ALTERNATIVA ALGORÍTMICA PARA LA GENERACIÓN DE FIXTURES DEL FPC



**Julio Mario Daza-Escorcia**

**DESCRIPCIÓN DE LAS ETAPAS PARA EL DESARROLLO DEL SOFTWARE**

**DEL FIXTURE DEL FPC**

A continuación, se presenta la descripción de las etapas que requiere el desarrollo de un software como son: el análisis, el diseño, la implementación y la validación.

1. **ANÁLISIS**

El siguiente software, Fixture del FPC usando cuadrados latinos, se enfoca en la optimización de la generación del fixture del FPC, el cual pretende reducir los costos de la organización de los torneos en temas de viaje y transporte de los jugadores. Estos sistemas están teniendo un auge considerable en los últimos años, debido a que ofrecen una alternativa a la organización de los encuentros optimizando de esta forma, los recursos de la Federación y ayuda a reducir el agotamiento mental que sufren los jugadores.

En esta aplicación, se pretende dar respuesta a los problemas crear fixtures aleatorios, con el modelo actual que usa la Federación Colombiana de Futbol. Una vez integrados, se desarrollan algoritmos para optimizar los encuentros de los equipos en el fixture para reducir costos de transporte y estrés en sus jugadores.

Ante el dinamismo actual, la misión de lograr nuevas ventajas competitivas y/o fortalecer las existentes, apunta a resolver práctica, ágil y satisfactoriamente muchos de los problemas organizacionales. Dentro de las empresas, la logística es una fuente inagotable de opciones estratégicas, por la enorme oportunidad de reducir costos (y con esto, su posicionamiento en la mente del cliente) y aprovechar de mejor forma los medios a disposición. El siguiente árbol de causas y efectos establece estas oportunidades en la gestión del transporte, que es un área identificada como crucial para la optimización de sistemas logísticos, y el problema de enrutamiento que se tiene actualmente en la creación de los fixtures.

Deficiencias en la gestión del transporte (específicamente en el enrutamiento de vehículos)

Bajo nivel de servicio de la compañía

Alto costo de la gestión de transporte

Poco aprovechamiento de los recursos

Poca competitividad de la organización y de su sistema logístico

Metodologías ineficientes e inapropiadas

Políticas de gerencia inconsistentes

Decisiones basadas en información no confiable

Desconocimiento de alternativas de solución

Desconocimiento de las necesidades del área

Sistema de información inadecuado

Problema central

Figura 1. Árbol de causas y efectos del problema general.

Enrutar una flota de vehículos con capacidad restringida heterogénea, para mover entre cada estación y el depósito la cantidad de bicicletas o carros para restaurar balancear el sistema, es una variante práctica perteneciente a la familia de problemas de enrutamiento de vehículos. Esta variante del problema estándar involucra una compañía cuya flota de vehículos debe abastecer completamente a un conjunto conocido de estaciones con distinta ubicación geográfica y demandas de bicicletas o carros. La flota de vehículos, con la totalidad de la carga a distribuir, parte de una bodega o almacén central y balancea el sistema, antes de regresar eventualmente al almacén. El objetivo del problema es hallar rutas para los vehículos de tal modo que las estaciones puedan balancearse al mínimo costo (en general, expresado en términos de la distancia total recorrida o el tiempo trascurrido), sin violar las restricciones de capacidad de los vehículos.

