este capitulo, “Documento de Requerimientos” , se encuentra en el capitulo siguiente: “Análisis de Requerimientos”, el cual es realizado completamente por el grupo de Taller V, con la posterior aprobación del usuario.

Problemática

Se pretende desarrollar un prototipo de software genérico S.A.R. (Sistema Automatizado de Ruteo) a través de Internet que resuelva la mayor cantidad posible de problemas de ruteo, logrando de esta forma un paquete genérico, y que permita además una buena interacción con el usuario final.
Para lograr completamente estos objetivos se desea además integrar la resolución de estos problemas a un SIG (Sistema de Información Geográfica)

Soluciones Existentes

Actualmente se cuenta con la solución ofrecida por el Taller 99 "Ruteo de Vehículos Genérico", (ver apéndice de Definición de Problemas) el cual soluciona de forma parcial los siguientes problemas:

- TSP
- MTSP
- VRP
- MDVRP
- VRP con ventanas de tiempo (VRPTW)

- MDVRP con ventanas de tiempo (MDVRPTW)
- VRP para flota heterogénea (VRPHF)
- CPP
- DARP

Se parte pues de dicho proyecto, cubriendo y ampliando las funcionalidades que este ofrece, no descartando una reingeniería del mismo.

Requerimientos

Funcionalidades de Ruteo

Se piden, además de las ofrecidas por el proyecto anterior, las siguientes funcionalidades de ruteo:

- SVRP. Demanda probabilística de los clientes, donde todos los clientes tienen la misma distribución de la demanda. Nos limitaremos al caso de oferta determinística por parte del deposito.

Funcionalidades geográficas

Se desea que el software pueda correr para usuarios de diferentes ciudades o zonas. Estas zonas son representadas por mapas diferentes, por lo que se debe lograr independencia del mapa que se utilice.
Debe permitir además determinadas funciones sobre el mapa, como moverse sobre él, hacer zoom en cierta región o realizar consultas sobre datos de una zona (por ejemplo ver una ruta por su nombre).
Se asumirá que el punto que representa al cliente sobre el mapa ya existe, obviando el problema de address matching (el problema de address matching se estudia con mas detenimiento en el apéndice de Geocodificación).
Se impone como restricción que los datos que se van a usar para manejar los mapas tienen que estar en formato Shapefile.

Funcionalidades extra

El software debe tener la capacidad de reconocer que tipo de problemas debe resolver, y resolverlo, a partir del conjunto de datos de entrada que recibe.
Debe retroalimentarse a si mismo, esto es, permitir su actualización a través de la contraposición de datos extraídos de la realidad e ingresados al software contra los calculado. Esto va a permitir un control y una optimización del producto.
Esta funcionalidad no se ofrece.

Interfase gráfica

Se debe realizar una interfase gráfica para la entrada y salida de datos.
Esta debe permitir ingresar los datos seleccionando los archivos a utilizar, visualizar gráficamente los datos ingresados y visualizar la solución obtenida.
Debe presentar funcionalidades de reportes para la salida de los resultados.

Configuración del producto

Se debe tener una configuración inicial a la hora de la instalación a través de un código. Este servirá para establecer cuales de todas las soluciones que el software brinda va a tener acceso el usuario. A partir de ese código el producto solo ofrecerá solución a determinados problemas de ruteo.

Durante la configuración se debe establecer manualmente la relación entre los campos que utiliza el producto y los nombres bajo los cuales el usuario tiene almacenada la información requerida. Esta información va a ser referida tanto a, por ejemplo, vehículos o pedidos, como a los campos asociados a los mapas. No existen restricciones con respecto a los nombres de las tablas originales. (Ver el apéndice F: Reporte de incidencia).

Manejo de los datos de entrada y de salida

Para administrar los datos de los vehículos, se debe manejar una BD propia o establecer una interfase con una ya existente.
El software debe devolver reportes. Se imprimirá un reporte con la ruta correspondiente a cada vehículo. Además un reporte por cada parada (cliente), con ruta hasta esa parada, hora de llegada, etc.

Modularizacíon

Para resolver los distintos problemas, se deben dividir las soluciones ofrecidas en distintos módulos (.dll). (Ver el apéndice F: Reporte de incidencia).
Dichas .dll deben tener parámetros ocultos.

Aplicación en Internet

Es un requerimiento del Taller el desarrollo o mapeo de la aplicación a Internet.

Herramientas

Se determina que las herramientas a utilizar serán Visual Basic, Visual C++, Visual InterDev, MapObject 2.0 (proporcionado por el usuario), MapObject IMS (proporcionado por el usuario, para la implementación en Internet)
Para implementar funcionalidades que trabajen sobre el mapa (por ejemplo hacer zoom, o moverse sobre él), se recomienda usar Moview de MapObject.
Cada una de las herramientas fue sugerida por las facilidades que brinda para implementar las diferentes partes que incluye el proyecto:
Los datos geográficos serán representados utilizando la herramienta MapObjects, desarrollada para ello.
La interfase gráfica se desarrollará en Visual Basic dado que permite óptimos resultados visuales y gran interacción con MapObjects.
El motor de la base de datos y la resolución de los distintos algoritmos se hará en Visual C++.
El mapeo de la aplicación a Internet se realizará a través de MapObjects IMS.

Investigación

Investigar en el área de “Address Matching”. Si bien en este proyecto se asume esta problemática como resuelta, esta es un área muy importante y poco resuelta relacionada con el tema de VRP, que puede llegar a constituir una limitante para nuestro trabajo.

Capítulo
3 Análisis de Requerimientos

Análisis de Requerimientos

Este capitulo es la conclusión y síntesis del capitulo anterior, "Documento de Requerimientos", el mismo es realizado por parte de los integrantes del proyecto con la posterior aprobación del usuario.
Este Análisis es realizado al inicio del proyecto, cuando el objetivo final era una aplicación de escritorio, previo a la decisión por parte de los usuario de desarrollar un sitio Web.

Introducción

Propósito

El propósito del sistema es poder lograr en forma fácil e intuitiva (y también eficiente) el Ruteo de Vehículos entre depósitos y clientes, diminuyendo el costo total del traslado, sin violar las restricciones impuestas por el usuario.
Dado el alto grado de información utilizada para dicha optimización, es indispensable no sólo manipular de forma eficiente dichos datos, sino actualizar bajo la forma de realimentación dichos datos, para tener un sistema dinámico y realista.

Alcance

El alcance del producto se enmarcó en dos categorías principales: ***básicos*** y ***especiales*** [11]***.***
Donde los ***básicos*** son aquellos de los cuales el sistema no puede carecer y los ***especiales*** son extras que implican mejoras del producto.

Básico

- Calcular rutas de forma aproximada – El sistema debe poder calcular rutas de bajo costo.
- Manipulación intuitiva de los datos – La información que se debe desplegar debe ser lo suficientemente simple como para que un usuario no especializado (aunque interiorizado en el tema) pueda comprenderla.
- Retroalimentación – El sistema debe ofrecer la posibilidad de aprender y actualizar sus costos en función de la experiencia.
- Instalación a medida – Ofrecer distintas licencias del producto, que limiten de distinta forma las funcionalidades del sistema, dependiendo de las necesidades del cliente.

Especial

- Internet – Una aplicación para Internet que proporcione parte de las funcionalidades del producto.
- Performance – Proporcionar respuestas rápidas al calculo de rutas.
- Actualización simple – Capacidad de actualizar la información accesible por el producto a medida que ésta se actualiza en la base de datos original del usuario.
- Address Matching – Proporcionar una codificación geográfica aceptable para los datos.
- Portabilidad (distintas BD, plataforma) – Debería poder partir de cualquier tipo de base de datos ODBC.

Definiciones, acrónimos y abreviaciones

Ruteo – Hallar una ruta.
Address Matching – asignar una coordenada geográfica a una dirección natural (ver el apéndice de Geocodificación)
Shapefile – Tipo de archivo que encierra datos geográficos.
Empresa – Entidad que utiliza el sistema.

Referencias

- Este documento se basa en la norma IEEE 830 para especificación de requerimientos.
- Además se utilizó el formato presentado por Jacobson para el Análisis de requerimientos como agregado para la realización de este documento.
- Estándar UML.

Vista general

Este documento describe en forma general las funcionalidades del producto, así como los actores, y objetos del dominio del problema que forman parte del sistema. Además de esto, se analiza también las restricciones externas que pesan sobre el producto.

Descripción general

Perspectiva del Producto

A continuación se presentarán los productos de los que depende el programa y con los cuales interactuará en algún momento.

El producto depende de:

- El sistema operativo Windows 2000 Advanced Server^{MR} o superior.
- Un estándar ODBC para el acceso a los datos.
- Un software (o similar) capaz de realizar el address matching.
- Mapobject 2.0 y Mapobject IMS.

El producto podría interactuar con:

- Una base de datos del cliente para tomar alguno de los datos utilizados por el producto.
- Una impresora para la presentación de reportes.
- Navegador web.

En el caso de los cuatro primeros, los productos son indispensables para la correcta ejecución del programa, y están en un nivel jerárquico inferior al del producto, ya que éste depende directamente de éstos para su funcionamiento.

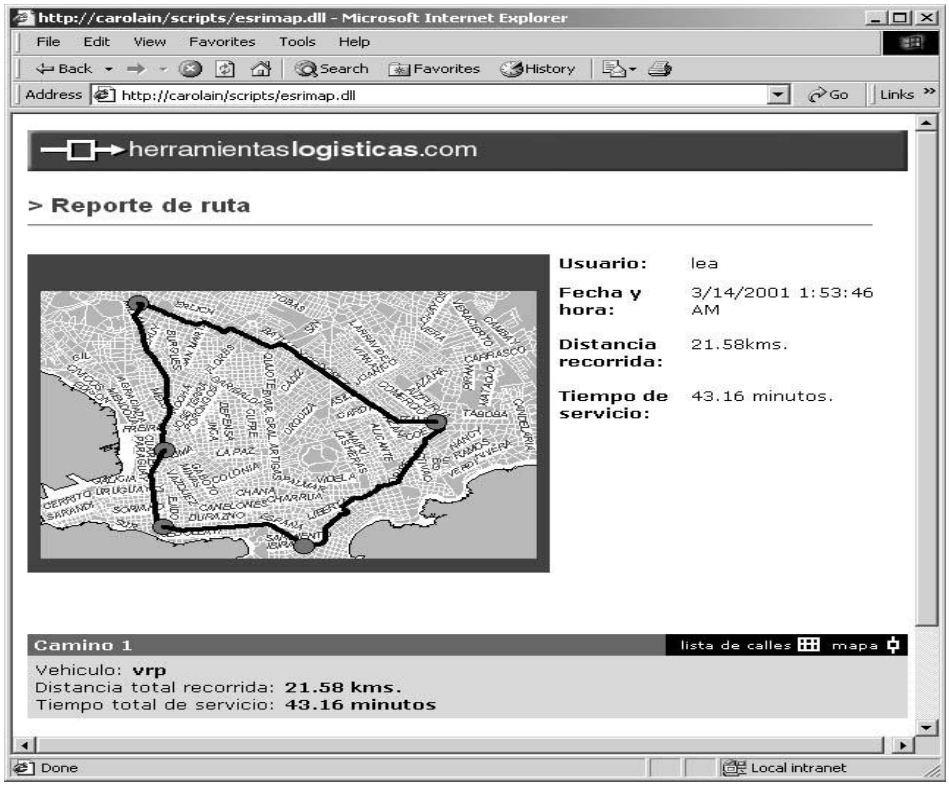
En los tres últimos casos los productos con los que interactuaría estarían en un nivel jerárquico superior, ya que no son indispensables.

Requerimientos de Hardware

Cualquier computador que soporte un entorno Windows.

Ejemplo de interfaz visual

A continuación se presenta una posible interfaz visual del producto, aunque su complejidad excede las expectativas del proyecto.



Funcionalidades

Los requerimientos funcionales del producto se presentan agrupados en forma jerárquica, en función de su similitud, para un mejor entendimiento de los mismos.
Además se presenta una clasificación de dichos requerimientos según: ***Oculta, Evidente*** y ***Opcional.***

1 Mapas y datos geográficos.

1.1 Elegir mapas de calles.

En el mapa de calles se encuentra información imprescindible para el funcionamiento del sistema. Por lo tanto, contar con buena información geográfica es una restricción importante del producto.

La elección del mapa de calles se realiza seleccionando el archivo shapefile correspondiente.

Evidente

1.2 Seteo rápido de propiedades de calles.

Considerando la carencia de datos posibles en la zona donde va a ser utilizado el producto. Para que éste no sea obsoleto se debe tener un seteo rápido de las propiedades imprescindibles para el funcionamiento del sistema, como tiempos medios de recorrida de calles, etc.

Esta funcionalidad tiene como principal motivo que el producto funcione, siendo vital su corta duración, es decir asumirá gran cantidad de información.

Vale aclarar que la intención siempre debe ser la de no utilizar esta funcionalidad, y que los datos sean proporcionados por un proveedor de datos geográficos.

Opcional.

1.3 Realimentar los datos de calles.

Esta funcionalidad es clave para un buen desempeño del producto.

El sistema debe poder utilizar los datos arrojados por una ruta realizada, como por ejemplo el tiempo de traslado, para actualizar los datos geográficos.

Oculta

1.4 Editar propiedades de calles y cruces.

La funcionalidad 1.2 también puede ser realizada de forma explicita por el usuario mediante la edición y modificación de las propiedades de las calles e intersecciones.

Se pretende que esta funcionalidad sea poco utilizada puesto que es tediosa.

Un ejemplo de propiedad de calles es el limite de velocidad.

Evidente

1.5 Realizar el AddressMatching.

El address matching o geocodificación es la determinación de las coordenadas geográficas X,Y a partir de una dirección proporcionada por el usuario. Dicha asignación debe ser realizado para los clientes y depósitos del usuario.

La geocodificación no es un problema de simple solución, y escapa al alcance de este sistema, por lo que se supondrá que está dada.

Es importante señalar que esta funcionalidad es una limitante para la utilización del producto.

Opcional.

2 Vehículos

Los vehículos del sistema son aquellos utilizados por la empresa para cubrir la demanda de los clientes, es decir, son aquellos que van a ser ruteados.

2.1 Agregar un vehículo

Agrega un vehículo al sistema, configurando todas las propiedades relevantes del mismo. Dichas propiedades son de vital importancia, porque son utilizadas para el calculo de la ruta seudo optima.

Por supuesto estas propiedades pueden ser modificadas (ver 2.4).

Evidente

2.2 Borrar un vehículo

Todo vehículo puede ser borrado del sistema. Esto debe realizarse cuando el mismo no va a utilizarse mas.

Es importante borrar un vehículo que no va a ser utilizado mas, puesto que sino puede ser considerado, por error, en los cálculos de optimización.

Evidente

2.3 Mostrar las asignaciones realizadas a un vehículo

Luego de calcular la ruta para un vehículo se puede ver, la carga que trasladará el mismo, la ventana de tiempo de funcionamiento esperada del mismo, etc.

Además se presentaran las propiedades estáticas del vehículo (propiedades que no depende de la ruta asignada), como por ejemplo el nombre del conductor.

Evidente

2.4 Editar las propiedades de un vehículo

Luego de ser agregado un vehículo, podemos editar y modificar sus propiedades.

Evidente

3 Clientes

Esta funcionalidad permite determinar los clientes de la empresa, si la empresa ya tiene un software que maneja los clientes en una base de datos es conveniente utilizar la funcionalidad de importación [3.4].

3.1 Agregar un cliente

Al agregar un cliente se comienza determinando su nombre, dirección (*), teléfono, fax, mail, etc. Luego se determinan otros datos como la ventana de tiempo admisible (esta ventana es la ventana por defecto para toda nueva orden del cliente, pero puede ser modificada en cada orden en particular), comentarios, etc..

(*) Es importante recalcar que esta funcionalidad debe utilizar la funcionalidad de geocodificación [1.5], para establecer la ubicación geográfica del cliente.

Evidente

3.2 Editar un cliente

Luego de ser agregado un cliente, podemos editar y modificar sus propiedades.

Evidente

3.3 Borrar un cliente

Todo cliente puede ser borrado del sistema. Esto debe realizarse cuando sabemos con certeza que el mismo no realizará mas ordenes.

Evidente

3.4 Importar clientes

El sistema permite la importación de clientes de otras bases de datos, para que dichas bases puedan ser importadas debe estar declarado el driver ODBC correspondiente. El formato ODBC es aceptado por la mayoría de las bases de datos comerciales, como Microsoft Access (.mdb).

3.4.1 Determinar un nuevo perfil de importacion de clientes

La intención de esta funcionalidad es establecer la relación entre la base de datos del sistema y la base de datos donde están los clientes a importar. Supuestamente esta funcionalidad debe ser utilizada solo una vez, en la instalación del producto.

Básicamente los pasos a seguir en la misma serán los siguientes: se comenzará especificando la base de datos (directorio y nombre) a ser importada, luego se seleccionará la tabla donde están los clientes, y finalmente se realizará la correspondencia entre los campos asociados.

Opcional.

3.4.2 Editar el perfil de importacion de clientes

Considerando eventuales cambios en el software de la empresa, puede ser necesario modificar el perfil de importación.
Opcional.

3.4.3 Importar clientes usando el perfil

La importación de los clientes se realiza con esta funcionalidad. Dado un perfil de importación, simplemente esta funcionalidad actualiza los clientes del sistema.
Es importante recalcar que la geocodificación es realizada por la funcionalidad [1.5] y que posiblemente deba realizarse manualmente para cada cliente importado.
Opcional.

4 Ordenes

Las ordenes representan las demandas de los clientes. En primera instancia las ordenes solo son ingresadas en el sistema, y no son satisfechas hasta que son agregadas a alguna "carpeta de ordenes" (ver funcionalidades 5).

4.1 Determinar ordenes manualmente

Esta funcionalidad permite el poder determinar las ordenes de un cliente manualmente, si la empresa ya tiene un software que maneja las ordenes de los clientes en una base de datos es conveniente utilizar la funcionalidad de importación [4.2].

4.1.1 Agregar una orden

Al agregar una orden se comienza determinando a que cliente esta dirigida. Luego se determinan otros datos como la ventana de tiempo admisible (TW), el tiempo de atención esperado, el peso (o volumen) de la orden para cada producto enviado, deadline de la orden, comentarios, etc..
Como observación importante es que si el cliente no ha sido previamente ingresado, éste se podrá agregar desde esta funcionalidad llamando a la funcionalidad [3.1].
Evidente

4.1.2 Editar una orden

Luego de ser agregada una orden, podemos editar y modificar sus propiedades.
Evidente

4.1.3 Borrar una orden

Toda orden puede ser borrada del sistema. Esto debe realizarse cuando la misma no va a ser llevada a cabo.
Es importante borrar una orden que no va a ser realizada, puesto que sino será considerada en los cálculos de optimización.
Evidente

4.2 Importar ordenes

El sistema permite la importación de ordenes de clientes de otras bases de datos, para que dichas bases puedan ser importadas debe estar declarado el driver ODBC correspondiente. El formato ODBC es aceptado por la mayoría de las bases de datos comerciales, como Microsoft Access (.mdb).

4.2.1 Determinar un nuevo perfil de importacion de ordenes

La intención de esta funcionalidad es establecer la relación entre la base de datos del sistema y la base de datos donde están las ordenes a

importar. Supuestamente esta funcionalidad debe ser utilizada solo una vez, en la instalación del producto.
Básicamente los pasos a seguir en la misma serán los siguientes: se comenzará especificando la base de datos (directorio y nombre) a ser importada, luego se seleccionará la tabla donde están las ordenes, y finalmente los campos asociados.
Evidente

4.2.2 Editar el perfil de importacion de ordenes

Considerando eventuales cambios en el software de la empresa, puede ser necesario modificar el perfil de importación.
Opcional.

4.2.3 Importar ordenes usando el perfil

La importación de las ordenes se realiza con esta funcionalidad. Dado un perfil de importación, simplemente esta funcionalidad actualiza las ordenes utilizada por el sistema.
Evidente

4.3 Satisfacer una orden

Para satisfacer una orden, ésta debe encontrarse dentro de alguna "carpeta de ordenes". Este ultimo es el conjunto de ordenes que son consideradas para el calculo de las rutas optimas, como se describe en [5].

4.3.1 Asignar una orden a una carpeta de ordenes

La asignación de las ordenes se realiza con esta funcionalidad. El administrador debe asignar a la misma carpeta todas las ordenes a ser ruteadas al mismo tiempo(las carpetas de ordenes se describe a continuación).
Lógicamente existe la función inversa, que desasigna una orden, para poderla asignar a otra carpeta (ver funcionalidad [5.5])
Evidente

4.3.2 Mostrar las ordenes no asignadas

Para simplificar la elección de las ordenes a incluir en una carpeta de orden dada, debe ser posible poder observar solo las ordenes no asignadas. Además es de utilidad que aparezcan resaltadas las ordenes que se encuentran en su deadline (ultimo día para ser satisfecha).
Opcional.

4.4 Buscar una orden

Puede ser necesario encontrar una orden dada, para ello se puede buscar un texto determinado entre todas las ordenes del sistema.
Opcional.

5 Carpetas de ordenes

Las carpetas de ordenes tienen una importancia significativa en el sistema, puesto que para cada una de ellas se realiza el calculo de las rutas.
El administrador debe utilizar las carpetas de ordenes de acuerdo a los requerimientos de la empresa, donde una carpeta de ordenes puede representar un día de la semana (si todos los días se realizan recorridos), un periodo de tiempo arbitrario, una zona, etc.

5.1 Crear una carpeta de ordenes

Agrega una carpeta de ordenes al sistema, configurando todas las propiedades relevantes de la misma. Dichas propiedades son el nombre (en general la fecha de realización) y los vehículos permitidos para ruteo.
Por supuesto estas propiedades pueden ser modificadas (ver [5.2]).
Evidente

5.2 Editar las propiedades de una carpeta de ordenes

Luego de ser creada una carpeta, podemos editar y modificar sus propiedades.
Opcional

5.3 Borrar una carpeta de ordenes

Una carpeta puede ser borrada del sistema. Esto debe realizarse cuando la misma no va a utilizarse mas o no hay interés en conservarla.
Opcional

5.4 Mostrar las asignaciones de ordenes realizadas a una carpeta

Siempre es posible ver las ordenes presentes en una carpeta de ordenes, puesto que son estas las que van a ser consideradas en las rutas.
Evidente

5.5 Modificar las asignaciones de ordenes realizadas a una carpeta

Agregar (con [4.3.1]) y quitar una orden presentes en una carpeta de ordenes es imprescindible para el funcionamiento ágil e intuitivo del producto.
Evidente

5.6 Exportar una carpeta de ordenes y sus rutas

Para tener un respaldo de estas carpetas, o para poderlos utilizar en otra aplicación se plantea la necesidad de exportarlas como base de datos, shapefiles, u otro formato útil.

Al exportar una carpeta de ordenes además se esta exportando las rutas utilizadas en la misma, por lo que es una información valiosísima para futuros análisis de la empresa.
Opcional.

6 Rutas

La función de rutear, es la funcionalidad mas importante del producto, y depende de la misma el éxito del sistema.

Para cada carpeta de ordenes se presenta un conjunto de rutas asociadas, que cubren la demanda de todas las ordenes.

6.1 Calcular las rutas para una carpeta de ordenes

Esta funcionalidad calcula las rutas optimas para satisfacer las ordenes de la carpeta, utilizando los vehículos destinados para la misma.

Pueden aparecer problemas que hagan imposible resolver las rutas para una carpeta de ordenes dada. Como ejemplo podemos citar: capacidad (volumen o peso) superada, superar el tiempo que los vehículos tiene disponibles (considerando los tiempos de atención), etc. Para solucionar estos problemas siempre se pueden quitar ordenes de la carpeta, o agregar mas vehículos a ser ruteados.

Previo al calculo de las rutas para una carpeta de ordenes, es necesario determinar el tipo de problema a optimizar, por ejemplo: TSP, VRP, SVRP, etc. (ver apéndice F); para esto se utiliza la funcionalidad [6.2].
Evidente.

6.2 Determinar el tipo de problema de ruteo

Previo al calculo de las rutas para una carpeta de ordenes dada, es necesario determinar el tipo de problema a optimizar. Con este sistema es posible resolver los siguientes problemas: TSP, MTSP, VRP, MDVRP, flota heterogénea (VRPHF), VRPTW, MDVRPTW, CPP, DARP, SVRP, etc. Por mas información ver el apéndice Definición de Problemas.
Oculta.

6.3 Asignar manualmente una orden a una ruta

Es posible elegir una orden (o un conjunto de ordenes) y asignarlas explícitamente a alguna ruta.
Opcional.

6.4 Quitar una orden de una ruta

Es posible quitar una orden (o un conjunto de ordenes) de una ruta. Esto puede realizarse de forma explicita (para asignar dicha orden a otra ruta usando la funcionalidad [6.3]), o implícitamente cuando se quita una orden de una carpeta de ordenes (usando la funcionalidad [4.3.1]).
Opcional.

6.5 Mejorar las rutas ya calculadas

Para poder optimizar las rutas ya calculadas podemos hacer un recalculo selectivo de las rutas optimas, permitiendo que algunas ordenes cambien de ruta o de orden, mientras que otras se mantengan fijas en la ruta actual.
Opcional.

7 Reportes

Los reportes desplegados son una parte muy importante del sistema, puesto que la comunicación de algunos actores con el sistema será solo a través de ellos.

7.1 Imprimir reporte para el despachador

Este reporte es utilizado por el despachador, en el se presenta un mapa de la ruta a realizar, las direcciones de los clientes a ser visitados, al igual que su demanda (carga y/o volumen), hora esperada de llegada, tiempo de ventana admisible (TW), etc.
Es importante incluir para cada cliente un espacio para comentarios para informar de cualquier eventualidad.
Evidente

7.2 Imprimir reporte para el administrador

Este reporte es utilizado por el administrador para analizar la eficiencia del sistema. En él se debe presentar un sumario (por ejemplo durante un mes) por vehículo donde se detallen el costo, la carga admisible, la cantidad de clientes visitados, la distancia recorrida, las horas de trabajo, el comportamiento de los tiempos de atención (clientes por hora, violaciones, etc), etc.
Evidente

7.3 Configuración de un nuevo reporte

Sin perder de vista que el sistema es un producto genérico, se puede presentar determinada realidad donde aparezca la necesidad de reportes particulares. Para esto se presenta esta funcionalidad, donde se puede crear un nuevo reporte a medida.
Opcional.

7.4 Exportar un reporte.

Para tener un respaldo de estos reportes, o para poderlos utilizar en otra aplicación se plantea la necesidad de exportarlos en varios formatos, como planilla electrónica, documento de texto, etc..
Opcional.

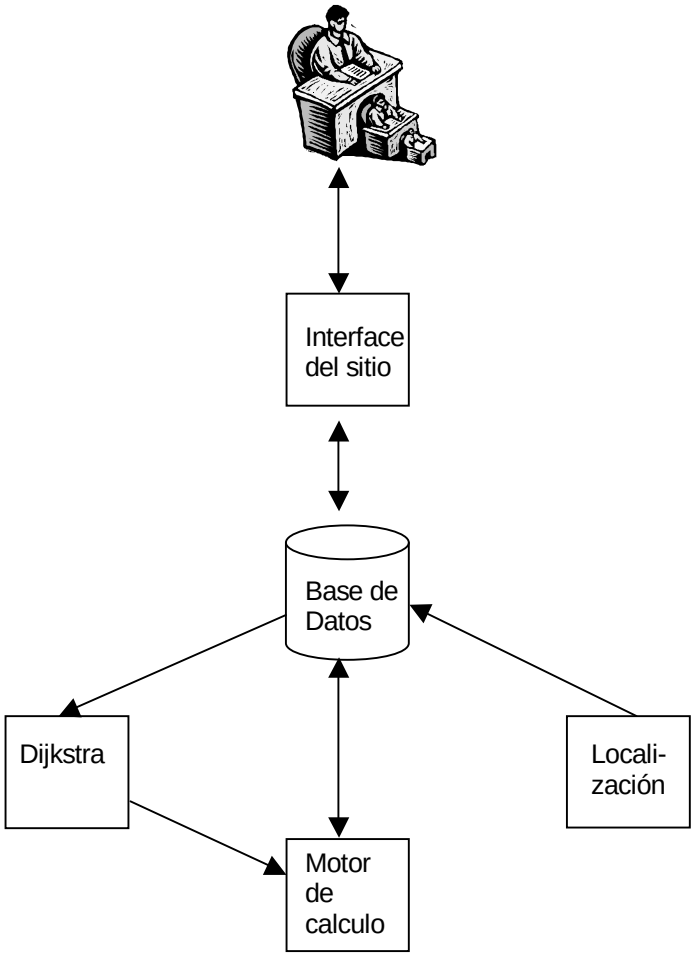
Las funcionalidades presentadas con anterioridad están sujetas a cambios en los requerimientos por parte del usuario. Dichos cambios se encuentran documentados en el apéndice F: Reportes de incidencias.

Actores

Usuarios finales – Son los actores primarios del producto, es decir, aquellos que actúan, a diario, directamente con las principales funcionalidades finales del producto. En este caso son quienes realizarán las consultas de rutas, piden reportes, etc.

Administrador – Es el actor secundario del sistema, o sea, aquel que supervisa y mantiene el producto. En este caso el actor secundario es aquella persona encargada de la instalación, lograr la mayor automatización posible en la realimentación; mientras que usuario final es el encargado de calcular, en sí, las rutas.

Dominio del Problema



Restricciones generales

- Plazo de Tiempo – El producto puede estar siendo lanzado por varias compañías, por lo que no se puede demorar un día más del plazo indicado.
- Costos – Suponiendo un financiamiento del producto por alguna entidad, seguramente ésta tenga un límite, por lo que el producto también está limitado a sus costos.
- Hardware – Las limitaciones de hardware, evidentemente, determinan un mejor desempeño del producto.
- Dificultad del proceso de Address Matching – Podría ser tan complejo el proceso de geocodificación, que imposibilite que éste se lleve a cabo por un proceso automatizado.

Asunciones y dependencias

- Se asume que el usuario tendrá un conocimiento mínimo sobre la información que se está manejando.
- Así también, se espera una mínima noción sobre manejo de interfaces visuales.
- El sistema debe correr sobre cualquier plataforma servidora de Microsoft, por ejemplo: Windows 2000 Server[MR].

Capítulo 4

Evaluación del Código Disponible

Evaluación del Código Disponible

El porqué de una evaluación

Como en el general de los casos, este proyecto no solo consta del trabajo específico para los objetivos propuestos, sino también de una serie de recursos disponibles de emprendimientos anteriores que hacen en gran parte al proyecto actual.
En nuestro caso particular, además de documentaciones e investigación previa, el principal recurso disponible es el trabajo codificado en sí mismo.
Se desprende de las propias características del proyecto la capacidad de incidencia del código en el desarrollo del mismo.
Es por esto que se decide realizar una evaluación minuciosa del mismo a los efectos de una mejor proyección y desarrollo del proyecto.

Evaluación

Introducción

Cualquier codificación es evaluable desde una amplia serie de características, pero a los efectos de la utilidad de dicho código en el proyecto actual, resumiremos esta evaluación entorno a las siguientes:

- Legibilidad
- Modularidad
- Correctitud
- Eficiencia
- Genericidad

De las características aquí planteadas se desprenden otras de igual importancia como la extensibilidad, adaptabilidad, robustez, etc.

Legibilidad

El código carece de reglas generales que favorezcan la legibilidad del mismo. Desde reglas de indentación hasta reglas de nomenclatura.
Más allá de las reglas, carece de prolijidad alguna. Son comunes las apariciones de 40 a 50 líneas corridas de código que resultan estar marcadas como comentarios, así

como también indentaciones en escalera que hacen imposible una lectura continua del mismo.

Modularidad

La modularidad del código es dispar. Mientras algunas características están correctamente modularizadas (ruta, grafo), otras son todo lo contrario. Son comunes las declaraciones de tipos de datos (no clases) que no solo son para uso interno de una clase, sino que se exportan a módulos externos tanto como tipo en sí así como resultado o parámetro de algún método.
Esto implica la perdida total de independencia del código.

Ejemplo:

```
(CODIGO) LisCli devolverClientes( .......)
```

Al hacer un llamado a dicha función se recibe la variable (no objeto) de tipo LisCli. Ahora, al acceder a dicha variable (suponiendo, para conocer al primer cliente de la misma), surge, entre otros, el siguiente problema:
Ya que LisCli no dispone de ninguna funcionalidad definida ¿Cómo hago para acceder al primer cliente de la lista?. No existe más remedio que involucrarse con la estructura interna del tipo LisCli, por lo que se debe acceder de la forma:

```
(CODIGO) LisCli->sig.idCliente
```

Aunque hubiera sido mucho más intuitivo, e independiente,

```
(CODIGO) LisCli.getPrimerCliente().getIdCliente()
```

Correctitud

Entendiendo por correctitud la cualidad del código que "mide" en que medida el software responde a los objetivos funcionales requeridos, evaluamos que el código es básicamente, de acuerdo a los requerimientos, no correcto.

Por más que el código resuelva los problemas planteados, se pierde de vista una de las características más importantes requeridas (sino la más importante) que es la genericidad del mismo.

Esto implica que el código (o mejor dicho, la librería creada) NO puede ser utilizada desde otras aplicaciones que no sea la presentada con el mismo.

Eficiencia

Quizás a expensas de las demás características consideradas "deseables" para el código, se encuentra que el mismo es satisfactoriamente eficiente.

En comparación con otros proyectos realizados se encuentra que, a modo de ejemplo, el problema del TSP (Traveling Salesman Problem) para 20 nodos se resuelve en un 18% del tiempo total consumido por el proceso en dicho proyecto.

Genericidad

Dados los requerimientos puntuales del código, nos referiremos a genericidad como el grado de independencia que mantiene dicho código con la aplicación sobre la cual se ejecuta. A continuación detallaremos puntos en los cuales esta independencia es poco clara.

- El código resuelve los problemas dado un particular formato de archivos de entrada. De la transformación de los datos geográficos a dichos formatos se debe encargar el software anfitrión.
- Se delega la responsabilidad del control de identificadores (tanto de nodos como aristas) al anfitrión. Al implementarse las estructuras como un array indexado por dichos identificadores, el código es muy susceptible a las características de los mismos. Como ejemplo, se asume que los identificadores de las aristas comienzan en 1 y culminan con el número total de ellas.
- El resultado del código consta de información apenas aprovechable. Esta información consiste en un archivo en formato texto donde solo se dispone de la información necesaria para "pintar" las rutas en el gráfico (específicamente, consiste de una secuencia de identificadores de esquinas).
- Se hace un mal uso de la memoria RAM, donde no se libera toda la memoria utilizada. Por lo tanto no es posible utilizar dicho código de forma sostenida en una aplicación cualquiera e imposible en un sistema operativo servidor.

Conclusiones

De la evaluación de desprende una clara confusión entre producto reusable y producto final. Dada las características del código se nota claramente que, en determinado momento de su desarrollo, el producto reusable se desvirtuó a modo de cumplir los requisitos mínimos impuestos por la aplicación anfitrión y no los requerimientos del usuario.
En lo que respecta a la utilización de dicho código, se puede prever una mayor dedicación a la esperada de acuerdo a la especificación del proyecto.
Cabe aclarar que hubiera sido ampliamente productivo un estudio previo más minucioso de el producto en cuestión a modo de una mejor especificación del proyecto.

Capítulo
5

Desarrollo Teórico

Desarrollo Teórico

Introducción

En éste capítulo se buscan las soluciones teóricas a los problemas propuestos, las cuales serán la justificación y la base para su posterior implementación.
Previo a dicho desarrollo, se hace una introducción genérica a la problemática del Ruteo de Vehículos. Por un estudio mas detallado en este punto se puede consultar el apéndice: Definición de Problemas.

Modelado

El Problema de Ruteo de Vehículos suele modelarse como un Grafo con costos en las aristas y/o en los nodos. En estos términos se está en condiciones de definir un problema básico de Ruteo de Vehículos. Se tiene un conjunto de nodos y/o arcos que deben ser servidos por una flota de vehículos. No hay restricciones sobre cuándo o en qué orden estas entidades deben ser servidas. El problema es construir un conjunto factible de rutas de bajo costo (una para cada vehículo).
Distintas variantes de las restricciones se presentan a este modelo básico.

Complejidad

Problemas NP-Duros

Una consideración importante en la formulación y solución de problemas de ruteo es el esfuerzo computacional asociado a las técnicas de solución para estos problemas. Este esfuerzo computacional crece en forma exponencial con el tamaño del problema. Si este crecimiento es demasiado rápido, el esfuerzo computacional se vuelve prohibitivo aun para problemas pequeños limitando la aplicación de una técnica de solución. Un algoritmo polinomialmente acotado para un problema es un procedimiento cuyo esfuerzo computacional crece polinomialmente con el tamaño del problema en el peor caso. La clase de problemas para los cuales se conocen algoritmos polinomialmente acotados de denomina P.
En contraste a la clase P, existe una gran clase de problemas en redes y combinatorios para los cuales aun no se conoce un algoritmo polinomailemtne

acotado. Esta clase de problemas se denomina NP-duros. Dado que la mayoría de los problemas de ruteo son NP-duros, los métodos para resolver estos problemas en forma optima tienen un crecimiento exponencial con el tamaño del problema. Enfrentados a un problema NP-duro, se usa algún método heuristico en la búsqueda de soluciones. Una heuristica es un procedimiento que usa la estructura del problema en forma matemática e intuitiva y brinda soluciones aceptables

Problemas de ruteo

Los problemas de ruteo son NP-Duros , por lo cual interesa tener buenas heurísticas que los resuelvan (o encontrar un algoritmo polinomialmente acotado que lo resuelva) y es importante conocer la complejidad de los algoritmos a utilizar.
La mayoría de los problemas de ruteo pueden ser formulados como problemas de redes; una medida del tamaño del problema puede ser la cantidad de nodos (y arcos) de la red correspondiente.

Problemas clásicos

Para la resolución de los problemas clásicos: TSP, MTSP, VRP, MDVRP, VRP con ventanas de tiempo (VRPTW), MDVRP con ventanas de tiempo (MDVRPTW), VRP para flota heterogénea (VRPHF), CPP, DARP se utilizan algoritmos elegidos por el usuario y tutor. Los mismos surgen de un análisis detallado realizado con anterioridad por la tesis de 1999.

Allí se eligió argumentando motivos de eficiencia y reusabilidad el algoritmo de los ahorros de Clark y Wright [2] para resolver los distintos tipos de problemas de ruteo. Por ejemplo, el algoritmo para solucionar el TSP es un caso particular del algoritmo para solucionar el VRP, que a su vez son casos particulares del MDVRP.

A continuación se transcribe dicho algoritmo:

El algoritmo, en el caso general, consta de los siguientes pasos:

1) *Dado un depósito y n clientes que serán servidos por ese depósito, crear un ciclo entre el depósito y el cliente para cada cliente.*
2) *Calcular los ahorros, esto es cuanto se ahorra al ir del depósito al cliente A, luego al cliente B y luego al depósito, en lugar de ir y volver a cada cliente desde el depósito.*
   Formalizando, $s_{ij} = c_{oi} + c_{oj} - c_{ij}$, siendo c_{ij} la distancia (o el costo) entre los nodos i y j. El nodo o representa el depósito.
3) *Ordenar los ahorros de mayor a menor.*
4) *Comenzando por el ahorro mayor, unir los nodos i y j correspondientes y quitar los arcos i-o y o-j. Repetir este paso hasta llegar a la solución.*

El algoritmo utilizado para calcular los caminos más cortos entre cada par de clientes es Dijkstra.

Los problemas clásicos se tratan básicamente de igual forma que en la tesis anterior, contemplando en este caso una generalidad mayor, pasando de un prototipo a una librería de gran porte.

Una de las modificaciones que es necesario realizarles es que acepten demandas no deterministas, como se detalla a continuación.

VRP Estocástico

Introducción

El SVRP (por sus siglas en ingles: Stochastic Vehicle Routing Problem) es una generalización del problema clásico de ruteo VRP, al que se le realizan las siguientes modificaciones:

- La demanda de los clientes es una variable aleatoria con una distribución de probabilidad conocida.
- Las rutas deben ser diseñadas antes de conocer la demanda real de cada cliente.

El objetivo del problema es minimizar el costo del viaje esperado, aunque se puede incurrir en ciertos tipos de costos altos si un cliente no es servido en un determinado viaje [2].

Un claro ejemplo de aplicación de este problema es la recolección de residuos.

En este capitulo desarrollamos una solución genérica para este tipo de problema. Primero se comienza por una especificación formal de problema, luego se realiza un desarrollo teórico, concluyendo en un método completo para este tipo de problemas.

Descripción Formal

Definiciones previas:

Sean:
Cij : costo de ir del nodo i al nodo j.

$$x_{\mathrm{i\,j\,k}} = \begin{cases} 1 \text{ Si el arco i - j se encuentra en la ruta de vehiculo k} \\ 0 \text{ En otro caso} \end{cases}$$

n : Numero de clientes
m: Numero de vehículos
Q_k : Es la capacidad del vehículo k
α : Es la máxima probabilidad de que alguna ruta falle.
d_i : Variable aleatoria que representa la demanda del cliente i, donde se conoce su función de distribución asociada

Problema:

Minimizar $\sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ijk}$

Sujeto a: $P\left(\sum_{i=1}^{n} \sum_{j=1}^{n} d_i x_{ijk} \le Q_k \right) \ge 1 - \alpha \qquad \forall k : 1..m$ Ec.1

$x = \{x_{ijk}\} \in S$ Para que la solución cumpla todas las demás restricciones del problema de ruteo.
(ver el apéndice de Definición de Problemas)

La Ec.1 representa que la probabilidad de obtener una ruta satisfactoria sea mayor a un umbral configurable. Que una ruta sea satisfactoria significa que la capacidad del vehículo asociado a la ruta, pueda satisfacer la demanda de los clientes de la misma. Básicamente en esta ecuación se concentra el concepto de demanda no determinista, y es la clave para resolver el problema.