1. **DISEÑO**

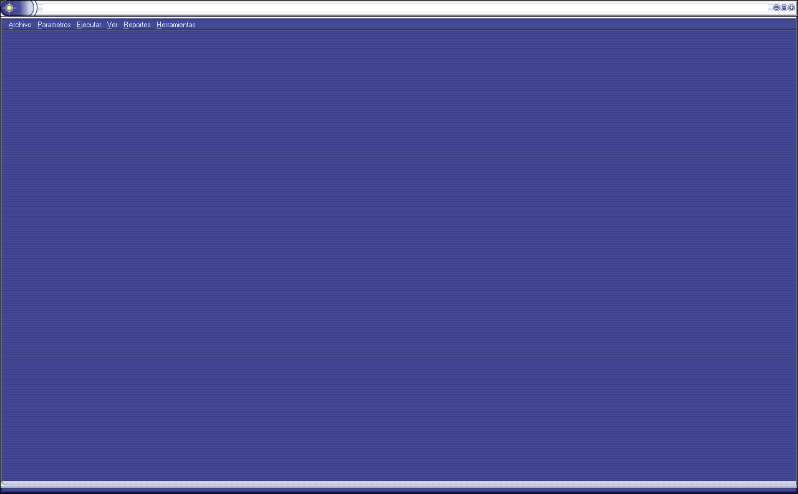
RV-SmartCities es un software, diseñado para el tratamiento de problemas de ruteo y planificación de vehículos con restricciones de capacidad, uní producto y flota homogénea. El software presenta, una interface grafica sencilla, didáctica, adaptable y robusta. El presente documento sirve de material de consulta para un eventual proceso de aprendizaje de la utilización del paquete informático. A continuación, se realiza una explicación del software y los procedimientos necesarios para su correcta utilización:

**2.1. Ejecución de la aplicación**

Al ejecutarse **Inicio/Programa/RV-SmartCities v1.0**, aparecerá en pantalla, la ventana de apertura de RV-SmartCities. Presione **Click** para ingresar a la aplicación principal.



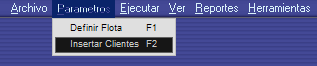
Al presionar “Click”, la ventana de apertura se cerrará e inmediatamente se abrirá la ventana principal de la aplicación, esta ventana puede ser apreciada en la siguiente figura:



En la parte superior de la ventana, es posible apreciar el menú principal, en este se realizan las tareas relacionadas como guardar y cargar los clientes a rutear.

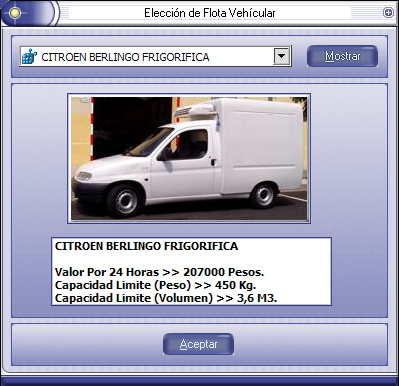


De la misma manera es posible acceder a las secciones de definición de la flota vehicular y a la de presentación de los reportes. Para abrir tales secciones o módulos, se debe presionar el puntero del ratón sobre la opción seleccionada.

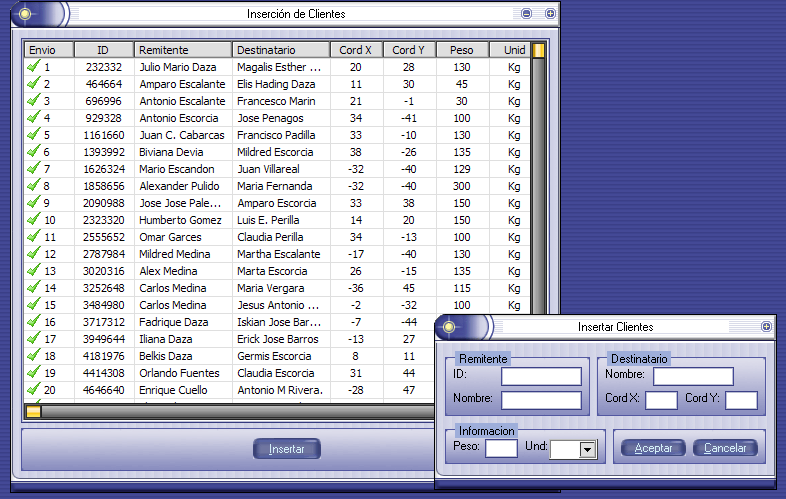


**2.2. Módulos de inicialización del sistema**

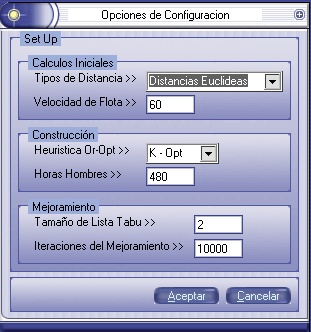
El primer módulo disponible, es la definición de la flota, para acceder a este, se debe seguir la siguiente ruta **Parámetros/Definir Flota.** En este módulo, se le suministra al software la información del tipo de la flota vehicular a utilizar (de carga general o frigorífica), su capacidad límite y su costo.



El segundo módulo disponible, es la definición de clientes, para acceder a este, se debe seguir la siguiente ruta **Parámetros/Insertar Clientes**. En este módulo, se le suministra al software la información de cantidad de clientes a rutear, sus direcciones y la carga a transportar.

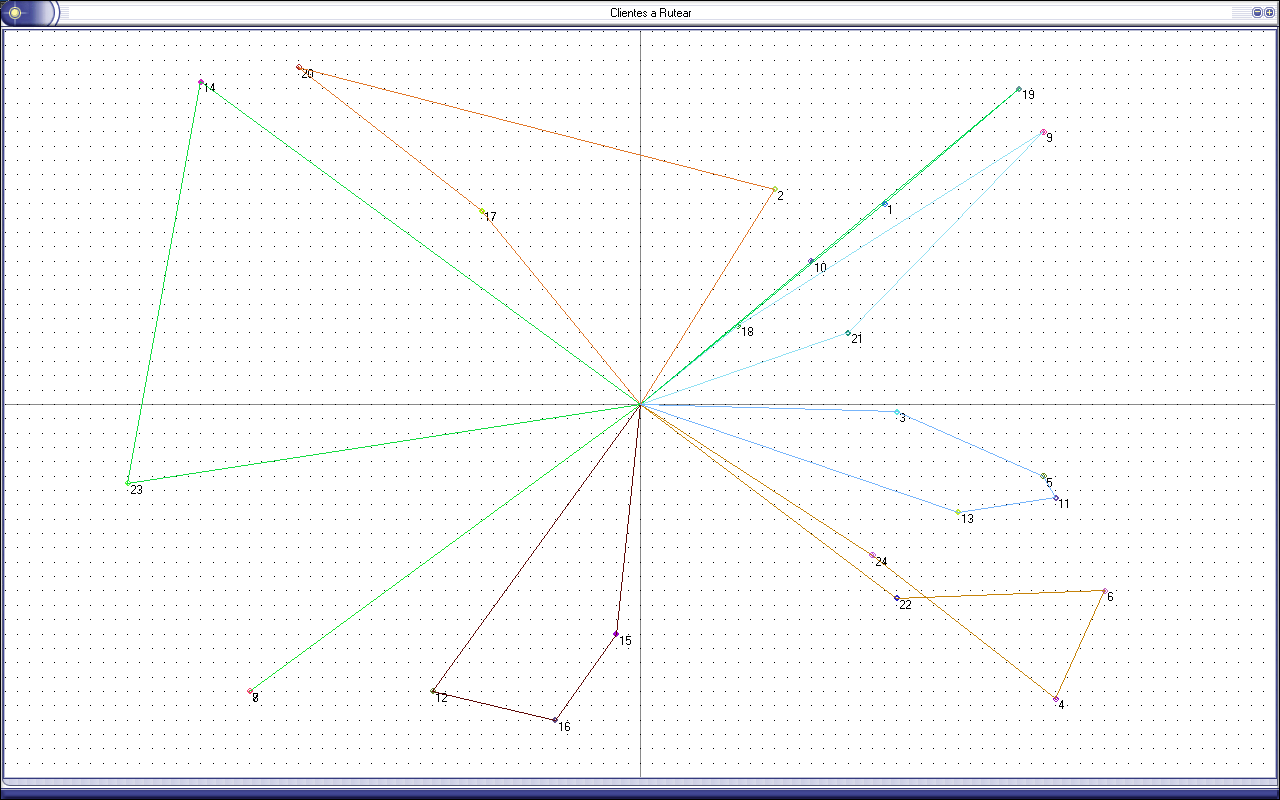
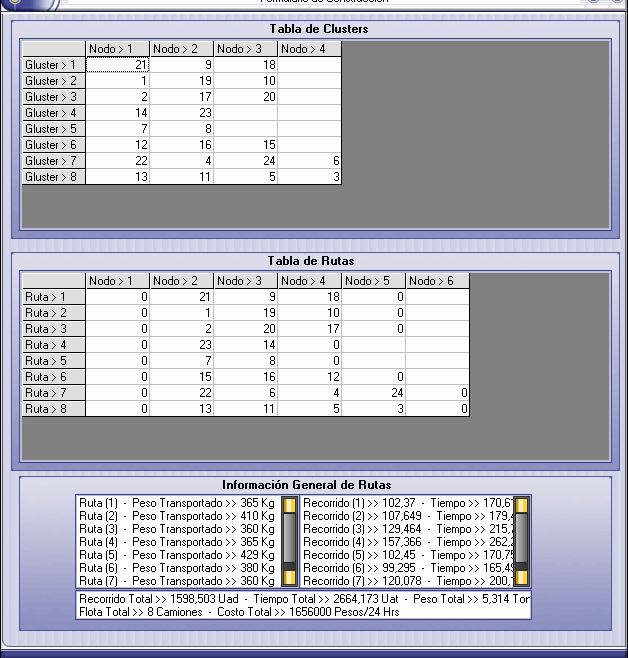