Solución planteada

Idea preliminar:

La idea es transformar el SVRP a un VRP, y utilizar así los algoritmos y todo el estudio ya realizado para el caso de problemas deterministas. Esto se realizará transformado la Ec.1 en una restricción determinista, es decir, encontrando una capacidad artificial para los vehículos, la cual va a ser utilizada en el algoritmo de VRP (junto con una demanda artificial determinista de cada cliente), dicha capacidad será menor a la capacidad real de los vehículos y esta dada por el siguiente teorema.

Formalización

Teorema [2]
Hipótesis:

$\{a_i\}$ son variables aleatorias independientes e idénticamente distribuidas $\forall i \in \{0..n\}$.

$\{y_i\}$ son variables de decisión $y_i \in \{0,1\} \quad \forall i \in \{0..n\}$

$b, \alpha \in \Re$

$T \in \Re \quad \text{tal que} \quad P(r \le T) = 1 - \alpha$, con $r \approx N(0,1)$

Tesis:

Podemos aproximar la expresión no determinista $\boxed{P\left(\sum_{i=1}^{n} a_i y_i \le b\right) \ge 1-\alpha}$ por una expresión determinista (en general no lineal) $\boxed{T \le \frac{b - m_z}{s_z}}$.

Demostración

Sea $P\left(\sum_{i=1}^{n} a_i y_i \le b\right) \ge 1-\alpha$ donde $\{a_i\}$ son independientes e idénticamente distribuidos

$\{y_i\}$ son variables de decisión

$b, \alpha \in \Re$

Debido a que las variables aleatorias $\{a_i\}$ son variables aleatorias idénticamente distribuidas podemos definir su esperanza y varianza de forma independiente de i :

$$m_a = E\{a_i\}$$

$$s_a{}^2 = E\{(a_i - m_a)^2\} = E\{a_i{}^2\} - m_a{}^2$$

Ahora definiremos la variable aleatoria $z = \sum_{i=1}^{n} a_i y_i$ donde por linealidad del valor esperado tenemos que :

$$m_z = E\left\{\sum_{i=1}^{n} a_i y_i\right\} = \sum_{i=1}^{n} y_i E\{a_i\} = \sum_{i=1}^{n} y_i m_a = m_a \sum_{i=1}^{n} y_i$$

$$s_z{}^2 = E\left\{\left(\sum_{i=1}^{n} a_i y_i - m_z\right)^2\right\} = E\left\{\left(\sum_{i=1}^{n} a_i y_i\right)^2\right\} - m_z{}^2 = E\left\{\left(\sum_{i=1}^{n} a_i y_i\right)\left(\sum_{j=1}^{n} a_j y_j\right)\right\} - m_z{}^2 =$$

$$= E\left\{\sum_{i,j=1}^{n} a_i y_i a_j y_j\right\} - m_z{}^2 = \sum_{i,j=1}^{n} y_i y_j E\{a_i a_j\} - m_z{}^2 = \sum_{\substack{i,j=1 \\ i\neq j}}^{n} y_i y_j E\{a_i a_j\} + \sum_{k=1}^{n} y_k{}^2 E\{a_k{}^2\} - m_z{}^2 =$$

$$= \sum_{\substack{i,j=1 \\ i\neq j}}^{n} y_i y_j E\{a_i\} E\{a_j\} + \sum_{k=1}^{n} y_k{}^2 \left(m_a{}^2 + s_a{}^2\right) - m_z{}^2 = \sum_{\substack{i,j=1 \\ i\neq j}}^{n} y_i y_j m_a{}^2 + \sum_{k=1}^{n} y_k{}^2 m_a{}^2 + \sum_{k=1}^{n} y_k{}^2 s_a{}^2 - m_z{}^2 =$$

$$= \sum_{i,j=1}^{n} y_i y_j m_a{}^2 + \sum_{k=1}^{n} y_k{}^2 s_a{}^2 - m_z{}^2 = \left(\sum_{i=1}^{n} y_i m_a\right)\left(\sum_{j=1}^{n} y_j m_a\right) + \sum_{k=1}^{n} y_k{}^2 s_a{}^2 - m_z{}^2 = m_z{}^2 + \sum_{k=1}^{n} y_k{}^2 s_a{}^2 - m_z{}^2 =$$

$$= \sum_{k=1}^{n} y_k{}^2 s_a{}^2 = s_a{}^2 \sum_{k=1}^{n} y_k{}^2$$

Volviendo a nuestra ecuación $P\left(\sum_{i=1}^{n} a_i y_i \le b\right) \ge 1-\alpha$ mediante simples transformaciones lineales encontramos una ecuación equivalente

$$P\left(\frac{\left(\sum_{i=1}^{n} a_i y_i\right) - m_z}{s_z} \le \frac{b - m_z}{s_z}\right) \ge 1-\alpha$$

Donde si utilizamos el teorema central del limite, podemos aproximar la variable aleatoria $r = \dfrac{\left(\sum_{i=1}^{n} a_i y_i\right) - m_z}{s_z}$ por una de distribución normal $r \approx N(0,1)$

Y si llamamos T al real, que cumple que $P(r \le T) = 1-\alpha$ considerando $r \approx N(0,1)$, entonces, por monotonía de la densidad de probabilidad, sabemos que cuando se cumple la expresión determinista $T \le \dfrac{b - m_z}{s_z}$ se cumple la inicial : $P\left(\sum_{i=1}^{n} a_i y_i \le b\right) \ge 1-\alpha$. Obteniendo la tesis del teorema.

Teorema Central del limite. [9]

Si las variables aleatorias $x_1, x_2, .., x_n, ...$ son independientes equidistribuidas, con esperanza *m* y varianza s^2 , entonces para cada x se cumple :

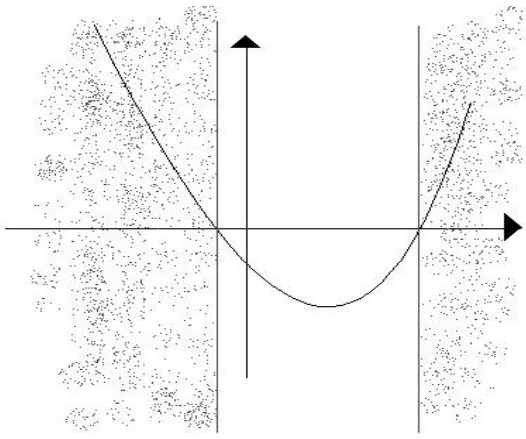
$$\lim_{n \to \infty} P\left(\frac{x_1 + x_2 + ... + x_n - nm}{\sqrt{n}s} \leq X \right) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{X} e^{-t^2/2} dt$$

Suposición adicional

Lamentablemente la expresión $T \leq \frac{b - m_z}{s_z}$, en general, no es lineal, por lo que no puede ser tratada con los métodos tradicionales. Para poder encontrar una expresión lineal debemos hacer una nueva suposición

$$s_z^{\,2} = \lambda m_z$$

Con esta suposición nuestra desigualdad $m_z + s_z T \leq b$ se convierte en $m_z + \sqrt{\lambda m_z} T \leq b$ y operando llegamos a que: $f(m_z) = m_z^{\,2} + b^2 - (2b + \lambda T^2) m_z \geq 0$ donde se puede observar gráficamente que nuestras áreas de interés son las siguientes (área sombreada):



Debido a la realidad de nuestro problema la zona de la derecha no tiene significado en nuestra realidad puesto que impondría la condición adiciona de que la demanda debe ser lo suficientemente grande. Este punto quedará en evidencia mas adelante, cuando interpretemos mas claramente los resultados.

Por lo tanto, como las raíces $f(m_z) = 0$, se encuentran en $m_z = \frac{2b + \lambda T^2 \pm T\sqrt{\lambda^2 T^2 + 4b\lambda}}{2}$, y nuestra zona de interés es la de la izquierda, encontramos que se debe cumplir que:

$$m_z \leq \frac{2b + \lambda T^2 - T\sqrt{\lambda^2 T^2 + 4b\lambda}}{2}$$

Interpretación de resultados

Podemos reunir todos los resultados anteriores y utilizarlos para solucionar nuestro problema inicial, el SVRP.

En primer lugar la restricción $s_z^{\,2} = \lambda m_z$, con $z = \sum_{i=1}^{n} a_i y_i$ es en realidad una restricción sobre las posibles formas de las variables aleatorias $\{a_i\}$ de la forma:

$$s_a^{\,2} = \lambda m_a$$

Como $\{y_i\}$ son variables de decisión $y_i \in \{0,1\} \quad \forall i \in \{0..n\}$, sabemos que $\sum_{k=1}^{n} y_k^{\,2} = \sum_{i=1}^{n} y_i$.

Además como las $\{a_i\}$ son independientes e idénticamente distribuidas $\forall i \in \{0..n\}$, en el teorema se pudo encontrar una expresión equivalente para $m_z = m_a \sum_{i=1}^{n} y_i$ y para $s_z^{\,2} = s_a^{\,2} \sum_{k=1}^{n} y_k^{\,2}$. Por lo que si se cumple $s_a^{\,2} = \lambda m_a$ se debe cumplir $s_z^{\,2} = \lambda m_z$, que es la ecuación sobre la que se baso nuestro análisis.

Por otro lado debemos considerar algunos temas de notación para poder utilizar nuestro análisis anterior. Considerando que partimos de $P\left(\sum_{i=1}^{n}\sum_{j=1}^{n} d_i x_{ijk} \le Q_k\right) \ge 1-\alpha \quad \forall k : 1..m$ y que nuestro desarrollo se realizo para $P\left(\sum_{i=1}^{n} a_i y_i \le b\right) \ge 1-\alpha$, se observa que si tomamos $a_i = d_i$, $b = Q_k$, y $\sum_{j=1}^{n} x_{ijk} = y_i$ las expresiones son equivalentes.

Por lo que podemos aplicar las deducciones anteriores a nuestro problema original.

Concluyendo que la restricción $m_z \le \dfrac{2b + \lambda T^2 - T\sqrt{\lambda^2 T^2 + 4b\lambda}}{2}$ tiene una expresión equivalente de la forma

$\sum_{i=1}^{n}\sum_{j=1}^{n} d_i x_{ijk} \le \dfrac{2Q_k + \lambda T^2 - T\sqrt{\lambda^2 T^2 + 4Q_k\lambda}}{2}$ lo cual nos da un método para resolver el SVRP, el cual se resume en la siguiente sección.

Resumen

Recordando que nuestra idea inicial es llevar el problema SVRP a un problema determinista VRP, por lo que debemos encontrar una capacidad ficticia $\overline{Q_k}$ que al cumplir una restricción de la forma: $\sum_{i,j=1}^{n} demanda_i x_{ijk} \le \overline{Q_k} \quad \forall k : 1..m$ nos asegure (con un error posible de α) que cumpla $P\left(\sum_{i=1}^{n}\sum_{j=1}^{n} d_i x_{ijk} \le Q_k\right) \ge 1-\alpha \quad \forall k : 1..m$

Tenemos que $$\overline{Q_k} = \frac{2Q_k + \lambda T^2 - T\sqrt{\lambda^2 T^2 + 4Q_k\lambda}}{2}$$

Donde Q_k es la capacidad del camión k

$T \in \Re \quad \text{tal que} \quad P(r \le T) = 1-\alpha$, con $r \approx N(0,1)$.

$\lambda \in \Re \quad \text{tal que} \quad s_k{}^2 = \lambda m_k$ donde

$$m_k = \sum_{i,j=1}^{n} x_{ijk} E\{d_i\}$$

$$s_k{}^2 = \sum_{i=1}^{n}\left(\sum_{j=1}^{n} x_{ijk}\right)^2 \left(E\{d_i{}^2\} - E\{d_i\}^2\right)$$

$\{d_i\}$ son las demandas independientes e idénticamente distribuidas de los clientes, deben cumplir cierta propiedad (*)

$\{d_i\}$ Para que la solución cumpla todas las demás restricciones del problema de ruteo. (ver el apéndice de Definición de Problemas)

Por ultimo se utiliza como criterio para el establecimiento de las demandas deterministas: $demanda_i$, el valor medio de las demandas aleatorias, es decir en nuestra notación:

$$demanda_i = E\{d_i\}$$

(*) Las variables aleatorias mas frecuentes que cumplen dicha propiedad son

Nombre	Función de distribución, densidad, o cuantía	esperanza	varianza
Uniforme	$f_x(x) = \begin{cases} \frac{1}{b-a} & a \le x \le b \\ 0 & \text{en otro caso} \end{cases}$	$\frac{b+a}{2}$	$\frac{(b+a)^2}{3} - \frac{b+a}{2}$
Binomial	$P(x=h) = C_h^n p^h (1\text{-}p)^{n-h}$	np	$np(1-p)$
Poisson	$P(x=h) = \gamma^h e^{-\gamma/h!} \quad h \in \mathrm{N}$	γ	γ

Gamma	$f_x(x) = \frac{\gamma^n x^{n-1} e^{-\gamma x}}{(n-1)!} \quad x > 0$	n/γ	n/γ^2

A los efectos de la tesis en la que se enmarca este documento, y por requerimientos del usuario, la solución solo se implementará para el caso de una variable aleatoria Uniforme, bajo las hipótesis que se presentaron en esta sección.

Capítulo
6

Diseño e Implementación de los Algoritmos de Ruteo

Diseño e Implementación de los Algoritmos de Ruteo

En el presente capitulo se pretende especificar los algoritmos de ruteo implementados, el proceso de desarrollo y los motivos que llevaron al mismo. Así mismo se presenta una especificación técnica exhaustiva de la biblioteca de ruteo implementada.

El diseño del sistema en su globalidad se detalla en el próximo capitulo, titulado: "Diseño e Implementación del Sistema". En el mismo no se detallan consideraciones pertinentes a los problemas de ruteo, y simplemente se hace referencia a la biblioteca descripta en este capitulo.

Algoritmos de Cálculo

Consideraciones Generales

Si bien los problemas detallados a se definen en general como el encontrar un camino *optimo* sobre un conjunto de nodos en una red con ciertas características adicionales, en la práctica los problemas se reducen a encontrar soluciones en cierta forma *buenas* (bajo ciertas consideraciones) que respondan a los requerimientos de una forma aceptable y en un tiempo de cálculo razonable.

En su formulación natural (solicitando la optimalidad), dichos problemas entran dentro de la categoría *NP-Completos* con lo cual sus tiempos de cómputo serían altamente inaceptables a medida que aumenta la dimensión de los datos del problema.

Es por esto que para solucionar dichos problemas se recurre a métodos heurísticos capaces de llegar a una solución empíricamente aceptable y en un tiempo de respuesta razonable en proporción a la dimensión de los datos.

Como base de dichas soluciones utilizamos el algoritmo de Clarke y Wright (conocido como algoritmo de los ahorros) con ciertas variaciones de acuerdo a las características de cada problema [2].

Clarke & Wright Aplicado

Este algoritmo consiste básicamente en, dado un conjunto inicial de rutas (factible), analizar la posibilidad de unir pares de dichas rutas evaluando cuantitativamente la ganancia de efectuar dicha unión. Esto nos resulta en otro conjunto de rutas analíticamente "mejorado" con respecto al inicial.

Para la aplicación práctica del algoritmo se elige partir del conjunto factible de rutas que incluyen el depósito y cada uno de los clientes por separado (es decir, una ruta por cliente).

Ahora definiremos el concepto de *ahorro* como sigue:

Sean *i, j* clientes, *D* el deposito y $c_{x\,y}$ el costo de ir de *x* a *y* que se supone simétrico. Se define S_{ij}, el ahorro entre *i* y *j* como:

$$S_{ij} = (2\,c_{Di} + 2\,c_{Dj}) - (c_{Di} + c_{Dj} + c_{ij})$$

Si el costo no es simétrico, la expresión es:

$$S_{ij} = ((c_{Di} + c_{iD}) + (c_{Dj} + c_{jD})) - (c_{Di} + c_{jD} + c_{ij})$$

En ambos casos la expresión se simplifica a:

$$S_{ij} = c_{iD} + c_{Dj} - c_{ij}$$

Intuitivamente, el ahorro queda definido como lo que evito gastar si, en vez de del cliente *I* ir al depósito y luego al *J*, voy directamente del *I* al *J.*

Se detallan los pasos del algoritmo:

1. Construir para cada cliente una ruta unitaria (solo participa ese cliente y el deposito).
2. Para cada par de clientes *i,j* calcular el correspondiente ahorro S_{ij}.
3. Ordenar los ahorros en forma decreciente.
4. Considerar el máximo ahorro S_{ij}. Se intentara unir en una única ruta a los clientes *i*, *j*, solo en el caso en que se encuentran, ambos, en algún extremo de las rutas en las que participan (es el primero o el último cliente en ser visitado). En otro caso no se considera la posibilidad de reestructurar las rutas. Además, se debe verificar que la nueva ruta cumpla las restricciones de capacidad de carga del vehículo, del depósito, y otras si existiesen.
5. Descartar el S_{ij} recién considerado. Si el siguiente ahorro a considerar es $S_{ij} \leq 0$ o no hay mas ahorros que considerar, terminar, sino ir a 4.

A continuación detallaremos la utilización de este algoritmo (así como de consideraciones especiales) para la resolución de cada uno de los problemas planteados.

TSP y VRP

Para la resolución del **VRP** se utiliza el algoritmo de los ahorros en su forma expuesta, mientras que para el **TSP** se eliminan las consideraciones de las capacidades de los vehículos y del depósito. En la práctica, se consideran los depósitos y los vehículos con capacidades infinitas y los clientes con demanda 0.

MTSP

Para la resolución del **MTSP** se hace una pequeña modificación al algoritmo de los ahorros utilizado para el **TSP**. En este caso se agrega una cláusula de finalización más restrictiva: el algoritmo termina de forma habitual (si la cantidad de clientes es inferior a la cantidad de vehículos) o cuando se tengan la misma cantidad de rutas que de vehículos.

MDVRP

Para poder reducir el problema del **MDVRP** a **VRP** se pueden tomar dos enfoques distintos: asignar y rutear, o bien, rutear y asignar (entendiendo por asignar el designar un depósito para cada cliente).

En nuestro caso optamos por una solución intermedia en la cual los clientes que están "notoriamente cerca" de un depósito son asignados a éste automáticamente mientras que la asignación de los que no están tan "notoriamente cerca" se trata de una manera especial.

De esta forma, luego de la asignación parcial inicial, se corre el **VRP** para cada subconjunto definido de datos. Paso seguido, se recorren los clientes "en duda" analizando la posibilidad de insertarlos entre las parejas de clientes (continuos) de cada ruta, hasta hallar la mejor opción.

VRPTW

Para resolver el **VRPTW** se usa el algoritmo de Clark y Wright adaptado para problemas con ventanas de tiempo. Dicha adaptación se reduce al cálculo de los ahorros utilizando una función de costos que pondera la distancia y el tiempo según una constante (llamada: *costo de una unidad de tiempo*) calibrada según la topología de la red y la naturaleza de los problemas con ventanas de tiempo a resolver.

Considerando esta implementación del algoritmo, la solución es un compromiso entre una ruta eficiente en distancia y la no violación de las restricciones de tiempos impuestas por el problema.

Por claridad y simplicidad se resuelve el problema con ventanas de tiempo simple (esto es, de un solo período cada una), pero considerando una generalización del algoritmo para aceptar ventanas de tiempo compuestas (varios períodos por ventana).

Por ultimo se advierte la inherente complicación que existe al buscar una solución factible para este tipo de problemas que incluyen variables temporales, que se refleja en este algoritmo con intrincadas restricciones sobre la posible unión de dos rutas (paso 4 del algoritmo de Clark y Wright).

MDVRPTW

Se comienza por una asignación de clientes a depósitos con la particularidad de que no solo se chequean las factibilidad con respecto a la demanda sino que también se analizan desde el punto de vista de las ventanas de tiempo. Se tiene en cuenta que el marco de tiempo máximo del cliente más el tiempo necesario para llegar del depósito a dicho cliente sea menor o igual que el marco de tiempo máximo del depósito.

Luego de esta división en sub-soluciones factibles para cada sub-conjunto de datos, se aplica el algoritmo del **VRPTW**.

VRPHF

Este algoritmo se resuelve mediante la asignación previa de vehículos a clientes. Los mismos se asignan de forma tal que cumplan las restricciones de capacidad del vehículo y, lo más significativo en este caso, que el vehículo que sirva a cada cliente sea el que cumpla su restricción de demanda con menos holgura (es decir, el vehículo que, dadas la capacidad y las asignaciones anteriores, disponga de menos oferta de producto disponible).

Opuesto a las generalidades, este algoritmo no depende del **VRP** dado que su función de costos debe considerar, además, los costos de los vehículos. De esta forma, se utiliza una variación del algoritmo de Clarke y Wright, conocida como Combined Savings (ahorros combinados) en la cual la función de costos se deduce como sigue:

$$s(i,j) = c(i,d) + c(d,j) - c(i,j).$$

$$s'(i,j) = s(i,j) + F(z_i) + F(z_j) - F(z_i + z_j).$$

$$s''(i,j) = s'(i,j) + \{F'(P(z_i + z_j) - z_i - z_j)\}.\gamma$$

Donde:

- z_i : Es la demanda de la ruta que pasa por el nodo i.
- F(n): Devuelve el costo fijo del vehículo que satisface la demanda n con menor holgura.
- F'(n): Devuelve el costo fijo del mayor vehículo cuya capacidad es menor o igual que n.
- P(n): Devuelve la capacidad del vehículo que satisface la demanda n con menor holgura.
- γ: Es 1 si (($z_i < z_j$) y ($P(z_i + z_j) - P(z_j) \neq 0$) y vale cero en caso contrario.

Dada esta ecuación de ahorros, el algoritmo sigue la misma estructura que el **VRP**.

DARP

Se resuelve un caso simple del problema DARP, donde se tiene solo un deposito, un vehículo con capacidad limitada, tiempo mínimo de *pickup* (l_i) y máximo de *delivery* (F_i) para cada cliente. Además debe cumplirse que ($l_i + T_i <= F_i$) para todo cliente, siendo T_i el tiempo mínimo de ir desde el punto de *pickup* al punto de *delivery.*

Se considera inaceptable (solución no valida) el hecho de llegar a un punto de *pickup* antes que sea su tiempo mínimo y esperar inactivamente que el mismo se cumpla, es decir, siempre se llegará a los clientes después de el tiempo mínimo establecido.

Para resolver este problema se siguen los siguientes pasos:

Primero se aplica el algoritmo **VRPTW** a los puntos de *pickup* de todos los clientes, donde se utiliza la demanda de cada cliente del **DARP**. Y como ventana de tiempo para cada cliente, se considera como tiempo inicial al tiempo mínimo del problema original (l_i) y tiempo final igual al tiempo máximo de delivery (F_i) menos el tiempo de ir desde el pickup al delivery (T_i), es decir, ($F_i - T_i$).

Obtenida la ruta solución para dicho **VRPTW** deben ser insertados todos los puntos de *delivery* de los clientes en el mismo orden en que fueron insertados los puntos de *pickup* en la ejecución del problema **VRPTW** previamente resuelto.
Como criterio para la incorporación de cada punto de delivery en particular se opta por minimizar el tiempo utilizado.

Se aclara que las restricciones sobre las que está desarrollado este algoritmo pueden no encontrar solución a problemas DARP de aparente simple complejidad.

CPP

Este algoritmo se resuelve de forma simple luego de realizar las siguientes consideraciones.
Se considera una extensión del problema clásico CPT, donde los arcos que se tienen que recorrer tienen demanda. Además como hay arcos que pueden tener varias demandas se opta por resolver este problema utilizando el algoritmo **VRP**, donde cada cliente se corresponde con una demanda de arco (dichos clientes se insertan a mitad del arco correspondiente).

SVRP

Para resolver el **SVRP** se usa el algoritmo de Clark y Wright adaptado para problemas con demanda no determinista. Dicha adaptación se explica en el capitulo capítulo anterior: “Desarrollo Teórico” y básicamente consiste en adaptar el problema al **VRP**.

Por ultimo se advierte la inherente complicación que surge de que una solución factible para el algoritmo pueda no cumplir con la demanda de los clientes en la realidad. Esto surge de la no certeza sobre la demanda de los clientes, que puede hacer que la capacidad de un vehículo no logré satisfacer la demanda de los clientes que él tiene asignado.
Por supuesto se incluye un parámetro de calibración para este algoritmo, que representa la probabilidad de que alguna ruta falle. Ajustando el mismo se puede disminuir la probabilidad de que el algoritmo arroje una solución que no cubra la demanda, en desmero de una peor utilización de los vehículos.

Diseño

El diseño actual del sistema no se puede enmarcar dentro de un estilo de diseño específico ya que se nutre de características de varios.

Para aclarar los conceptos, se podría decir que el motor de cálculo fue diseñado mediante una estructura *three-tier* en la que las distintas capas se corresponden con los módulos de la siguiente manera:

Capa de usuario: La capa de usuario esta dada por la interfaz de entrada y salida especificada en su propio capítulo.

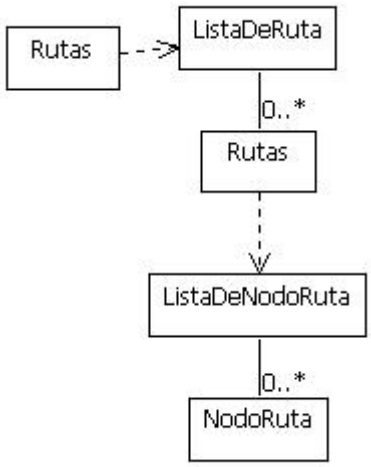
Capa de servicios: Aquí encontraríamos el motor de cálculo en sí mismo. Los módulos que, interactuando, resuelven las características algorítmicas de los problemas.

Capa de datos: En la capa de datos se situarían un conjunto de clases y módulos que cumplen la función de almacenadores de información, tanto para los datos de entrada como para los datos de salida, como para estructuras intermedias necesarias.

A continuación se presenta un diagrama híbrido de diagrama de clases, diagrama de utilización y diagrama de in/out que refleja el funcionamiento general del sistema de acuerdo a lo anteriormente especificado. Detallaremos los sub-diagramas que luego compondrán el todo.

Rutas.

La clase rutas se crea esencialmente para lograr compatibilidad con los algoritmos recibidos de proyectos anteriores. Su finalidad es la de almacenar los distintos recorridos solución del problema -recordar que dado el algoritmo de los ahorros, el resultado consta de un conjunto factible de rutas.



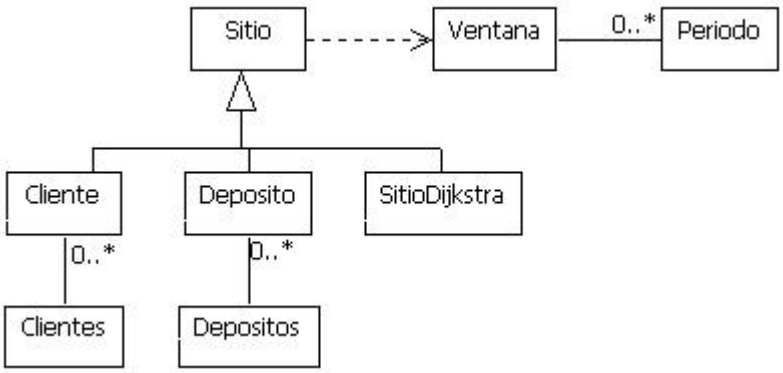
Caminos.

La lista de caminos cumple una finalidad similar a la de rutas. El almacenamiento de datos resultado.

Su cometido interno consiste en almacenar los nodos intermedios que se encuentran entre dos nodos problema (clientes, depósitos, etc.). Es mediante la identificación de éstos que se identifica a cada camino.

Dada la particularidad de que el sistema puede resolver los problemas (dado sus requerimientos de entrada y salida), esta clase generalmente solo almacena costos, tiempos e identificadores.

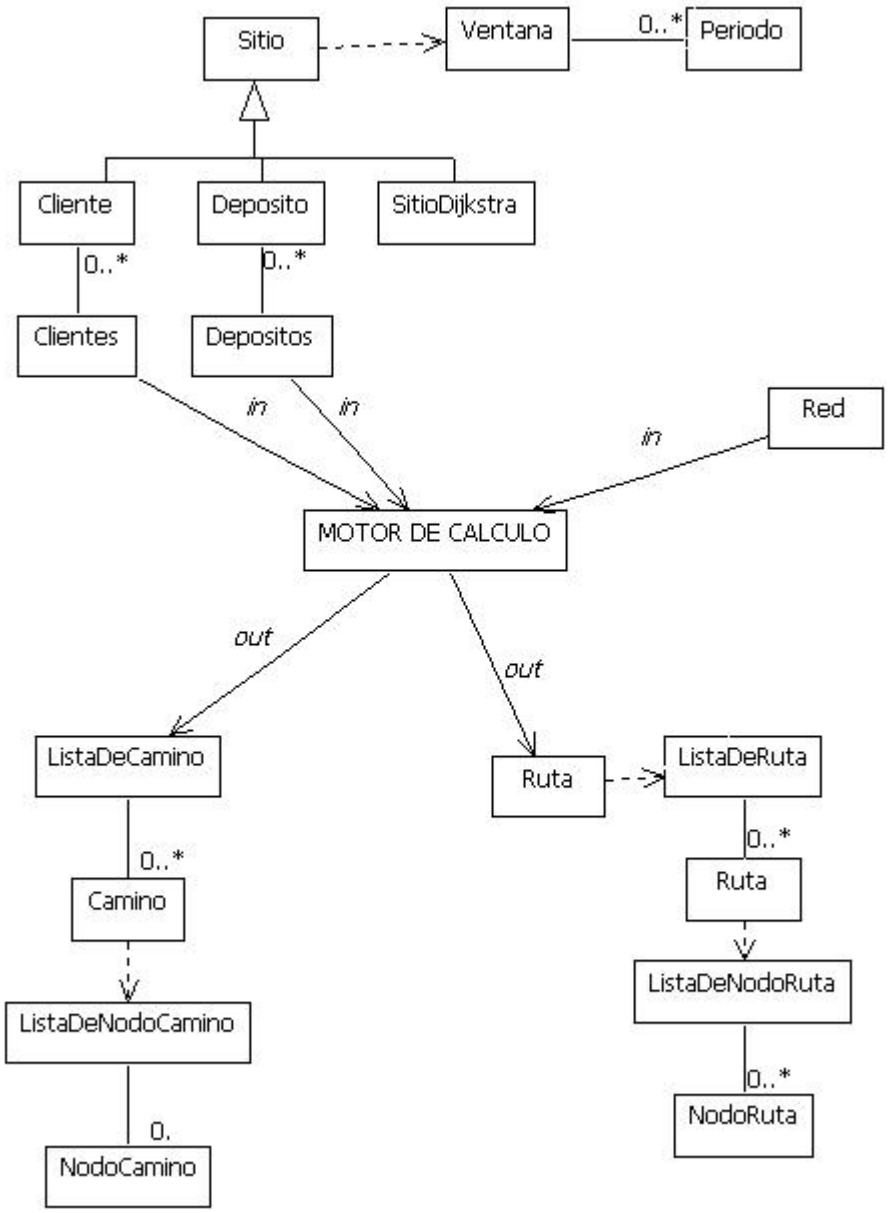
Sitios



La estructura de sitios se utiliza para almacenar todo lo referente a los distintos nodos de interés para el problema (clientes, depósitos, pick-up, delivery, etc.)

Aquí se utiliza el concepto de herencia debido a que se necesita que cada nodo identificado como “sitio” cumpla con ciertas características como ser un identificador, una ventana de tiempo, etc.

Diagrama General.



Nota: Claro está que para la especificación de este diagrama se pasan por alto decenas de clases y módulos que se utilizan en pos de la solución. Por más detalles consultar el apéndice sobre Implementación.

Especificación Técnica de la Biblioteca de Vínculos Dinámicos : Router

La biblioteca Router es la encargada de resolver los problemas de ruteo, por tanto es una parte medular del sistema.
En esta sección se describe las generalidades, interfase de entrada y salida , y las funciones exportadas de dicha biblioteca.

Por información mas detallada sobre dicha biblioteca se recomienda la lectura del apéndice K: "Biblioteca de Vínculos Dinámicos: Router". A continuación se presenta un resumen de dicho apéndice.

Introducción

El objetivo de esta biblioteca de vínculos dinámicos (y del software en general), es resolver una gama importante de problemas de ruteo. Se toma la decisión de encapsular la solución de dichos problemas en una librería para lograr independencia entre la interfase y la implementación, pensando en futuros cambios y mejoras sobre la biblioteca en si misma.

Los problemas de ruteo resueltos por esta librería son:

- **VRP** (Vehicle Routing Problem). En este problema, dado un depósito y uno o varios clientes, se determina la cantidad de vehículos (todos con la misma capacidad de carga) y la ruta tales el recorrido de los vehículos sea mínimo.
- **MDVRP** (Multi-Depot Vehicle Routing Problem). Este problema es similar al anterior, pero la cantidad de depósitos es mayor a 1.
- **VRPTW** (Vehicle Routing Problem with Time-Windows). En este problema, cada cliente tiene una ventana de tiempo (o time-window), es decir, un rango de tiempo en el que el cliente puede ser atendido. Este problema considera un solo depósito que tiene asociado una ventana de tiempo que indica el horario en que presta servicios.
- **MDVRPTW** (Multi-Depot Vehicle Routing Problem with Time-Windows). Ídem anterior, pero se consideran varios depósitos.
- **VRPHF** (VRP Flota Heterogénea). Ídem VRP, pero en este caso los vehículos tienen diferente capacidad. Por lo tanto ya no son itercambiables entre si. Se debe generar una ruta para cada vehiculo.
- **TSP** (Travelling Salesman Problem). Este problema es un caso particular del VRP, donde se tiene un sólo depósito y un solo vehículo, y donde la demanda de los clientes es cero.
- **MTSP**
- **DARP** (Dial-a-ride Problem). Este problema resuelve el caso en que cada cliente tiene un punto de carga y un punto de entrega, todo dentro de una ventana de tiempo. El problema es, como en los casos anteriores, minimizar el costo para servir a todos los clientes, partiendo desde un depósito dado.
- **CPP** (Chinese Postman Problem). El problema del CPP es hallar un recorrido tal que pase por todos los clientes y su costo sea mínimo, sin embargo la variante del CPP resuelta por este software recibe las demandas de los clientes, no de los arcos.
- **SVRP** (Stochastic Vehicle Routing Problem). Es una generalización del problema clásico VRP, donde los clientes tienen demanda no determinista.

Por mas detalles sobre los tipos de problemas se puede consultar el apéndice: Definición de Problemas.

Con respecto a la comunicación de la librería con el resto del sistema, la misma se reduce a un archivo de entrada y otro de salida, como se especifica en la sección correspondiente.

Arquitectura de Comunicación: Interfase

Debido a que el motor de cálculo es programado en forma independiente de la interfase de usuario (y en otro lenguaje de programación, C++), es necesario implementar algún mecanismo de comunicación entre estos.

La interfase es pensada, en este caso, buscando:

- **Claridad**. Se pretende que se los datos de entrada y salida puedan ser interpretados intuitivamente.
- **Simplicidad**. Se busca simplicidad en el desarrollo de programación.
- **Generalidad**. Es necesario una interfase lo suficientemente maleable como para adaptarse a nuevos tipos de problemas o cambios en los mismos.
- **Minimizar Información**. Se pretende eliminar la redundancia en los datos de entrada y salida, para favorecer la claridad antes mencionada.

Como solución a estos requerimientos se presenta como opción mas clara la utilización de archivos de texto. Para eliminar la dificultad que puede encerrar el manejo de los mismos se utiliza el **formato .INI.** Este formato permite el manejo rápido y simple de archivos de texto y fue desarrollado por Microsoft (por mas información ver la ayuda de las API de Windows).

Como conclusión de esta sección se tiene que la interfase de la biblioteca se realiza mediante dos archivos en formato .INI. La entrada.INI donde se especifican los datos del problema a resolver, y la Salida.INI donde se devuelve la solución al problema de ruteo.

Entrada de la Biblioteca

Generalidades

Esta interfase pretende especificar completamente la entrada de la DLL para los problema: VRP, MDVRP, VRPTW, MDVRPTW, VRPHF, TSP, MTSP, DARP, CPP y SVRP.

Básicamente la entrada es un conjunto de clientes (llamados también sitios) a los cuales hay que satisfacer su demanda, partiendo de uno o varios depósitos. Restringiendo la solución a las particularidades de cada tipo de problema.

Como antes se mencionó, la entrada del motor de calculo (.dll) se limita a un archivo de texto en formato .INI.
Dicho archivo de texto, de ahora en mas referido como Entrada.INI, tiene el nombre y ubicación (path) que el usuario del motor determino en la entrada (ver: Funciones Exportadas).

Especificación

Los datos que aparecen en el archivo de entrada dependen en gran forma del problema de ruteo que se esta resolviendo. Por ejemplo para el caso de flota heterogénea se especifican varios tipos de vehículos, en cambio para el TSP es necesario no especificar ninguno. Para el caso de ventana de tiempo se especifica la ventana de tiempo admisible para los clientes y depósitos, etc.

Principales identificadores

A continuación se detalla una tabla de los principales identificadores utilizados, según grupo y clave del formato .INI.

El archivo de entrada siempre está encabezado por el grupo [PROBLEMA], que contiene los datos mas relevantes necesario para la definición del problema de ruteo.

Posteriormente los grupos [SITIO_Z] y [DEPOSITO_Z] definen los clientes y depósitos del problema, numerados de forma creciente, comenzando en 1 y terminando en la cantidad definida en [PROBLEMA].

Problema:

[PROBLEMA]	Definición del problema
TIPO=	Tipo de problema (Entero) 1– **VRP** Con todos los caminos precalculados. 2- VRP Donde los caminos NO precalculados se calculan utilizando Dijkstra. 3- VRP Donde los caminos NO precalculados se calculan utilizando BFS. 4- VRP Donde los caminos NO precalculados se calculan utilizando DFS. 10– **MDVRP** Con todos los caminos precalculados. 20– **VRPTW** Con todos los caminos precalculados. 30– **MDVRPTW** Con todos los caminos precalculados. 40– **VRPHFE** Con todos los caminos precalculados. 50– **TSP** Con todos los caminos precalculados. 60– **DARP** Con todos los caminos precalculados. 70– **CPP** Con todos los caminos precalculados. 80– **MTSP** Con todos los caminos precalculados. 90– **SVRP** Con todos los caminos precalculados.
SITIOS=	Cantidad de clientes (Entero)
DEPOSITOS=	Cantidad de depósitos (Entero)
VEHICULOS=	Cantidad de tipos (Entero)
RED=	Cantidad de arcos de la red del CPP(Entero)
PARES =	Cantidad de pares de sitios carga-entrega para el DARP (Entero)
PATH_CAMINOS =	Dirección y nombre de archivo completo, donde se encuentran los caminos (en el formato especificado en la SALIDA) precalculados.

```
                    (Atención: puede ser este mismo archivo o el
                    archivo de salida).(String)
```

Los sitios son los clientes a los cuales se debe abastecer (para la mayoría de los problemas planteados).

Sitios:

```
[SITIO_Z]       Definición del sitio numero Z
ID=             identificador externo del sitio (Entero)
X_NODO_ANT=     Coordenada X del nodo anterior (Float)
Y_NODO_ANT=     Coordenada Y del nodo anterior (Float)
X_NODO_POS=     Coordenada X del nodo posterior (Float)
Y_NODO_POS=     Coordenada Y del nodo posterior (Float)
DEMANDA=        Demanda del sitio (Float)
VENTANA=        identificador de la ventana (que figura en el
                mismo archivo INI) (Entero)

TIEMPO_SERVICIO=      Tiempo de servicio del sitio (Float)

DIST_ANT=       Distancia al nodo anterior (Float)
DIST_POST=      Distancia al nodo posterior (Float)
DEM_ANT=        Demora en tiempo al nodo anterior (Float)
DEM_POST=       Demora en tiempo al nodo posterior (Float)

CAMINO_HACIA_S_k      Camino de ir desde el sitio actual (Z) al
                      sitio k (Entero)
CAMINO_HACIA_D_k      Camino de ir desde el sitio actual (Z) al
                      deposito k (Entero)
```

Para todos los problemas es necesario tener definido un deposito , citándose como caso particular los problemas de múltiple deposito, donde es necesario tener varios.

```
  Depósitos:

  [DEPOSITO_Z]    Definición del deposito Z
  ID=             identificador externo del deposito (Entero)
  X_NODO_ANT=           Coordenada X del nodo anterior (Float)
  Y_NODO_ANT=           Coordenada Y del nodo anterior (Float)
  X_NODO_POS=           Coordenada X del nodo posterior (Float)
  Y_NODO_POS=           Coordenada Y del nodo posterior (Float)

  VENTANA=          identificador de la ventana (que figura en
                  el
                     mismo archivo INI) (Entero)
  CAPACIDAD=            Capacidad del deposito (Float)

  DIST_ANT=       Distancia al nodo anterior (Float)
  DIST_POST=            Distancia al nodo posterior (Float)
  DEM_ANT=        Demora en tiempo al nodo anterior (Float)
  DEM_POST=       Demora en tiempo al nodo posterior (Float)

  CAMINO_HACIA_S_k      Camino de ir desde el deposito actual
                        (Z) al sitio k (Entero)
```

	Para el caso del CPP, debe interpretarse como el camino al punto medio del arco k CAMINO_HACIA_D_k Camino de ir desde el deposito actual(Z) al deposito k (Entero)

En la mayoría de los algoritmos es necesario tener definido algún tipos de vehículos. Y para el caso de flota heterogénea es necesario definir varios (mas de uno) tipos de vehículos.

Vehículos:

[TIPO_VEHICULO_Z]	Definición de todos los vehículos de tipo_z
CANTIDAD=	Cantidad de vehículos de este tipo (Entero)
ID=	identificador externo del tipo_vehiculo
CAPACIDAD=	Capacidad de los vehículos del tipo z (Float)
COSTO_FIJO=	Costo fijo de los vehículos del tipo z (Float)
COSTO_VAR=	Costo variable de los vehículos del tipo z (Float)

Para los problemas con ventanas de tiempo es necesario el grupo [VENTANA].

Ventana:

[VENTANA_Z]	Definición del la ventana de tiempo Z
CANTIDAD_PERIODOS=	Cantidad de periodos que hay en la ventana (Entero)
PERIODO_INICIO_Z=	Hora de inicio del periodo Z (Float)
PERIODO_FINAL_Z=	Hora de final del periodo Z (Float)

Para los problemas tipo CPP, se especifican los arcos. Dichos arcos se pueden pensar como los clientes, es decir, los sitios, de este tipo de problema.