El tercer módulo disponible, es la configuración, para acceder a este, se debe seguir la siguiente ruta **Herramientas/Opciones de Configuración**. En este módulo, se le suministra al software la información de tipo de Heurística Or-Opt a utilizar, las HH que componen el jornal, la velocidad media de la flota, el tamaño de la lista tabú, y el número de iteraciones para el mejoramiento.

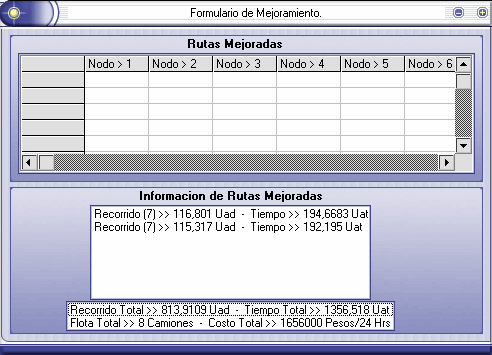


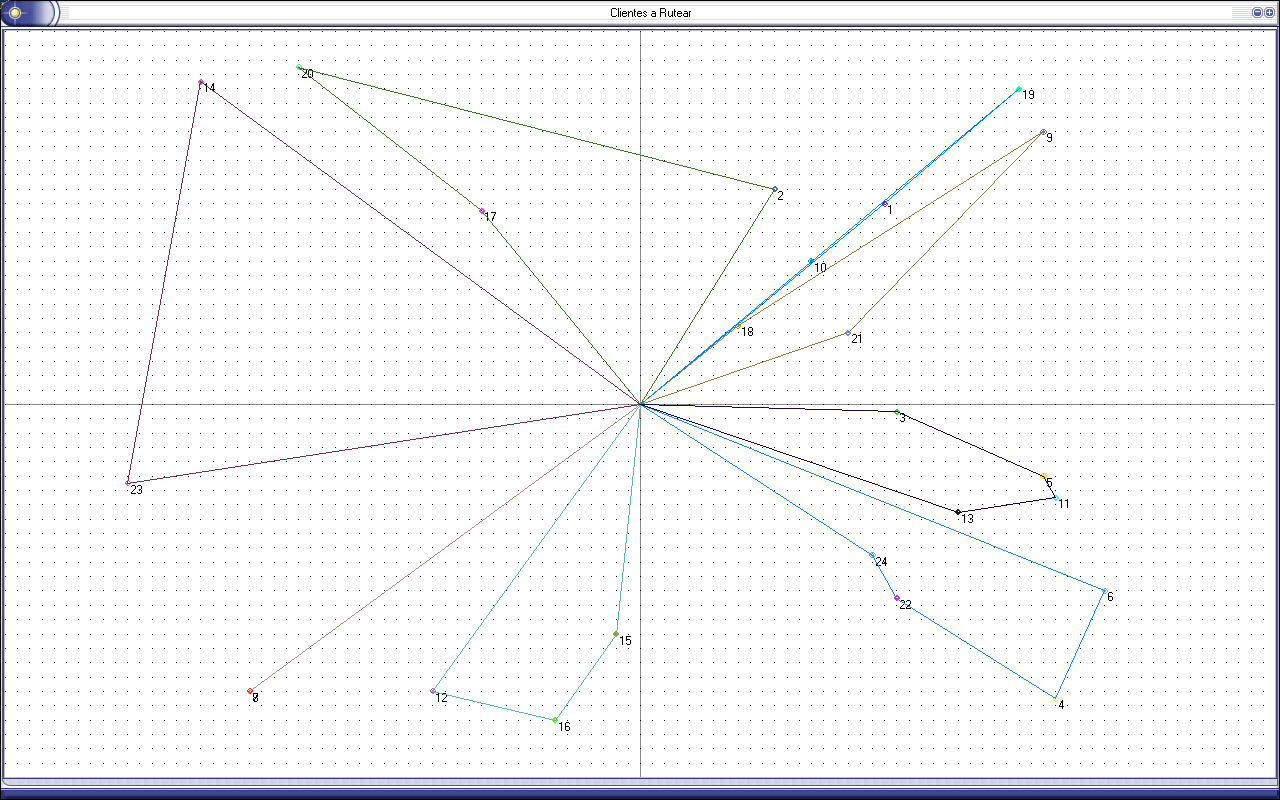
**2.3. Funcionamiento de RV-SmartCities**

Luego de completar los módulos anteriores se procede a ejecutar el ruteo y la posterior planificación, para esto seguimos la ruta, **Ejecutar/Ruteo/Construcción**, aquí el software muestra la construcción de las rutas con sus respectivos tiempos, distancia, capacidad y costo, de manera simultánea puede observar el resultado de la construcción gráficamente, para esto seguimos la ruta **Ver/Mapa**.



Posterior a esto se procede a ejecutar el mejoramiento de las rutas construidas, para esto seguimos la ruta, **Ejecutar/Ruteo/Mejoramiento**, aquí el software muestra el mejoramiento de las rutas previamente construidas con sus respectivos tiempo, distancia, capacidad y costo, de manera simultánea puede observar el resultado del mejoramiento gráficamente, para esto seguimos la ruta **Ver/Mapa**.

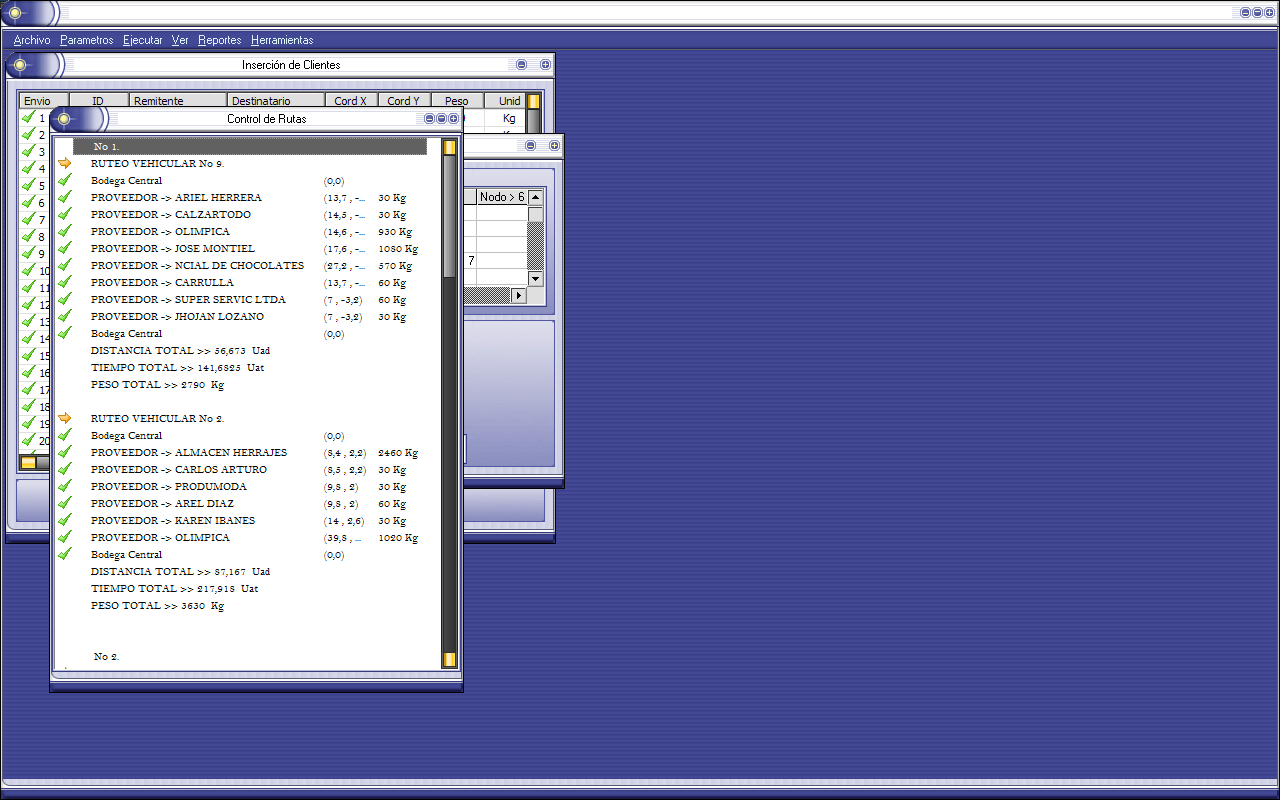




A continuación se procede a ejecutar la planificación de las rutas construidas y mejoradas en el ruteo, para esto seguimos la ruta, **Ejecutar/Planificación/Planificar**, aquí el software muestra gráficamente mediante un diagrama de Gantt los conjuntos de rutas que pueden abastecerse en un tiempo determinado para maximizar así la utilización de la capacidad instalada.



Por último, se puede observar esta planificación a manera de reporte para mayor aclaración de los conjuntos de rutas a abastecer, para esto seguimos la ruta, **Ver/Reporte/Control de Rutas**.



**3. IMPLEMENTACIÓN DEL SOFTWARE RV-SMARTCITIES**

La implementación del diseño se realizó a través de programación orientada a eventos, utilizando el lenguaje de programación Microsoft Visual Basic 6.0, haciendo iteraciones conforme a la metodología adoptada, permitiendo que se desarrollara el prototipo de manera incremental e iterativa.

A continuación, se muestra el código fuente para alternativa algorítmica para problemas de enrutamiento de vehículos en smart cities.