Arco:

[ARCO_Z]	Definición del arco Z
ID=	identificador externo del arco (Entero)
COSTO=	Costo del arco (Float)
TIEMPO=	Tiempo del arco (Float)
X_NODO_ANT=	Coordenada X del nodo anterior (Float)
Y_NODO_ANT=	Coordenada Y del nodo anterior (Float)
X_NODO_POS=	Coordenada X del nodo posterior (Float)
Y_NODO_POS=	Coordenada Y del nodo posterior (Float)

CAMINO_HACIA_S_k	Camino de ir desde el punto medio del arco actual (Z) al punto medio del arco k (Entero)
CAMINO_HACIA_D_k	Camino de ir desde el punto medio del arco actual (Z) al deposito k (Entero)

Para los problemas tipo DARP, donde cada cliente tiene un sitio de pickup y un sitio de delivery, se especifican los pares. Cada sitio pertenece a algún par, como sitio de carga o como sitio de entrega. Por tanto para este tipo de problemas hay el doble de sitios que de pares.

Pares:

[PAR_Z]	Definición del par carga-entrega Z
SITIO_CARGA=	identificador del sitio de carga(que figura en el mismo archivo INI)
SITIO_ENTREGA=	identificador del sitio de entrega(que figura en el mismo archivo INI)

Para los problemas con demanda no determinista (SVRP), se define una variable aleatoria que cumple los efectos de dicha demanda. Esta variable aleatoria es única, y la misma para todos los clientes [2].

Demanda No Determinista (variable aleatoria):

[VA_DEMANDA]	Definicion de la variable aleatoria que representa la demanda de cada cliente para el SVRP.
TIPO =	0-Uniforme, 1-Normal (Entero)
MEDIA=	Valor medio de la variable aleatoria (Float)
VARIANZA=	Varianza de la variable aleatoria (Float)

Salida de la Biblioteca

Generalidades

Esta interfase pretende especificar completamente la salida de la DLL para los problema: VRP, MDVRP, VRPTW, MDVRPTW, VRPHF, TSP, MTSP, DARP, CPP y SVRP.

Básicamente la salida de la biblioteca es una ruta, o conjunto de rutas. Dichas rutas son la solución completa al problema de ruteo que se le a presentado al motor de calculo como entrada.

[2] Las demandas de los clientes forman un IID: variableas aleatorias independientes e idénticamente distribuidas.

Al igual que la entrada, la salida del motor de calculo (.dll) se limita a un archivo de texto en formato .INI.
Dicho archivo de texto, de ahora en mas referido como Salida.INI, tiene el nombre y ubicación (path) que el usuario del motor determino en la entrada (ver: Funciones Exportadas).

Especificación

Con el fin de esclarecer los términos y la notación utilizada se realizaran una serie de puntualizaciones.
Existen tres tipos básicos de nodos a distinguir: los "depósitos", los "clientes" y las "esquinas".
Una "ruta" es el recorrido desde un deposito a otro deposito pasando por varias esquinas y /o clientes.
Un "camino" es el recorrido solo a través de esquinas de un deposito (o cliente) a un deposito (o cliente).

Por tanto, la salida es un conjuntos de rutas, las cuales a su vez están formadas por un conjunto de caminos.

Los datos que aparecen en el archivo salida dependen en gran forma del problema de ruteo que se esta resolviendo. Por ejemplo para el caso de flota heterogénea cada ruta tiene un vehículo asignado. Para el caso de ventana de tiempo cada ruta y camino tiene una ventana de tiempo en la cual se debe realizar su recorrido, etc.

Principales identificadores

A continuación se detalla una tabla de los principales identificadores utilizados, según grupo y clave del formato .INI.

El archivo de salida siempre está encabezado por el grupo [SOLUCION], que contiene los datos mas relevantes de la solución al problema de ruteo.

Solución:

[SOLUCION]	Datos de la solución
RUTAS=	Cantidad de rutas (Entero)
DEMANDA=	Demanda de la solución (Float)
COSTO=	Costo de la solución (Float)
NODOS=	Cantidad de nodos (clientes, depósitos o esquinas) que componen la solución (Entero)
VENTANA=	identificador de la ventana de la solución (que figura en el mismo archivo INI). La ventana de la solución, representa el periodo mínimo inicial y el periodo máximo final, considerando todas las rutas (Entero).

Posteriormente al grupo [SOLUCION], se presentan las rutas de la solución, numeradas de forma creciente, comenzando en 1 y terminando en la cantidad de rutas determinadas en [SOLUCION].

Rutas:

```
[RUTA_Z]        Definición de la ruta numero Z
VEHICULO=       identificador externo del vehículo que realiza
                la Ruta (Entero)
DEMANDA=        Demanda satisfecha por la ruta(Float)
COSTO=          Costo de la ruta (Float)
TIEMPO=         Tiempo de la ruta (Float)
VENTANA=        identificador de la ventana (que figura en el
                mismo archivo INI) (Entero)
ID_DEPOSITO_INICIAL= El id del deposito inicial (Entero)
ID_DEPOSITO_FINAL=   El id del deposito final (Entero)

CAMINO_INICIAL=El numero de camino inicial en esta ruta
                (Entero)
CAMINO_FINAL=  El numero de camino final en esta ruta (Entero)
```

Cada [RUTA_Z], está formada por un conjunto de caminos, comenzando en CAMINO_INICIAL y terminando en CAMINO_FINAL, especificados en el propio grupo [RUTA_Z].

Caminos:

```
[CAMINO_Z]      Definición del camino numero Z
COSTO=          Costo del camino (Float)
TIEMPO=         Tiempo del camino (Float)
VENTANA=        identificador de la ventana (que figura en el
                mismo archivo INI) (Entero)
TIPO_INICIAL=   0-Cliente y 1-Deposito (Entero)
ID_INICIAL=     El id del deposito o cliente inicial (Entero)
OPERACIÓN_INICIAL=   0-De paso, 1-Despachar y 2-Cargar
                (Entero)
TIPO_FINAL=     0-Cliente y 1-Deposito (Entero)
ID_FINAL=       El id del deposito final (Entero)
OPERACIÓN_FINAL= 0-De paso, 1-Despachar y 2-Cargar (Entero)


NODO_INICIAL=   El numero de nodo inicial en este camino
                (Entero)
NODO_FINAL=     El numero de nodo final en este camino (Entero)

CALCULAR_CAMINO=     Entero que indica (1) si hay calcular el
                     camino por que se manejo un costo exacto,
                     o (0) si hay que leerlo de este mismo
                     archivo y grupo.
```

Al igual que como las rutas están formadas de caminos, los caminos están formados de nodos.
Los nodos son necesarios para el caso en que la biblioteca además de realizar la solución al problema de ruteo, también se utilice para hallar los caminos mínimos entre dos depósitos (o clientes).

Nodo:

```
[NODO_Z]        Definición del nodo Z
TIPO=           0-Esquina, 2-Comienzo ArcoCPP 3-Fin ArcoCPP
                (Entero)
X_NODO=         Coordenada X del nodo (Float)
Y_NODO=         Coordenada Y del nodo (Float)
VENTANA=        Para el caso de depósitos y clientes se debe
                especificar la ventana de atención. (Entero)
```

Al igual que en la entrada es necesario, para los problemas con ventanas de tiempo el grupo [VENTANA] **.**

Ventana:

```
[VENTANA_Z]           Definición del la ventana de tiempo Z
CANTIDAD_PERIODOS=    Cantidad de periodos que hay en la
                      ventana (Entero)
PERIODO_INICIO_Z=     Hora de inicio del periodo Z (Float)
PERIODO_FINAL_Z=      Hora de final del periodo Z (Float)
```

Funciones Exportadas

La funciones que exporta la biblioteca son: ***"CargarEstructura"*** , ***"Calcular"*** y ***"Dijkstra"***.

A pesar de que la biblioteca esta pensada para su uso exclusivo en la resolución de problemas de ruteo mediante la función ***"Calcular"***, ésta también incluye dos funcionalidades de extrema utilidad: ***"CargarEstructura"*** y ***"Dijkstra".*** Dichas funcionalidades se incluyeron para lograr una solución integral y una autonomía total de la biblioteca. A trabes de ***"Dijkstra"*** se puede calcular las mejores rutas para una topología de red genérica cargada mediante ***"CargarEstructura"***.
La interface de estas dos funcionalidades se detalla en las secciones: "Entrada para Cargar de la Red" y "Interface para el Calculo del Dijkstra"

A continuación se detallan cada una de las funciones exportadas por la biblioteca: ***"CargarEstructura"*** , ***"Calcular"*** y ***"Dijkstra".***

Cargar Estructura

Descripción
Carga la estructura (red o ciudad) en memoria.
Se debe ejecutar una sola vez, luego de declarada la librería.
Su ejecución demora un tiempo considerable, de aproximadamente un minuto para 15.000 nodos.
La utilidad de esta función se limita, si posteriormente se ejecuta alguna llamada a la función "Dijkstra" o a la función "Calcular" sin caminos precalculados.
Declaración

```
Declare Function "CargarEstructura" Lib " t5vrp.dll" (ByVal
IniEstructura As String) As Long
```

Ejemplo de uso:

```
Call CargarEstructura("C:\tempVRP\theme2.ini")
```

Parámetros

IniEstructura : Recibe un String conteniendo la dirección y nombre del archivo .INI de la red. La especificación de dicho archivo .INI se encuentra en: Entrada para Cargar la Red.

Valor de Retorno

Devuelve un entero que representa un código de error. Dichos códigos de error se encuentran especificados en ConstantesError.h

Calcular

Descripción

Calcula los problemas de ruteo

Dependiendo del tipo de problema, es necesario la carga de la estructura previa

Declaración

```
Declare Function Calcular Lib "t5vrp.dll" (ByVal IniEntrada As
String, ByVal IniSalida As String) As Double
```

Ejemplo de uso:

```
Call Calcular("c:\tempVRP\entradaVRP.ini", "c:\tempVRP\
salidaVRP.ini")
```

Parámetros

IniEntrada : Recibe un String conteniendo la dirección y nombre del archivo INI de entrada. La especificación de dicho archivo .INI se encuentra en: Entrada de la biblioteca.

IniSalida: Recibe un String conteniendo la dirección y nombre del archivo INI de salida. La especificación de dicho archivo .INI se encuentra en: Salida de la biblioteca.

Valor de Retorno

Devuelve un entero que representa un código de error. Dichos códigos de error se encuentran especificados en ConstantesError.h

Dijkstra

Descripción

Ejecuta el Dijkstra

Se debe ejecutar posteriormente de la carga de la estructura

Declaración

```
Declare Function Dijkstra Lib "t5vrp.dll" (ByVal IniEntrada As
String, ByVal IniSalida As String) As Double
```

Ejemplo de uso:

```
Call Dijkstra("c:\tempVRP\entradaDijkstra.ini", "c:\tempVRP\
salidaDijkstra.ini")
```

Parámetros

IniEntrada : Recibe un String conteniendo la dirección y nombre del archivo INI de entrada. La especificación de dicho archivo .INI se encuentra en: Interface para el Calculo del Dijkstra.

IniSalida: Recibe un String conteniendo la dirección y nombre del archivo INI de salida. La especificación de dicho archivo .INI se encuentra en: Interface para el Calculo del Dijkstra.

Valor de Retorno

Devuelve un entero que representa un código de error. Dichos códigos de error se encuentran especificados en ConstantesError.h

Códigos de Error

Al ejecutar cualquier funcionalidad exportada por la biblioteca, la misma devuelve un código de error. Los códigos de error son un entero (Integer) que representa algún tipo de dificultad presentada durante la ejecución de alguna funcionalidad exportada por la biblioteca.

Se puede encontrar un detalle de los códigos de error en el archivo: ***ConstantesError.h*** o en el apéndice K: “Biblioteca de Vínculos Dinámicos: Router”.

Capítulo
7
Diseño e Implementación del Sistema

Diseño e Implementación del Sistema

Introducción

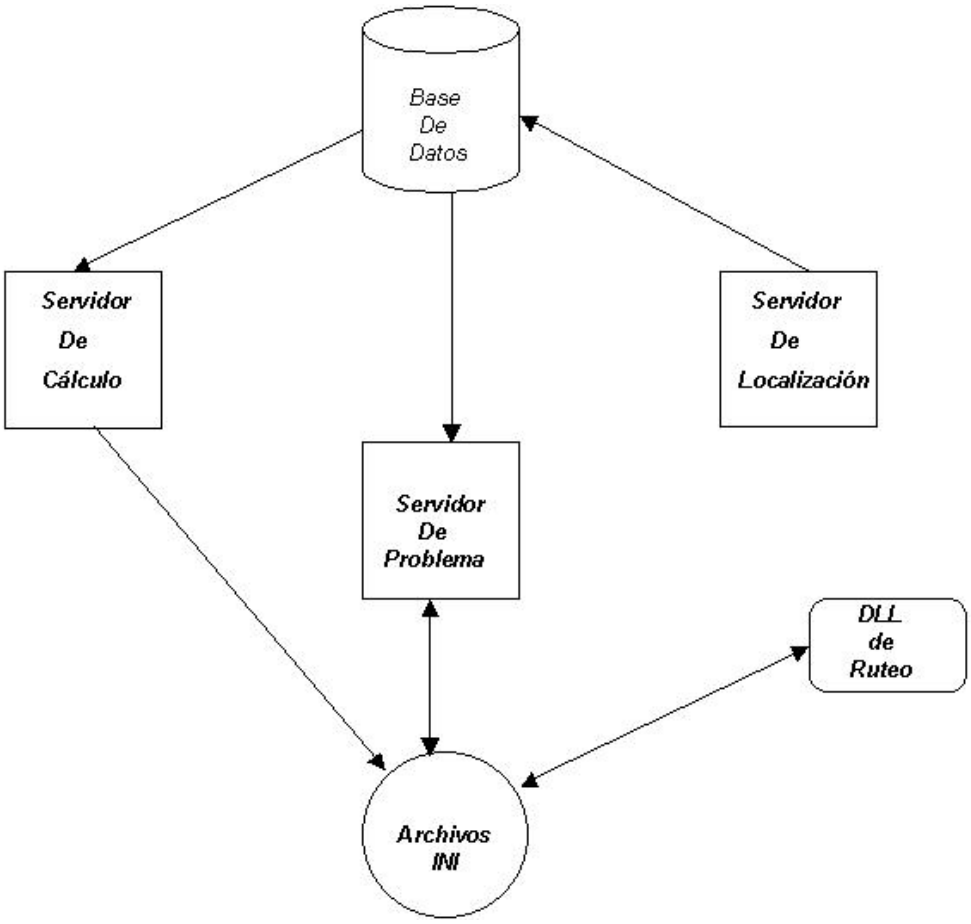
En esta sección se modela la realidad planteada logrando un diseño de alto nivel para el sistema en su globalidad.
En primer lugar se agrupan las funcionalidades en distintos núcleos según su objetivo en el desarrollo del procesamiento de los datos. así se obtuvieron cuatro grandes grupos: las funcionalidades de problema propiamente dicho, las de localización, las de cálculos previos y las que constituyen el proyecto en ASP (paginas web). Con la correcta integración de ellos se obtiene todo el ciclo de procesamiento de los datos.

Servidores

Dada esta división, la mejor forma de implementarla es la creación de distintos servidores, los cuales actúan de forma totalmente independiente.
Una arquitectura de servidores permite la escalabilidad del sistema en varios sentidos, este es un requisito fundamental para un sistema que aspira a ampliarse.

A continuación se presenta un diagrama que describe la interacción entre los servidores, mostrando el flujo de información de los datos. El diagrama incluye la librería de ruteo de forma explicita para mostrar su participación en el sistema. La interacción con la misma es a través de archivos de texto (por mas detalles ver apéndice k: Biblioteca de vínculos dinámicos: Router).

Sobre todos los servidores aquí presentados se encuentra el servidor Web, que por simplicidad no se incluye en este diagrama.



Servidor Problema

En este servidor se resuelve específicamente el problema de ruteo, construyéndose la solución. Se levanta al inicio del servicio.
Veremos ahora una descripción informal de la resolución del problema:
Este servidor se levanta cuando empieza a correr el servicio, y queda esperando por una solicitud de problema. Esta solicitud es enviada una vez que el usuario ha definido completamente los elementos que intervienen en el problema que desea resolver. Estos datos fueron almacenados en las tablas correspondientes de la Base de Datos, de donde van a ser leídos. Una vez que se recogieron todos estos datos, se crea el archivo ini que corresponde a la interfase de entrada con la Dll de Cálculo (ver apéndice k: Biblioteca de vínculos dinámicos: Router) y se la invoca. El servidor queda esperando que la dll termine. Hecho esto, lee el archivo ini de salida de la DLL (ver apéndice k: Biblioteca de vínculos dinámicos: Router) y construye un objeto Ruta, que contiene todos los datos que constituyen la solución del usuario. En base a este objeto se construyen los Reportes que contienen la solución que va a ser desplegada.

Realidad

El servidor Problema será el encargado de resolver el problema de ruteo.
Primero se arma el problema, teniendo en cuenta sus distintos elementos que lo componen.

Los datos para definir los componentes del problema están almacenados. Se leen estos y se arma el problema a resolver de tal modo que el motor de cálculo lo interprete. Una vez que este devuelve la solución, esta se interpreta a su vez y se le agrega al problema su ultimo componente: la ruta solución.
Veamos una descripción de todos los posibles elementos de un problema y sus propiedades.
Separamos los componentes en primarios y secundarios, en base a sí son usados para definir directamente el problema o para definir otros componentes de este.

Nodo

Define un punto del mapa a través de sus coordenadas x e y. Tiene además asociado un tipo que indica si le corresponde a la ubicación de un cliente o de un deposito.
Secundario

Punto

Define un punto del mapa a través de los dos nodos más cercanos a él en la red. Contiene además la distancia y el tiempo entre ellos.
Secundario

Ventana

Define una ventana de tiempo tanto para un sitio una ruta, un camino o un Trayecto. Una ventana de tiempo es un o varios periodos de tiempo definidos por una hora inicial y una hora final. Es durante ese periodo que el cliente debe ser visitado y atendido. Esta solución es para el caso de un solo periodo, pero podría extenderse a n.
Secundario

Sitio

Como sitios se manejan tanto los clientes como los depósitos. Los sitios tiene asociado un tipo que indica si es cliente “C” o deposito “D”. Contiene un nodo y un punto que determinan su ubicación en la red. Se indica damas cual es la demanda, en caso de ser un cliente, o la capacidad si es un deposito. Utiliza una ventana de tiempo, que impone restricciones en el horario de atención para el sitio, y se le asocia además un tiempo de servicio, que es la demora para atender al sitio (cargar o descargar según corresponda). Se explicita también la distancia y la demora en tiempo desde el sitio a los nodos anterior y posterior que su usan para definirlo.
Primario

Arco

Un arco queda determinado por dos nodos, el origen y el destino del arco.
Como propiedades asociadas tiene el tiempo y el costo entre esos nodos.
Secundario

Red

Una red es un conjunto de arcos que determinan una zona para el CPP.
Cada red tiene asociada la cantidad de arcos que la conforman y los arcos en sí.
Primario.

Par

Definimos como par al conjunto de dos sitios, donde uno es de carga y otro es de entrega. Es un conjunto de pares lo que define el DARP.
Primario.

Tipo_Vehiculo

Se almacenan los vehículos agrupados por tipo. Dentro de un mismo tipo están todos los vehículos que pertenecen a la misma flota, es decir, que tienen las mismas propiedades. Estas propiedades son capacidad, costo fijo y costo variable. Se indica también la cantidad de vehículos del tipo.
Primario

Trayecto

Un trayecto determina el recorrido que debe hacerse desde un sitio al siguiente. Tiene asociado el sitio del que sale y al que llega. Cada camino tiene un costo y un tiempo total, así como la demanda total que cubre. También la ventana de tiempo en la cual el trayecto se recorre. Se almacena además el recorrido del trayecto como una línea de MapObjects para poder desplegarla posteriormente en el mapa.
Secundario

Camino

Un camino determina el recorrido que debe hacerse dentro de la ruta cuando se sale de un deposito y se vuelve a un deposito. Un camino esta formado por los distintos trayectos que van entre cada sitio. Tiene asociado el deposito del que sale y al que llega el camino, así como todos los sitios por los que pasa. Se indica el tipo de vehículo que lo recorre (esto tiene sentido para Flota Heterogénea). Cada camino tiene un costo y un tiempo total, así como la demanda total que cubre. También la ventana de tiempo en la cual el camino se transita. Se almacena además el recorrido del camino como una línea de MapObjects para poder desplegarla posteriormente en el mapa.
Primario

Ruta

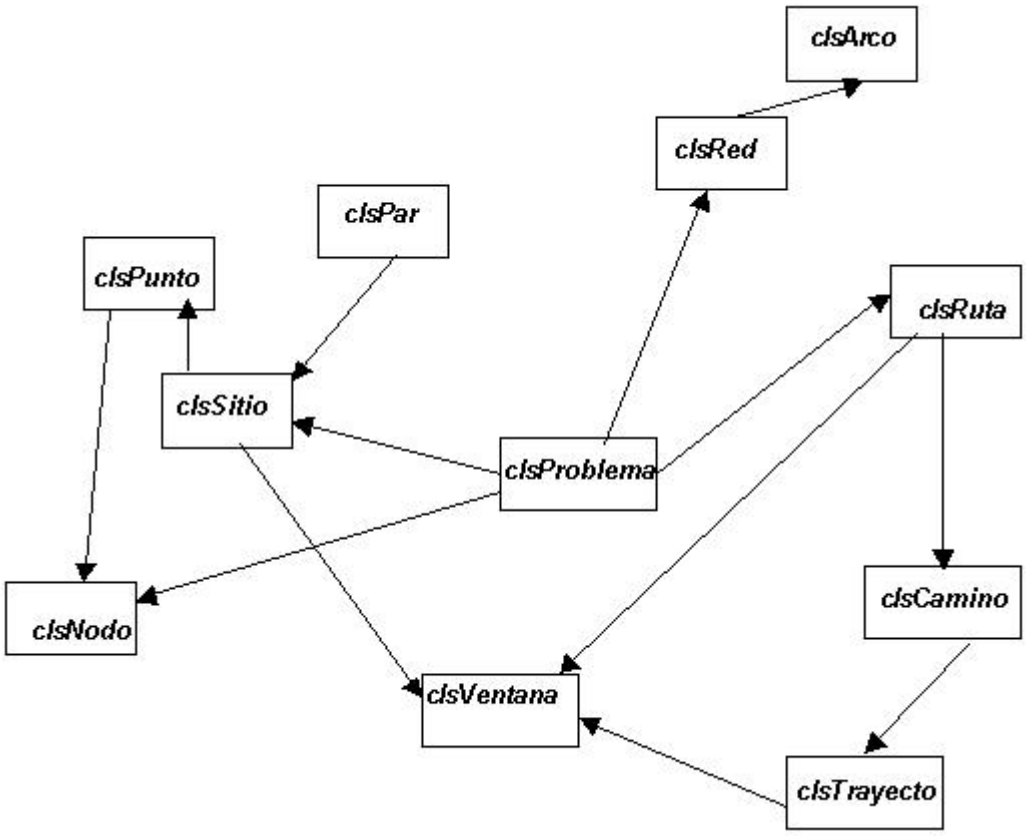
Una ruta es la solución al problema de ruteo. Esta formada por distintos caminos, que indican cada vez que se sale y se llega al deposito, como debe hacerse esa recorrida. La ruta tiene asociado un costo y un tiempo total, así como la demanda total que cubre. También la ventana de tiempo en la cual se completa. Se almacena además el recorrido del de la ruta como una línea de MapObjects para poder desplegarla posteriormente en el mapa.
Primario

Problema

Un problema contiene toda la especificación de los datos necesarios para resolver la solicitud del usuario, así como ciertos datos de este y la solución que se le devuelve. En primer lugar se almacena el tipo de problema, lo cual condiciona a sus demás componentes. Se le asocia el usuario a quien pertenece y la hora a la cual este empezó a correr el problema. Para poder resolverlo, se indica cuales son sus elementos. En el caso que el problema sea un CPP estos van a ser una red y vehículos con sus propiedades

correspondientes (ver apéndice E: Definición de problemas).
Si es un TSP va a manejar sitios (en donde se engloban clientes y depósitos).
Para el DARP los componentes serán pares de sitios, y para el resto de los tipos de problema los componentes son sitios y vehículos con las propiedades asociadas pertinentes. Por ultimo un problema contiene la ruta solución. Si se decide almacenar este problema, se le asocia un escenario.
Primario.

MOR



TDN

Formularios

- **Frm_mapa**

  USES clsProblema
  USES clsRuta

USES clsReportes

PRIVATE

Procedure WebLink1_Request(arguments: out Object, values: out Object);
// Recibe desde el Web una lista de argumentos y sus valores correspondientes, que conforman los datos usuario , tipo_problema y comando del problema a resolver. Resuelve dicho problema y arma los reportes correspondientes.

Procedure actualizarProblem: in();
//Quita del arreglo de problem: in los que hallan estado por m: in de un máximo de minutos.

Procedure actualizarUsuarios()
//Quita de la tabla de usuarios conectados aquellos que hayan relizado su ultima operacion hace m: in de un máximo de minutos.

Procedure Tipo_Problema(problema: in String, Tipo: out Integer)
//Dado un problema como string, devuelve un entero que indica el tipo

EXPORTS

Procedure problema(Usuario: in String, problema: in String);
// Este procedimiento va a resolver el problema especificado en problema, con los //datos del Usuario

Procedure Datos(punto: in MapObjects2.Point, Distancia1: out Float, Distancia2: out //Float, Demora1: out Float, Demora2: out Float, Nod_ori: out MapObjects2.Point, //nod_fin: out MapObjects2.Point)
//Recibe un pmto de MapObjects2 y devuelve los 2 puntos de la red de los que está //más cerca, la Distancia y la Demora a cada uno de ellos.
//Como los cálculos del DijksTra : inocian el punto a la esquina más cercana, esta //función devuelve dicha esquina y un punto muy próximo.

Módulos

• Declaraciones

USES reddll_nuevo.dll
USES kernel32.dll
USES t5vrp.dllPublic

EXPORTS
Function EscribirINI_STR(grupo: in String, Elemento: in String, Valor: in String, Archivo: in String) :Integer
//Escribe en el archivo .ini especificado en ARCHIVO, en el grupo y en el //elemento indicados, el valor . de VALOR Devuleve -1 si la escritura fue correcta

Function LeerINI_Int(grupo : in String, Elemento : in String, Archivo : in String, Por_Defecto : in Integer) : Integer
//Lee del archivo .ini especificado en ARCHIVO, en el grupo y en el elemento
//indicados, el entero correspondiente. En c: ino de que no haya nada en el archivo,
//devuelve el valor indicado en POR DEFECTO.

Function LeerINI_Str(grupo: in String, Elemento: in String, Archivo: in String, Por_Defecto: in String) : String

//Lee del archivo .ini especificado en ARCHIVO, en el grupo y en el elemento indicados, el string correspondiente. En c: ino de que no haya nada en el archivo, devuelve el valor indicado en POR DEFECTO.

Function LeerINI_Float(grupo: in String, Elemento: in String, Archivo: in String, Por_Defecto: in String) : Float
//Lee del archivo .ini especificado en ARCHIVO, en el grupo y en el elemento indicados, el flota correspondiente. En c: ino de que no haya nada en el archivo, devuelve el valor indicado en POR DEFECTO.

Function ComoArray_Numeros(cadena : in String, separador : in String) : in Integer[]
// Transforma un string en un array de números

Function ComoArray_String(cadena: in String, separador: in String) : in String[]
// Transforma un string en un array de strings

Procedure QuickSort(vdata[] : in String, Low : in Long, Hi: in Long)
// Ordena un array usando el algoritmo burbuja

Function Extraer(Lista: string[], Indice: in integer) : in string
//Devuelve el string que ocupa la posición indicada por índice en el arreglo de strings lista

Function Cantidad(Lista: string[]): Integer
//Devuelve la cantidad de "," que hay en un arreglo de strings

- **Reportes**

USES clsProblema

EXPORTS

Procedure Reporte(Usuario: in String, Tipo: in String, Optional : in Integer)
// Crea el tipo de reporte indicado en TIPO para el usuario indicado en USUARIO

PRIVATE

Procedure Rep_Cliente(elProblema: in clsProblema, índice: in Integer)
//Crea el Reporte por Cliente para la ruta del problema indicada en índice.

Procedure Rep_Usuario(Usuario : in String)
//Crea el Reporte por usuario para el indicado en USUARIO.

Procedure Rep_TipoReporte(problema: in clsProblema)
//Crea el reporte TipoReporte para el problema indicado

Procedure Rep_Escenario(problema: in clsProblema)
//Crea el reporte Escenario para el problema indicado

Procedure Rep_FechaHora(elProblema: in clsProblema)
//Crea el reporte FechaHora para el problema indicado

Procedure Rep_NombreVehiculo(key_vehiculo: in Integer)
//Crea el reporte FechaHora para el problema indicado

Procedure Rep_MapaRuta(problema: in clsProblema, índice: in Integer)
//Crea el Reporte por Mapa para la ruta del problema indicada en índice.

Procedure Rep_MapaRutaTotal(problema: in clsProblema)
//Crea el Reporte por Mapa para la ruta del problema indicada en índice.

Procedure Guardar_Escenario(elProblema: in clsProblema)
//Crea el reporte Guardar_Escenario para el problema indicado

Procedure Rep_Chequear_Demanda(elProblema : in clsProblema)
//Chequea que la demanda de los clientes no supere la del depossito. Si no es : ini, se le avisa al usuario y de tod: in form: in se devuelve la ruta solución.

- **Rutear**

USES reddll.dll

EXPORTS

Function Rutea_2Puntos(Origen : IN Object, Destino: in Object) :in MapObjects2.Line
//Devuelve, como una línea de Mapobjects2, el camino más corto entre los dos puntos de entrada

Function Archivo2Linea(Archivo: in String, camino: in Integer) : MapObjects2.Line
// Esta función permite crear a partir de un archivo de salida de la DLL
// un array de puntos. Recibe el nombre del archivo (string) y el numero
// del camino (integer). Revisar la estructura del archivo.

Módulos de clases

- **clsNodo**

EXPORTS

Procedure Let Tipo(vdata: in Integer)
//Setea el tipo con el valor indicado.
//Tipo indica si el nodo pertenece a un cliente ("C") o a un deposito ("D")

Function Get Tipo() : Integer
//Devuelve el tipo del nodo

Procedure Let X(vdata: in float)
//Setea la coordenada X con el valor indicado.

FunctionGet X() : float
//Devuelve la coordenada X del nodo

Procedure Let X(vdata: in float)
//Setea la coordenada X con el valor indicado.

FunctionGet X() : float
//Devuelve la coordenada X del nodo

- **clsPunto**

USES clsNodo

EXPORTS

Function Get distancia() : Float
//Devuelve la distancia entre el nodo posterior y el anterior del punto

Procedure Let tiempo(vdata: out float)
//Setea la demora en tiempo entre el nodo posterior y el anterior del punto

Function Get tiempo() : float
//Devuelve la demora en tiempo entre el nodo posterior y el anterior del punto

Procedure Set nodo_destino(vdata: in clsNodo)
//Setea el nodo destino del punto

Function Get nodo_destino() : clsNodo
//Devuelve el nodo destino del punto

Procedure Set nodo_origen(vdata: clsNodo)
//Setea el nodo origen del punto

Function Get nodo_origen() : clsNodo
//Devuelve el nodo origen del punto

- **clsVentana**

EXPORTS

Function Get Cantidad_Periodos() : Integer
//Devuelve la cantidad de periodos de la ventana

Procedure Agregar_Periodo(Hora_Inicio: in Float, Hora_Final: in Float)
//Agrega el período de Hora_inicio a Hora_final a la ventana

Function Get Periodos(I: in Integer) : in ArregloPriodos
//Devuelve el arreglo de los periodos de la ventana

- **clsSitio**

USES clsNodo
USES clsPunto
USES clsVentana

EXPORTS

Procedure Let Tipo(vdata: in String)
//Setea el tipo de sitio en cliente ("C") o depósito ("D")

Function Get Tipo(): String
//Devuelve el tipo del sitio

Procedure Set nodo(vdata: in clsNodo)
//Setea el nodo : inociado al sitio

Function Get nodo(): clsNodo

//Devuelve el nodo : inociado al sitio

ProcedureLet Tiempo_Servicio(vdata: in Float)
//Setea el tiempo de servicio del sitio
Function Get Tiempo_Servicio(): Float
//Devuelve el tiempo de servicio del sitio

Procedure Set ventana(vdata: in clsVentana)
//Setea la ventana : inociada al sitio

Function Get ventana() : clsVentana
//Devuelve la ventana del sitio

Procedure Let demanda(vdata: Float)
//Setea la demanda del sitio en caso de que sea un cliente

Function Get demanda() : float
//Devuelve la demanda del sitio en caso de que sea un cliente

Function Let Capacidad(vdata : Float)
//Setea la Capacidad del sitio en caso de que sea un deposito

Function Get Capacidad() : in Float
//Setea la Capacidad del sitio en caso de que sea un deposito

Procedure Set punto(l vdata : in clsPunto)
//Setea el punto del sitio con el valor indicado

Function Get punto() : in clsPunto
//Devuelve el punto del sitio

Procedure Letid(vdata : in Integer)
//Setea el id del sitio con el valor indicado

Function Get id() : in Integer
//Devuelve el id del sitio

Function Get Dist_Ant() : in Float
//Devuelve la distancia del sitio al nodo anterior

Procedure LetDist_Ant(vdata : in Float)
//Setea la distancia del sitio al nodo anterior

Function Get Dist_Post() : in Float
//Devuelve la distancia del sitio al nodo posterior

Procedure LetDist_Post(vdata : in Float)
//Setea la distancia del sitio al nodo posterior

Function Get Dem_Ant() : in Float
//Devuelve la demora en tiempo del sitio al nodo anterior

Procedure LetDem_Ant(vdata : in Float)
//Setea la demora en tiempo del sitio al nodo anterior

Function Get Dem_Post() : in Float

//Devuelve la demora en tiempo del sitio al nodo anterior

Procedure LetDem_Post(vdata : in Float)
//Setea la demora en tiempo del sitio al nodo anterior

- **clsTipo_Vehículo**

EXPORTS

Procedure Letid(vdata : in Integer)
//Setea el id del tipo de vehiculo

Function Get id() : in Integer
//Devuelve el id del tipo de vehiculo

Procedure Agregar_Vehiculo(vehiculo : in clstipo_Vehiculo)
//Agrega un vehiculo al tipo de vehiculo

Function Get los_Vehiculos() : in Vehiculos[]
// Devuelve los vehiculos del tipo de vehiculo

Procedure Letcantidad(vdata : in Integer)
//Setea la cantidad de vehiculos del tipo

Function Get cantidad() : in Integer
//Devuelve la cantidad de vehiculos del tipo

Procedure LetCosto_Variable(vdata : in Float)
//Setea el costo variable del tipo de vehiculo

Function Get Costo_Variable() : in Float
//Devuelve el costo variable del tipo de vehículo

Procedure LetCosto_Fijo(vdata : in Float)
//Setea el costo fijo del tipo de vehiculo

Function Get Costo_Fijo() : in Float
//Devuelve el costo fijo del tipo de vehiculo

Procedure LetCapacidad(vdata : in Float)
//Setea la capacidad de cada vehículo del tipo

Function Get Capacidad() : in Float
//Devuelve la capacidad de cada vehículo del tipo

Procedure LetId_key(vdata : in Integer)
//Setea el id del tipo de vehiculo

Function Get Id_key() : in Integer
//Devuelve el id del tipo de vehículo

- **ClsArco**

EXPORTS

Procedure Lettiempo(vdata : in Float)
//Setea el tiempo asociado al arco

Function Get tiempo() : in Float
//Devuelve el tiempo asociado al arco

Procedure Letcosto(vdata : in Float)
//Setea el costo asociado al arco

Function Get costo() : in Float
//Devuelve el costo asociado al arco

Procedure Set nodo_destino(vdata : in clsNodo)
//Setea el nodo destino del arco

Function Get nodo_destino() : in clsNodo
//Devuelve el nodo destino del arco

Procedure Set nodo_origen(vdata : in clsNodo)
//Setea el nodo origen del arco

Function Get nodo_origen() : in clsNodo
//Setea el nodo origen del arco

- **ClsRed**

USES clsArco
USES clsSitio

EXPORTS

Function Get Arcos(i : in Integer) : in clsArco[]
//Devuelve los arcos asociados a la red

Procedure Letcantidad(vdata : in Integer)
//Setea la cantidad de arcos que tiene la red

Function Get cantidad() : in Integer
//Devuelve la cantidad de arcos que tiene la red

Public Procedure Agregar_Arco(Arco : in clsArco)
//Agrega el arco indicado a la red

- **ClsPar**

USES clsSitio

EXPORTS

Procedure Set sitio_entrega(vData : in clsSitio)
//Setea el sitio de entrega del DARP

Function Get sitio_entrega() : clsSitio
//Devuelve el sitio de entrega del DARP

Procedure Set sitio_carga(ByVal vData As clsSitio)
//Setea el sitio de carga del DARP

Function Get sitio_carga() : clsSitio
//Setea el sitio de carga del DARP

- **ClsTrayecto**

USES clsNodo

EXPORTS

Procedure LetSitioDestino(vdata : in Integer)
//Setea el id del sitio destino del trayecto

Function Get SitioDestino() : in Integer
//Devuelve el id del sitio destino del trayecto

Procedure Set Recorrido(vdata : in MapObjects2.Line)
//Setea el recorrido del trayecto

Function Get Recorrido() : in MapObjects2.Line
//Devuelve el recorrido del trayecto

Procedure Set nodo_final(vdata : in clsNodo)
//Setea el nodo final del trayecto

Function Get nodo_final() : in clsNodo
//Devuelve el nodo final del trayecto

Procedure Set nodo_inicial(vdata : in clsNodo)
//Setea el nodo inicial del trayecto

Function Get nodo_inicial() : in clsNodo
//Devuelve el nodo inicial del trayecto
Procedure Letoperacion_final(vdata : in Integer)
//Setea la operacion final

Function Get operacion_final() : in Integer
//Devuelve la operacion final

Procedure Letid_final(vdata : in Integer)
//Setea el id del sitio final

Function Get id_final() : in Integer
//Devuleve el id del sitio final

Procedure Lettipo_final(vdata : in Integer)
//Setea el tipo del sitio final

Function Get tipo_final() : in Integer
//Devuelve el tipo del sitio final

Procedure Letoperacion_inicial(vdata : in Integer)
//Setea la operacion inicial

Function Get operacion_inicial() : in Integer
//Devuelve la operacion inicial

Procedure Letid_inicial(vdata : in Integer)
//Setea el id del sitio inicial

Function Get id_inicial() : in Integer
//Devuleve el id del sitio inicial

Procedure Lettipo_inicial(vdata : in Integer)
//Setea el tipo del sitio inicial

Function Get tipo_inicial() : in Integer
//Devuelve el tipo del sitio inicial

Procedure Set ventana(vdata : in clsVentana)
//Setea la ventana de tiempo del trayecto

Function Get ventana() : in clsVentana
//Devuelve la ventana de tiempo del trayecto

Procedure Lettiempo(vdata : in Float)
//Setea el tiempo asociado el trayecto

Function Get tiempo() : in Float
//Devuelve el tiempo asociado el trayecto

Procedure Letcosto(vdata : in Float)
//Setea el costo asociado el trayecto

Function Get costo() : in Float
//Devuelve el costo asociado el trayecto

- **ClsCamino**

USES clsTrayecto
USES clsNodo
USES clsVentana

EXPORTS

Procedure AgregarTrayecto(trayecto : in clsTrayecto)
//'Agrega un trayecto al camino y une al recorrido del camino el del trayecto

Function Get Cantidad_Trayectos() : in Integer
//Devuelve la cantidad de trayectos del camino

Function Get Trayectos() : in clsTrayecto[]
//Devuelve los trayectos del camino

Procedure Lettrayecto_final(vdata : in Integer)

//Setea el trayecto final del camino

Function Get trayecto_final() : in Integer
//Devuelve el trayecto final del camino

Procedure Lettrayecto_inicial(vdata : in Integer)
//Setea el trayecto inicial del camino

Function Get trayecto_inicial() : in Integer
//Devuelve el trayecto inicial del camino

Procedure Letid_deposito_inicial(vdata : in Integer)
//Setea el id del deposito inicial del camino

Function Get id_deposito_inicial() : in Integer
//Devuelve el id del deposito inicial del camino

Procedure Lettipo_inicial(vdata : in Integer)
//Setea el tipo inicial

Function Get tipo_inicial() : in Integer
//Devuelve el tipo inicial

Procedure Set nodo_final(vdata : in clsNodo)
//Setea el nodo final

Function Get nodo_final() : in clsNodo
//Devuelve el nodo final

Procedure Set nodo_inicial(vdata : in clsNodo)
//Setea el nodo inical

Function Get nodo_inicial() : in clsNodo
//Devuelve el nodo inicial

Procedure Set ventana(vdata : in clsVentana)
//Setea la ventana de tiepo del camino

Function Get ventana() : in clsVentana
//Devuelve la ventana de tiempo dle camino

Procedure Lettiempo(vdata : in Float)
//Setea el tiempo asociado al camino
Function Get tiempo() : in Float
//Devuelve el tiempo asociado al camino

Procedure Letdemanda(vdata : in Float)
//Setea la demanda total del camino

Function Get demanda() : in Float
//Devuelve la demanda del camino

Procedure Set Recorrido(vdata : in MapObjects2.Line)
//Setea el recorrido del camino

Function Get Recorrido() : in MapObjects2.Line

//Devuelve el recorrido del camino

Procedure Letcosto(vdata : in Float)
//Setea el costo dle camino

Function Get costo() : in Float
//Devuelve el costo del camino

Procedure Letvehiculo(vdata : in Integer)
//Setea el vehiculo asociado al camino

Function Get vehiculo() : in Integer
//Devuleve el vehiculo asociado al camino

Procedure Letid_deposito_final(vdata : in Integer)
//Setea el id del deposito final

Function Get id_deposito_final() : in Integer
//Devuelve el id del deposito final

- **ClsRuta**

USES clsVentana
USES clsCamino

EXPORTS

Procedure Set ventana(vdata : in clsVentana)
//Setea La vantana de la ruta

Function Get ventana() : in clsVentana
//Devuelve la ventana de la ruta

Function Get Recorrido() : in MapObjects2.Line
//Devuelve el recorrido de la ruta

Function Get Cantidad_Caminos() : in Integer
//Devuleve la cantidad de caminos de la ruta

Procedure LetCantidad_Nodos(vdata : in Integer)
//Setea la cantidad de nodos de la ruta

Function Get Cantidad_Nodos() : in Integer
//Devuelve la cantidad de nodos de la ruta

Procedure Lettiempo(vdata : in Float)
//Setea el tiempo asociado a la ruta

Function Get tiempo() : in Float
//Devuelve el tiempo asociado a la ruta

Procedure Letdemanda(vdata : in Float)
//Setea la demanda total de la ruta

Function Get demanda() : in Float
//Devuelve la demanda total de la ruta

Function Get Caminos() : in clsCamino
//Devuelve los caminos que forman la ruta

Procedure Letcosto(vdata : in Float)
//Setea el costo asociado a la ruta

Function Get costo() : in Float
//Devuelve el costo asociado a la ruta

Procedure Agregar_Camino(camino : in clsCamino)
//Agrega un camino a la ruta

- **ClsProblema**

USES clsSitio
USES clsVentana

EXPORTS

Procedure LetTipo(Tipo : in String)
//Setea el tipo de problema que se resuelve

Function Get Tipo() : in String
//Devuelve el tipo de problema que se resuelve

Procedure LetUsuario(vdata : in String)
//Setea el usuario

Function Get Usuario() : in String
//Devuelve el usuario

Function Get hora() : in Fecha
//Setea la hora en que se empezó correr el problema

Procedure Lethora(vdata : in Fecha)
//Devuelve la hora en que se empezó a correr el problema

Procedure Agregar_Cliente(Cliente : in clsSitio)
//agrega el cliente indicado a los clientes del problema

Function Get Cantidad_Clientes() : in Integer
//Devuelve la cantidad de clientes con que corre el problema

Function Get Depositos() : in ClsSitio[]
//Devuelvelos depositos del problema

Procedure Agregar_Depositos(deposito : in clsSitio)
//Agrega un deposito al problema

Function Get Cantidad_Depositos() : in Integer
//Devuleve la cantidad de depositos del problema

Procedure Set Tipo_Vehiculos(vdata : in clsTipo_Vehiculos [])

//Setea los tipo de vehículos del problema

Function Get Tipo_Vehiculos() : in clsTipo_Vehiculos []
//Devuelve los tipo de vehículos del problema

Function Get Cantidad_TipoVehiculos() : in Integer
//Devuelve la cantidad de tipos de vehículos del problema

Procedure Agregar_Tipo_Vehiculo(tipovehiculo : in clstipo_Vehiculo)
//Agrega un tipo de vehículo al problema

Function Get Pares() : in clsPar()
//Devuelve los pares asociados al problema (si es un DARP)

Procedure Agregar_Par(par : in clsPar)
//Agrega un par al problema

Procedure Set red(vdata : in clsRed)
//Setea la red del problema

Function Get red() : in clsRed
//Devuelve la red del problema (si es un CPP)

Procedure Agregar_Red(red : in clsRed)
//Agrega una red al problema

Function Get Ruta() : in clsRuta
//Devuelve la ruta solución

Procedure Crear_Ruta()
//Crea la ruta solución (invocando a la dll de ruteo)

Function ConstruyeRuta(Archivo_ini_Salida : in String) : in clsRuta
//Arma la ruta solución a partir de los datos devueltos por la dll de ruteo en el archivo especificado

Servidor de Cálculo

Este servidor calcula usando el algoritmo de Dijkstra los caminos de costo y tiempo mínimos entre los sitios de un usuario y almacena esos datos en un archivo ini. Se levanta al inicio del servicio y queda esperando una solicitud.
Cada vez que un usuario declara un nuevo sitio para el problema que está definiendo, se hace una llamada a este servidor, quien calcula los caminos de costo y tiempo mínimos del nuevo sitio a todos los demás del usuario y de todos los demás a este.
La ventaja de tener un módulo aparte para los cálculos auxiliares mejora enormemente la performance, ya que especifica las funciones de la dll de cálculo, independiente de cómo se calculen los Dijkstra.

Realidad

Para minimizar el costo de los caminos se reduce la información de los sitios a su ubicación en la red. Para eso se manejan los sitios como nodos.

Nodo

Define la ubicación de un sitio en el mapa a través de sus coordenadas x e y. Tiene además asociado el identificador del sitio.
Primario.

TDN

Módulos

- **Declaraciones**

USES reddll_nuevo.dll
USES kernel32.dll
USES t5vrp.dllPublic

EXPORTS

Function EscribirINI_STR(grupo: in String, Elemento: in String, Valor: in String, Archivo: in String) :Integer
//Escribe en el archivo .ini especificado en ARCHIVO, en el grupo y en el elemento indicados, el valor . de VALOR Devuleve -1 si la escritura fue correcta

Function LeerINI_Int(grupo : in String, Elemento : in String, Archivo : in String, Por_Defecto : in Integer) : Integer
//Lee del archivo .ini especificado en ARCHIVO, en el grupo y en el elemento indicados, el entero correpondiente. En c: ino de que no haya nada en el archivo, devuelve el valor indicado en POR DEFECTO.

Function LeerINI_Str(grupo: in String, Elemento: in String, Archivo: in String, Por_Defecto: in String) : String
//Lee del archivo .ini especificado en ARCHIVO, en el grupo y en el elemento indicados, el string correpondiente. En c: ino de que no haya nada en el archivo, devuelve el valor indicado en POR DEFECTO.

Function LeerINI_Float(grupo: in String, Elemento: in String, Archivo: in String, Por_Defecto: in String) : Float
//Lee del archivo .ini especificado en ARCHIVO, en el grupo y en el elemento indicados, el flota correpondiente. En c: ino de que no haya nada en el archivo, devuelve el valor indicado en POR DEFECTO.

Function ComoArray_Numeros(cadena : in String, separador : in String) : in Integer()
// Transforma un string en un array de números

Function ComoArray_String(cadena: in String, separador: in String) : in String[]
// Transforma un string en un array de strings

Módulos de clases

- **clsNodo**

EXPORTS

Procedure Let id(vdata: in Integer)
//Setea el identificador del nodo con el valor indicado.

Function Get id() : Integer
//Devuelve el identificador del nodo

Procedure Let X(vdata: in float)
//Setea la coordenada X con el valor indicado.

FunctionGet X() : float
//Devuelve la coordenada X del nodo

Procedure Let X(vdata: in float)
//Setea la coordenada X con el valor indicado.

FunctionGet X() : float
//Devuelve la coordenada X del nodo

- **clsRaiz**

PRIVATE

Procedure Calcula_Rutas(elUsuario : in string)
//Calcula los costos asociados a los caminos entre sitios de un usuario.
Esos costos se almacenan en un archivo .ini asociado al usuario.

Servidor de Localización

Este Servidor fue cedido íntegramente por la empresa ICA..
Se encarga de realizar la geocodificación (requerimientos que escapan al taller pero necesarios para el producto)
Recibe una calle y una esquina, y devuelve el par x,y correspondiente a esa esquina en la red.

Manejo de los datos

Se va a manejar una tabla por cada componente primario del problema. En estas tablas se guardan todos los datos asociados a los componentes relevantes a la hora de armar el problema.

Al usuario se le permite tener sus datos previamente registrados en el sitio. Así, cuando quiere correr un problema puede usar datos ya registrados, con eventuales modificaciones, y datos nuevos. Para discriminar entre estos do tipos de datos se manejan dos tablas por cada componente, una temporal y otra registrada. Los elementos de la tabla temporal son eliminados una vez que el usuario se desconecta. Estas tablas difieren solo en algunos campos que no inciden en la definición del problema.
Los datos que se van a usar para construir la solución son los que están almacenados en la tabla temporal. Para eso se hacen los traspasos de datos correspondientes si se van a usar elementos registrados.

Comunicación entre Servidores

La comunicación entre los distintos servidores se realiza a través de la base de datos o a través de datos almacenados en memoria (se usan archivos .inis).

Servidor Web

Para la construcción del sitio Web se desarrollo un proyecto ASP, ACTIVE SERVER PAGES. Una Active Server Page es un archivo de extensión .ASP que contiene una combinación de sentencias HTML y scripts lógicos. Para la implementación del sistema cliente-servidor se usa el Internet Information Server.
Cuando el IIS recibe una solicitud http de un archivo ASP, la respuesta HTML final es generada dinámicamente desde las sentencias del HTML estático más la inserción de algún HTML generado por el script.

Realidad

El proyecto ASP se encarga de la comunicación entre el usuario y los servidores como un sistema cliente-servidor.
Constituye la interfase por la cual se van a ingresar los datos que definen el problema y la forma de devolver los resultados.

Los datos principales que se traspasan entre las páginas, y en base a los cuales se va a determinar los html dinámicos son:

Usuario
Determina el usuario que está conectado.

Problema
Indica que problema se quiere resolver.

Elemento
Indica que elemento del problema se esta definido. Se toma como elemento los depósitos, clientes o vehiculos.
Los datos asociados a cada elemento se almacenan en la base de datos.

Implementación

En este proyecto se agrupan las páginas según las funcionalidad que cubren.

General

Index.asp

Está página conforma la presentación del sitio. Es la página inicial y continúe la información general del servicio.

Si quien la visita es un usuario registrado, ingresa su loggin y su clave para poder acceder a sus servicios. Si fue así se chequea que el usuario y su contraseña sean correctos. Estos datos están almacenados en la tabla RtblClientes.

En ese caso de que le usuario esté autorizado se pasa el usuario a la página problema y se despliega esta. De lo contrario se despliega una página de error.

Problema

Problema.asp

La página de definición del problema está dividida en varios frames, que se van cargando según se vayan ingresando los datos que los determinan. Todos los frames están contenidos en la misma página

Recibe el nombre del usuario conectado a través de un formulario y lo ingresa en la tabla de usuarios conectados. Según el tipo de usuario que sea, se despliegan los problemas a los que tiene acceso, así como la entrada al mantenimiento de sus datos registrados.

Una vez que el usuario seleccionó resolver un problema o actualizar su base, problema.asp se llama a si misma, enviando a través de un formulario el usuario y la acción correspondiente, y despliega dinámicamente los elementos que correspondan según el problema seleccionado. Si los elementos ya están definidos, se hace la solicitud web de resolución del problema al servidor problema. Si no es así, se selecciona el elemento a definir y se llama a la página de mantenimiento, pasándole por un formulario el usuario, el problema y el elemento.

Mantenimiento

Mantenimiento.asp

La pagina de Mantenimiento recibe el problema que se quiere resolver, el elemento que se esta definiendo y el usuario al que pertenece. Esto corre en el servidor.

Esta página se conforma con varios frames definidos en diferentes páginas, las cuales se indican en Contenido.htm. Cuando se llama a la pagina de Mantenimiento se despliegan todos los frames que la conforman, produciéndose luego una comunicación dinámica entre ellos en base a la interacción con el usuario, quien va a definir así las propiedades de los elementos.

Contenido.htm

Aquí se asocian las frames que conforman Mantenimiento.asp con las páginas que los definen. Estas páginas son, en orden de frames, La_lista.asp, Nada_registro.asp y Los_elegidos.asp

La_lista.asp

Esta es la primer página de las que constituyen la pagina de mantenimiento. Toma a través de las variables de *Session* el usuario, el problema y el elemento, con lo cual consulta en la base de datos registrados si el usuario en cuestión tiene datos ya cargados para ese problema. Si es así estos se visualizan a través de su nombre, permitiendo al usuario seleccionar uno. Si se selecciona un elemento ya registrado,

se llama a la pagina siguiente pasándole por parámetro dentro del formulario que viaja la identificación de dicho elemento. Se define también en esta pagina el botón de retorno, que permite volver a la pagina de definición de problema. Para esto se envía el formulario correspondiente a Problema.asp.
El acceso a la base de datos registrados y la manipulación de los datos se hace en el servidor, mientras que el envío de los formularios a las demás paginas corre en el cliente.

Nada_registro.asp
En este frame es donde el usuario va a definir las propiedades del elemento. Toma a través de las variables de *Session* el usuario, el problema y el elemento. En caso de ser invocada por La_lista.asp, toma del formulario que esta le envía el parámetro que indica que elemento ya registrado debe tomar.
Si se usa un elemento ya registrado, se toma de la base de datos registrada sus propiedades y se despliegan para poder ser modificadas. Cuando se decide agregar ese elemento se inserta en la base de datos temporal.
Si se va a definir un nuevo elemento, el usuario edita las propiedades que quiere asignarle. En caso de estar definiéndose un sitio, se debe indicar su localización. Para eso se envía un web request al servidor de localización, quien realiza la geocodificación y devuelve las coordenadas de la dirección ingresada.
Una vez que todos los datos están ingresados se envía un formulario con la orden de dar el alta a los_elegidos.asp.
En esta pagina se chequea que no se supere la cantidad máxima de elementos permitida por el problema. Si es así no se hace la solicitud de alta y se le avisa al usuario.

Los_elegidos.asp
Este frame despliega todos los elementos del tipo que se esta considerando definidos por el usuario para este problema.
Esta pagina maneja las altas, bajas y modificaciones de la base.
Cada vez que se define un nuevo elemento, esta página recibe la orden de dar un lata con los datos correspondientes. Realizado esto despliega los datos del nuevo elemento.
Acá se le permite al usuario además eliminar elementos.

Mantenimiento BD

MantenimientoBD.asp
Esta página es análoga a la de mantenimiento, con pequeñas modificaciones.
Por un lado no hay una cantidad máxima a ingresar para ningún tipo de elemento.
Para ingresar los datos de los elementos, se despliegan los asociados a todos los problemas que puede resolver el usuario.
Los frames que la constituyen son análogos a los de Mantanimeinto.asp.

ContenidoBD.htm
La_listaBD.asp
Nada_registroBD.asp
Los_elegidosBD.asp

Escenarios

Un escenario es una "fotografía" de un problema. A través de estas opciones puede almacenar y recrear situaciones que involucren depósitos, clientes y vehículos.

Definir.asp
Definido y resuelto un problema, esta pagina es invocada si se decidió guardarlo como un escenario, para volver a correrlo más tarde.
Recibe en un formulario el identificador del problema.
Así pasa los datos asociados a esa problema de la tabla temporal a la tabla registrada.
Se almacenan también los cálculos de los costos asociados a los caminos.

Recuperar_Escenario
Si en la página de problema se seleccionan los escenarios, esta pagina es invocada, recibiendo en un formulario el tipo de problema y el usuario.
Despliega todos los escenarios de problemas de ese tipo del usuario en cuestión.
Una vez que se elige un problema , se hace un submit a Cargar_Escenario.asp

Cargar_Escenario.asp
Recibe en las variables de sesión el usuario y el tipo de problema, y en un formulario cual es el escenario seleccionado.
Así, traspasa los datos del escenario de la base registrada a la temporal, y hace un submit a la pagina problema.asp.
A partir de ahí el usuario puede ver los elementos, modificarlos si lo desea, y correr el problema.

localización

Localizar.htm
La página de Localización es quien realiza la conexión con el servidor de localización. Corre en el servidor.
Esta página se conforma con varios frames definidos en diferentes páginas, las cuales se indican en Contenido.htm. Cuando se llama a la pagina de Localización se despliegan todos los frames que la conforman, produciéndose luego una comunicación dinámica entre ellos en base a la interacción con el usuario, quien va a definir la ubicación del sitio

Contenido.htm
Aquí se asocian las frames que conforman localizar.htm con las páginas que los definen. Estas páginas son, en orden de frames, el_input.htm, Nombre_de_calle.htm, Nombre_de_esquina.htm y Mapa.htm

el_input.htm
En esta pagina el usuario ingresa un nombre de calle para localizar un sitio.
Se toma ese valor y se realiza el link con el servidor de localización, que devuelve la todas las calles que contienen exactamente el valor ingresado, y todas las esquinas de esas calles.
Se pasa a Nombre_de_calle.htm esas calles.

Nombre_de_calle.htm
Recibe de el_input.htm las calles candidato, las despliega y le permite al usuario seleccionar una.

Nombre_de_esquina.htm
Recibe el nombre de calle seleccionado y despliega todos los nombres de calle esquinas correspondientes a esa calle.
El usuario selecciona una esquina y queda así conformada la dirección.

Se hace el link con el servidor de localización que devuelve los x,y correspondientes a ese sitio.

Mapa.htm
Dados los x,y, despliega el mapa de la zona en la que están situados y ese punto marcado.

Reportes

Reporte_tipo.htm
Recibe del servidor problema los datos de la soluciona l problema ingresado, y los despliega.
Estos datos están organizados por caminos.
Existen links a los datos de cada camino.

Includes

En estas páginas se definen funciones que usan la mayoría de las paginas que conforman el sitio. Por eso se definen una sola vez aquí, y se incluyen cuando es necesario usarlas. Corren en el servidor.

Capítulo
8

Testeo

Testeo

Plan de Verificación y Validación

Objetivos

El plan de verificación y validación del proyecto tiene los siguientes objetivos:

- Detallar las actividades requeridas para preparar y conducir la verificación y validación del proyecto.
- Definir las herramientas de testeo y el ambiente necesario para conducir la verificación y validación del proyecto.

Alcance

Este plan de verificación y validación cubre una completa verificación del proyecto propuesto. Esto incluye documentaciones, actas, informes, software desarrollado, y todo objeto relacionado con el desarrollo del proyecto.

Referencias

Los siguientes documentos fueron usados como fuentes de información para elaborar el plan de verificación y validación:

IEEE Std 1063-1987, IEEE Standard for Software User Documentation.
Normas para la Documentación (ver Apéndice C, Normas de Documentación)
IEEE Std 830-1984, IEEE Standard for Software Requirements Specifications
IEEE Std 829-1983, IEEE Standard for Software Test Documentation.

Ítems de Testeo

Todos los ítems que conforman el proyecto serán testeados durante la verificación y validación del proyecto.
La verificación va a comprobar que el proyecto cumpla satisfactoriamente con sus objetivos. Para realizar esta operación correctamente es necesario enfrentar los resultados obtenidos con los esperados. Para conocer estos últimos nos basamos en los siguientes documentos :

- Análisis de requerimientos para el proyecto

- Descripción del diseño del proyecto
- Estándares internos definidos para los documentos

Software

Se realiza el testeo del software en diferentes etapas secuenciales:

- Módulos de Programa
- Integración del Sistema
- Sistema

El testeo de cada módulo será de Caja Negra.
Luego que cada uno de ellos haya sido testeado y aprobado y luego un testeo de integración.
Una vez que todos los componentes se hayan acoplado correctamente, se hará el testeo del sistema final.

Documentos

Los documentos que forman parte del proyecto constituyen la descripción en lenguaje natural de este, cualidad que los convierte en la vía de comunicación entre quienes desarrollan el producto y el usuario, así como con los tutores y con los encargados de posibles extensiones sobre él.
Además son utilizados por la propia verificación para validar el sistema.
Por estos motivos es importante asegurar su corrección.
Los documentos que serán verificados son los siguientes:

- Especificación del análisis de requerimientos
- Descripción del diseño del sistema

Las actas de reuniones se asumen correctas cuando todos los integrantes de las reuniones que describen las aprueban.

Características que No Serán Testeadas

Se asume la correctitud de determinados documentos, por ser estos o bien de manejo interno propiamente, o bien correspondientes a la distribución del trabajo.
Se establece a su vez que la verificación es correcta.
En base a estos criterios los siguientes documentos no serán verificados:

- Informes de reuniones
- Resúmenes de reuniones de requerimientos
- Planillas de actividades
- La verificación no será verificada.

Enfoque

Se usará la documentación del proyecto para preparar todos los diseños de testeo, casos y especificación de procedimientos. Este enfoque verificará la exactitud y la comprensibilidad de la información en la documentación en aquellas áreas cubiertas por las pruebas.

Criterios de Aceptación y No Aceptación de los Ítems de Testeo

Los siguientes criterios serán tomados en cuenta en el momento de decidir si un ítem aprueba o no aprueba el testeo:

- Número de instancias de prueba sin fallo para los distintos casos de prueba.
- El cumplimiento de las funcionalidades especificadas por el análisis de requerimientos y/o la descripción del diseño del sistema.
- La performance será un factor crítico de aprobación o no del producto, salvo que esté especificado en los documentos de requerimientos y/o diseño del sistema.
- La robustez será, según el caso en mayor o menor medida, un punto a tener en cuenta.

Criterios de Suspensión y Requerimientos de Reasunción

Criterios de Suspensión

Introducción

En esta sección se establecen los criterios bajo los cuales cierto módulo o documento no es considerado correcto, por lo cual se suspende su verificación (criterios de suspensión). La suspensión de la verificación implica una etapa de corrección del componente inmediatamente posterior a la detección del error.
Se definen también que requerimientos debe cumplir ese componente antes de que su verificación sea retomada (criterios de reasunción)

Software

El encontrar alguna falla en alguna de las características que son verificadas en los módulos, en su integración o en el sistema, que impida el testeo en forma correcta de las demás características a verificar será motivo suficiente para que la verificación de la porción del sistema que causa el fallo y las partes que dependan de él, se vea suspendida.

Documentos

Cuando algún documento presente inconsistencias graves en la redacción, incomprensibilidad en lo que intenta describir o ambigüedad en sus términos, su verificación será suspendida.

Requerimientos de Reasunción

Software

Cuando una nueva versión del ítem que causó la suspensión de la verificación esté disponible, luego de que se provocara la suspensión, la verificación será reanudada.

Documentos

Ídem para software.

Plan de prueba del sistema

A continuación se detallan los procedimientos elaborados para la evaluación de los distintos módulos del sistema final.

Router

El testeo de la librería de ruteo **Router** se desarrolla partiendo de un prototipo funcional continuando por un versionado incremental a medida que sea requerido, tanto por reportes de testeo como por agregación de funcionalidades.

Dada la complejidad de la especificación manual de parámetros de entrada para el router, en su fase de implementación el testeo se realiza con caso minimales que cubran la mayor cantidad de casos límite del problema. Luego de esto se procede al testeo a fondo a medida que se integra con la interfase visual de especificación de problema.

Dicho lo anterior es que la fase de testeo del Router se desarrolla en dos instancias en constante ciclo: Prueba mínima en implementación y Prueba exhaustiva en integración.
La primera tiene lugar a medida que la implementación avanza. Es realizada por los implementadores y consta de casos limites de pequeño tamaño, con el único fin de corregir errores de implementación y no para testear el rendimiento de los algoritmos. La prueba exhaustiva en integración la realizan los usuarios del sistema, emulando de la mejor forma situaciones reales.

Dado que la selección de los algoritmos a utilizar y su aprobación fue realizada en una etapa anterior a este taller, no es necesario testear la performance de dichos algoritmos con bibliotecas estándar comparativas (para ver testeos sobre los algoritmos utilizados referirse a [5]).

Interfase

Mientras que el desarrollo del Router partía de una versión funcional, esto es distinto en el caso de la interfase, la cual se desarrolla en versiones por funcionalidad.

Dado esto, el testeo se realiza simultáneamente con el desarrollo de las distintas funcionalidades de forma indirecta durante la constante integración del Sistema.

Entorno de prueba del sistema

Router

El entorno de testeo para la librería Router es muy variado. El motivo principal para tal decisión es el de lograr amplia utilización de la librería de ruteo para futuras aplicaciones, requerimiento del actual proyecto3.
Los entornos utilizados se clasifican según la instancia del ciclo de desarrollo en el que se encuentre el sistema:

Instancia	Software	Hardware
Prueba mínima en implementación	Microsoft Windows 98	Pentium II 350 Mhz, 64 Mb Ram.
	Microsoft Windows 98	Pentium II 400 Mhz, 128 Mb Ram.
	Microsoft Windows NT 4.0	Pentium I 250 Mhz, 32 Mb Ram.
	Microsoft Windows 2000	Pentium II 350 Mhz, 64 Mb Ram.
	Microsoft Windows 2000	Pentium II 400 Mhz, 128 Mb Ram.
Prueba exhaustiva en integración	Microsoft Windows 2000	Pentium II 350 Mhz, 128 Mb Ram.
	Microsoft Windows 2000	Pentium II 400 Mhz, 128 Mb Ram.

Interfase

El entorno utilizado para testear la interface se redujo mayoritariamente al entorno de desarrollo (Microsoft Windows 2000, Pentium II 450 Mhz, 128 Mb Ram.).
Se presentaron dos excepciones a lo dicho anteriormente, donde el sistema completo se instaló, de forma exitosa, en otros entornos.

Especificación de caso de testeo

La especificación de los casos de testeo corresponde a la empresa ICA (Ingenieros Consultores Asociados) partiendo de casos reales de distribución/recolección de bienes y servicios y a experiencias anteriores con sistemas similares.

Prueba mínima en implementación

Dada la complejidad de la especificación manual de parámetros de entrada para el router, en su fase de implementación el testeo se realiza con caso minimales que cubran la mayor cantidad de casos límite del problema. Para esta etapa los desarrolladores realizaron sus propios casos de testeos, siendo en mayor o menor medida exhaustivos en función de la complejidad del problema.
En esta etapa se considera satisfactorio el resultado de un testeo cuando la solución devuelta por el algoritmo respeta todas las restricciones correspondientes al problema en cuestión y coincide con la solución teórica esperada.

[3] Dado que la librería tiene el formato de Dll (dinamic link library) quedan excluidos otros sistemas operativos , como Linux.

A continuación se resumen los testeos realizados en esta instancia según el tipo de problema:

VRP (Vehicle Routing Problem).

Dado que este problema es el mas utilizado por el resto de los algoritmos se testea exhaustivamente los casos limites, cubriendo satisfactoriamente la totalidad de las combinaciones de capacidad de los vehículos, capacidad del deposito, demanda de los clientes, ubicación geográfica, etc. (siempre considerando un deposito y dos clientes.)
Además de estos casos se incluyeron otros donde se variaba aleatoriamente la cantidad de clientes y depósitos conduciendo también a buenos resultados.

Cantidad de pruebas de casos limites:	15
Tiempo de corrida medio:	0.25 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	Ninguno

Una observación importante es que este problema es también testeado durante la prueba de los problemas que dependen de él.

MDVRP (Multi-Depot Vehicle Routing Problem).

Este problema es similar al anterior, pero la cantidad de depósitos es mayor a 1. Se testea exhaustivamente el caso de dos depósitos

Cantidad de pruebas de casos limites:	10
Tiempo de corrida medio:	0.40 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	(1)

VRPTW (Vehicle Routing Problem with Time-Windows).

En este problema, cada cliente tiene un rango de tiempo en el que el cliente puede ser atendido.
Debido a que es la base para los demás problemas de ruteo con ventana de tiempo este problema se estudia profundamente en la esta etapa de desarrollo. Se prueban innumerables cantidad de opciones, con un máximo de cuatro clientes (principalmente con dos) cambiando posición geográfica, ventana de tiempos, demanda, capacidad del deposito y de los vehículos, etc

Cantidad de pruebas de casos limites:	30
Tiempo de corrida medio:	0.40 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	Ninguno

Una observación importante es que este problema es también testeado durante la prueba de los problemas que dependen de él.

MDVRPTW (Multi-Depot Vehicle Routing Problem with Time-Windows).

Este problema es similar al anterior solo que considera varios depósitos. Es esto ultimo en lo que se basan los principales casos de testeo de esta instancia.

Cantidad de pruebas de casos limites:	3
Tiempo de corrida medio:	0.50 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	(ver Apéndice E: Definición de Problemas)

VRPHF (VRP Flota Heterogénea).

Ídem VRP, pero en este caso los vehículos tienen diferente capacidad. Este problema presenta gran cantidad de posibles errores por lo que se realiza un prueba prefunda de las principales variantes (siempre considerando casos sencillos, con un máximo de tres tipos de vehículos).

Cantidad de pruebas de casos limites:	15
Tiempo de corrida medio:	0.25 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	Ninguno

TSP (Travelling Salesman Problem).

Este problema es un caso particular del VRP, donde se tiene un sólo depósito y un solo vehículo, y donde la demanda de los clientes es cero. Sin embargo, y pensando en futuras divergencias entre ambos algoritmos se opta por una implementación por separado. Por tanto este algoritmo se testea de forma profunda

Cantidad de pruebas de casos limites:	15
Tiempo de corrida medio:	0.15 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	Ninguno

MTSP (M - Travelling Salesman Problem).

El **MTSP** es una pequeña variante del **TSP** en donde la cantidad de rutas de la solución esta dada por la cantidad de vehículos del problema. Dada esta situación se opta en esta instancia por testear este problema solo en función de la cantidad de vehículos. De esta forma los casos testeados se remitieron a cuatro clientes (o menos) y una variación de cantidad de vehículos en todo el rango útil:

Cantidad de pruebas de casos limites:	20
Tiempo de corrida medio:	0.10 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	(1)

(1). Un problema conocido de este algoritmo es que las ultimas rutas tienen siempre menos clientes que las primeras.

DARP (Dial-a-ride Problem).

Este problema resuelve el caso en que cada cliente tiene un punto de carga y un punto de entrega, todo dentro de una ventana de tiempo. La solución implementada para este problema presenta casos patológicos donde los resultados no son los esperados. Dichos casos son conocidos y aceptados por parte del usuario (a efectos de este proyecto) y no se incluyen en este análisis.
Al igual que casos anteriores este problema deriva del **VRPTW**, por lo que gran parte del testeo se realiza en esa instancia
Se testea el caso de dos clientes (dos de carga y dos de descarga)

Cantidad de pruebas de casos limites:	4
Tiempo de corrida medio:	0.15 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	(ver Apéndice E: Definición de Problemas)

CPP (Chinese Postman Problem).

El problema del CPP es hallar un recorrido tal que pase por todos los clientes y su costo sea mínimo, sin embargo la variante del CPP resuelta por este software recibe las demandas de los clientes, no de los arcos. Dada esta variante, este algoritmo mayoritariamente se encuentra resuelto por el **VRP** por lo que gran parte del testeo se realiza en tal momento. Para este problema en particular se realizan solamente los siguientes casos (de cuatro clientes como máximo):

Cantidad de pruebas de casos limites:	4
Tiempo de corrida medio:	0.15 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	Ninguno

SVRP (Stochastic Vehicle Routing Problem).

Es una generalización del problema clásico VRP, donde los clientes tienen demanda no determinista. Dada la inexistencia de experiencia y de casos de prueba conocidos por parte del usuario, el testeo y calibración de este problema se realiza solo de forma superficial. De todas formas este algoritmo mayoritariamente utiliza el problema **VRP** por lo que gran parte del testeo se realiza en tal momento.

Cantidad de pruebas de casos limites:	3
Tiempo de corrida medio:	0.15 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	Ninguno

DIJKSTRA.

Se presentan distintas variantes del dijkstra, donde se pueden resaltar: con ponderación de ángulos, pseudo-óptimos (DFS, BFS), completo(de uno a todos) o parcial (de uno a algunos), con mejoras de eficiencias.
Esta funcionalidad de la librería es secundaria y no utilizada actualmente por el sistema. Por esta razón, y dada la heterogeneidad de las opciones realizadas, se presentó inviable en tiempo realizar un testeo profundo; por lo que en acuerdo con el usuario se da prioridad a la ponderación del ángulo. Para este caso tenemos (red de Montevideo, de la fuente a solo un destino):

Cantidad de pruebas de casos limites:	5
Tiempo de corrida medio:	1 seg.
Resultado:	Satisfactorio.
Errores conocidos, no solucionados:	(2)

(2) Se considera que los tiempos de ejecución son inaceptables (hecho que no es relevante porque nuestro sistema no utiliza esta funcionalidad).

Prueba exhaustiva en integración.

En la prueba exhaustiva de integración se prueban los problemas de ruteo de la librería dinámica y la interface grafica de forma simultanea. Esto permite que el testeo pueda ser realizado por personas no calificadas, desplegando un resultado de fácil interpretación.

En particular se realizaron innumerables pruebas de testeo basándose en casos reales por parte del usuario, en el propio ambiente final del producto, dando resultados altamente satisfactorios tanto en performance como en optimización.

Informes del testeo

Router

A continuación se detalla la sucesión de reportes de testeo, especificación del error.

Fecha	Problema	Descripción	Observaciones
19/10/2000	MDVRP	No se está considerando correctamente el radio especificado para la asignación de los clientes "frontera"	Se soluciona
23/10/2000	VRP	Considerar el hecho que un vehículo pueda realizar más de un viaje	Se decide que la asignación de vehículos se realice por el sistema exterior.
31/10/2000	TSP	En algunos casos devuelve más de una ruta	Se soluciona
31/10/2000	MTSP	Error en la concepción del problema	Se recodifica
20/11/2000	TODOS	El sistema se cae con más de 10 clientes	Se soluciona
25/11/2000	VRP	No contempla correctamente las capacidades de los depósitos.	Error de testeo
15/12/2000	VRP	No considera correctamente las capacidades de los vehículos	Se soluciona
14/12/2000	MTSP	Resultados no satisfactorios	Se mejora
18/12/2000	VRPHF	Resultados no satisfactorios	Se mejora
18/12/2000	MDVRP	No asigna de forma satisfactoria clientes a depósitos	Se corrige mediante el radio de asignación
23/12/2000	VRPTW	Asigna rutas de forma no satisfactoria	El error radica en la especificación de las unidades de costo y tiempo en el sistema exterior.

Capítulo
9

Conclusiones

Conclusiones

Resumen de la Contribución del Documento

Este documento detalla de forma sucinta el desarrollo y conclusiones obtenidas durante el transcurso del Taller V del grupo de ingeniería en computación integrado por: Adolfo Agorio, Lea Kaufman y Pablo Rodríguez. En el se encuentran detallados los fundamentos teóricos desarrollados y aprendidos por el grupo, al igual que todo lo referente al producto realizado, como ser: el análisis de requerimientos, el diseño, la implementación, etc.

Resultado Principal

Como principal logro de este documento se tiene el desarrollo completo e integral de una solución para Internet.
Esto representa la implementación de un sitio Web que ofrece una amplia gama de soluciones a problemas de ruteo. Prácticamente no existen en el mundo servicios tan completos a través de Internet.
Dicha solución abarca la mayoría de los requerimientos presentados por el usuario. Además la arquitectura de la misma esta pensada para su fácil adaptabilidad a un software monousuario de oficina.

También se desarrolla en este documento una solución completa al problema SVRP (Stochastic Vehicle Routing Problem) y la implementación de la misma.

Aplicaciones del Resultado

Las aplicaciones que tiene el producto realizado son muy importantes y variadas.
La aplicación mas importante y directa que se puede citar, es la comercialización de un sitio completo para Internet con las mismas funcionalidades y arquitectura desarrolladas por el grupo de taller.

Además, tanto el análisis de requerimientos como la modularización del software contemplan de forma completa todas las funcionalidades necesarias por un software de oficina, Pudiendo ampliar en esta dirección el producto actual, partiendo de un estudio detallado y código fuente reutilizable.

Por ultimo, y desde un punto de vista teórico, se tiene mayor cobertura y conocimiento de problemas de ruteo resueltos, incluyendo el problema de demanda estocástica (SVRP) aplicable a realidades muy variadas como la recolección de residuos, la venta puerta a puerta de productos portables, etc.

Dirección de Investigaciones Futuras y Cuestiones Abiertas

La principales investigaciones futuras deben realizarse con la intención de realizar una aplicación para monousuario de escritorio, cumpliendo totalmente los requerimientos de una aplicación de esta naturaleza.
En el capitulo "Análisis de Requerimientos" se plantea un análisis de requerimientos completo para una aplicación de este tipo. El motivo de este documento es ser lo suficientemente genérico como para sentar un antecedente y no perder el fin de todo el informe, que es el realizar una aplicación comercial.
No todos los puntos descritos en ese Análisis han sido cumplidos, ya que se sufrió un cambio de requerimientos sobre la marcha, que enfoco al taller al desarrollo de una aplicación en Internet y no a una de oficina, para la cual estaba pensado el Análisis inicial.
A su vez, el producto desarrollado para Internet, y la arquitectura sobre la cual se basa, son pensados a partir de dicho análisis, y con el fin de alcanzar el mismo.

También se puede investigar para mejorar (en tiempos de respuesta) el calculo de los caminos óptimos entre dos puntos de la red, es decir, mejorar el desempeño del algoritmo de Dijkstra en la .dll. Dicha mejora permite unificar el servidor problema (encargado de resolver los problemas de ruteo) y el servidor de calculo (encargado de calcular los caminos óptimos entre los clientes y depósitos), obteniendo una solución mas compacta y maleable. Este objetivo se pospuso para el futuro debido a que la aplicación tiene una arquitectura para Internet, y es importante la separación en módulos para mejorar los tiempos de respuesta, mientras que si pensamos en una aplicación de oficina dicha unión sería muy útil por modularidad.

En cuanto a los algoritmos de ruteo, estos han sido pensados para la integración de los mismos entre sí, pudiendo obtenerse de esta manera, un motor mas potente o adaptar el producto a las demandas particulares de un cliente. A continuación se detalla una tabla donde se especifican los problemas que pueden ser integrados de forma muy simple:

	TSP	MTSP	VRP	SVRP	VRPTW	VRPHF	MDVRP	MDVRPTW	CPP	DARP
TSP	-	-	Si	Si	Si	Si	Si	Si	No	No
MTSP	-	-	Si	Si	Si	Si	Si	Si	No	No
VRP	Si	Si	-	-	-	-	-	-	No	No
SVRP	Si	Si	-	-	Si	Si	Si	Si	No	No
VRPTW	Si	Si	-	Si	-	Si	Si	-	No	No
VRPHF	Si	Si	-	Si	Si	-	Si	Si	Si	Si
MDVRP	Si	Si	-	Si	Si	Si	-	-	Si	Si
MDVRPTW	Si	Si	-	Si	-	-	Si	-	No	No
CPP	No	No	No	No	No	Si	Si	No	-	No
DARP	No	No	No	No	No	Si	Si	No	No	-

Tabla de Integración

De todas formas se mantuvo una separación de los problemas para desminuir complicaciones.

Por ultimo, con respecto a nuevos problemas de ruteo por resolver, se deja como pendiente el VRP con múltiples compartimentos.

Se implementa el sistema de forma tal que sea fácilmente extensible para la resolución de otras aplicaciones de logística, como ser mantenimiento de stock, ruteo por demanda.

Otro posible punto a tratar es la detección automática del problema a solucionar por parte del sistema, a partir de los datos definidos por el usuario.

Bibliografía

[1] JAILLET,P; ODONI, A. R.. "*The Probabilistic Vehicle Routing Problem*", smd.

[2] BODIN, LAWRENCE; GOLDEN, BRUCE; ASSAD, ARJANG. *"Routing and Scheduling of vehicles and crews: the state of the art", 1989,* smd

[3] VALDURIEZ, PATRICK. Project Rodin (INRIA); *"Some hints to improve Writting of technical papers" ;* smd

[4] ALONZO , IGNACIO; BURGUEÑO, FERNANDA*; "Informe Final Estructura Genérica", 1999*

[5] PEDEZERT, CARLOS; BARREIRO, JAVIER; SOSA, NICOLÁS; *"Informe Final Ruteo de Vehículos", 1999.*

[6] IEEE Std 1063-1987, IEEE Standard for Software User Documentation.

[7] IEEE Std 830-1984, IEEE Standard for Software Requirements Specifications

[8] IEEE Std 829-1983, IEEE Standard for Software Test Documentation.

[9] CABAÑA, E:M. *"Probabilidad y estadística*", Oficina de publicaciones CEI, Montevideo, 1986.

[10] University of Karlsruhe *"Software Library for Operations Research"* http://www.wior.uni-karlsruhe.de/Bibliothek/Software_for_OR/Vehicle_Routing/com/aacom.html (4/99)

[11] ArcLogistics Route, Evaluation Edition; *www.esri.com*

Apéndice
A

Apéndice A: Control de Reuniones

Control de Reuniones

Primera Reunión Administrativa

Fecha

7/4/00

Participantes

Omar Viera, Leonardo Loureiro, Adolfo Agorio, Lea Kaufman, Pablo Rodríguez

Objetivo

Definición de los requerimientos para el proyecto por parte del usuario representante de ICA Leonardo Loureiro.

Temas Tratados

Funcionalidades de Ruteo

Agregar a las funcionalidades del taller 99 de ruteo de vehículos la resolución de los siguientes problemas:

- Demanda probabilística de los clientes, donde todos los clientes tienen la misma distribución de la demanda.
- Ruteo con compartimentos.

Funcionalidades Extra

- Reconocer que tipo de problemas debe resolver el software a partir del conjunto de datos de entrada que recibe.
- Retroalimentación: permitir la actualización del software a través de la contraposición de datos extraídos de la realidad e ingresados al software contra lo calculado.

Configuración del producto

- Configuración inicial a través de un código para establecer, a la hora de la instalación, a cuáles de todos las soluciones que el software brinda el usuario va a tener acceso.
- Durante la configuración se debe establecer manualmente la relación entre los campos que utiliza el producto y los nombres bajo los cuales el usuario tiene almacenada la información requerida. Esta información va a ser referida tanto a, por ejemplo, vehículos o pedidos, como a los campos asociados a los mapas. No existen restricciones con respecto a los nombres de las tablas originales.

Manejo de los mapas

- El software deberá ser independiente del mapa. Esto es, de que ciudad o zona se represente.
- Restricción: los datos que se van a usar para manejar los mapas tienen que estar en formato Shapefile.
- Se asumirá que el punto que representa al cliente sobre el mapa ya existe.

Aplicación en Internet

Mapeo de la aplicación a través de IMS.

Herramientas

Preferencia por Visual Basic y Visual C++

Tareas Mediatas Requeridas

- Presentación de un plan de trabajo.
- Presentación de un acta de la reunión con el usuario.

Segunda Reunión Administrativa

Fecha

27/4/00

Participantes

Omar Viera, Leonardo Loureiro, Luis Calderón, Adolfo Agorio, Lea Kaufman, Pablo Rodríguez

Objetivo

Definición de los requerimientos para el proyecto por parte del usuario representante de ICA Leonardo Loureiro. Diagramación general de los grandes temas involucrados en el proyecto.