Option Explicit

Rem Vectores Tipo String

Public Id() As String

Public Remite() As String

Public Destina() As String

Public VectCargo() As Variant

Public VectPerse() As Variant

Public Modelo As String

Rem Vectores Tipo Single

Public dirx() As Single

Public diry() As Single

Public PesoKg() As Single

Public Radio() As Single

Public RadioAux() As Single

Public MtzDist() As Single

Public MtzTime() As Single

Public DistRutaTotal() As Single

Public TimeRutaTotal() As Single

Public DistTotalRuteo As Single

Public TimeTotalRuteo As Single

Public MtzPenalDist() As Single

Rem Vectores Tipo Integer

Public Nodos() As Integer

Public NodosAux() As Integer

Public NodosNew() As Integer

Public PesoCargaAux() As Integer

Public ClientesAsingAux() As Integer

Public ClientesAsignados() As Integer

Public PesoDeLaCarga() As Integer

Public TruckAsignado() As Integer

Public MejorSec() As Integer

Public SecuenciasAux() As Integer

Public SecuenciasLocales() As Integer

Rem Vectores Tipo Double

Public CordPolar() As Double

Public CordPolarAux() As Double

'Declaracion de Constantes

Rem Constantes Tipo Integer

Public Const Infi As Integer = 10000

'Declaracion de Variables

Rem Variables Tipo Integer

Public TamVect As Integer

Public PesoTotal As Variant ' se dana en integer

Public TamDeFlota As Integer

Public TypeDist As Integer

Public TypeOpt As Integer

Public VelTruck As Double

Public TListTabu As Integer

Public Max As Integer

Public Opt As Integer

Public HH As Integer

Public Gantt() As Integer

Rem Variables Tipo Single

Public CapacityTruck As Single

Public MaxWC As Single

Public MaxC As Single

Public MaxCx As Single

Public MaxCy As Single

Public TimeXRuta() As Single

Public Times() As Single

Public PosTimes() As Single

Public Tciclo As Single

Public Ruta As Integer

Public TiempoTotal As Single

Rem Variables Tipo Variant

Public Lvc As Variant

Rem Variables Tipo Long

Public Iteraciones As Long

Public CosteTruck As Long

Public Sub DistEuclidianas()

Dim i As Integer, j As Integer