Temas Tratados

Nuevos Requerimientos

- Vehículos: Administración de los datos de los vehículos. Manejar una BD propia o establecer una interfase con una ya existente
- Clientes: Investigar en el área de "AddressMatching", y como esto puede ser una limitante para nuestro trabajo.
- Funcionalidades geográficas: Moverme en el mapa. Zoom sobre cierta zona del mapa. Consulta sobre datos de cierta zona, por ejemplo ver una ruta por su nombre. Se recomienda usar Moview de MapObject para implementarlas .
- Estudio de la estructura para representar las calles: sentido, etc
- Impresión de reportes: Un reporte por cada vehículo, con su ruta correspondiente. Además un reporte por cada parada (cliente), con ruta hasta esa parada, hora de llegada, etc

Modularizacíon

Dividir los problemas en distintos módulos (.dll). Dichas .dll deben tener parámetros ocultos.

SVRP

Nos limitaremos al caso de oferta determinística.

Consideraciones Generales

- Usar como base para este proyecto la tesis de estructura genérica. (va a ser proporcionado por el usuario) no la tesis anterior de algoritmos genéricos de ruteo.
- Nombre del producto S.A.R. (Sistema Automatizado de Ruteo).
- Resaltar el hecho que nuestro proyecto es un trabajo novedoso (resolvemos problemas de ruteo nuevos)

Bibliografía recomendada

Object Orient Análisis, Modelling and Design. (J. Rumbauj)

Herramientas

Se determino que las herramientas a utilizar serán Visual Basic, Visual C++, MapObject 2.0 (proporcionado por el usuario), MapObject IMS (proporcionado por el usuario, para la implementación en Internet).

Tareas Mediatas Requeridas

- Rehacer un plan de trabajo.
- Presentación de un acta de la reunión con el usuario.
- Presentar una primera solución al problema SVRP.
- Estudio de la tesis anterior: "Estructura genérica para algoritmo de ruteo de vehículos"
- Interiorizarse con el problema de vehículos compartimentados.

Geocodificación (AddressMatching)

Se pretende una investigación en este tema, donde aparezcan lo siguiente:

- Plantear problemática
- Marco teórico (básicamente dos partes: analizador lexicográfico y representación grafica)
- Herramientas implementadas para resolver este problema.
- Importancia de la correctitud de las fuentes de datos y sus costos.

Tercera Reunión Administrativa

Fecha

10/5/00

Participantes

Omar Viera, Leonardo Loureiro, Luis Calderón, Adolfo Agorio, Lea Kaufman, Pablo Rodríguez

Objetivo

Prioridades en los requerimientos para el proyecto por parte del usuario.

Temas Tratados

Consideraciones Generales

- Se presento a discusión las herramientas a utilizar para desarrollar el producto.

Tareas Mediatas Requeridas

- Realizar en forma exhaustiva, por parte del grupo de taller, un análisis de los requerimientos presentados por el usuario.
- Presentación de un acta de la reunión con el usuario.

Cuarta Reunión Administrativa

Fecha

4/8/00

Participantes

Omar Viera, Leonardo Loureiro, Luis Calderón, Pablo Rodríguez

Objetivo

Prioridades en los requerimientos para el proyecto por parte del usuario.

Temas Tratados

Consideraciones Generales

- Usar MapObject 2.0 dentro de Visual C++ para lograr una interfase con el usuario a través del formato .shp y .dbf.
- Es necesario, por parte del usuario, resolver el problema MTSP por lo que se requiere la dll del problema VRP rápidamente.

Herramientas

Se determino la utilización de MapObject 2.0 dentro de Visual C++.

Tareas Mediatas Requeridas

- Terminar la .dll del problema VRP.
- Presentación de un acta de la reunión con el usuario.
- Reunión con Luis Calderón para especificar la interface de la .dll.

Apéndice **B**

Apéndice B: Plan de Trabajo y Avances del Proyecto

Plan de Trabajo y Avances del Proyecto

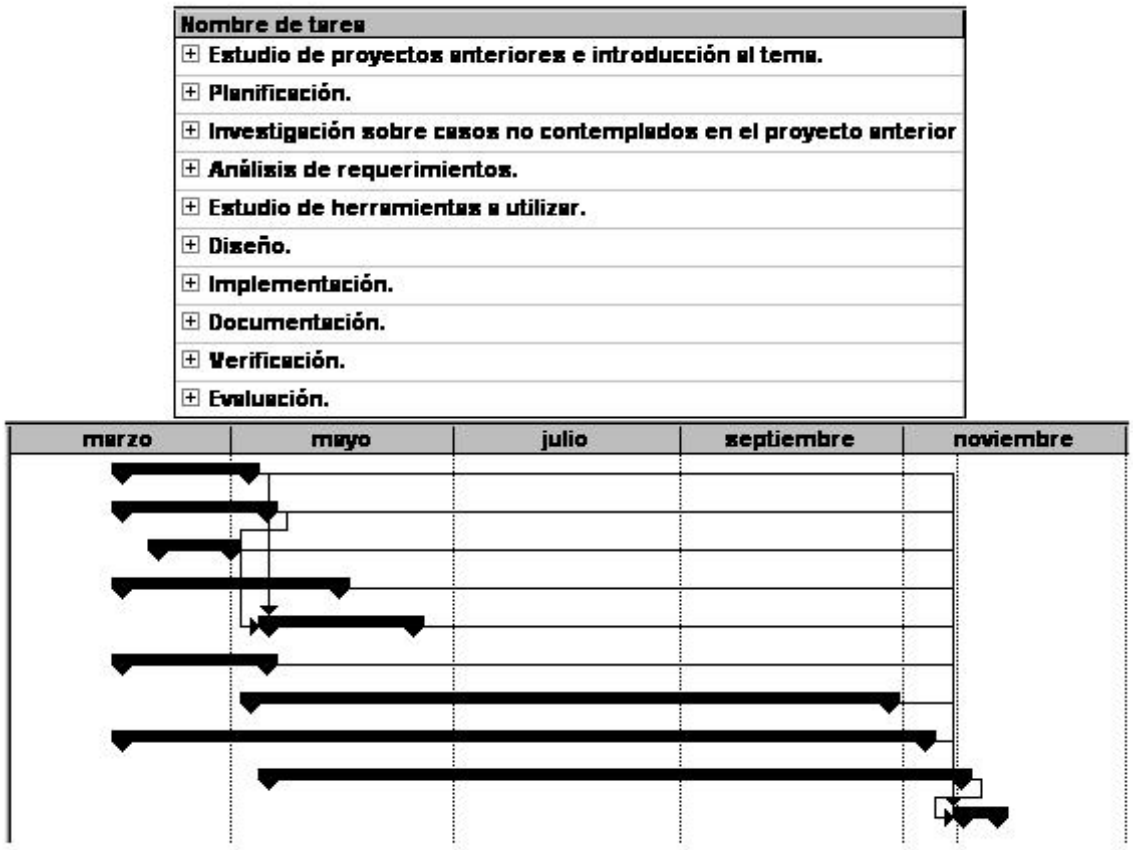
Identificación de tareas

Tarea	Duración	Comienzo	Fin
1. Estudio de proyectos anteriores e introducción al tema.	7 sems	01-Abr	05-May
1.1 Estudio de los distintos casos del problema del VRP contemplados en el proyecto anterior.	2 sems	01-Abr	10-Abr
1.2 Estudio de proyecto de estructura genérica	2 sems	01-Abr	10-Abr
1.3 Introducción a los SIG.	2 sems	01-Abr	10-Abr
1.3.1 Estudio del formato ShapeFile de ESRI y .dbf.	1 sem	01-Abr	05-Abr
1.3.2 Familiarización con el uso de MapObjects 2.0.	2 sems	01-Abr	10-Abr
1.3.3 Familiarización con el uso de MapObjects IMS.	2 sems	01-Abr	10-Abr
1.4 Análisis de las estructuras de datos y algoritmos utilizadas en el proyecto anterior.	5 sems	11-Abr	05-May
2 Planificación.	8 sems	01-Abr	10-May
2.1 Identificación de tareas.	4 sems	01-Abr	20-Abr
2.2 Planificación preliminar.	1 sem	06-May	10-May
2.3 Análisis del desarrollo del proyecto.	0 sems	01-Abr	01-Abr
3 Investigación sobre casos no contemplados en el proyecto anterior.	4 sems	11-Abr	30-Abr
3.1 Demanda no determinística.	4 sems	11-Abr	30-Abr
3.1.1 Especificación de problema (requerimientos).	2 sems	11-Abr	20-Abr
3.1.2 Desarrollo de una solución.	4 sems	11-Abr	30-Abr
3.2 Vehículos compartimentados.	4 sems	11-Abr	30-Abr
3.2.1 Especificación de problema (requerimientos)	2 sems	11-Abr	20-Abr
3.2.2 Desarrollo de una solución.	4 sems	11-Abr	30-Abr
4 Análisis de requerimientos.	12 sems	01-Abr	30-May
4.1 Captación de requerimientos del usuario	12 sems	01-Abr	30-May
4.2 Especificación de los requerimientos de usuario.	8 sems	01-Abr	10-May
4.2.1 Especificación no formal de los requerimientos de usuario.	0 sems	01-Abr	01-Abr
4.2.2 Especificación formal de los requerimientos de usuario.	8 sems	01-Abr	10-May
4.2.2.1 Análisis de las funcionalidades necesarias.	6 sems	01-Abr	30-Abr
4.2.2.2 Síntesis de las funcionalidades .	2 sems	01-May	10-May
4.3 Compatibilización Requerimientos-Herramientas.	1 sem	11-May	15-May
5 Estudio de herramientas a utilizar.	8 sems	11-May	19-Jun
5.1 Estructura genérica vs. Tesis anterior	4 sems	11-May	30-May
5.2 MapObjects 2.0.	4 sems	11-May	30-May
5.3 MapObjects IMS.	4 sems	31-May	19-Jun
5.4 Desarrollo de Dll's en visual C++.	3 sems	11-May	25-May

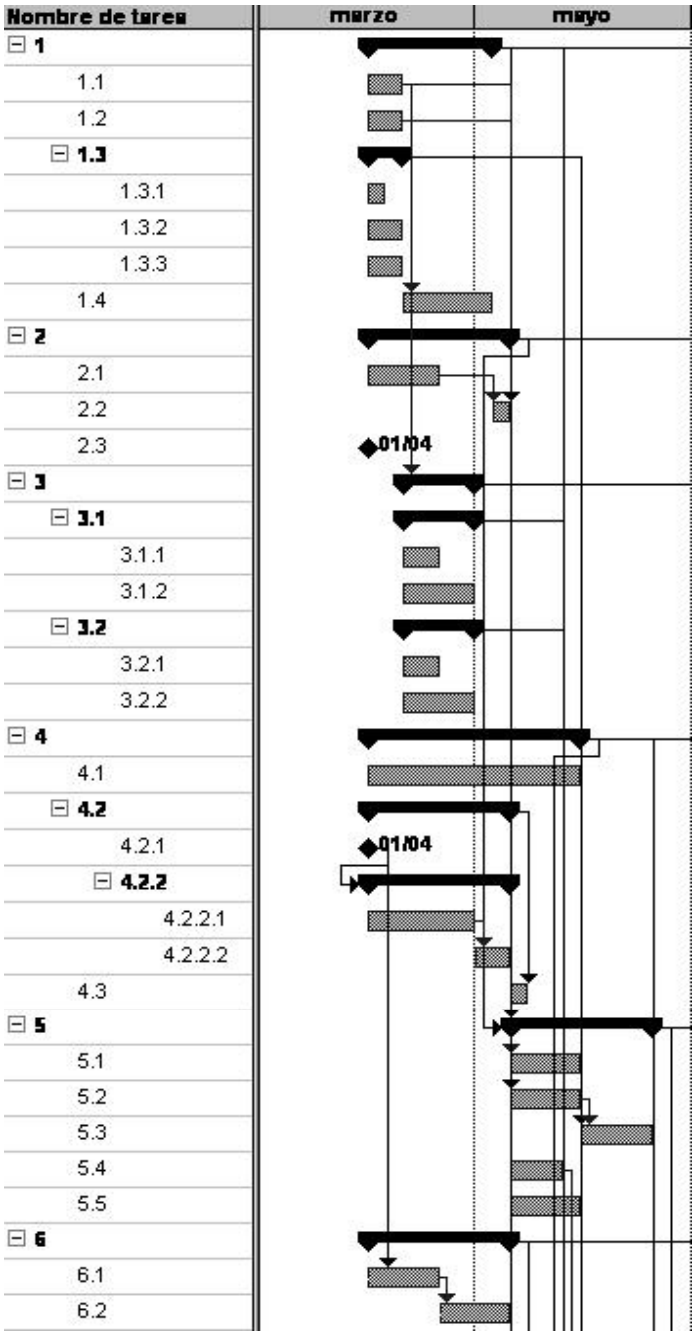
5.5 Puesta a punto en Visual Basic.	4 sems	11-May	30-May
6 Diseño.	8 sems	01-Abr	10-May
6.1 Diseño conceptual (MOR, diagramas de bloques, casos de uso, etc.)	4 sems	01-Abr	20-Abr
6.2 Diseño de implementación.	4 sems	21-Abr	10-May
7 Implementación.	35 sems	06-May	27-Oct
7.1 Desarrollo de una interface shp -> tesis anterior en VC++.	4 sems	06-May	25-May
7.2 Implementación del algoritmo svrp en una Dll.	4 sems	26-May	14-Jun
7.3 Implementación del algoritmo compartimentado en una Dll.	6 sems	26-May	24-Jun
7.4 Implementación de *.Dll's.	7 sems	26-May	29-Jun
7.5 Implementación para PC.	18 sems	31-May	28-Ago
7.5.1 Implementación de interfaz gráfica.	18 sems	31-May	28-Ago
7.5.2 Implementación de funcionalidades intermedias.	4 sems	31-May	19-Jun
7.6 Implementación para Internet.	12 sems	29-Ago	27-Oct
7.6.1 Implementación de interfaz gráfica.	12 sems	29-Ago	27-Oct
7.6.2 Implementación de funcionalidades intermedias.	4 sems	29-Ago	17-Sep
8 Documentación.	44 sems	01-Abr	06-Nov
8.1 Elaboración de los estándares de documentación (planilla Word, etc).	1 sem	01-Abr	05-Abr
8.2 Elaboración de la documentación de usuario.	6 sems	20-Jun	19-Jul
8.3 Elaboración de la documentación técnica (informe).	0 sems	01-Abr	01-Abr
8.4 Elaboración de la presentación.	2 sems	28-Oct	06-Nov
8.5 Elaboración de informes de reuniones con el usuario.	0 sems	01-Abr	01-Abr
8.6 Elaboración de estándares de implementación.	1 sem	20-Jun	24-Jun
8.7 Evaluación del código fuente de la tesis anterior.	5 sems	06-May	30-May
9 Verificación.	38 sems	11-May	16-Nov
9.1 Armado del plan de verificación.	2 sems	11-May	20-May
9.2 Verificación de los algoritmos.	4 sems	28-Oct	16-Nov
9.2.1 Algoritmo svrp en una Dll.	2 sems	28-Oct	06-Nov
9.2.2 Algoritmo compartimentado en una Dll.	4 sems	28-Oct	16-Nov
9.2.3 Las *.Dll's.	2 sems	28-Oct	06-Nov
9.3 Verificación interfaz ShapeFile - Txt	2 sems	26-May	04-Jun
9.4 Verificación de módulos.	1 sem	21-May	25-May
9.5 Verificación de interfase.	3 sems	21-May	04-Jun
9.6 Verificación del producto final.	2 sems	21-May	30-May
10 Evaluación.	2 sems	17-Nov	26-Nov
10.1 Evaluación final del plan de acción.	2 sems	17-Nov	26-Nov
10.2 Evaluación del cumplimiento de los requerimientos de usuario.	2 sems	17-Nov	26-Nov
10.3 Evaluación sobre los objetivos del proyecto.	2 sems	17-Nov	26-Nov
10.4 Desarrollo a futuro.	2 sems	17-Nov	26-Nov

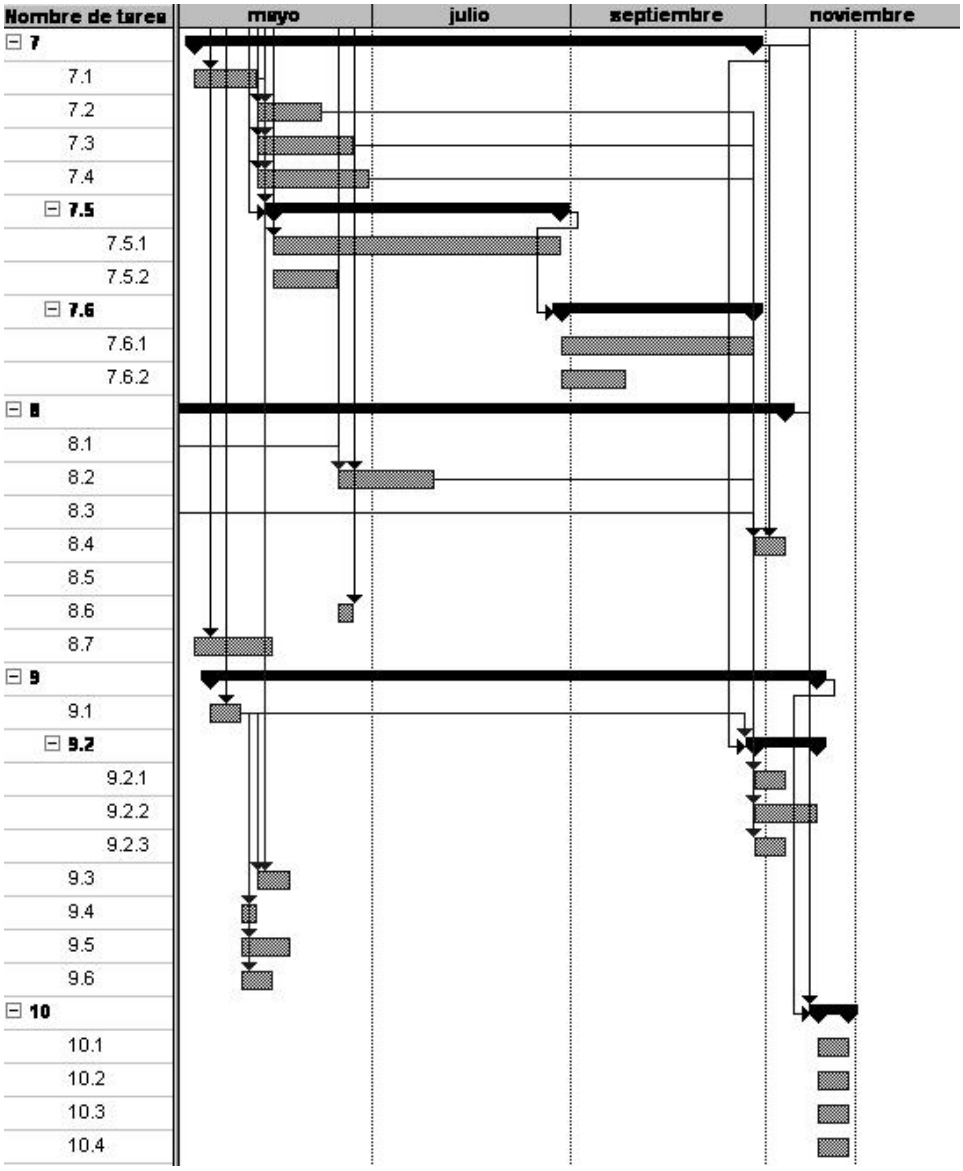
Diagramas de precedencia (Gantt)

Especificación Global



Especificación por Sub-Tareas




Nombre de tarea
mayo
julio
septiembre
noviembre
7
7.1
7.2
7.3
7.4
7.5
7.5.1
7.5.2
7.6
7.6.1
7.6.2
8
8.1
8.2
8.3
8.4
8.5
8.6
8.7
9
9.1
9.2
9.2.1
9.2.2
9.2.3
9.3
9.4
9.5
9.6
10
10.1
10.2
10.3
10.4

Apéndice C

Apéndice C: Normas y Estándares de Documentación

Normas y Estándares de Documentación

La estructura de la documentación se basa en el paper para tal función desarrollado por Patrick Valduriez, Project Rodin, INRIA, Rocquencourt, France [3].

Apéndice D

Apéndice D: Normas y Estándares de Implementación

Normas y Estándares de Implementación

Alcance

El presente documento especifica las normas y convenciones que deberán utilizar los implementadores a la hora de escribir y/o documentar código.
Mediante el cumplimiento de las normas especificadas se pretende lograr un código bien documentado, prolijo y homogéneo.

Contenido

Normas de Programación

En este capítulo se describen las pautas generales para escribir el código del proyecto.

Convenciones de Nomenclatura

- Todos los identificadores deben ser descriptivos y mnemotécnicos.
- Los nombres de las clases deben comenzar con letra mayúscula. Si se componen de más de una palabra, las palabras subsiguientes también deben comenzar con mayúsculas, y no se debe utilizar separadores (por ej. '_').
- Los nombres de las variables privadas de la clase deben comenzar con mv
- El nombre de las variables locales a los métodos y los nombres de los métodos debe comenzar con minúscula, en caso de ser más de una palabra, las siguientes comienzan con mayúscula.
- Los nombres de las funciones selectoras y modificadoras, es decir, de solicitud de una propiedad o cambio de una propiedad, deben ser nombradas ***getNombreFunción*** y ***setNombreFunción*** respectivamente.
- Los nombres de las constantes debe ser escritos en mayúsculas.
- Se recomienda la descripción del objetivo de las variables al ser declaradas, si esto no está implícito en su nombre.

Diagramado del Código

- La indentación debe ser apropiada y consistente, respetando el nivel de anidación. El cuerpo de un método debe estar indentado un nivel más que el encabezado.
- Para indentar se utilizará el tabulador (Tab).
- Los comentarios de una sentencia o un bloque de sentencias debe preceder a las mismas. Se permiten comentarios al final de la línea si esto favorece al mejor entendimiento del código. Todos los comentarios deben estar escritos en forma impersonal, y sin errores ortográficos.
- El indicador de fin de bloque debe indicar qué bloque cierra, debe encontrarse al mismo nivel de indentación del comienzo del bloque y debe estar solo (excepto por comentarios) en la línea.

Estructura de la Clase

La utilización de clases es una de las herramientas más potentes para lograr la modularidad en la programación.
Los comentarios previos a la declaración de la clase tendrán el siguiente formato

- Nombre de la clase, versión actual y fecha de la versión.
- Descripción general y funcionalidad de la clase.
- Lista de versiones de la clase incluyendo en cada una:
- Versión
- Autor
- Fecha
- Modificaciones a la versión anterior

La estructura propiamente dicha será la siguiente:

- Importaciones
- Declaración de la clase
- Declaración de variables públicas. En lo posible, se debe evitar el uso de estas e implementar una interfaz con primitivas ***get*** y ***set.***
- Declaración de variables privadas.
- Definición de constructores.
- Definición de rutinas públicas.
- Definición de rutinas privadas.

Estructura de las Rutinas

Comentarios previo al encabezado de cada función o procedimiento:

- Versión
- Autor
- Fecha
- Funcionalidad
- Precondiciones
- Aclaración del significado de los parámetros
- Aclaración del resultado devuelto, si se aplica
- Relación a otros procedimientos o funciones

Técnicas de Programación

Se mantendrá, dentro de lo posible, un estilo prolijo de programación, evitando por ejemplo el uso de saltos en el código, o utilización de miembros Protected para facilitar el acceso a los mismos, cuando en realidad debieran ser privados.
No se prohíbe el uso de estas técnicas cuando realmente sean de ayuda para lograr un desempeño más eficiente, o en algunos casos un mejor entendimiento, pero, considerando que éste no es generalmente el caso, no debieran ser usadas.

Apéndice **E**

Apéndice E: Definición de Problemas

Definición de Problemas

A continuación se realiza una introducción al Problema de Ruteo de Vehículos, definiendo términos y conceptos que se usan en este documento.

Problemas de Ruteo de Vehículos

El problema de ruteo consiste en obtener rutas que sean "buenas" (en algún sentido) para que recorra una flota de vehículos visitando una sola vez cada punto de interés. Es decir que se debe decidir que recorrido hará cada vehículo (en general cumpliendo una función durante el mismo).
Los problemas de ruteo surgen fundamentalmente en el ámbito de la distribución / recolección tanto de bienes como de servicios. Éstos problemas implican trazar una o varias rutas a recorrer con determinadas restricciones, tratando de optimizar una determinada función objetivo.

El problema clásico de Ruteo de Vehículos (VRP) se define como: "Determinar un conjunto de rutas de costo mínimo para una flota homogénea de vehículos que sirve a un conjunto de clientes dispersos geográficamente".
Modificando las condiciones o restricciones de dicho problema se pueden generar infinitas variantes. Es sobre un conjunto de las variantes más importantes qua trabajamos.
En la definición de requerimientos de este proyecto se entrego una lista de problemas que abarcan los casos principales y más utilizados en la práctica y pueden ser ampliados para satisfacer las necesidades puntuales planteadas por un problema determinado, así también como dos de los casos menos estudiados, o por lo menos sin soluciones conocidas.

Se describen ahora los problemas a tratar:

TSP

El TSP (Traveling Salesman Problem por sus siglas en inglés) se refiere al caso en que tenemos un solo depósito, un solo vehículo, N nodos que requieren servicio ubicados en una red (que puede ser dirigida o no), sin restricciones de capacidad del vehículo, sin restricciones de demanda de los clientes y sin restricciones de tiempo.

El objetivo es minimizar la distancia recorrida pasando por todos los nodos exactamente una vez volviendo al nodo de origen (depósito).
Podemos definir la red como G = [N, A, C] siendo N el conjunto de nodos, A el conjunto de arcos y C la matriz de costos que determina la distancia de un nodo a otro.

Si Cij = dij entonces la distancia que hay que recorrer para ir del nodo i al nodo j es dij. Lo que se debe hallar es el Ciclo Hamiltoniano de costo mínimo. Si los costos son simétricos, o sea, si el costo de viajar de un lugar (nodo) a otro no depende de la dirección en que se viaje, estamos frente a un TSP simétrico; en caso contrario el TSP se llama asimétrico o dirigido.

Un claro ejemplo de aplicación de este problema es el recorrido que debe hacer un visitador médico relevando los pedidos.

Formulación Matemática.

Consideramos cij = cji = costo de ir del nodo i al nodo j y viceversa.

x ij = $\begin{cases} 1 & \text{si el arco i-j se encuentra en la ruta obtenida como solución final} \\ 0 & \text{en otro caso} \end{cases}$

Problema:

Minimizar $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$

Sujeto a $\sum_{i=1}^{n} x_{ij}$ = bj = 1 (j = 1..n)Asegura que a cada nodo llega una sola arista.

$\sum_{j=1}^{n} x_{ij}$ = ai = 1 (i = 1..n)Asegura que de cada nodo sale una sola arista.

xij = 0 o 1 (i,j = 1..n)

Estas ecuaciones no son suficientes para definir completamente el problema dado que no impiden la formación de sub-tours (ver Fig 1). El objetivo del problema es minimizar la distancia recorrida satisfaciendo la demanda de los nodos con un ciclo único. Con las 3 restricciones vistas anteriormente se puede presentar el siguiente caso:
N={1,2,3,4,5,6} conjunto de nodos
A={x12, x23, x31, x45, x56, x64}
En este caso tenemos 2 sub-tours o ciclos, formados uno por los nodos 1,2 y 3 y el otro formado por los nodos 4, 5 y 6.

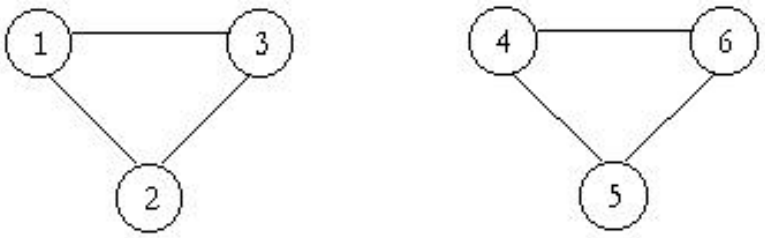


Fig 1

Para evitar esto debemos agregar la siguiente restricción al problema:

$$\sum_{i\in Q}\sum_{j\notin Q} x_{ij} \geq 1$$

para todo Q subconjunto no vacío de N

Esta ecuación nos asegura que de todo punto que pertenezca a un subconjunto de N sale una arista que termina en un nodo que no pertenece a dicho subconjunto. De esta forma aseguramos que existe un camino para ir de cualquier nodo a todos los otros, asegurando de esta forma que la solución hallada es un ciclo único.

MTSP

El "multi traveling salesman problem" (MTSP) es una generalización del TSP (un solo vehículo). La diferencia con el TSP es que se tiene una flota de M vehículos y un depósito. M vehículos deben visitar N nodos buscando minimizar la distancia total recorrida por los M vehículos. Cada vehículo debe visitar un sub-ciclo de nodos que comience y termine en el depósito común, y cada nodo debe ser visitado exactamente una vez por un vehículo.

Formulación Matemática.

Problema:

Minimizar $\sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$

Sujeto a

$$\sum_{i=1}^{n} x_{ij} = b_j = \begin{cases} 1 & \text{si } j = 2,3, \ldots ,n \\ M & \text{si } j = 1 \end{cases}$$

Asegura que a cada cliente llega una sola arista y que al depósito llegan M aristas

$$\sum_{j=1}^{n} x_{ij} = a_i = \begin{cases} 1 & \text{si } i = 2,3, \ldots ,n \\ M & \text{si } i = 1 \end{cases}$$

Asegura que de cada cliente sale una sola arista y que del depósito salen M aristas.

$x_{ij} = 0 \text{ o } 1 \qquad (i,j = 1..n)$

Esta formulación matemática es la adaptación de la formulación matemática del TSP al MTSP

VRP Clásico

El VRP clásico es una generalización del MTSP (varios vehículos), lo cual lo hace aplicable a mayor cantidad de casos de la realidad. Las diferencias con el MTSP consisten en que en este caso los clientes tienen requerimientos de servicio que deben ser cumplidos (demanda), la flota de vehículos es limitada y los vehículos tienen capacidad limitada. Cada vehículo debe visitar un sub-ciclo de nodos que comience y termine en el depósito común, y cada nodo debe ser visitado exactamente una vez por un vehículo.

Un ejemplo de este problema es el reparto de productos que debe hacer una empresa a sus clientes distribuidos geográficamente. Cada cliente tiene una determinada demanda y los vehículos deben comenzar y finalizar el día en el depósito.
Formulación Matemática.
Podemos basarnos en la formulación matemática del MTSP pero faltan varias restricciones. Aquí no aparecen restricciones de cumplimiento de la demanda de los clientes, no exceder la capacidad de los vehículos, etc.
La formulación matemática completa del VRP clásico es la siguiente:

Formulación matemática:

Problema:

Minimizar $\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{v=1}^{NV} c_{ij}x_{ij}^{v}$ (a)

Sujeto a $\sum_{i=1}^{n}\sum_{v=1}^{NV} x_{ij}^{v} = 1$ (j = 2,....,n) (b)

$\sum_{j=1}^{n}\sum_{v=1}^{NV} x_{ij}^{v} = 1$ (i = 2,....,n) (c)

$\sum_{i=1}^{n} x_{ip}^{v} - \sum_{j=1}^{n} x_{pj}^{v} = 0$ (v = 1,....,NV; p = 1,....,n) (d)

$\sum_{i=1}^{n} d_i (\sum_{j=1}^{n} x_{ij}^{v}) <= K_v$ (v = 1,....,NV) (e)

$\sum_{i=1}^{n} t_i^{v} \sum_{j=1}^{n} x_{ij}^{v} + \sum_{i=1}^{n}\sum_{j=1}^{n} t_{ij}^{v} x_{ij}^{v} <= T_v$ (v = 1,....,NV) (f)

$\sum_{j=2}^{n} x_{1j}^{v} <= 1$ (v = 1,....,NV) (g)

$\sum_{i=2}^{n} x_{i1}^{v} <= 1$ (v = 1,....,NV) (h)

$x_{ij}^{v} =$ 0 or 1 para todo i, j, v (i)

Donde n = número de nodos, NV = número de vehículos, Kv = capacidad del vehículo v, Tv = máximo tiempo posible para la ruta del vehículo v, di = demanda del nodo i (d1 = 0), tiv = tiempo de servicio requerido por el vehículo v en el nodo i (t1v=0), tijv = tiempo de viaje del vehículo v del nodo i al nodo j, cij = costo del viaje del nodo i al nodo j, xijv = 1 si el arco i-j es atravesado por el vehículo v y 0 en otro caso.
La ecuación (a) indica que el objetivo es minimizar la distancia total recorrida. También se pueden minimizar costos sustituyendo cij por un coeficiente cijv que depende del vehículo. Las ecuaciones (b) y (c) que la demanda de cada nodo es cumplida exactamente por un vehículo. La continuidad de la ruta es asegurada por la

ecuación (d), si un vehículo entra a un nodo con demanda, debe salir de dicho nodo. La ecuación (e) establece la restricción de capacidad de los vehículos y la ecuación (f) las restricciones de tiempo total que demora en recorrerse una ruta. Las ecuaciones (g) y (h) aseguran que la disponibilidad de vehículos no es superada.

Al VRP clásico se le pueden aplicar diferentes extensiones o variantes. En este caso consideraremos los problemas con mas de un depósito, con flota heterogénea y con ventanas de tiempo.

MDVRP

Este tipo de problema es similar al VRP (ruteo de N vehículos con un depósito único y clientes con demanda preestablecida y sin ventanas de tiempo) con la diferencia de que se tiene un conjunto de depósitos y no un depósito único como ocurría en el caso anterior. De esta forma tenemos una flota de M vehículos que salen de D depósitos y deben satisfacer la demanda de N nodos. Los vehículos deben salir de un depósito y volver al mismo.
Los ejemplos prácticos de aplicación son similares a los del VRP pero para empresas más grandes que tengan varios depósitos. Un ejemplo sería un supermercado que en una ciudad tenga varias sucursales y recepcione (por Internet por ej.) pedidos de sus clientes. Dada la demanda y ubicación geográfica de cada cliente se formarán las rutas a recorrer por cada vehículo de reparto de mercadería que saldrán de las diferentes sucursales del supermercado.

Formulación Matemática.

Pequeñas variaciones a la formulación matemática (ecuaciones (a) a la (i)) vista para el VRP clásico permiten establecer la formulación matemática para el MDVRP. Consideramos los nodos 1,2,....,M como depósitos, obtenemos las nuevas ecuaciones modificando el índice en las restricciones (b) y (c) a (j = M+1,....,n) y (i = M+1,....,n) respectivamente, y modificando las restricciones (g) y (h) por las siguientes:

$$\sum_{i=1}^{M} \sum_{j=M+1}^{n} x_{ij}{}^{v} <= 1 \qquad \text{(v = 1,....,NV)} \qquad \text{(g')}$$

$$\sum_{p=1}^{M} \sum_{i=M+1}^{n} x_{ip}{}^{v} <= 1 \qquad \text{(v = 1,....,NV)} \qquad \text{(h')}$$

VRPHF: Problemas con Flota Heterogénea

Ahora consideraremos una situación en que el distribuidor decide, por conveniencia y confiabilidad, alquilar, en lugar de comprar, vehículos por un cierto tiempo. El número de vehículos de cada tipo necesario para satisfacer la demanda en forma efectiva debe ser determinado con una política de ruteo coherente para esos vehículos. Asumimos que cada tipo de vehículo tiene un costo de alquiler fijo y un costo de ruteo variable que es proporcional a la distancia recorrida. Cada tipo de vehículo tiene su capacidad y su costo fijo. El componente del costo variable depende del combustible, mantenimiento y mano de obra.

Para resolver este tipo de problemas utilizaremos el algoritmo de Clarke y Write (CW) tradicional [b], algoritmo de los ahorros modificando la función utilizada para calcular los ahorros. En este algoritmo dos rutas conteniendo clientes i y j pueden ser unidas si: a) i y j pertenecen a rutas diferentes, b) i y j no son puntos interiores a las rutas, c) la demanda de la combinación de ambas rutas no excede la capacidad de los vehículos.
Recordamos que la decisión sobre cuáles rutas unir se basa en el cálculo del ahorro que dicha unión permite obtener. Los ahorros se calculan como s(i,j) = c(0,i) + c(0,j) – c(i,j), siendo c(i,j) el costo de ir desde el punto i al punto j y siendo 0 un depósito. El problema que plantea este método para el caso de flota heterogénea es que no considera los costos fijos de los vehículos. Debemos ampliar el concepto de ahorro incluyendo ahora dichos costos además de los costos de ruteo. A este método se le llama método "Combined Savings" (CS) [b].
Sea F una función en los reales que dada z (demanda de una ruta) devuelve el costo fijo del menor vehículo capaz de satisfacer la demanda z. Consideremos una ruta I que tiene a i como nodo "límite" (nodos conectados con el depósito) y una ruta J que tiene a j como nodo "límite". Si la demanda total de estas rutas es zi y zj el ahorro obtenido al unir dichas rutas es: s'(i,j) = s(i,j) + F(zi) + F(zj) – F(zi + zj). Este método tiene el inconveniente que los ahorros calculados son inmediatos.
Por ejemplo supongamos que tenemos las rutas I, J y K con una demanda de 100 cada una servidas por vehículos de capacidad 100. Supongamos también que el próximo vehículo más grande tiene una capacidad de 300 y su costo fijo es considerablemente mayor al de los vehículos de capacidad 100. Puede ocurrir que la combinación de las tres rutas siendo servidas por un vehículo de capacidad 300 sea más barata que las tres rutas por separado. Pero el algoritmo CS encontrará dicha solución? No necesariamente. Dado que las rutas se combinan de a dos es probable que al combinar las rutas I y J por ej. el ahorro sea negativo ya que se requiere un vehículo da capacidad 300 el cual tiene un costo fijo elevado por lo que dicha combinación sería descartada. En este caso CS no considera que el vehículo de capacidad 300 tiene capacidad de 100 unidades inutilizada, la cual puede ser útil a la hora de unir una nueva ruta sin aumentar los costos fijos.

Esto genera una variante de este algoritmo, llamado Opportunity Savings Algorithms [b]. A los ahorros vistos en el punto anterior se agregan los "Opportunity savings" (OS), los cuales son una función de la capacidad no utilizada del vehículo que cubre la nueva ruta formada.
Un tipo de Opportunity Saving Algorithm es el Optimistic Opportunity Savings (OOS). En este caso los OS se definen como el costo fijo del menor vehículo que puede servir la capacidad inutilizada del nuevo vehículo. Es decir que si tenemos vehículos de capacidades 50, 100 y 200 y formamos una nueva ruta con demanda 120 servida por un vehículo con capacidad 200 el ahorro s'' = s' + OS siendo OS el costo fijo del vehículo de capacidad 100 por ser el menor vehículo capaz de cubrir las 80 unidades de capacidad inutilizada. La explicación a esta formula está dada por el hecho de que se asume que se encontrará una ruta de demanda 80 la cual se agregará a la ruta recién formada ahorrándose el costo del vehículo de 100 que servía a la ruta de 80. Sea P(z) la capacidad del menor vehículo capaz de satisfacer la demanda de la ruta z entonces: s''(i,j) = s(i,j) + F(zi) + F(zj) – F(zi + zj) + F(P(zi + zj)- zi - zj) = s'(i,j) + F(P(zi + zj)- zi - zj).

Otro tipo de Opportunity Saving Algorithm es el Realistic Opportunity Savings (ROS). Sea F'(z) análoga a F(z) salvo que F'(z) representa los costos fijos del mayor vehículo cuya capacidad es menor o igual a z. Sean I y J dos rutas con demanda zi y zj y sean i y j sus puntos terminales. Asumimos que zi >= zj.
Si F(zi + zj) = F(zi) el ahorro es el mismo que el calculado en el caso del CS, es decir s'(i,j). Si F(zi + zj) > F(zi) debemos considerar los OS por lo que s'''(i,j) = s'(i,j) + F'(P(zi + zj)- zi - zj). Datos empíricos demuestran que ROS es superior a OOS y CS dado

que brinda mejores resultados en 7 de 12 muestreos obteniendo mejores tiempos en los 12 muestreos.

En resumen tenemos los siguientes métodos y sus respectivos cálculos de ahorros:

Algoritmo	Ahorro	Fórmula para calcular el ahorro
CW	s(i,j)	$c(0,i) + c(0,j) – c(i,j)$
CS	s'(i,j)	$s(i,j) + F(z_i) + F(z_j) – F(z_{i+}z_j)$
OOS	s''(i,j)	$s'(i,j) + F(P(z_{i+}z_j)- z_i - z_j)$
ROS	s'''(i,j)	$s'(i,j) + F'(P(z_{i+}z_j)- z_i - z_j)$

VRPTW: Problemas con Ventanas de Tiempo

Los distintos puntos del sistema (ya sean depósitos, puntos intermedios o destinos) pueden tener restricciones de tiempo. El manejo más común de las restricciones de tiempo se realiza a través de las ventanas de tiempo.

Una ventana de tiempo de dos dimensiones [s,t] para una entidad determina que dicha entidad debe ser visitada en el intervalo de tiempo que va desde s hasta t. Si estamos hablando por ejemplo de un comercio de venta de ropa podemos establecer que la mercadería puede llegar al mismo entre las 10hs. y las 18hs., por lo que su ventana de tiempo será [10,18].

También podemos expresar solamente un tiempo mínimo o máximo. Una ventana de tiempo de una dimensión es [-∞,t] o [t,+∞]. La primera ventana de tiempo implica que el servicio debe ser llevado a cabo antes del tiempo t y la segunda implica que el servicio debe ser llevado a cabo luego del tiempo t.

Una determinada entidad puede estar asociada a más de una ventana de tiempo. Por ejemplo, un comercio que está abierto de 8hs. a 12hs. y de 14hs. a 18hs. y solo en ese horario puede recibir la mercadería.

Las ventanas de tiempo pueden representar también días de la semana, por ejemplo se puede decir que una entidad esta disponible sólo los martes, jueves y sábados. Una vez que los días de la semana son asignados, podría haber ventanas de tiempo y restricciones de precedencia entre tareas para las tareas que se realizarán durante esos días.

Algunas aplicaciones de los problemas con ventanas de tiempo son:

- Ruteo de ómnibus escolares
- Ruteo de camiones con carrocería con cargas parciales o totales
- Ruteo de vendedores callejeros

CPP

El problema del cartero chino requiere la determinación de un ciclo de costo mínimo que pase por todos los arcos de la red al menos una vez (la demanda está en los arcos). Puede ser dirigido o no dirigido, según si los arcos son dirigidos o no. Las dos variaciones pueden resolverse por algoritmos con tiempo de ejecución polinómico.
El problema del cartero chino mixto tiene algunos arcos dirigidos y otros no, este problema es NP-duro.

En nuestro caso trataremos el problema del CPP no dirigido dado que los demás están fuera del alcance de este taller.

Algunas aplicaciones del problema del cartero chino son: ruteo de vendedores callejeros, carteros, inspectores de las líneas eléctricas, etc. Como en algunas situaciones la cantidad de clientes que requieren servicio es muy alta, identificarlos individualmente sería muy engorroso; en estos casos consideramos al problema del cartero chino como la variación continua del TSP discreto. Aquí un arco sustituye a una cantidad de clientes.

Los arcos tienen requerimientos de servicio que deben ser cumplidos, la flota de vehículos es limitada y los vehículos tienen capacidad limitada.

DARP

En problemas del tipo DARP (dial-a-ride-problem por sus siglas en inglés) los clientes llaman al "despachador" solicitando un servicio. Cada cliente especifica el punto de carga y entrega de lo que se vaya a transportar y quizás también el período de tiempo en que desea que sea realizada cada acción. Si todos los clientes reclaman servicio inmediato estamos frente a un DARP dinámico o DARP en tiempo real. Si todos los clientes llaman con anticipación de forma tal que la base de datos con los requerimientos de los clientes este completa antes de comenzar a hallar la solución estamos frente a un DARP estático.
En algunos casos el horario de carga (pickup) o entrega (delivery) es especificado con anticipación y la otra acción debe ser realizada dentro de un período de tiempo determinado desde el horario ya dado. Es decir, se fija uno de los extremos y el período máximo de tiempo que puede transcurrir entre la carga y entrega en lugar del caso tradicional en que se indican horario de carga y horario de entrega. En este caso estamos frente a un problema con ventanas de tiempo de dos dimensiones.

SVRP

El SVRP (Stochastic vehicle routing) es una generalización del problema clásico de ruteo VRP, al que se le realizan las siguientes modificaciones:

- La demanda de los clientes es una variable aleatoria con una distribución de probabilidad conocida
- Las rutas deben ser diseñadas antes de conocer la demanda real de cada cliente.

El objetivo del problema es minimizar el costo del viaje esperado, aunque se puede incurrir en ciertos tipos de costos si un cliente no es servido en un determinado viaje.

Un claro ejemplo de aplicación de este problema es la recolección de residuos.

Formulación Matemática.

Definiciones previas:
Sean:
Cij : costo de ir del nodo i al nodo j.

$$x_{ijk} = \begin{cases} 1 & \text{Si el arco } i-j \text{ se encuentra en la ruta de vehiculo } k \\ 0 & \text{En otro caso} \end{cases}$$

n : Numero de clientes
m: Numero de vehículos

Q_k : Es la capacidad del vehículo k

α : Es la máxima probabilidad de que alguna ruta falle.

d_i : Variable aleatoria que representa la demanda del cliente i, donde se conoce su función de distribución asociada

Problema:

Minimizar $\sum_{k=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ijk}$

Sujeto a: $P\left(\sum_{i=1}^{n} \sum_{j=1}^{n} d_i x_{ijk} \leq Q_k \right) \geq 1 - \alpha \qquad \forall k : 1..m$ Ec.1

$x = \{x_{ijk}\} \in S$ Para que la solución cumpla todas las demás restricciones del problema de ruteo

Ruteo con compartimentos

El problema de ruteo con compartimentos es una variación del problema de ruteo clásico, en donde se considera ahora que cada vehículo está compartimentado en secciones disociadas, que le permiten llevar más de un tipo de mercadería sin mezclarlas. Los clientes pueden tener demandas asociadas a todas las mercaderías que el depósito ofrece o solo a algunas de ellas.

Apéndice F

Apéndice F: Reporte de Incidencias

A lo largo del desarrollo de un proyecto siempre se presentan elementos que afectan ese desarrollo.
Esos elementos pueden venir de parte del usuario, cambios en el diseño o problemas tecnológicos.

Se enumeraran ahora algunas de las incidencias ocurridas a lo largo de este taller:

Problemas de ruteo:

Los problemas de ruteo a resolver por parte de este proyecto incluían el problema de ruteo con compartimentos. Posteriormente se elimina por parte de la empresa este tipo de problemas en pro de lograr una biblioteca mas robusta y en menor tiempo.

Modularización:

Si bien en un principio los requerimientos proponían el desarrollo de una dll para cada problema, finalmente estos cambiaron y se desarrollo una única dll que recibe un código indicando que problema debe resolver. De esta forma se integran todos los problemas en una única librería, que reúna mas funcionalidades y sea mas robusta.

Sitio Web:

Los requerimientos indicaban una investigación sobre una posible aplicación en Internet.
Sin embargo, sobre la marcha del proyecto los usuarios de la empresa plantearon su necesidad de tener corriendo un sistema que permitiera rutear a través de la red.
Así, la aplicación monousuario de oficina planteada originalmente se cambio por un sitio Web.

Este cambio tuvo grandes implicancias sobre los requerimientos del producto, y en el "Análisis de Requerimientos" deben descartarse o modificarse varias funcionalidades, dentro de las que se destacan el manejo y obtención de datos a partir de una base de datos del cliente.

Apéndice **G**

Apéndice G: Geocodificación (address matching)

Geocodificación (address matching)

introducción

La "geocodificación" es un campo de investigación académica de importancia en la actualidad. Se enmarca dentro de la teoría de analizadores semánticos, en el área de la computación.
La importancia se debe al uso y necesidad mas frecuente de utilizar datos geográficos para la toma de decisiones por parte de las instituciones; desde el Estado hasta la pequeña empresa. El incorporar datos geográficos a las bases de datos de las instituciones puede ser una tarea ardua y la geocodificación pretende simplificar y/o automatizar dicho proceso.
Se estima que el 70% de todos los datos que se encuentran en las bases de datos de las instituciones a nivel mundial pueden ser geocodificados. Los sistemas que permiten la incorporación y utilización de datos geográficos son conocidos como S.I.G. (Sistema de Información Geográfica) y se espera que tengan un gran auge en los años venideros..

El objetivo de este documento es introducir la problemática de la geocodificación. En particular discutir como este problema influye sobre un software de ruteo genérico.

Definición del Problema

La "geocodificación o address matching" se podría definir como el proceso por el cual se determinan las coordenadas geográficas (X,Y) a partir de una dirección en lenguaje natural.
En general la dirección en lenguaje natural consta de un nombre de calle, un numero, y datos auxiliares como las esquinas, el numero de apartamento.
El ingreso de la dirección pudo ser realizado en un tiempo pasado , por lo tanto aparece un problema agregado y de consideración, que es el cambio de nombres de las calles a lo largo del tiempo.

Solución Planteada

La primera instancia de la geocodificación es dado la dirección ingresada en lenguaje natural, un string de caracteres, determinar un nombre de calle y numero de puerta

valido dentro del universo de nombre de calles y numero de puertas permitido por el sistema.
Para esto se puede utilizar un analizador lexicográfico y sintáctico. Un método, simple y práctico, muy utilizado y de relativo éxito, es el de extraer el nombre de la calle de la cadena de caracteres y este compararlo con el nombre de las calles de nuestro universo conocido de calles, eligiendo el mas parecido. Con esto reducimos nuestro problema a una simple comparación de cadenas de caracteres.
El algoritmo mas simple y flexible que nos dice que tanto se parecen dos secuencias de caracteres es el Soundex. .

Una segunda instancia de la geocodificación es dado un nombre de calle y un numero de puerta del universo de nombre de calles y numero de puertas permitido por el sistema, determinar las coordenadas geográficas (X,Y).
La solución genérica, y mas aceptada a este subproblema se detalla a continuación. Se tiene una red de arcos representativo de las calles (varios arcos seguidos representan una calle). Cada arco tiene asociado el nombre de la calle a la que pertenece y un numero de puerta inicial y final. Entonces el problema se reduce a una simple búsqueda del arco correcto.
Notar que el éxito de esta estrategia depende directamente de la correctitud de los datos (numero inicial y final de cada arco) de la red, con la cual no siempre podemos contar. Además el mayor problema de esta solución es determinar el nombre de calle dentro de nuestro universo a partir de un string.
Finalmente podemos aproximar la coordenada geográfica (X,Y) a través de una interpolación, sobre el arco, del numero de puerta real entre el numero de puerta inicial y final.

Variantes

Ciertas variantes del método puro pueden ser aplicadas para un mejor desempeño en casos particulares. A continuación se resumen algunos de estos casos.

A veces, aparece la necesidad de tener asociado a cada arco una numeración inicial a al izquierda, numeración inicial a al derecha, numeración final a al izquierda y numeración final a al derecha, en lugar de simplemente un numero de puerta inicial y final. Esta red es mas compleja pero el método es esencialmente el mismo.

Otra variante es tener la distancia al eje, que básicamente dado el punto (X,Y) donde determinamos la dirección, nos dice la distancia extra que hay que recorrer para llega a la dirección especifica. Esta mejora tiene importante resultados en el caso de geocodificación rural.

La solución planteada fue ideada y se adapta a otras realidades, diferentes a la uruguaya, como por ejemplo a la de EEUU donde los nombres de las calles son poco cambiantes con el tiempo, y en general son nombres numéricos (ej. la quinta avenida).
Para el caso de Uruguay es necesario en primera instancia tener un diccionario de calles, como un diccionario de sinónimos donde la comparación de string se realice con cada elemento de ese diccionario. Un ejemplo ilustrativo de elementos en este diccionario seria: es asignar Propios como un sinónimo de Batlle y Ordóñez.
Como ventaja de la realidad uruguaya, tenemos que el nomenclátor de calles es chico, aproximadamente 3.000.

Consideraciones

La instancia mas difícil de la solución planteada es determinar el nombre de la calle a partir de una secuencia de caracteres en lenguaje natural. Para esto existen algunas reglas básicas que se resumen a continuación.
En la secuencia pueden aparecer datos adicionales que deben ser omitidos como por ejemplo: las esquinas ("esq." Y la secuencia posterior), el numero de apartamento ("ap.", "Ap.", "apartamento", etc.), etc.
Además en general se eliminan los artículos ("la", "los", etc.) que no aportan información sobre el nombre de la calle.
Por ultimo en general no se considera (porque lleva a equivocaciones) el tipo de calle, como por ejemplo: avenida ("av.", "Av."), bulevar , etc.

Conclusiones

La geocodificación es un problema relevante en el mundo de los negocios.
La solución de la misma no es trivial, y es necesario para la misma un amplio conocimiento y una base de datos geográficos correctos, lo cual, en general, no es fácil de obtener.
El éxito y viabilidad de un software de ruteo genérico depende directamente de una solución aceptable para la geocodificación.

Referencias

Entrevista al Ing. Luis Calderón, encargado de la geocodificación en la empresa ICA.

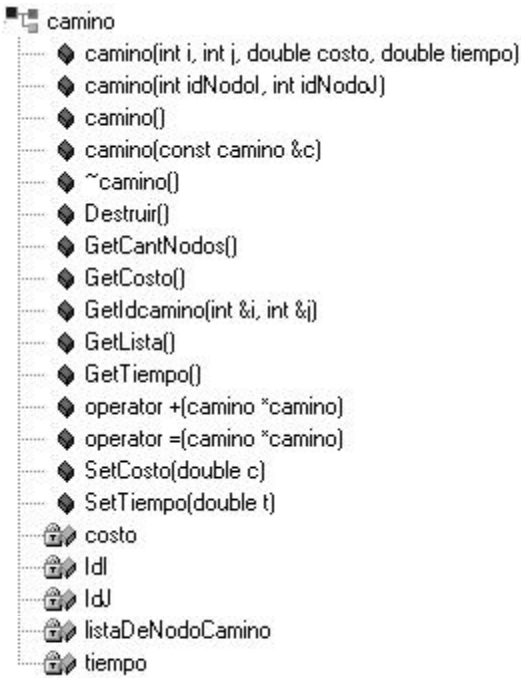
Apéndice
H

Apéndice H: Implementación y Esquema de Clases

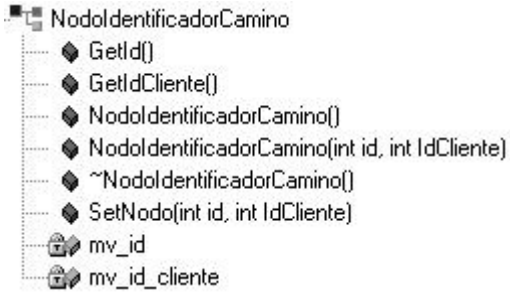
Implementación y Esquema de Clases

A continuación se detallan las interfaces (así como sus elementos privados) de cada una de las clases que intervienen de forma significativa en el sistema. Naturalmente, se dejan de lado clases de uso auxiliar.

Caminos



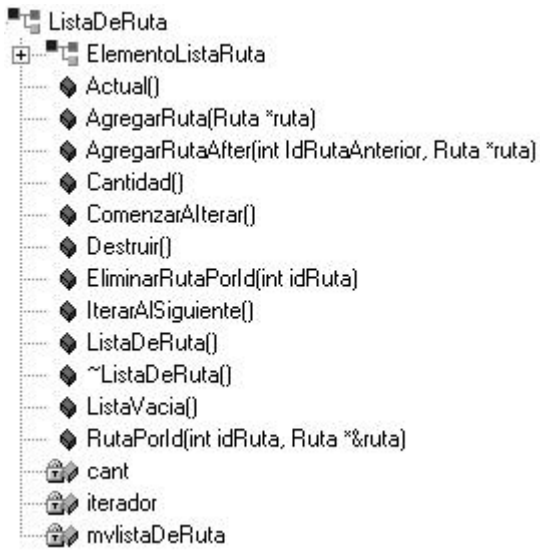

ListaDeCamino
ElementoListaCamino
Actual()
AgregarCamino(camino *c)
AnexarListaCamino(ListaDeCamino *l)
CaminoPorId(int idNodoI, int idNodoJ, camino *&c)
ComenzarAIterar()
Destruir()
EliminarCaminoPorId(int idNodoI, int intidNodoJ)
IterarAlSiguiente()
ListaDeCamino()
~ListaDeCamino()
ListaVacia()
cant
iterador
mvlistaDeCamino
ListaDeIdentificadorCamino
ElementoListaNodoIdentificadorCamino
Actual()
AgregarNodoIdentificadorCamino(NodoIdentificadorCamino *nodoIdentificadorCamino)
AgregarNodoIdentificadorCaminoAfter(int IdNodoIdentificadorCaminoAnterior, NodoIdentificadorCamino
AgregarNodoIdentificadorCaminoPrimero(NodoIdentificadorCamino *nodoIdentificadorCamino)
Cantidad()
ComenzarAIterar()
Destruir()
IterarAlSiguiente()
ListaDeIdentificadorCamino(const ListaDeIdentificadorCamino &)
ListaDeIdentificadorCamino()
~ListaDeIdentificadorCamino()
ListaVacia()
operator =(const ListaDeIdentificadorCamino &)
cant
iterador
mvlistaDeNodoIdentificadorCamino
ListaDeNodoCamino
ElementoListaNodoCamino
Actual()
AgregarNodoCamino(NodoCamino *nodoCamino)
AgregarNodoCaminoAfter(int IdNodoCaminoAnterior, NodoCamino *nodoCamino)
AgregarNodoCaminoPrimero(NodoCamino *nodoCamino)
Cantidad()
ComenzarAIterar()
Destruir()
IterarAlSiguiente()
ListaDeNodoCamino(const ListaDeNodoCamino &)
ListaDeNodoCamino()
~ListaDeNodoCamino()
ListaVacia()
operator =(const ListaDeNodoCamino &)
cant
iterador
mvlistaDeNodoCamino


NodoIdentificadorCamino
GetId()
GetIdCliente()
NodoIdentificadorCamino()
NodoIdentificadorCamino(int id, int IdCliente)
~NodoIdentificadorCamino()
SetNodo(int id, int IdCliente)
mv_id
mv_id_cliente


NodoCamino
GetIdInt()
NodoCamino()
~NodoCamino()
SetIdInt(int interno)
mv_idInterno

Rutas

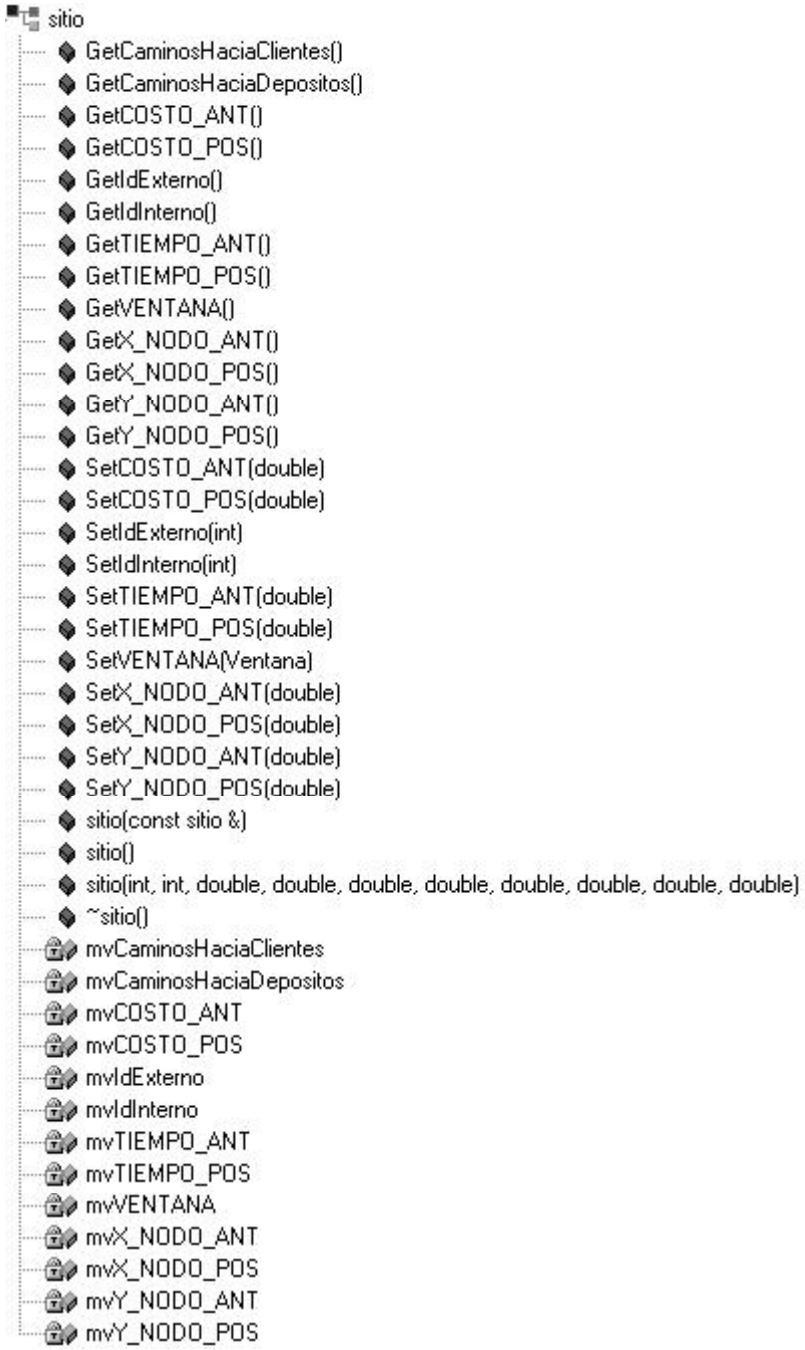

ListaDeNodoRuta
ElementoListaNodoRuta
Actual()
AgregarNodoRuta(NodoRuta *nodoRuta)
AgregarNodoRutaAfter(int IdNodoRutaAnterior, NodoRuta *nodoRuta)
AgregarNodoRutaPrimero(NodoRuta *nodoRuta)
Cantidad()
ComenzarAIterar()
CopiarDe(ListaDeNodoRuta *lista)
DemandaLista()
Destruir()
IterarAlSiguiente()
ListaDeNodoRuta()
~ListaDeNodoRuta()
ListaVacia()
NodoPorId(int idNodo, NodoRuta *nodoRuta)
operator =(const ListaDeNodoRuta &)
SiguienteId()
SiguienteVacia()
cant
iterador
mvlistaDeNodoRuta

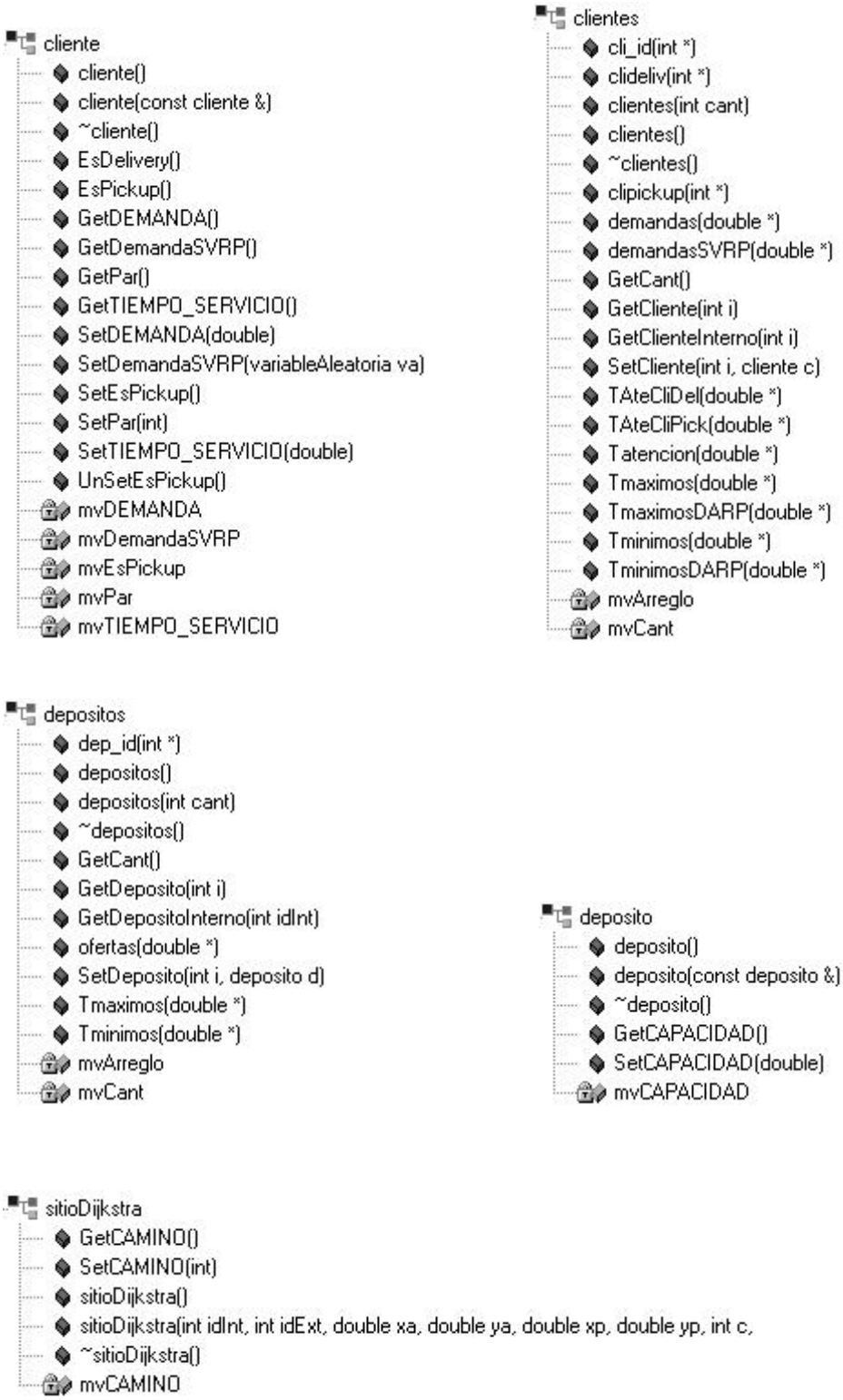

ListaDeRuta
ElementoListaRuta
Actual()
AgregarRuta(Ruta *ruta)
AgregarRutaAfter(int IdRutaAnterior, Ruta *ruta)
Cantidad()
ComenzarAIterar()
Destruir()
EliminarRutaPorId(int idRuta)
IterarAlSiguiente()
ListaDeRuta()
~ListaDeRuta()
ListaVacia()
RutaPorId(int idRuta, Ruta *&ruta)
cant
iterador
mvlistaDeRuta

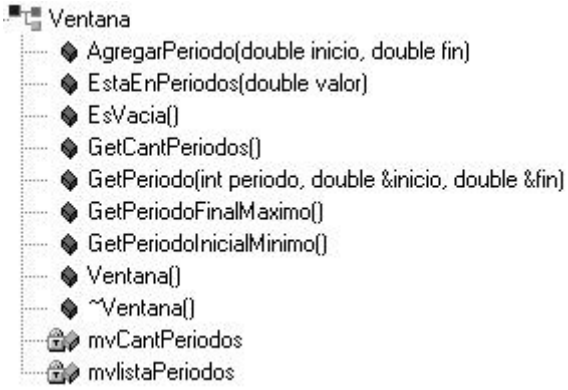

NodoRuta
EsCliente()
GetDemanda()
GetIdNodo()
GetTAte()
GetTEA()
GetTMax()
GetTMin()
NodoRuta(bool esCliente, int IdNodo, int TMin, int Tmax, int TAte, int TEA, double Demanda)
NodoRuta()
~NodoRuta()
operator =(NodoRuta nodoRuta)
SetDemanda(double value)
SetIdNodo(int value)
SetTAte(int value)
SetTEA(int value)
SetTMax(int value)
SetTMin(int value)
Demanda
IdNodo
mvEsCliente
TAte
TEA
TMax
TMin
Ruta
CopiarDe(Ruta *ruta)
Destruir()
GetCantEsq(ListaDeCamino &listaDeCamino)
GetCantNodos()
GetCosto(ListaDeCamino &listaDeCamino)
GetDemanda()
GetDeposito()
GetIdRuta()
GetIdVehiculo()
GetLista()
GetTiempoAntesDeCamino(int i, int j, clientes &c, depositos &d, ListaDeCamino &listaDeCamino)
GetVentana(clientes &c, depositos &d, ListaDeCamino &listaDeCamino)
operator +(Ruta *ruta)
operator =(Ruta *ruta)
Ruta(const Ruta &ruta)
Ruta()
Ruta(int idRuta, int idVehiculo, NodoRuta *deposito)
~Ruta()
SetDeposito(NodoRuta *dep)
SetIdRuta(int id)
SetIdVehiculo(int)
Deposito
IdRuta
IdVehiculo
listaDeNodoRuta


Rutas
AcomodarIdRutas()
AsignarId(int id)
Cant_Rutas()
CantNodosTotal(ListaDeCamino &listaCamino)
CantRutDep(int dep)
CostoTotal(ListaDeCamino &listaCamino)
CrearRuta(int r1, int dep, double oferta, int cli, double dem)
CrearRuta(int r1, int dep, double oferta, int cli, int v, double dem)
CrearRutaLibre(int dep, double oferta, int cli, double dem)
CrearRutaVacia(int r1, int dep)
DemandaTotal()
Destruir()
GetLista()
GetRuta(int r)
GetVentana(clientes &c, depositos &d, ListaDeCamino &listaDeCamino)
InsEnRuta(int r1, int cli1, int cliins, double dem)
InsPrimero(int r1, int cli, double dem)
InsRutas(Rutas *r)
InsUltimo(int r1, int cli, double dem)
PrimerCli(int r)
RutaCli(int cli)
RutaDep(int n, int dep)
Rutas()
~Rutas()
UltimoCli(int r)
UnirRutas(int r1, int r2)
listaDeRuta
MaxId

Sitios

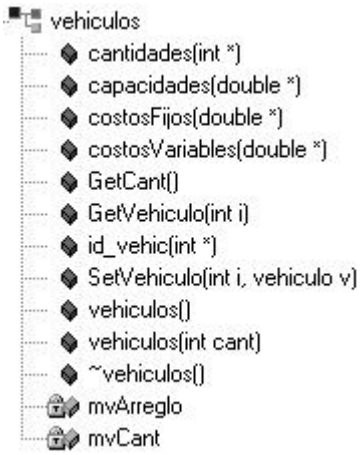

sitio
GetCaminosHaciaClientes()
GetCaminosHaciaDepositos()
GetCOSTO_ANT()
GetCOSTO_POS()
GetIdExterno()
GetIdInterno()
GetTIEMPO_ANT()
GetTIEMPO_POS()
GetVENTANA()
GetX_NODO_ANT()
GetX_NODO_POS()
GetY_NODO_ANT()
GetY_NODO_POS()
SetCOSTO_ANT(double)
SetCOSTO_POS(double)
SetIdExterno(int)
SetIdInterno(int)
SetTIEMPO_ANT(double)
SetTIEMPO_POS(double)
SetVENTANA(Ventana)
SetX_NODO_ANT(double)
SetX_NODO_POS(double)
SetY_NODO_ANT(double)
SetY_NODO_POS(double)
sitio(const sitio &)
sitio()
sitio(int, int, double, double, double, double, double, double, double, double)
~sitio()
mvCaminosHaciaClientes
mvCaminosHaciaDepositos
mvCOSTO_ANT
mvCOSTO_POS
mvIdExterno
mvIdInterno
mvTIEMPO_ANT
mvTIEMPO_POS
mvVENTANA
mvX_NODO_ANT
mvX_NODO_POS
mvY_NODO_ANT
mvY_NODO_POS


cliente
cliente()
cliente(const cliente &)
~cliente()
EsDelivery()
EsPickup()
GetDEMANDA()
GetDemandaSVRP()
GetPar()
GetTIEMPO_SERVICIO()
SetDEMANDA(double)
SetDemandaSVRP(variableAleatoria va)
SetEsPickup()
SetPar(int)
SetTIEMPO_SERVICIO(double)
UnSetEsPickup()
mvDEMANDA
mvDemandaSVRP
mvEsPickup
mvPar
mvTIEMPO_SERVICIO
clientes
cli_id(int *)
clideliv(int *)
clientes(int cant)
clientes()
~clientes()
clipickup(int *)
demandas(double *)
demandasSVRP(double *)
GetCant()
GetCliente(int i)
GetClienteInterno(int i)
SetCliente(int i, cliente c)
TAteCliDel(double *)
TAteCliPick(double *)
Tatencion(double *)
Tmaximos(double *)
TmaximosDARP(double *)
Tminimos(double *)
TminimosDARP(double *)
mvArreglo
mvCant
depositos
dep_id(int *)
depositos()
depositos(int cant)
~depositos()
GetCant()
GetDeposito(int i)
GetDepositoInterno(int idInt)
ofertas(double *)
SetDeposito(int i, deposito d)
Tmaximos(double *)
Tminimos(double *)
mvArreglo
mvCant
deposito
deposito()
deposito(const deposito &)
~deposito()
GetCAPACIDAD()
SetCAPACIDAD(double)
mvCAPACIDAD
sitioDijkstra
GetCAMINO()
SetCAMINO(int)
sitioDijkstra()
sitioDijkstra(int idInt, int idExt, double xa, double ya, double xp, double yp, int c,
~sitioDijkstra()
mvCAMINO

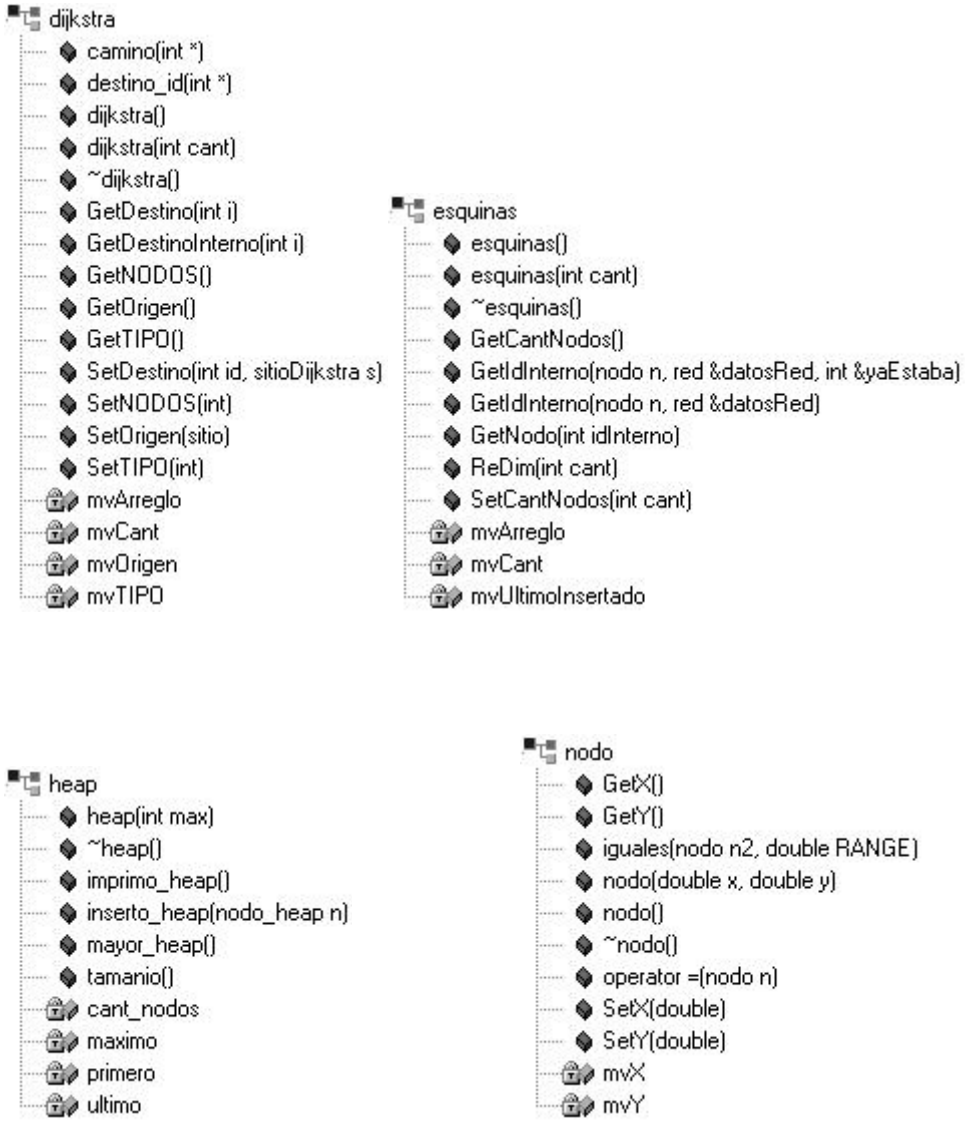

Ventana
AgregarPeriodo(double inicio, double fin)
EstaEnPeriodos(double valor)
EsVacia()
GetCantPeriodos()
GetPeriodo(int periodo, double &inicio, double &fin)
GetPeriodoFinalMaximo()
GetPeriodoInicialMinimo()
Ventana()
~Ventana()
mvCantPeriodos
mvlistaPeriodos

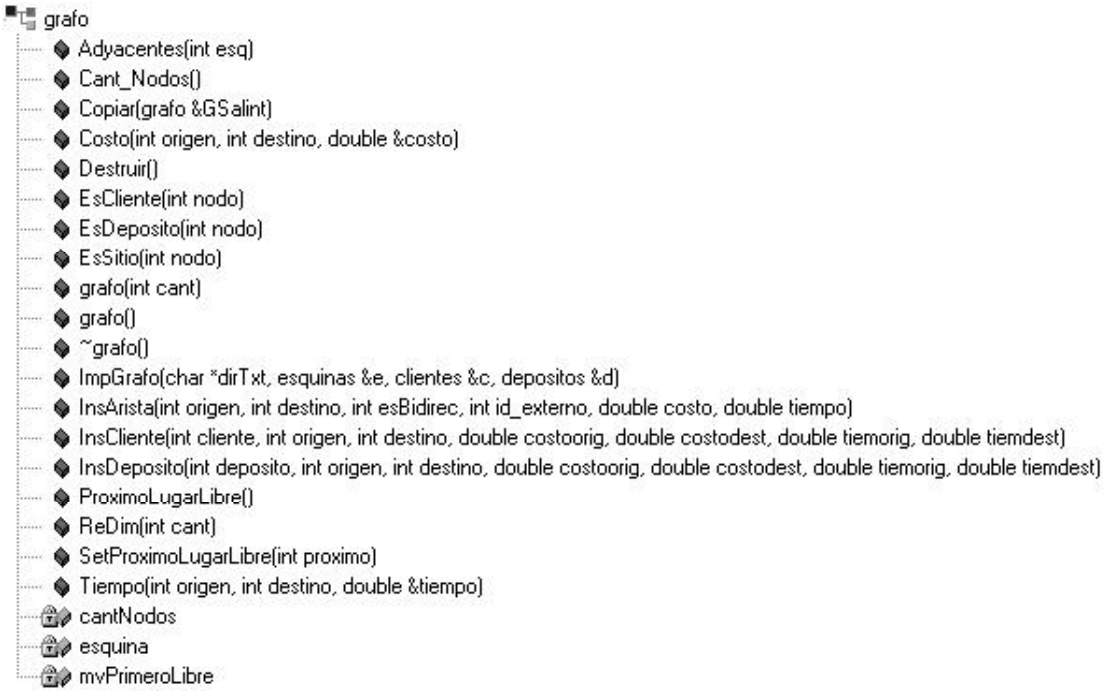
Vehículos


vehiculo
GetCANTIDAD()
GetCAPACIDAD()
GetCOSTO_FIJO()
GetCOSTO_VARIABLE()
GetIdExterno()
GetIdInterno()
operator =(vehiculo v)
SetCANTIDAD(int)
SetCAPACIDAD(double)
SetCOSTO_FIJO(double)
SetCOSTO_VARIABLE(double)
SetIdExterno(int)
SetIdInterno(int)
vehiculo()
vehiculo(int idExt, int idInt, int cant, double cap, double cf, double cv)
~vehiculo()
mvCANTIDAD
mvCAPACIDAD
mvCOSTO_FIJO
mvCOSTO_VARIABLE
mvIdExterno
mvIdInterno


vehiculos
cantidades(int *)
capacidades(double *)
costosFijos(double *)
costosVariables(double *)
GetCant()
GetVehiculo(int i)
id_vehic(int *)
SetVehiculo(int i, vehiculo v)
vehiculos()
vehiculos(int cant)
~vehiculos()
mvArreglo
mvCant

Varios útiles




grafo
Adyacentes(int esq)
Cant_Nodos()
Copiar(grafo &GSalint)
Costo(int origen, int destino, double &costo)
Destruir()
EsCliente(int nodo)
EsDeposito(int nodo)
EsSitio(int nodo)
grafo(int cant)
grafo()
~grafo()
ImpGrafo(char *dirTxt, esquinas &e, clientes &c, depositos &d)
InsArista(int origen, int destino, int esBidirec, int id_externo, double costo, double tiempo)
InsCliente(int cliente, int origen, int destino, double costoorig, double costodest, double tiemorig, double tiemdest)
InsDeposito(int deposito, int origen, int destino, double costoorig, double costodest, double tiemorig, double tiemdest)
ProximoLugarLibre()
ReDim(int cant)
SetProximoLugarLibre(int proximo)
Tiempo(int origen, int destino, double &tiempo)
cantNodos
esquina
mvPrimeroLibre

Apéndice
I

Apéndice I: Sistemas de Información Geográfica

Sistemas de Información Geográfica (GIS)

A continuación se realiza una introducción al concepto de Sistemas de Información Geográfica, definiendo términos y conceptos que se usan en este documento.

Los Sistemas de Información Geográfica (GIS), pueden ser usados para investigaciones científicas, administración de recursos, simulación de eventos, planificación de desarrollo, así como otros.

Definición de un GIS

Se han dado muchas definiciones de lo que es un GIS pero en un sentido estricto se puede decir que un GIS en un sistema informático capaz de reunir, almacenar, manipular y visualizar información geográficamente referenciada, es decir datos identificados por su ubicación en un mapa.

Forma de Trabajo de un GIS

Relacionando Información de Distintos Orígenes

Por lo general un GIS tomará un mapa como base de trabajo sobre la cuál almacenará y desplegará datos. Es común que el mapa y los datos no provengan siempre de la misma fuente, será aquí en donde el GIS deberá relacionar las distintas fuentes de información para unificar todo dentro del mapa, por ejemplo podrían vincularse fotos áreas de un satélite con los porcentajes de lluvia.

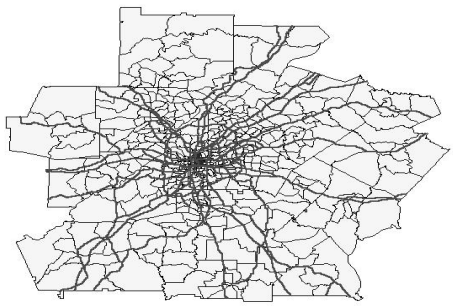


Figura 1. Mapa de Atlanta que viene con Arcview

Captura de Datos

Si la información a ser utilizada no está ya digitalizada, la misma debe ser convertida a un formato en el que la misma pueda ser reconocida por una computadora, los mapas pueden ser digitalizados, o dibujados utilizando programas destinados a estos efectos y de esta forma capturar las coordenadas de los distintos elementos.
Un GIS puede ser utilizado para enfatizar la relación espacial que existe entre los distintos elementos de un mapa, por ejemplo mientras que en un mapa digital una calle será seguramente representada y vista como una línea, en un GIS podrá reconocerse esa calle como el borde entre un área de bosques y un área urbanizada, o porque no como el enlace que une dos ciudades.
El proceso de la captura de datos, es por lo general una de las etapas que consume mayor cantidad de tiempo dentro de la ardua tarea que consiste en la creación de un GIS.

Integración de los Datos

Un GIS hace posible la vinculación o integración de información que de otra manera resultaría muy difícil de vincular. Por ejemplo con la utilización de un GIS una compañía de agua podrá determinar que cantidad de material séptico deberá destinar para la potabilización del agua consumida en una determinada área, esto se hará vinculando a un GIS el consumo de agua de una determinada zona.

Estructuras de Datos

Cómo ya mencionamos una de las principales propiedades de un GIS es que permite vincular información que de otra forma resultaría prácticamente imposible vincular. Para poder hacer esta vinculación un GIS deberá tener definidas las estructuras de datos necesarias para poder efectuar está tarea, por ejemplo será necesario identificar y vincular distintos puntos del mapa digital como pertenecientes a una misma zona, por lo cual son necesarias estructuras de datos que permitan hacer esta vinculación.

Modelado de Datos

Una de las principales características de un GIS se encuentra vinculada al hecho de que el poder almacenar información relacionada con un mapa digital permite representar la idea tridimensional de la Tierra, por ejemplo a través de un GIS se puede conocer los contornos de las zonas con igual índices de lluvias, o con igual índices de temperaturas, es decir se puede desplegar distintos tipos de información basados en el mismo mapa base, de acuerdo a la información que hayamos vinculado al mismo.

Función de un GIS

La forma en qué los datos son almacenados y desplegados permiten que un GIS sea una herramienta que resulta especialmente adecuada para la realización de complejos análisis.

Devolución de Información

La manera típica en que un GIS devuelve información es la siguiente, cuando una persona cliquea sobre un punto específico del mapa la información almacenada relacionada ese punto es desplegada, también se podría pedir que coloreando el mapa el GIS pinte de un mismo color todas aquellas áreas en donde variables específicas tienen el mismo valor.

Modelo Topológico

A través de un GIS pueden manejarse relaciones de espaciales entre los distintos objetos que conforman nuestro sistema, por lo que un GIS nos permitirá determinar condiciones de adyacencia, proximidad, inclusión, entre los distintos objetos que forman parte del mapa digital.

Redes

Un GIS permite almacenar información referente a distintas redes, como por ejemplo una red de distribución, de esta forma se pueden simular distintos eventos como por ejemplo el cálculo del tiempo total que llevaría un recorrido en la antes mencionada red de distribución.

Salida

Por lo general la salida de un GIS resultará un gráfico en pantalla, que permitirá el análisis de las personas que deben tomar decisiones de la situación existente reflejada en ese mapa. Es entonces un aspecto crítico de un GIS la forma en que las respuestas son mostradas al usuario, teniendo en cuenta que información será más relevante para el usuario, dejando de lado aquella que no aporte nada en el caso específico que se esta tratando.

Aplicaciones de GIS

GIS a Través de la Historia.

En las paredes de las cavernas de Lascaux, Francia, el hombre de Cro-Magnon dibujo imágenes de los animales que cazaba, asociado a estos dibujos hay historias que relatan la migración de las especies en las distintas épocas del año. En esencia estos dibujos con sus respectivas historias siguen los dos aspectos fundamentales de un GIS, un archivo gráfico vinculado con objetos de una base de datos; en este caso el dibujo de cada animal vinculado a su ciclo migratorio.

Creación de Mapas

Actualmente los investigadores están trabajando para incorporar la experiencia de los cartógrafos en área de GIS, para la producción automática de mapas.

Selección de Lugares Estratégicos

Una de las más recientes aplicaciones de GIS se encuentra en el área de toma de decisiones, en particular en el campo de la selección de lugares estratégicos, por ejemplo volvamos a nuestra compañía de agua, ahora quiere establecer una nueva planta potabilizadora, es en caso en donde la información almacenada en el GIS le permitirá a nuestra empresa determinar la locación de la nueva planta con respecto a las distintas áreas de consumo buscando disminuir los gastos de distribución del agua.

Planificación de Sistemas de Emergencia

Por ejemplo un GIS puede ser utilizado para combinar la red vial, con información geológica, para analizar en caso de un terremoto el tiempo de respuesta de las unidades de emergencia.
Para poder llevar a cabo está planificación información especifica del problema deberá ser almacenada en nuestro GIS.

Simulación de Efectos Climáticos y Desastres Ambientales

Conociendo características especiales de los terrenos y redes fluviales que atraviesan los mismos, es que se puede simular el efecto de un determinado cambio climático o desastre ambiental, por ejemplo conociendo el modelo de mareas de un río, es que se puede simular y de esta forma prever el efecto de un derramamiento de petróleo en un lugar específico.

El Futuro de los GIS

Muchas disciplinas se han visto beneficiadas por la aplicación de las técnicas de GIS, pero aún existe una amplia gama de problemas en donde aún no se ha incorporada esta tecnología, pero sería de gran importancia su incorporación, además a los ya existentes GIS, se encuentre actualmente en estudio la posibilidad de expandirlos agregando nuevas características a los mismos.

Dos nuevos campos en donde se está buscando la incorporación de las tecnologías de GIS son los siguientes:

Historia y Cambio Global del Clima

La comunidad científica internacional ha reconocido y admitido las consecuencias ambientales de la actividad humana, es en este campo que la tecnología de GIS ha cobrado importante relevancia, como una herramienta de análisis, para comprender este proceso global de cambio. Un variado número de mapas e imágenes de satélites pueden ser combinados para simular la interacción de variados y complejos sistemas naturales.

Incorporando el Tiempo

La tecnología de GIS da a los investigadores la habilidad de examinar las variaciones de la Tierra con el correr del tiempo, para ello debe incorporarse a los GIS tradicionales la posibilidad de vincular a cada punto no un único registro, sino varios registros idénticos cuya variable independiente será la fecha u hora en cual dicha información fue vinculada con el punto del mapa.

Conclusiones

Los GIS tienen gran campo de aplicación, y su principal ventaja sobre la cartografía y otros métodos tradicionales se basa en su característica dinámica, lo que le permite adaptarse con gran facilidad a los cambios topológicos de las áreas en estudio.
El campo de aplicación de los GIS no tiene límites, y puede ser tanto en el área de las investigaciones, como en el área comercial.

Apéndice **J**

Apéndice J: Estudio de Soluciones Existentes

Estudio de Soluciones Existentes

En este apéndice se incluye un estudio de las soluciones comerciales existentes en el área del ruteo de vehículos. Dicho estudio se realizó en 1999 por Carlos Pedezert, Javier Barreiro y Nicolás Sosa en el contexto de Taller V.
Aquí solo se presentan las principales conclusiones, teniendo que referirse a la documentación original por mas información.

A continuación se presenta una lista de productos, obtenida de la biblioteca de software de investigación de operaciones de la Universidad de Karlsruhe [10].

Soluciones Existentes

DynaRoute

DynaRoute es un sistema de ruteo de vehículos y scheduling para el manejo de despacho en tiempo real de vehículos. El sistema determina asignaciones óptimas entre vehículos y nuevas órdenes tomando en cuenta la posición del vehículo, horas de disponibilidad del vehículo y horas de disponibilidad del cliente.
http://www.saitech-inc.com (4/99)

GeoRoute 5

GeoRoute 5 usa una base de datos para representar la red de calles en forma detallada junto con un conjunto de algoritmos de ruteo especializado que producen eficientes rutas punto a punto. Puede ser usado diariamente o periódicamente para generar rutas basadas en pedidos de clientes o servicios periódicos. En ambos casos, GeoRoute 5 ayuda a reducir los costos operativos y a respetar los compromisos con los clientes.
Para cada cliente a ser visitado, se puede especificar distintos tipos de vehículos, cantidad que será entregada o retirada, tiempo que demorará el servicio y la posibilidad de ingresar ventanas de tiempo.
http://www.giro.ca (4/99)

GeoRoute Postal

El producto GeoRoute Postal provee organización postal con funciones adicionales. Esto incluye la habilidad de planificar y optimizar la entrega de correspondencia como también operaciones motorizadas tales como recolección de mailbox, distribución de paquetes, etc.

Este sistema permite también definir los recursos disponibles y los modos de viaje (a pie, bicicleta, auto, etc.). Para cada modo, se puede especificar la velocidad de viaje y si las restricciones viales (calles flechadas, por ej.) deben ser respetadas o no. Los modos pueden combinarse, como en el caso que se use un vehículo para trasladarse a determinada zona y luego el reparto se haga a pie.
http://www.giro.ca (4/99)

RouteSmart

RouteSmart evalúa las diversas formas en las cuales el conductor (o conductores) pueden viajar a través de las calles y servir a los clientes en su ruta, teniendo en cuenta diversas restricciones operacionales como capacidad de los vehículos, horario de servicio de los clientes, calles flechadas, etc.
http://www.routesmart.com (4/99)

SHIPCONS II

SHIPCONS II es un sistema designado para asistir a aquellas personas que deben planificar el despacho de productos para tomar mejores decisiones acerca de los recursos de transporte.
SHIPCONS II recibe información acerca de las órdenes que serán despachadas y los vehículos disponibles (con su capacidad, costo, etc.). Esta información es procesada para generar un cierto número de rutas factibles, las cuales son la entrada para un optimizador que seleccionará el conjunto de rutas que aseguren que todas las órdenes serán despachadas al menor costo.
http://www.insight-mss.com (4/99)

TransCAD

TransCAD es un sistema de información geográfica diseñado específicamente para ser usado por profesionales del área de transporte para almacenar, mostrar, manejar y analizar datos de transporte.
TransCAD puede ser usado para todos los modos de transporte, a cualquier escala o nivel de detalle.
Este sistema provee un poderoso sistema de información geográfica con extensiones especiales para el área de transporte, herramientas de mapeo y visualización diseñadas para aplicaciones de transporte y módulos de aplicación para ruteo.
http://www.caliper.com (4/99)

Trapeze-Taxi

Este sistema es una aplicación de ruteo y manejo diseñada para compañías de taxi.
Con Trapeze-taxi se puede despachar taxis en forma instantánea y notificar a los conductores los cambios, construir una base de datos de clientes con información estadística, generar reportes, etc.
www.trapezesoftware.com (4/99)

Direct Route

El poderoso motor de ruteo de DirectRoute diseña rutas basadas en la ubicación del cliente, capacidad de los vehículos y horarios del cliente. Esto asegura el aprovechamiento máximo de los vehículos, la minimización de la distancia recorrida y la puntualidad del servicio.
http://www.appianlogistics.com (4/99)

GeoRoute-Municipal

Geo-Route Municipal es un poderoso software de ruteo que está específicamente diseñado para la creación automática de rutas para servicios en las ciudades, tales como recolección de residuos, limpiado de calles, remoción de nieve, etc. El software provee todas las herramientas necesarias para preparar los datos, crear y optimizar las rutas y mostrar los resultados en forma de mapas o reportes.
http://www.giro.ca (4/99)

LoadExpress Plus

LoadExpress Plus for Windows ofrece una forma simple, poderosa y flexible de construir y optimizar rutas para la realización de entregas a clientes. Este sistema soporta el ingreso de restricciones tales como tamaño variable de los vehículos, horarios de entrega, etc.
http://www.mile.com/loadexpress.htmi (4/99)

RiMMS

El software de ruteo y scheduling RiMMS permite a las empresas manejar mejor sus recursos de transporte, mejorando la productividad y la eficiencia, reduciendo los costos operacionales.
RiMMS ayuda a los usuarios a diseñar rutas de entrega y asignar tareas a los recursos. El sistema integra tecnología de mapeo para crear rápidamente mapas a color con instrucciones de ruta precisas, basándose en varios parámetros de la base de datos del usuario.
http://www.lightstone.com (4/99)

RoutePro Dispatcher

RoutePro Dispatcher permite construir rutas a partir de la información de los clientes o adaptar rutas existentes a fluctuaciones diarias.
Entre otros parámetros, es posible definir horarios para cada cliente, especificar prioridades para los mismos, definir distintos tipos de vehículos con costos asociados, especificar horarios para los vehículos, etc.
http://www.caps.com (4/99)

Routronics 2000

El sistema Routronics 2000 es un despachador en tiempo real de vehículos. Las ventajas de su uso radican en el ahorro de tiempo y combustible, así como la satisfacción del cliente.
Routronics 2000 además brinda un seguimiento continuo de los conductores. Cuando el despachador recibe un pedido, se le brindan distintas rutas posibles. Este hace su elección, y la información va directamente al conductor. Los pedidos no asignados, o no confirmados, son mostrados en pantalla para que el despachador tome la acción correspondiente.
http://www.carrierlogistics.com (4/99)

STARS

STARS (Smart Truck Assignment and Routing System) determina cargas de productos y rutas óptimas para vehículos de entrega. Como una herramienta de uso diario, STARS puede reducir substancialmente los costos operativos minimizando las distancias recorridas y el tiempo.
http://www.saitech-inc.com (4/99)

Optrak

Optrak reduce los costos de distribución planificando eficientemente las cargas y las rutas de los vehículos, haciendo el mejor uso de los recursos. Sus sofisticadas optimizaciones dan como resultado mejores soluciones a una amplia variedad de problemas.
http://www.optrak.co.uk (4/99)

Truckstops for Windows

Truckstops es un programa de ruteo que recibe tres tipos de dato:
- información de las paradas a realizar: clientes con la cantidad de bultos que se recogerán o entregarán, horarios, etc.
- información de los vehículos: costos, capacidad máxima
- información general: por ejemplo, cantidad de depósitos.

http://www.bestroutes.com (4/99)

ArcLogistics Route

ArcLogistics Route es un sistema que resuelve problemas complejos de ruteo de vehículos y scheduling. Este sistema determina que vehículo debe servir a cada cliente, y la mejor secuencia de servicio para cumplir con los requisitos temporales de los mismos, minimizando la distancia (o tiempo) recorrida.
http://www.esri.com/software/arclogistics/whatsnew.html(10/00)

Resumen

A continuación se resumen en una tabla los casos que resuelven los paquetes comerciales mencionados anteriormente.

Producto	TSP	TSPTW	TSPHF	VRP	VRPTW	VRPHF	MDVRP	CPP	DARP
DynaRoute	-	-	-	-	-	-	-	-	Si
GeoRoute 5	Si	Si	Si	Si	Si	Si	Si	-	-
GeoPostal	-	-	-	-	-	-	-	Si	-
RouteSmart	Si	Si	Si	Si	Si	Si	Si	-	-
SHIPCONS II	Si	Si	Si	Si	Si	Si	Si	-	-
TransCad	Si	Si	Si	Si	Si	Si	Si	Si	-
Trapeze-Taxi	-	-	-	-	-	-	-	-	Si
Direct Route	Si	Si	Si	Si	Si	Si	S/i	-	-
GeoRoute-Municipal	-	-	-	-	-	-	-	Si	-
LoadExpress Plus	Si	Si	Si	Si	Si	Si	Si	-	-
RiMMS	Si	s/i	s/i	Si	s/i	s/i	S/i	-	-
RoutePro Dispatcher	Si	Si	Si	Si	Si	Si	Si	-	-
Routronics 2000	-	-	-	-	-	-	-	-	Si
STARS	Si	Si	Si	Si	Si	Si	Si	-	-
Optrak	Si	s/i	s/i	Si	s/i	s/i	Si	-	-
TruckStops for Windows	Si	Si	Si	Si	Si	Si	Si	-	-
ArcLogistics Route	Si	Si	Si	Si	Si	Si	Si	-	-

Apéndice **K**

Apéndice K: Biblioteca de Vínculos Dinámicos : Router

Biblioteca de Vínculos Dinámicos : Router

La biblioteca Router es la encargada de resolver los problemas de ruteo, por tanto es una parte medular del sistema.
En esta sección se describe: Generalidades, Características del Usuario, Herramienta de desarrollo, Interfase, Entrada de la biblioteca, Salida de la biblioteca Funciones Exportadas, Códigos de Error y Ejemplos.

Introducción

El objetivo de esta biblioteca de vínculos dinámicos (y del software en general), es resolver una gama importante de problemas de ruteo. Se toma la decisión de encapsular la solución de dichos problemas en una librería para lograr independencia entre la interfase y la implementación, pensando en futuros cambios y mejoras sobre la biblioteca en si misma.

Los problemas de ruteo resueltos por esta librería son:

- **VRP** (Vehicle Routing Problem). En este problema, dado un depósito y uno o varios clientes, se determina la cantidad de vehículos (todos con la misma capacidad de carga) y la ruta tales el recorrido de los vehículos sea mínimo.
- **MDVRP** (Multi-Depot Vehicle Routing Problem). Este problema es similar al anterior, pero la cantidad de depósitos es mayor a 1.
- **VRPTW** (Vehicle Routing Problem with Time-Windows). En este problema, cada cliente tiene una ventana de tiempo (o time-window), es decir, un rango de tiempo en el que el cliente puede ser atendido. Este problema considera un solo depósito que tiene asociado una ventana de tiempo que indica el horario en que presta servicios.
- **MDVRPTW** (Multi-Depot Vehicle Routing Problem with Time-Windows). Ídem anterior, pero se consideran varios depósitos.
- **VRPHF** (VRP Flota Heterogénea). Ídem VRP, pero en este caso los vehículos tienen diferente capacidad. Por lo tanto ya no son itercambiables entre si. Se debe generar una ruta para cada vehiculo.
- **TSP** (Travelling Salesman Problem). Este problema es un caso particular del VRP, donde se tiene un sólo depósito y un solo vehículo, y donde la demanda de los clientes es cero.
- **MTSP** (Multi-Vehicle Travelling Salesman Problem) Este problema es un caso particular del VRP, donde se tiene un sólo depósito y varios vehículos-, y donde la demanda de los clientes es cero.

- **DARP** (Dial-a-ride Problem). Este problema resuelve el caso en que cada cliente tiene un punto de carga y un punto de entrega, todo dentro de una ventana de tiempo. El problema es, como en los casos anteriores, minimizar el costo para servir a todos los clientes, partiendo desde un depósito dado.
- **CPP** (Chinese Postman Problem). El problema del CPP es hallar un recorrido tal que pase por todos los clientes y su costo sea mínimo, sin embargo la variante del CPP resuelta por este software recibe las demandas de los clientes, no de los arcos.
- **SVRP** (Stochastic Vehicle Routing Problem). Es una generalización del problema clásico VRP, donde los clientes tienen demanda no determinista.

Por mas detalles sobre los tipos de problemas se puede consultar el apéndice: definición de Problemas.

Con respecto a la comunicación de la librería con el resto del sistema, la misma se reduce a un archivo de entrada y otro de salida, como se especifica en la sección correspondiente.

Características del Usuario

En el caso de esta librería, el usuario final, es un programador o una persona con sólidos conocimientos en computación. Esta biblioteca está pensada para ser utilizada en el sistema desarrollado o de forma independiente para otras aplicaciones donde sea necesario resolver problemas de ruteo.

Se acuerda con el usuario que los errores en la llamada a funciones y /o en el pasaje de parámetros deben ser detectados por la biblioteca, es decir son responsabilidad, de la misma. Como método de manejo de errores se adopta la devolución de un código de error, como se explica mas adelante.

Herramienta de Desarrollo

Debido a la característica netamente algorítmica de la biblioteca, se elige como lenguaje de programación C++ para su implementación.
Además, se opta por Visual C++ por requerimiento explicito del usuario, y por la familiarización previa de los programadores con el ambiente de desarrollo.

Arquitectura de Comunicación: Interfase

Debido a que el motor de cálculo es programado en forma independiente de la interfase de usuario (y en otro lenguaje de programación, C++), es necesario implementar algún mecanismo de comunicación entre estos.

La interfase es pensada, en este caso, buscando:

- **Claridad**. Se pretende que se los datos de entrada y salida puedan ser interpretados intuitivamente.
- **Simplicidad**. Se busca simplicidad en el desarrollo de programación.

- **Generalidad**. Es necesario una interfase lo suficientemente maleable como para adaptarse a nuevos tipos de problemas o cambios en los mismos.
- **Minimizar Información**. Se pretende eliminar la redundancia en los datos de entrada y salida, para favorecer la claridad antes mencionada.

Como solución a estos requerimientos se presenta como opción mas clara la utilización de archivos de texto. Para eliminar la dificultad que puede encerrar el manejo de los mismos se utiliza el **formato .INI.** Este formato permite el manejo rápido y simple de archivos de texto y fue desarrollado por Microsoft (por mas información ver la ayuda de las API de Windows).

Como conclusión de esta sección se tiene que la interfase de la biblioteca se realiza mediante dos archivos en formato .INI. La entrada.INI donde se especifican los datos del problema a resolver, y la Salida.INI donde se devuelve la solución al problema de ruteo.

Entrada de la Biblioteca

Generalidades

Esta interfase pretende especificar completamente la entrada de la DLL para los problema: VRP, MDVRP, VRPTW, MDVRPTW, VRPHF, TSP, MTSP, DARP, CPP y SVRP.

Básicamente la entrada es un conjunto de clientes (llamados también sitios) a los cuales hay que satisfacer su demanda, partiendo de uno o varios depósitos. Restringiendo la solución a las particularidades de cada tipo de problema.

Como antes se mencionó, la entrada del motor de calculo (.dll) se limita a un archivo de texto en formato .INI.
Dicho archivo de texto, de ahora en mas referido como Entrada.INI, tiene el nombre y ubicación (path) que el usuario del motor determino en la entrada (ver: Funciones Exportadas).

Especificación

Los datos que aparecen en el archivo de entrada dependen en gran forma del problema de ruteo que se esta resolviendo. Por ejemplo para el caso de flota heterogénea se especifican varios tipos de vehículos, en cambio para el TSP es necesario no especificar ninguno. Para el caso de ventana de tiempo se especifica la ventana de tiempo admisible para los clientes y depósitos, etc.

Principales identificadores

A continuación se detalla una tabla de los principales identificadores utilizados, según grupo y clave del formato .INI.

El archivo de entrada siempre está encabezado por el grupo [PROBLEMA], que contiene los datos mas relevantes necesario para la definición del problema de ruteo.

Posteriormente los grupos [SITIO_Z] y [DEPOSITO_Z] definen los clientes y depósitos del problema, numerados de forma creciente, comenzando en 1 y terminando en la cantidad definida en [PROBLEMA].

Problema:

```
[PROBLEMA]      Definición del problema
TIPO=           Tipo de problema (Entero)
                1- VRP Con todos los caminos precalculados.
                2-    VRP Donde los caminos  NO precalculados
                      se calculan utilizando Dijkstra.
                3-    VRP Donde los caminos  NO precalculados
                      se calculan utilizando BFS.
                4-    VRP Donde los caminos  NO precalculados
                      se calculan utilizando DFS.
                10- MDVRP Con todos los caminos
                    precalculados.
                20- VRPTW Con todos los caminos
                    precalculados.
                30- MDVRPTW Con todos los caminos
                    precalculados.
                40- VRPHFE Con todos los caminos
                    precalculados.
                50- TSP Con todos los caminos precalculados.
                60- DARP Con todos los caminos precalculados.
                70- CPP Con todos los caminos precalculados.
                80- MTSP Con todos los caminos precalculados.
                90- SVRP Con todos los caminos precalculados.

SITIOS=         Cantidad de clientes (Entero)
DEPOSITOS=      Cantidad de depósitos (Entero)
VEHICULOS=      Cantidad de tipos (Entero)

RED=            Cantidad de arcos de la red del CPP(Entero)
PARES =         Cantidad de pares de sitios carga-entrega
                para el DARP (Entero)

PATH_CAMINOS =  Dirección y nombre de archivo completo, donde
                se encuentran los caminos (en el formato
                especificado en la SALIDA) precalculados.
                (Atención: puede ser este mismo archivo o el
                archivo de salida).(String)
```

Los sitios son los clientes a los cuales se debe abastecer (para la mayoría de los problemas planteados).

Sitios:

```
[SITIO_Z]       Definición del sitio numero Z
ID=             identificador externo del sitio (Entero)
```

```
X_NODO_ANT=     Coordenada X del nodo anterior (Float)
Y_NODO_ANT=     Coordenada Y del nodo anterior (Float)
X_NODO_POS=     Coordenada X del nodo posterior (Float)
Y_NODO_POS=     Coordenada Y del nodo posterior (Float)
DEMANDA=        Demanda del sitio (Float)
VENTANA=        identificador de la ventana (que figura en el
                mismo archivo INI) (Entero)

TIEMPO_SERVICIO=     Tiempo de servicio del sitio (Float)

DIST_ANT=       Distancia al nodo anterior (Float)
DIST_POST=      Distancia al nodo posterior (Float)
DEM_ANT=        Demora en tiempo al nodo anterior (Float)
DEM_POST=       Demora en tiempo al nodo posterior (Float)

CAMINO_HACIA_S_k     Camino de ir desde el sitio actual (Z) al
                     sitio k (Entero)
CAMINO_HACIA_D_k     Camino de ir desde el sitio actual (Z) al
                     deposito k (Entero)
```

Para todos los problemas es necesario tener definido un deposito , citándose como caso particular los problemas de múltiple deposito, donde es necesario tener varios.

Depósitos:

```
[DEPOSITO_Z]      Definición del deposito Z
ID=               identificador externo del deposito (Entero)
X_NODO_ANT=       Coordenada X del nodo anterior (Float)
Y_NODO_ANT=       Coordenada Y del nodo anterior (Float)
X_NODO_POS=       Coordenada X del nodo posterior (Float)
Y_NODO_POS=       Coordenada Y del nodo posterior (Float)

VENTANA=          identificador de la ventana (que figura en el
                   mismo archivo INI) (Entero)
CAPACIDAD=        Capacidad del deposito (Float)

DIST_ANT=         Distancia al nodo anterior (Float)
DIST_POST=        Distancia al nodo posterior (Float)
DEM_ANT=          Demora en tiempo al nodo anterior (Float)
DEM_POST=         Demora en tiempo al nodo posterior (Float)

CAMINO_HACIA_S_k      Camino de ir desde el deposito actual (Z) al
                      sitio k (Entero)
                      Para el caso del CPP, debe interpretarse
                      como el camino al punto medio del arco k
CAMINO_HACIA_D_k      Camino de ir desde el deposito actual
                     (Z) al deposito k (Entero)
```

En la mayoría de los algoritmos es necesario tener definido algún tipos de vehículos. Y para el caso de flota heterogénea es necesario definir varios (mas de uno) tipos de vehículos.

Vehículos:

[TIPO_VEHICULO_Z]	Definición de todos los vehículos de tipo_z
CANTIDAD=	Cantidad de vehículos de este tipo (Entero)
ID=	identificador externo del tipo_vehiculo
CAPACIDAD=	Capacidad de los vehículos del tipo z (Float)
COSTO_FIJO=	Costo fijo de los vehículos del tipo z (Float)
COSTO_VAR=	Costo variable de los vehículos del tipo z (Float)

Para los problemas con ventanas de tiempo es necesario el grupo [VENTANA].

Ventana:

[VENTANA_Z]	Definición del la ventana de tiempo Z
CANTIDAD_PERIODOS=	Cantidad de periodos que hay en la ventana (Entero)
PERIODO_INICIO_Z=	Hora de inicio del periodo Z (Float)
PERIODO_FINAL_Z=	Hora de final del periodo Z (Float)

Para los problemas tipo CPP, se especifican los arcos. Dichos arcos se pueden pensar como los clientes, es decir, los sitios, de este tipo de problema.

Arco:

[ARCO_Z]	Definición del arco Z
ID=	identificador externo del arco (Entero)
COSTO=	Costo del arco (Float)
TIEMPO=	Tiempo del arco (Float)
X_NODO_ANT=	Coordenada X del nodo anterior (Float)
Y_NODO_ANT=	Coordenada Y del nodo anterior (Float)
X_NODO_POS=	Coordenada X del nodo posterior (Float)
Y_NODO_POS=	Coordenada Y del nodo posterior (Float)
CAMINO_HACIA_S_k	Camino de ir desde el punto medio del arco actual (Z) al punto medio del arco k (Entero)
CAMINO_HACIA_D_k	Camino de ir desde el punto medio del arco actual (Z) al deposito k (Entero)

Para los problemas tipo DARP, donde cada cliente tiene un sitio de pickup y un sitio de delivery, se especifican los pares. Cada sitio pertenece a algún par, como sitio de

carga o como sitio de entrega. Por tanto para este tipo de problemas hay el doble de sitios que de pares.

Pares:

```
[PAR_Z]          Definición del par carga-entrega Z
SITIO_CARGA=     identificador del sitio de carga(que figura en
                 el mismo archivo INI)
SITIO_ENTREGA=   identificador del sitio de entrega(que figura
                 en el mismo archivo INI)
```

Para los problemas con demanda no determinista (SVRP), se define una variable aleatoria que cumple los efectos de dicha demanda. Esta variable aleatoria es única, y la misma para todos los clientes [4].

Demanda No Determinista (variable aleatoria):

```
[VA_DEMANDA]     Definicion de la variable aleatoria que
                 representa la demanda de cada cliente para el
                 SVRP.
TIPO =           0-Uniforme, 1-Normal (Entero)
MEDIA=           Valor medio de la variable aleatoria (Float)
VARIANZA=        Varianza de la variable aleatoria (Float)
```

Salida de la Biblioteca

Generalidades

Esta interfase pretende especificar completamente la salida de la DLL para los problema: VRP, MDVRP, VRPTW, MDVRPTW, VRPHF, TSP, MTSP, DARP, CPP y SVRP.

Básicamente la salida de la biblioteca es una ruta, o conjunto de rutas. Dichas rutas son la solución completa al problema de ruteo que se le a presentado al motor de calculo como entrada.

Al igual que la entrada, la salida del motor de calculo (.dll) se limita a un archivo de texto en formato .INI.
Dicho archivo de texto, de ahora en mas referido como Salida.INI, tiene el nombre y ubicación (path) que el usuario del motor determino en la entrada (ver: Funciones Exportadas).

Especificación

Con el fin de esclarecer los términos y la notación utilizada se realizaran una serie de puntualizaciones.

[4] Las demandas de los clientes forman un IID: variableas aleatorias independientes e idénticamente distribuidas.

Existen tres tipos básicos de nodos a distinguir: los “depósitos”, los “clientes” y las “esquinas”.
Una “ruta” es el recorrido desde un deposito a otro deposito pasando por varias esquinas y /o clientes.
Un “camino” es el recorrido solo a través de esquinas de un deposito (o cliente) a un deposito (o cliente).

Por tanto, la salida es un conjuntos de rutas, las cuales a su vez están formadas por un conjunto de caminos.

Los datos que aparecen en el archivo salida dependen en gran forma del problema de ruteo que se esta resolviendo. Por ejemplo para el caso de flota heterogénea cada ruta tiene un vehículo asignado. Para el caso de ventana de tiempo cada ruta y camino tiene una ventana de tiempo en la cual se debe realizar su recorrido, etc.

Principales identificadores

A continuación se detalla una tabla de los principales identificadores utilizados, según grupo y clave del formato .INI.

El archivo de salida siempre está encabezado por el grupo [SOLUCION], que contiene los datos mas relevantes de la solución al problema de ruteo.

Solución:

[SOLUCION]	Datos de la solución
RUTAS=	Cantidad de rutas (Entero)
DEMANDA=	Demanda de la solución (Float)
COSTO=	Costo de la solución (Float)
NODOS=	Cantidad de nodos (clientes, depósitos o esquinas) que componen la solución (Entero)
VENTANA=	identificador de la ventana de la solución (que figura en el mismo archivo INI). La ventana de la solución, representa el periodo mínimo inicial y el periodo máximo final, considerando todas las rutas (Entero).

Posteriormente al grupo [SOLUCION], se presentan las rutas de la solución, numeradas de forma creciente, comenzando en 1 y terminando en la cantidad de rutas determinadas en [SOLUCION].

Rutas:

[RUTA_Z]	Definición de la ruta numero Z
VEHICULO=	identificador externo del vehículo que realiza la Ruta (Entero)
DEMANDA=	Demanda satisfecha por la ruta(Float)
COSTO=	Costo de la ruta (Float)
TIEMPO=	Tiempo de la ruta (Float)
VENTANA=	identificador de la ventana (que figura en el mismo archivo INI) (Entero)
ID_DEPOSITO_INICIAL=	El id del deposito inicial (Entero)
ID_DEPOSITO_FINAL=	El id del deposito final (Entero)

```
CAMINO_INICIAL=El numero de camino inicial en esta ruta
               (Entero)
CAMINO_FINAL=  El numero de camino final en esta ruta (Entero)
```

Cada [RUTA_Z], está formada por un conjunto de caminos, comenzando en CAMINO_INICIAL y terminando en CAMINO_FINAL, especificados en el propio grupo [RUTA_Z].

Caminos:

```
[CAMINO_Z]      Definición del camino numero Z
COSTO=          Costo del camino (Float)
TIEMPO=         Tiempo del camino (Float)
VENTANA=        identificador de la ventana (que figura en el
                mismo archivo INI) (Entero)
TIPO_INICIAL=   0-Cliente y 1-Deposito (Entero)
ID_INICIAL=     El id del deposito o cliente inicial (Entero)
OPERACIÓN_INICIAL=   0-De paso, 1-Despachar y 2-Cargar
                (Entero)
TIPO_FINAL=     0-Cliente y 1-Deposito (Entero)
ID_FINAL=       El id del deposito final (Entero)
OPERACIÓN_FINAL= 0-De paso, 1-Despachar y 2-Cargar (Entero)


NODO_INICIAL=   El numero de nodo inicial en este camino
                (Entero)
NODO_FINAL=     El numero de nodo final en este camino (Entero)

CALCULAR_CAMINO=     Entero que indica (1) si hay calcular el
                     camino por que se manejo un costo exacto,
                     o (0) si hay que leerlo de este mismo
                     archivo y grupo.
```

Al igual que como las rutas están formadas de caminos, los caminos están formados de nodos.
Los nodos son necesarios para el caso en que la biblioteca además de realizar la solución al problema de ruteo, también se utilice para hallar los caminos mínimos entre dos depósitos (o clientes).

Nodo:

```
[NODO_Z]        Definición del nodo Z
TIPO=           0-Esquina, 2-Comienzo ArcoCPP 3-Fin ArcoCPP
                (Entero)
X_NODO=         Coordenada X del nodo (Float)
Y_NODO=         Coordenada Y del nodo (Float)
VENTANA=        Para el caso de depósitos y clientes se debe
                especificar la ventana de atención. (Entero)
```

Al igual que en la entrada es necesario, para los problemas con ventanas de tiempo el grupo [VENTANA] .

Ventana:

```
[VENTANA_Z]          Definición del la ventana de tiempo Z
CANTIDAD_PERIODOS=   Cantidad de periodos que hay en la
                     ventana (Entero)
PERIODO_INICIO_Z=    Hora de inicio del periodo Z (Float)
PERIODO_FINAL_Z=     Hora de final del periodo Z (Float)
```

Funciones Exportadas

La funciones que exporta la biblioteca son: ***"CargarEstructura"*** , ***"Calcular"*** y ***"Dijkstra"***.

A pesar de que la biblioteca esta pensada para su uso exclusivo en la resolución de problemas de ruteo mediante la función ***"Calcular"***, ésta también incluye dos funcionalidades de extrema utilidad: ***"CargarEstructura"*** y ***"Dijkstra".*** Dichas funcionalidades se incluyeron para lograr una solución integral y una autonomía total de la biblioteca. A través de ***"Dijkstra"*** se puede calcular las mejores rutas para una topología de red genérica cargada mediante ***"CargarEstructura"***.
La interface de estas dos funcionalidades se detalla en las secciones: "Entrada para Cargar de la Red" y "Interface para el Calculo del Dijkstra"

A continuación se detallan cada una de las funciones exportadas por la biblioteca: ***"CargarEstructura"*** , ***"Calcular"*** y ***"Dijkstra".***

Cargar Estructura

Descripción

Carga la estructura (red o ciudad) en memoria.
Se debe ejecutar una sola vez, luego de declarada la librería.
Su ejecución demora un tiempo considerable, de aproximadamente un minuto para 15.000 nodos.
La utilidad de esta función se limita, si posteriormente se ejecuta alguna llamada a la función "Dijkstra" o a la función "Calcular" sin caminos precalculados.

Declaración

```
Declare Function "CargarEstructura" Lib " t5vrp.dll" (ByVal
IniEstructura As String) As Long
```

Ejemplo de uso:

```
Call CargarEstructura("C:\tempVRP\theme2.ini")
```

Parámetros

IniEstructura : Recibe un String conteniendo la dirección y nombre del archivo .INI de la red. La especificación de dicho archivo .INI se encuentra en: Entrada para Cargar la Red.

Valor de Retorno

Devuelve un entero que representa un código de error. Dichos códigos de error se encuentran especificados en ConstantesError.h

Calcular

Descripción

Calcula los problemas de ruteo
Dependiendo del tipo de problema, es necesario la carga de la estructura previa

Declaración

```
Declare Function Calcular Lib "t5vrp.dll" (ByVal IniEntrada As
String, ByVal IniSalida As String) As Double
```

Ejemplo de uso:

```
Call Calcular("c:\tempVRP\entradaVRP.ini", "c:\tempVRP\
salidaVRP.ini")
```

Parámetros

IniEntrada : Recibe un String conteniendo la dirección y nombre del archivo INI de entrada. La especificación de dicho archivo .INI se encuentra en: Entrada de la biblioteca.

IniSalida: Recibe un String conteniendo la dirección y nombre del archivo INI de salida. La especificación de dicho archivo .INI se encuentra en: Salida de la biblioteca.

Valor de Retorno

Devuelve un entero que representa un código de error. Dichos códigos de error se encuentran especificados en ConstantesError.h

Dijkstra

Descripción

Ejecuta el Dijkstra
Se debe ejecutar posteriormente de la carga de la estructura

Declaración

```
Declare Function Dijkstra Lib "t5vrp.dll" (ByVal IniEntrada As
String, ByVal IniSalida As String) As Double
```

Ejemplo de uso:

```
Call Dijkstra("c:\tempVRP\entradaDijkstra.ini", "c:\tempVRP\
salidaDijkstra.ini")
```

Parámetros

IniEntrada :	Recibe un String conteniendo la dirección y nombre del archivo INI de entrada. La especificación de dicho archivo .INI se encuentra en: Interface para el Calculo del Dijkstra.
IniSalida:	Recibe un String conteniendo la dirección y nombre del archivo INI de salida. La especificación de dicho archivo .INI se encuentra en: Interface para el Calculo del Dijkstra.

Valor de Retorno

Devuelve un entero que representa un código de error. Dichos códigos de error se encuentran especificados en ConstantesError.h

Códigos de Error

Al ejecutar cualquier funcionalidad exportada por la biblioteca, la misma devuelve un código de error. Los códigos de error son un entero (Integer) que representa algún tipo de dificultad presentada durante la ejecución de alguna funcionalidad exportada por la biblioteca.

Se puede encontrar un detalle de los códigos de error en el archivo: ***ConstantesError.h***.

A continuación se presenta una breve reseña de los errores mas significativos:

Error Genérico

CE_ERROR_GENERICO = 1;
//Se debe a errores no considerados en particular

Error al leer un Sitio

// Un sitio es un cliente o un deposito
// Hay que sumar este valor en los casos que se detallan en las respectivas funciones
CE_INI_KEY_ID=1;
CE_INI_KEY_X_NODO_ANT=2;
CE_INI_KEY_Y_NODO_ANT=3;
CE_INI_KEY_X_NODO_POS=4;
CE_INI_KEY_Y_NODO_POS=5;
CE_INI_KEY_COSTO_ANT=6;
CE_INI_KEY_COSTO_POS=7;
CE_INI_KEY_TIEMPO_ANT=8;
CE_INI_KEY_TIEMPO_POS=9;

Errores correspondientes a CargarEstructura

```
// Errores correspondientes a levantar los datos de la red del .INI
        CE_RED_INI_KEY_NODOS=1002;
        //Se puede deber a:
        //              esta mal la seccion [RED],
        //              esta mal el key NODOS,
        //              no se encuentra el archivo .INI de la red,
        //              el archivo esta siendo usado de forma exclusiva por otra aplicacion

        CE_RED_INI_KEY_PATH_RED=1003;
        CE_RED_INI_KEY_ID=1004;
        CE_RED_INI_KEY_COSTO=1005;
        CE_RED_INI_KEY_VELOCIDAD=1006;
        CE_RED_INI_KEY_FLECHADA=1007;
        CE_RED_INI_KEY_X_NODO_ANT=1008;
        CE_RED_INI_KEY_Y_NODO_ANT=1009;
        CE_RED_INI_KEY_X_NODO_POS=1010;
        CE_RED_INI_KEY_Y_NODO_POS=1011;
        CE_RED_INI_KEY_CTE_RANGO=1012;
        CE_RED_INI_KEY_CTE_KG=1013;
        CE_RED_INI_KEY_COSTO_UNIDAD_TIEMPO=1014;
        CE_RED_INI_KEY_RADIO_MULTIPLE_DEPOT=1015;
        CE_RED_INI_KEY_PROBABILIDAD_ERROR_RUTA=1016;

// Errores correspondientes a levantar la red del DBF
        CE_NO_EXISTE_ARCHIVO_DBF = 1017;
        // Se debe a que no hay memoria RAM suficiente
        CE_MEMORIA_GRAFO = 1018;
        // Se debe a que no hay memoria RAM suficiente
        CE_CANT_ESQUINAS_GRAFO = 1019;
        // Se debe a que la cantidad de NODOS de la red esta mal
        CE_MEMORIA_ESQUINAS = 1020;
        // Se debe a que la cantidad de NODOS de la red esta mal
```

Errores correspondientes a calcular el Dikjstra

```
// Errores correspondientes a levantar los datos del Dijkstra del .INI
        CE_DIJKS_INI_KEY_NODOS = 2001;
        //Se puede deber a:
        //              esta mal la seccion [DIJKSTRA],
        //              esta mal el key NODOS,
        //              no se encuentra el archivo .INI de la red,
        //              el archivo esta siendo usado de forma exclusiva por otra aplicacion

        CE_DIJKS_INI_KEY_TIPO= 2002;
        // No es un tipo valido

        CE_DIJKS_INI_SECTION_NODO= 2003;
        CE_DIJKS_INI_KEY_CAMINO = 2004;
        // error al leer el camino de un destino

        CE_DIJKS_ORIGEN_ANT = 2005;
        //el nodo (x,y) anterior del origen NO EXISTE
```

```
CE_DIJKS_ORIGEN_POS = 2006;
//el nodo (x,y) posterior del origen NO EXISTE

CE_DIJKS_ORIGEN = 2100;
//cuando esta mal los datos del origen,
//hay que sumar este valor al de los errores al leer sitios

CE_DIJKS_DESTINO_ANT = 2007;
//el nodo (x,y) anterior de algún destino NO EXISTE

CE_DIJKS_DESTINO_POS = 2008;
//el nodo (x,y) posterior de algún destino NO EXISTE

CE_DIJKS_DESTINO = 2200;
//cuando esta mal los datos de algún destino,
//hay que sumar este valor al de los errores al leer sitios

CE_CANT_SITIOS_GRAFO=2300;
// Se debe a :
//                  la cantidad de NODOS de la red esta mal
//                  la cantidad de clientes esta mal
//                  la cantidad de depostos esta mal

// Errores correspondientes a guardar el Dijkstra en el .INI
CE_DIJKS_SAL_INI_KEY_CAMINOS=2401;
CE_DIJKS_SAL_INI_KEY_TIPO=2402;
CE_DIJKS_SAL_INI_KEY_COSTO=2403;
CE_DIJKS_SAL_INI_KEY_TIEMPO=2404;
CE_DIJKS_SAL_INI_KEY_ID_INICIAL=2405;
CE_DIJKS_SAL_INI_KEY_ID_FINAL=2406;
CE_DIJKS_SAL_INI_KEY_NODOS=2407;
CE_DIJKS_SAL_INI_KEY_X_NODO_Z=2408;
CE_DIJKS_SAL_INI_KEY_Y_NODO_Z=2409;
```

Errores correspondientes a los algoritmos de ruteo (función Calcular)

```
CE_ALGORITMO_NO_EXISTE= 3001;
// Se debe a que el algoritmo pedido en el TIPO de [PROBLEMA] no existe (o todavia no fue desarrollado)

CE_NO_HAY_UN_SOLO_DEPOSITO= 3002;
// El algoritmo solo acepta un deposito ([PROBLEMA] DEPOSITOS=1)

CE_NO_HAY_UN_SOLO_VEHICULO= 3003;
// El algoritmo solo acepta un vehiculo ([PROBLEMA] VEHICULOS=1)

CE_NO_HAY_NINGUN_VEHICULO= 3004;
// El algoritmo solo acepta ningun vehiculo ([PROBLEMA] VEHICULOS=0)

CE_NO_HAY_MAS_DE_UN_DEPOSITO= 3005;
// El algoritmo esta pensado para mas de un deposito([PROBLEMA] DEPOSITOS>1)

CE_NO_HAY_MAS_DE_UN_VEHICULO= 3006;
// El algoritmo esta pensado para mas de un vehículo([PROBLEMA] VEHICULOS>1)
```

```
 CE_VENTANA_INVALIDA_CLIENTE= 3100;
/*El algoritmo no acepta el tipo de ventana presentado.
Los ultimos dos digitos corresponden al identificador del cliente en cuestion*/

 CE_VENTANA_INVALIDA_DEPOSITO= 3200;
/*El algoritmo no acepta el tipo de ventana presentado.
Los ultimos dos digitos corresponden al identificador del deposito en cuestion*/

// Errores correspondientes a levantar los datos del problema del .INI
        CE_PROBLEMA_INI_KEY_TIPO=4001;
        //Se puede deber a:
        //              no se encuentra el archivo .INI de la red,
        //              el archivo esta siendo usado de forma exclusiva por otra aplicacion
        //              esta mal la seccion [PROBLEMA],
        //              esta mal el key TIPO
         CE_PROBLEMA_INI_KEY_SITIOS=4002;
        // Esta mal la cantidad de sitios del problema
         CE_PROBLEMA_INI_KEY_DEPOSITOS=4003;
        // Esta mal la cantidad de depositos del problema
         CE_PROBLEMA_INI_KEY_VEHICULOS=4004;
        // Esta mal la cantidad de vehiculos del problema
         CE_PROBLEMA_INI_KEY_RED_CPP=4005;
        // No se encuentra el key
         CE_PROBLEMA_INI_KEY_PARES_DARP=4006;
        // No se encuentra el key
         CE_PROBLEMA_INI_KEY_PATH_CAMINOS=4007;
        // No se encuentra el key.
        // Recordar que en el caso en que no se tengan caminos precalculados se debe pasar como
path_caminos el string vacio

// Errores correspondientes a levantar los datos de los clientes del .INI
// también son los errores correspondientes a levantar los arcos del CPP (por analogia)
        CE_CLIENTE=100000;
        // Los ultimos digitos corresponden al cliente donde ocurrio el error
         CE_CLIENTE_INI_KEY_ID=100100;
         CE_CLIENTE_INI_KEY_X_NODO_ANT=100200;
         CE_CLIENTE_INI_KEY_Y_NODO_ANT=100300;
         CE_CLIENTE_INI_KEY_X_NODO_POS=100400;
         CE_CLIENTE_INI_KEY_Y_NODO_POS=100500;
         CE_CLIENTE_INI_KEY_COSTO_ANT=100600;
         CE_CLIENTE_INI_KEY_COSTO_POS=100700;
         CE_CLIENTE_INI_KEY_TIEMPO_ANT=100800;
         CE_CLIENTE_INI_KEY_TIEMPO_POS=100900;
          CE_CLIENTE_INI_KEY_DEMANDA=110100;
          CE_CLIENTE_INI_KEY_VENTANA=110200;
          CE_CLIENTE_INI_KEY_TIEMPO_SERVICIO=110300;

// Errores correspondientes a levantar los datos de los depositos del .INI
         CE_DEPOSITO=200000;
        // La causa de estos errores es igual a la causa correspondiente en los clientes (errores
anteriores)
        // Los ultimos digitos corresponden al deposito donde ocurrio el error
         CE_DEPOSITO_INI_KEY_ID=200100;
         CE_DEPOSITO_INI_KEY_X_NODO_ANT=200200;
         CE_DEPOSITO_INI_KEY_Y_NODO_ANT=200300;
         CE_DEPOSITO_INI_KEY_X_NODO_POS=200400;
```

```
                CE_DEPOSITO_INI_KEY_Y_NODO_POS=200500;
                CE_DEPOSITO_INI_KEY_COSTO_ANT=200600;
                CE_DEPOSITO_INI_KEY_COSTO_POS=200700;
                CE_DEPOSITO_INI_KEY_TIEMPO_ANT=200800;
                CE_DEPOSITO_INI_KEY_TIEMPO_POS=200900;
                 CE_DEPOSITO_INI_KEY_CAPACIDAD=210100;
                 CE_DEPOSITO_INI_KEY_VENTANA=210200;

    // Errores correspondientes a levantar los datos de los vehiculos del .INI
            // Cada error de los siguietes corresponde a una mala lectura del key correspondiente
            // Los ultimos digitos corresponden al vehiculo donde ocurrio el error
             CE_VEHICULO_INI_KEY_CANTIDAD=300100;
             CE_VEHICULO_INI_KEY_ID=300200;
             CE_VEHICULO_INI_KEY_CAPACIDAD=300300;
             CE_VEHICULO_INI_KEY_COSTO_FIJO=300400;
             CE_VEHICULO_INI_KEY_COSTO_VARIABLE=300500;

    // Errores correspondientes a Levantar los caminos precalculados del .INI de caminos
            //Los ultimos dos digitos (unidades y decenas) corresponden al numero de camino que
genero el error
             CE_CAMINO_HACIA_CLIENTE_INI_KEY_COSTO= 400100;
             CE_CAMINO_HACIA_CLIENTE_INI_KEY_TIEMPO=400200;
             CE_CAMINO_HACIA_DEPOSITO_INI_KEY_COSTO=400300;
             CE_CAMINO_HACIA_DEPOSITO_INI_KEY_TIEMPO=400400;

    // Errores correspondientes al algoritmos de ruteo VRP
             CE_VRP_CAPACIDAD_VEHICULO_EXCEDIDA= 500100;
            //Los ultimos dos digitos corresponden al identificador del cliente que excede la capacidad del
vehiculo*/
             CE_VRP_CAPACIDAD_DEPOSITO_EXCEDIDA= 500200;
            //La suma de las demandas de los clientes debe ser menor a la capacidad del deposito

    // Errores correspondientes al algoritmos de ruteo MDVRP
            static const int CE_MDVRP_CAPACIDAD_DEPOSITO_EXCEDIDA = 600100;
            //La suma de las demandas de los clientes debe ser menor a la suma de las capacidades del
deposito

    // Errores correspondientes a Guardar el problema VRP, TSP,... en el .INI
            /*Se pospone la realizacion de estos errores.
            Por ahora se devuelve 1 cuando ocurre un error de este tipo*/
```

Ejemplos

A continuación se presentan archivos en formato .INI que son ejemplos representativos para los principales problemas resueltos por la biblioteca mediante la funcionalidades “CargarEstructura”, “Calcular” y “Dijkstra”.

CargarEstructura

```
[RED]
NODOS=1402
PATH_RED=F:\tempVRP\streets.shp
ID=Streets_
COSTO=Length
VELOCIDAD=Length
FLECHADA=Flechada
X_NODO_ANT=Fnode_
Y_NODO_ANT=Fnode_
X_NODO_POS=Tnode_
Y_NODO_POS=Tnode_
CTE_RANGO=0.1
CTE_KG=0
COSTO_UNIDAD_TIEMPO=1.11111
RADIO_MULTIPLE_DEPOT=0.11666666666
PROBABILIDAD_ERROR_RUTA=0.01
```

Dijkstra

```
[DIJKSTRA]
TIPO=1
ID=200000
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
NODOS=3

[NODO_1]
ID=200006
DIST_ANT=0.00059
DIST_POST=0.00078
DEM_ANT=0.00091
DEM_POST=0.00121
X_NODO_ANT=13026
Y_NODO_ANT=13026
X_NODO_POS=13027
```

```
Y_NODO_POS=13027
CAMINO=11

[NODO_2]
ID=200008
DIST_ANT=0.00058
DIST_POST=0.00082
DEM_ANT=0.00012
DEM_POST=0.00017
X_NODO_ANT=12386
Y_NODO_ANT=12386
X_NODO_POS=12248
Y_NODO_POS=12248
CAMINO=12

[NODO_3]
ID=200004
DIST_ANT=0.00214
DIST_POST=0.00103
DEM_ANT=0.00087
DEM_POST=0.00096
X_NODO_ANT=10994
Y_NODO_ANT=10994
X_NODO_POS=10932
Y_NODO_POS=10932
CAMINO=13
```

Calcular

TSP

```
[PROBLEMA]
TIPO=50
SITIOS=2
DEPOSITOS=1
VEHICULOS=0
RED=0
PARES=0
PATH_CAMINOS = f:\tempVRP\caminito.ini

[SITIO_1]
ID=200001
DEMANDA=-1
VENTANA=VENTANA_S_1
TIEMPO_SERVICIO=20
DIST_ANT=0.00068
DIST_POST=0.00084
DEM_ANT=0.00093
DEM_POST=0.00115
X_NODO_ANT=12079
Y_NODO_ANT=12079
X_NODO_POS=12080
Y_NODO_POS=12080
```

```
CAMINO_HACIA_S_2=1
CAMINO_HACIA_D_1=2

[SITIO_2]
ID=200002
DEMANDA=-1
VENTANA=VENTANA_S_2
TIEMPO_SERVICIO=5
DIST_ANT=0.00137
DIST_POST=0.00132
DEM_ANT=0.00172
DEM_POST=0.00167
X_NODO_ANT=13103
Y_NODO_ANT=13103
X_NODO_POS=13038
Y_NODO_POS=13038
CAMINO_HACIA_S_1=3
CAMINO_HACIA_D_1=4

[DEPOSITO_1]
ID=200000
CAPACIDAD=-1
VENTANA=VENTANA_D_1
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=5
CAMINO_HACIA_S_2=6

[TIPO_VEHICULO_1]
ID=55555
CANTIDAD=1
COSTO_FIJO=111
COSTO_VAR=222
CAPACIDAD=100
```

MDVRP

```
[PROBLEMA]
TIPO=10
SITIOS=2
DEPOSITOS=2
VEHICULOS=1
RED=0
PARES=0
PATH_CAMINOS = f:\tempVRP\caminitoMDVRP.ini

[SITIO_1]
ID=200001
DEMANDA=20
VENTANA=VENTANA_S_1
```

```
TIEMPO_SERVICIO=20
DIST_ANT=0.00068
DIST_POST=0.00084
DEM_ANT=0.00093
DEM_POST=0.00115
X_NODO_ANT=12079
Y_NODO_ANT=12079
X_NODO_POS=12080
Y_NODO_POS=12080
CAMINO_HACIA_S_2=1
CAMINO_HACIA_D_1=2
CAMINO_HACIA_D_2=7

[SITIO_2]
ID=200002
DEMANDA=50
VENTANA=VENTANA_S_2
TIEMPO_SERVICIO=5
DIST_ANT=0.00137
DIST_POST=0.00132
DEM_ANT=0.00172
DEM_POST=0.00167
X_NODO_ANT=13103
Y_NODO_ANT=13103
X_NODO_POS=13038
Y_NODO_POS=13038
CAMINO_HACIA_S_1=3
CAMINO_HACIA_D_1=4
CAMINO_HACIA_D_2=8

[DEPOSITO_1]
ID=200000
CAPACIDAD=100
VENTANA=VENTANA_D_1
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=5
CAMINO_HACIA_S_2=6
CAMINO_HACIA_D_2=9

[DEPOSITO_2]
ID=200003
CAPACIDAD=100
VENTANA=VENTANA_S_3
TIEMPO_SERVICIO=10
DIST_ANT=0.001456
DIST_POST=0.00028
DEM_ANT=0.0005
DEM_POST=0.00071
X_NODO_ANT=11689
```

```
Y_NODO_ANT=11689
X_NODO_POS=11549
Y_NODO_POS=11549
CAMINO_HACIA_S_1=10
CAMINO_HACIA_S_2=11
CAMINO_HACIA_D_1=12

[TIPO_VEHICULO_1]
ID=55555
CANTIDAD=1
COSTO_FIJO=111
COSTO_VAR=222
CAPACIDAD=100
```

VRPTW

```
[PROBLEMA]
TIPO=20
SITIOS=2
DEPOSITOS=1
VEHICULOS=1
RED=0
PARES=0
PATH_CAMINOS = f:\tempVRP\caminito.ini

[SITIO_1]
ID=200001
DEMANDA=20
VENTANA=1
TIEMPO_SERVICIO=20
DIST_ANT=0.00068
DIST_POST=0.00084
DEM_ANT=0.00093
DEM_POST=0.00115
X_NODO_ANT=12079
Y_NODO_ANT=12079
X_NODO_POS=12080
Y_NODO_POS=12080
CAMINO_HACIA_S_2=1
CAMINO_HACIA_D_1=2

[SITIO_2]
ID=200002
DEMANDA=50
VENTANA=2
TIEMPO_SERVICIO=5
DIST_ANT=0.00137
DIST_POST=0.00132
DEM_ANT=0.00172
DEM_POST=0.00167
X_NODO_ANT=13103
Y_NODO_ANT=13103
X_NODO_POS=13038
Y_NODO_POS=13038
CAMINO_HACIA_S_1=3
CAMINO_HACIA_D_1=4
```

```
[DEPOSITO_1]
ID=200000
CAPACIDAD=10
VENTANA=3
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=5
CAMINO_HACIA_S_2=6

[TIPO_VEHICULO_1]
ID=55555
CANTIDAD=1
COSTO_FIJO=111
COSTO_VAR=222
CAPACIDAD=100

[VENTANA_1]
CANTIDAD_PERIODOS=1
PERIODO_INICIO_1=480.01
PERIODO_FINAL_1=720.01
;480= 8 AM
;720= 12 AM

[VENTANA_2]
CANTIDAD_PERIODOS=2
PERIODO_INICIO_1=420.02
PERIODO_FINAL_1=720.02
PERIODO_INICIO_2=780.52
PERIODO_FINAL_2=1140.02
;1140= 19 PM

[VENTANA_3]
CANTIDAD_PERIODOS=3
PERIODO_INICIO_1=540.03
PERIODO_FINAL_1=660.03
PERIODO_INICIO_2=780.03
PERIODO_FINAL_2=960.03
PERIODO_INICIO_3=1080.03
PERIODO_FINAL_3=1200.03
;1080= 18 PM
;1200= 20 PM
```

MDVRPTW

```
[PROBLEMA]
TIPO=30
SITIOS=2
DEPOSITOS=2
VEHICULOS=1
```

```
RED=0
PARES=0
PATH_CAMINOS = f:\tempVRP\caminitoMDVRP.ini

[SITIO_1]
ID=200001
DEMANDA=20
VENTANA=1
TIEMPO_SERVICIO=20
DIST_ANT=0.00068
DIST_POST=0.00084
DEM_ANT=0.00093
DEM_POST=0.00115
X_NODO_ANT=12079
Y_NODO_ANT=12079
X_NODO_POS=12080
Y_NODO_POS=12080
CAMINO_HACIA_S_2=1
CAMINO_HACIA_D_1=2
CAMINO_HACIA_D_2=7

[SITIO_2]
ID=200002
DEMANDA=50
VENTANA=2
TIEMPO_SERVICIO=5
DIST_ANT=0.00137
DIST_POST=0.00132
DEM_ANT=0.00172
DEM_POST=0.00167
X_NODO_ANT=13103
Y_NODO_ANT=13103
X_NODO_POS=13038
Y_NODO_POS=13038
CAMINO_HACIA_S_1=3
CAMINO_HACIA_D_1=4
CAMINO_HACIA_D_2=8

[DEPOSITO_1]
ID=200000
CAPACIDAD=10
VENTANA=3
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=5
CAMINO_HACIA_S_2=6
CAMINO_HACIA_D_2=9

[DEPOSITO_2]
ID=200003
```

```
CAPACIDAD=100
VENTANA=3
DIST_ANT=0.001456
DIST_POST=0.00028
DEM_ANT=0.0005
DEM_POST=0.00071
X_NODO_ANT=11689
Y_NODO_ANT=11689
X_NODO_POS=11549
Y_NODO_POS=11549
CAMINO_HACIA_S_1=10
CAMINO_HACIA_S_2=11
CAMINO_HACIA_D_1=12

[TIPO_VEHICULO_1]
ID=55555
CANTIDAD=1
COSTO_FIJO=111
COSTO_VAR=222
CAPACIDAD=100

[VENTANA_1]
CANTIDAD_PERIODOS=1
PERIODO_INICIO_1=480.01
PERIODO_FINAL_1=720.01
;480= 8 AM
;720= 12 AM

[VENTANA_2]
CANTIDAD_PERIODOS=2
PERIODO_INICIO_1=420.02
PERIODO_FINAL_1=720.02
PERIODO_INICIO_2=780.52
PERIODO_FINAL_2=1140.02
;1140= 19 PM

[VENTANA_3]
CANTIDAD_PERIODOS=3
PERIODO_INICIO_1=540.03
PERIODO_FINAL_1=660.03
PERIODO_INICIO_2=780.03
PERIODO_FINAL_2=960.03
PERIODO_INICIO_3=1080.03
PERIODO_FINAL_3=1200.03
;1080= 18 PM
;1200= 20 PM

[VENTANA_4]
CANTIDAD_PERIODOS=1
PERIODO_INICIO_1=540.03
PERIODO_FINAL_1=660.03
```

MTSP

```
[PROBLEMA]
TIPO=80
```

```
SITIOS=5
DEPOSITOS=1
VEHICULOS=1
RED=0
PARES=0
PATH_CAMINOS = f:\tempVRP\caminito.ini

[SITIO_1]
ID=200001
DEMANDA=0
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0.00068
DIST_POST=0.00084
DEM_ANT=0.00093
DEM_POST=0.00115
X_NODO_ANT=12079
Y_NODO_ANT=12079
X_NODO_POS=12080
Y_NODO_POS=12080
CAMINO_HACIA_S_2=1
CAMINO_HACIA_S_3=2
CAMINO_HACIA_S_4=3
CAMINO_HACIA_S_5=4
CAMINO_HACIA_D_1=5

[SITIO_2]
ID=200002
DEMANDA=0
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0.00137
DIST_POST=0.00132
DEM_ANT=0.00172
DEM_POST=0.00167
X_NODO_ANT=13103
Y_NODO_ANT=13103
X_NODO_POS=13038
Y_NODO_POS=13038
CAMINO_HACIA_S_1=6
CAMINO_HACIA_S_3=7
CAMINO_HACIA_S_4=8
CAMINO_HACIA_S_5=9
CAMINO_HACIA_D_1=10

[SITIO_3]
ID=200003
DEMANDA=60
VENTANA=0
TIEMPO_SERVICIO=10
DIST_ANT=0.001456
DIST_POST=0.00028
DEM_ANT=0.0005
DEM_POST=0.00071
X_NODO_ANT=11689
Y_NODO_ANT=11689
```

```
X_NODO_POS=11549
Y_NODO_POS=11549
CAMINO_HACIA_S_1=11
CAMINO_HACIA_S_2=12
CAMINO_HACIA_S_4=13
CAMINO_HACIA_S_5=14
CAMINO_HACIA_D_1=15

[SITIO_4]
ID=200004
DEMANDA=5
VENTANA=0
TIEMPO_SERVICIO=2
DIST_ANT=0.00214
DIST_POST=0.00103
DEM_ANT=0.00087
DEM_POST=0.00096
X_NODO_ANT=10994
Y_NODO_ANT=10994
X_NODO_POS=10932
Y_NODO_POS=10932
CAMINO_HACIA_S_1=16
CAMINO_HACIA_S_2=17
CAMINO_HACIA_S_3=18
CAMINO_HACIA_S_5=19
CAMINO_HACIA_D_1=20

[SITIO_5]
ID=200005
DEMANDA=20
VENTANA=0
TIEMPO_SERVICIO=5
DIST_ANT=0.00112
DIST_POST=0.00116
DEM_ANT=0.0008
DEM_POST=0.00083
X_NODO_ANT=11793
Y_NODO_ANT=11793
X_NODO_POS=11705
Y_NODO_POS=11705
CAMINO_HACIA_S_1=21
CAMINO_HACIA_S_2=22
CAMINO_HACIA_S_3=23
CAMINO_HACIA_S_4=24
CAMINO_HACIA_D_1=25

[DEPOSITO_1]
ID=200000
CAPACIDAD=0
VENTANA=0
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
```

```
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=26
CAMINO_HACIA_S_2=27
CAMINO_HACIA_S_3=28
CAMINO_HACIA_S_4=29
CAMINO_HACIA_S_5=30

[TIPO_VEHICULO_1]
ID=55555
CANTIDAD=3
COSTO_FIJO=111
COSTO_VAR=222
CAPACIDAD=100
```

DARP

```
[PROBLEMA]
TIPO=60
SITIOS=4
DEPOSITOS=1
VEHICULOS=1
RED=0
PARES=2
PATH_CAMINOS = f:\tempVRP\caminito.ini

[SITIO_1]
ID=200001
DEMANDA=20
VENTANA=1
TIEMPO_SERVICIO=20
DIST_ANT=0.00068
DIST_POST=0.00084
DEM_ANT=0.00093
DEM_POST=0.00115
X_NODO_ANT=12079
Y_NODO_ANT=12079
X_NODO_POS=12080
Y_NODO_POS=12080
CAMINO_HACIA_S_2=1
CAMINO_HACIA_S_3=2
CAMINO_HACIA_S_4=3
CAMINO_HACIA_D_1=4

[SITIO_2]
ID=200002
DEMANDA=50
VENTANA=2
TIEMPO_SERVICIO=5
DIST_ANT=0.00137
DIST_POST=0.00132
DEM_ANT=0.00172
DEM_POST=0.00167
X_NODO_ANT=13103
Y_NODO_ANT=13103
X_NODO_POS=13038
```

```
Y_NODO_POS=13038
CAMINO_HACIA_S_1=5
CAMINO_HACIA_S_3=6
CAMINO_HACIA_S_4=7
CAMINO_HACIA_D_1=8

[SITIO_3]
ID=200003
DEMANDA=60
VENTANA=3
TIEMPO_SERVICIO=10
DIST_ANT=0.001456
DIST_POST=0.00028
DEM_ANT=0.0005
DEM_POST=0.00071
X_NODO_ANT=11689
Y_NODO_ANT=11689
X_NODO_POS=11549
Y_NODO_POS=11549
CAMINO_HACIA_S_1=9
CAMINO_HACIA_S_2=10
CAMINO_HACIA_S_4=11
CAMINO_HACIA_D_1=12

[SITIO_4]
ID=200004
DEMANDA=5
VENTANA=4
TIEMPO_SERVICIO=2
DIST_ANT=0.00214
DIST_POST=0.00103
DEM_ANT=0.00087
DEM_POST=0.00096
X_NODO_ANT=10994
Y_NODO_ANT=10994
X_NODO_POS=10932
Y_NODO_POS=10932
CAMINO_HACIA_S_1=13
CAMINO_HACIA_S_2=14
CAMINO_HACIA_S_3=15
CAMINO_HACIA_D_1=16

[DEPOSITO_1]
ID=200000
CAPACIDAD=10
VENTANA=5
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=17
CAMINO_HACIA_S_2=18
```

```
CAMINO_HACIA_S_3=19
CAMINO_HACIA_S_4=20

[TIPO_VEHICULO_1]
ID=55555
CANTIDAD=1
COSTO_FIJO=111
COSTO_VAR=222
CAPACIDAD=100

[VENTANA_1]
CANTIDAD_PERIODOS=1
PERIODO_INICIO_1=480.01
PERIODO_FINAL_1=720.01
;480= 8 AM
;720= 12 AM

[VENTANA_2]
CANTIDAD_PERIODOS=2
PERIODO_INICIO_1=420.02
PERIODO_FINAL_1=720.02
PERIODO_INICIO_2=780.52
PERIODO_FINAL_2=1140.02
;1140= 19 PM

[VENTANA_3]
CANTIDAD_PERIODOS=3
PERIODO_INICIO_1=540.03
PERIODO_FINAL_1=660.03
PERIODO_INICIO_2=780.03
PERIODO_FINAL_2=960.03
PERIODO_INICIO_3=1080.03
PERIODO_FINAL_3=1200.03
;1080= 18 PM
;1200= 20 PM

[VENTANA_4]
CANTIDAD_PERIODOS=1
PERIODO_INICIO_1=480.04
PERIODO_FINAL_1=720.04
;480= 8 AM
;720= 12 AM

[VENTANA_5]
CANTIDAD_PERIODOS=1
PERIODO_INICIO_1=480.05
PERIODO_FINAL_1=720.05
;480= 8 AM
;720= 12 AM

[PAR_1]
SITIO_CARGA=1
SITIO_ENTREGA=2

[PAR_2]
SITIO_CARGA=3
```

```
SITIO_ENTREGA=4
```

CPP

```
[PROBLEMA]
TIPO=70
SITIOS=0
DEPOSITOS=1
VEHICULOS=0
RED=2
PARES=0
PATH_CAMINOS = f:\tempVRP\caminito.ini

[DEPOSITO_1]
ID=200000
CAPACIDAD=0
VENTANA=VENTANA_D_1
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=1
CAMINO_HACIA_S_2=2

[ARCO_1]
ID=200001
X_NODO_ANT=25
Y_NODO_ANT=50
X_NODO_POS=1.5
Y_NODO_POS=1.2
COSTO=500
TIEMPO=500
CAMINO_HACIA_S_2=3
CAMINO_HACIA_D_1=4

[ARCO_2]
ID=200002
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=5
Y_NODO_POS=12
COSTO=50
TIEMPO=50
CAMINO_HACIA_S_1=5
CAMINO_HACIA_D_1=6
```

VRPHF

```
[PROBLEMA]
TIPO=40
SITIOS=9
```

```
DEPOSITOS=1
VEHICULOS=2
RED=0
PARES=0
PATH_CAMINOS=g:\tempVRP\fh11del03\DJ_umi.ini

[SITIO_1]
ID=1080
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=200
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_2=14
CAMINO_HACIA_S_3=32
CAMINO_HACIA_S_4=58
CAMINO_HACIA_S_5=5
CAMINO_HACIA_S_6=8
CAMINO_HACIA_S_7=74
CAMINO_HACIA_S_8=22
CAMINO_HACIA_S_9=44
CAMINO_HACIA_D_1=4

[SITIO_2]
ID=1082
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=400
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=16
CAMINO_HACIA_S_3=33
CAMINO_HACIA_S_4=59
CAMINO_HACIA_S_5=17
CAMINO_HACIA_S_6=18
CAMINO_HACIA_S_7=75
CAMINO_HACIA_S_8=23
CAMINO_HACIA_S_9=45
CAMINO_HACIA_D_1=15

[SITIO_3]
ID=1084
X_NODO_ANT=0
Y_NODO_ANT=0
```

```
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=600
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=35
CAMINO_HACIA_S_2=36
CAMINO_HACIA_S_4=60
CAMINO_HACIA_S_5=37
CAMINO_HACIA_S_6=38
CAMINO_HACIA_S_7=76
CAMINO_HACIA_S_8=39
CAMINO_HACIA_S_9=46
CAMINO_HACIA_D_1=34

[SITIO_4]
ID=1086
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=800
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=62
CAMINO_HACIA_S_2=63
CAMINO_HACIA_S_3=64
CAMINO_HACIA_S_5=65
CAMINO_HACIA_S_6=66
CAMINO_HACIA_S_7=77
CAMINO_HACIA_S_8=67
CAMINO_HACIA_S_9=68
CAMINO_HACIA_D_1=61

[SITIO_5]
ID=1079
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=100
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=6
```

```
CAMINO_HACIA_S_2=19
CAMINO_HACIA_S_3=40
CAMINO_HACIA_S_4=69
CAMINO_HACIA_S_6=9
CAMINO_HACIA_S_7=78
CAMINO_HACIA_S_8=24
CAMINO_HACIA_S_9=47
CAMINO_HACIA_D_1=2

[SITIO_6]
ID=1081
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=300
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=11
CAMINO_HACIA_S_2=20
CAMINO_HACIA_S_3=41
CAMINO_HACIA_S_4=70
CAMINO_HACIA_S_5=12
CAMINO_HACIA_S_7=79
CAMINO_HACIA_S_8=25
CAMINO_HACIA_S_9=48
CAMINO_HACIA_D_1=10

[SITIO_7]
ID=1089
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=1000
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=81
CAMINO_HACIA_S_2=82
CAMINO_HACIA_S_3=83
CAMINO_HACIA_S_4=84
CAMINO_HACIA_S_5=85
CAMINO_HACIA_S_6=86
CAMINO_HACIA_S_8=87
CAMINO_HACIA_S_9=88
CAMINO_HACIA_D_1=80

[SITIO_8]
```

```
ID=1083
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=500
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=27
CAMINO_HACIA_S_2=28
CAMINO_HACIA_S_3=42
CAMINO_HACIA_S_4=71
CAMINO_HACIA_S_5=29
CAMINO_HACIA_S_6=30
CAMINO_HACIA_S_7=89
CAMINO_HACIA_S_9=49
CAMINO_HACIA_D_1=26

[SITIO_9]
ID=1085
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
DEMANDA=700
VENTANA=0
TIEMPO_SERVICIO=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
DEM_POST=0
CAMINO_HACIA_S_1=51
CAMINO_HACIA_S_2=52
CAMINO_HACIA_S_3=53
CAMINO_HACIA_S_4=72
CAMINO_HACIA_S_5=54
CAMINO_HACIA_S_6=55
CAMINO_HACIA_S_7=90
CAMINO_HACIA_S_8=56
CAMINO_HACIA_D_1=50

[DEPOSITO_1]
ID=1078
X_NODO_ANT=0
Y_NODO_ANT=0
X_NODO_POS=0
Y_NODO_POS=0
CAPACIDAD=100000
VENTANA=0
DIST_ANT=0
DIST_POST=0
DEM_ANT=0
```

```
DEM_POST=0
CAMINO_HACIA_S_1=3
CAMINO_HACIA_S_2=13
CAMINO_HACIA_S_3=31
CAMINO_HACIA_S_4=57
CAMINO_HACIA_S_5=1
CAMINO_HACIA_S_6=7
CAMINO_HACIA_S_7=73
CAMINO_HACIA_S_8=21
CAMINO_HACIA_S_9=43

[TIPO_VEHICULO_1]
CANTIDAD=3
ID=841
ID_VEHICULO_1_1=1
ID_VEHICULO_1_2=2
ID_VEHICULO_1_3=3
ID_VEHICULO_1_4=4
COSTO_FIJO=1
COSTO_VAR=1
CAPACIDAD=600

[TIPO_VEHICULO_2]
CANTIDAD=3
ID=842
ID_VEHICULO_2_1=1
ID_VEHICULO_2_2=2
ID_VEHICULO_2_3=3
ID_VEHICULO_2_4=4
COSTO_FIJO=20
COSTO_VAR=20
CAPACIDAD=1000
```

SVRP

```
[PROBLEMA]
TIPO=90
SITIOS=2
DEPOSITOS=1
VEHICULOS=1
RED=0
PARES=0
PATH_CAMINOS = f:\tempVRP\caminito.ini

[SITIO_1]
ID=200001
DEMANDA=20
VENTANA=0
TIEMPO_SERVICIO=20
DIST_ANT=0.00068
DIST_POST=0.00084
DEM_ANT=0.00093
DEM_POST=0.00115
X_NODO_ANT=12079
Y_NODO_ANT=12079
X_NODO_POS=12080
```

```
Y_NODO_POS=12080
CAMINO_HACIA_S_2=1
CAMINO_HACIA_D_1=2

[SITIO_2]
ID=200002
DEMANDA=50
VENTANA=0
TIEMPO_SERVICIO=5
DIST_ANT=0.00137
DIST_POST=0.00132
DEM_ANT=0.00172
DEM_POST=0.00167
X_NODO_ANT=13103
Y_NODO_ANT=13103
X_NODO_POS=13038
Y_NODO_POS=13038
CAMINO_HACIA_S_1=3
CAMINO_HACIA_D_1=4

[DEPOSITO_1]
ID=200000
CAPACIDAD=10
VENTANA=0
DIST_ANT=0.00072
DIST_POST=0.00079
DEM_ANT=0.00018
DEM_POST=0.0002
X_NODO_ANT=12139
Y_NODO_ANT=12139
X_NODO_POS=12161
Y_NODO_POS=12161
CAMINO_HACIA_S_1=5
CAMINO_HACIA_S_2=6

[TIPO_VEHICULO_1]
ID=55555
CANTIDAD=1
COSTO_FIJO=111
COSTO_VAR=222
CAPACIDAD=100

[VA_DEMANDA]
TIPO=0
MEDIA=30
VARIANZA=1170
, se supuso U[0,60]
```

Caminito.ini

Este archivo es utilizado por los ejemplos de “Calcular “ antes mensionados.

```
[SOLUCION]
CAMINOS=30
```

```
[CAMINO_1]
COSTO=7.138000354e-002
TIEMPO=7.138000354e-002
[CAMINO_2]
COSTO=8.47e-003
TIEMPO=8.47e-003
[CAMINO_3]
COSTO=2.2e-002
TIEMPO=2.2e-002
[CAMINO_4]
COSTO=7.096000057e-002
TIEMPO=7.096000057e-002
[CAMINO_5]
COSTO=7.717000082e-002
TIEMPO=7.717000082e-002
[CAMINO_6]
COSTO=1.511e-002
TIEMPO=1.511e-002
[CAMINO_1]
COSTO=7.138000354e-002
TIEMPO=7.138000354e-002
[CAMINO_7]
COSTO=8.47e-003
TIEMPO=8.47e-003
[CAMINO_8]
COSTO=2.2e-002
TIEMPO=2.2e-002
[CAMINO_9]
COSTO=7.096000057e-002
TIEMPO=7.096000057e-002
[CAMINO_10]
COSTO=7.717000082e-002
TIEMPO=7.717000082e-002
[CAMINO_11]
COSTO=1.511e-002
TIEMPO=1.511e-002
[CAMINO_12]
COSTO=7.138000354e-002
TIEMPO=7.138000354e-002
[CAMINO_13]
COSTO=8.47e-003
TIEMPO=8.47e-003
[CAMINO_14]
COSTO=2.2e-002
TIEMPO=2.2e-002
[CAMINO_15]
COSTO=7.096000057e-002
TIEMPO=7.096000057e-002
[CAMINO_16]
COSTO=7.717000082e-002
TIEMPO=7.717000082e-002
[CAMINO_17]
COSTO=1.511e-002
TIEMPO=1.511e-002
[CAMINO_18]
COSTO=7.138000354e-002
```

```
TIEMPO=7.138000354e-002
[CAMINO_19]
COSTO=8.47e-003
TIEMPO=8.47e-003
[CAMINO_20]
COSTO=2.2e-002
TIEMPO=2.2e-002
[CAMINO_21]
COSTO=1.511e-002
TIEMPO=1.511e-002
[CAMINO_22]
COSTO=7.138000354e-002
TIEMPO=7.138000354e-002
[CAMINO_23]
COSTO=8.47e-003
TIEMPO=8.47e-003
[CAMINO_24]
COSTO=2.2e-002
TIEMPO=2.2e-002
[CAMINO_25]
COSTO=7.096000057e-002
TIEMPO=7.096000057e-002
[CAMINO_26]
COSTO=7.717000082e-002
TIEMPO=7.717000082e-002
[CAMINO_27]
COSTO=1.511e-002
TIEMPO=1.511e-002
[CAMINO_28]
COSTO=8.47e-003
TIEMPO=8.47e-003
[CAMINO_29]
COSTO=2.2e-002
TIEMPO=2.2e-002
[CAMINO_30]
COSTO=7.096000057e-002
TIEMPO=7.096000057e-002
```

Entrada para Cargar de la Red

Especificación

RED:

[RED]	Definición de la red
PATH_RED=	Dirección y nombre del archivo .DBF asociado.
NODOS=	Cantidad de nodos (esquinas) de la red. (String)
ID=	Nombre del campo del ID de cada arista del .DBF. (este Id debe ser un entero) (String)
COSTO=	Nombre del campo del costo de cada arista del .DBF. (String)
VELOCIDAD=	Nombre del campo de la velocidad de cada arista del .DBF.(String)
FLECHADA=	Nombre del campo del sentido de cada arista del .DBF. (String)
X_NODO_ANT=	Nombre del campo de la coordenada x del nodo origen de cada arista del .DBF. (String)
Y_NODO_ANT=	Nombre del campo de la coordenada y del nodo origen de cada arista del .DBF. (String)
X_NODO_POS=	Nombre del campo de la coordenada x del nodo destino de cada arista del .DBF. (String)
Y_NODO_POS=	Nombre del campo de la coordenada y del nodo destino de cada arista del .DBF. (String)
CTE_RANGO=	Es el rango dentro del cual una coordenada se considera como única. (float)
CTE_KG=	Es el factor de ponderación del peso de los giros para el calculo del Dijkstra. (float)
COSTO_UNIDAD_TIEMPO=	Para los algoritmos que utilizan el tiempo, es necesario saber el costo que tiene una unidad de tiempo. (float)
RADIO_MULTIPLE_DEPOT=	Para los algoritmos con multiple deposito es necesario definir el radio de decicion. (float)
PROBABILIDAD_ERROR_RUTA=	Para el problema SVRP, representa la probabilidad de que algun vehiculo no pueda satisfacer la demanda de los clientes de su ruta.

Interface para el Cálculo del Dijkstra

Entrada

La entrada de la función “Dijkstra” se encuentra en un archivo de formato .INI.

Especificación

Dijkstra:

[DIJKSTRA]	Definición del problema
TIPO=	Tipo de solución. 0-Dijkstra con ponderación de giro(Entero) 1-Dijkstra común 2-Dijkstra común, termina cuando se alcanzan todos los destinos(Entero)
ID=	Identificador externo del origen (partida del dijkstra) (Entero)
X_NODO_ANT=	Coordenada X del nodo anterior al origen (Float)
Y_NODO_ANT=	Coordenada Y del nodo anterior al origen (Float)
X_NODO_POS=	Coordenada X del nodo posterior al origen (Float)
Y_NODO_POS=	Coordenada Y del nodo posterior al origen (Float)
DIST_ANT=	Distancia al nodo anterior del origen (Float)
DIST_POST=	Distancia al nodo posterior del origen(Float)
DEM_ANT=	Demora en tiempo al nodo anterior del origen (Float)
DEM_POST=	Demora en tiempo al nodo posterior del origen (Float)
NODOS=	Cantidad de nodos, hacia donde se quiere encontrar un camino (Entero)

Nodo:

[NODO_Z]	Definición del nodo Z
ID=	Identificador externo del destino (llegada del dijkstra) (Entero)
X_NODO_ANT=	Coordenada X del nodo anterior al destino (Float)
Y_NODO_ANT=	Coordenada Y del nodo anterior al destino (Float)
X_NODO_POS=	Coordenada X del nodo posterior al destino (Float)
Y_NODO_POS=	Coordenada Y del nodo posterior al destino (Float)

DIST_ANT=	Distancia al nodo anterior del destino (Float)
DIST_POST=	Distancia al nodo posterior del destino (Float)
DEM_ANT=	Demora en tiempo al nodo anterior del destino (Float)
DEM_POST=	Demora en tiempo al nodo posterior del destino (Float)
CAMINO=W	Sección ([CAMINO_W]) donde se guarda el camino a este nodo, en el archivo de salida (Entero)

Salida

La salida de la función “Dijkstra” se encuentra en un archivo de formato .INI.

Especificación

Solución:	
[SOLUCION]	Datos de la solución
CAMINOS=	Cantidad de caminos. Para el caso en que los caminos no son consecutivos, esta variable representa el camino de identificador mayor. (Entero).

Camino:	
[CAMINO_W]	Definición del camino numero W
TIPO=	Determina el tipo de camino 0-Dijkstra (optimo) 1-Heurística(suboptimo) (Entero)
COSTO=	Costo del camino (Float)
TIEMPO=	Tiempo del camino (Float)
ID_INICIAL=	El id del origen (Entero)
ID_FINAL=	El id del destino(Entero)
NODOS=	Cantidad de nodos que forman el camino (Entero)
X_NODO_Z=	Coordenada X del nodo numero Z del (float)
Y_NODO_Z=	Coordenada Y del nodo numero Z del (float)