ReDim MtzDist(0 To TamVect, 0 To TamVect + 1)

ReDim MtzTime(0 To TamVect, 0 To TamVect + 1)

ReDim MtzPenalDist(0 To TamVect, 0 To TamVect + 1)

For i = 0 To TamVect

For j = 0 To TamVect

If (i <> j) Then

MtzDist(i, j) = Round(Sqr(((dirx(j) - dirx(i)) ^ 2) + ((diry(j) - diry(i)) ^ 2)), 3)

MtzTime(i, j) = Round((MtzDist(i, j) / VelTruck), 3) 'UNIDADES ADIMENCIONALES DE TIEMPO

MtzPenalDist(i, j) = MtzDist(i, j) 'PARA PENALIZACIONES TABU

Else

MtzDist(i, j) = Infi

MtzTime(i, j) = Infi

MtzPenalDist(i, j) = Infi 'PARA PENALIZACIONES TABU

End If

Next j: ReDim Preserve Radio(i + 1): Radio(i + 1) = MtzDist(0, i + 1)

Next i

End Sub

Public Sub DistManhatan()

Dim i As Integer, j As Integer

ReDim MtzDist(0 To TamVect, 0 To TamVect + 1)

ReDim MtzTime(0 To TamVect, 0 To TamVect + 1)

ReDim MtzPenalDist(0 To TamVect, 0 To TamVect + 1)

If (VelTruck = 0) Then VelTruck = (60 / 100)

For i = 0 To TamVect

For j = 0 To TamVect

If (i <> j) Then

MtzDist(i, j) = Round((Abs(dirx(j) - dirx(i))) + (Abs(diry(j) - diry(i))), 3)

MtzTime(i, j) = Round((MtzDist(i, j) / VelTruck), 3) 'UNIDADES ADIMENCIONALES DE TIEMPO

MtzPenalDist(i, j) = MtzDist(i, j) 'PARA PENALIZACIONES TABU

Else

MtzDist(i, j) = Infi

MtzTime(i, j) = Infi

MtzPenalDist(i, j) = Infi 'PARA PENALIZACIONES TABU

End If

Next j: ReDim Preserve Radio(i + 1): Radio(i + 1) = MtzDist(0, i + 1)

Next i

End Sub

Public Sub CalculoAngulos()

Dim i As Integer

For i = 1 To TamVect

ReDim Preserve CordPolar(i)

If ((dirx(i) = 0) And (diry(i) > 0)) Then

CordPolar(i) = 90

ElseIf ((dirx(i) = 0) And (diry(i) < 0)) Then

CordPolar(i) = 270

ElseIf ((dirx(i) > 0) And (diry(i) = 0)) Then

CordPolar(i) = 0

ElseIf ((dirx(i) < 0) And (diry(i) = 0)) Then

CordPolar(i) = 180

ElseIf ((dirx(i) > 0) And (diry(i) > 0)) Then

CordPolar(i) = Round((Atn((diry(i) / dirx(i)))) \* 57.2957795131, 3) '(180/Pi)=57.29577

ElseIf Round((dirx(i) > 0) And (diry(i) < 0), 3) Then

CordPolar(i) = Round(360 + ((Atn((diry(i) / dirx(i)))) \* 57.2957795131), 3) '(180/Pi)=57.29577

ElseIf ((dirx(i) < 0) And (diry(i) > 0)) Then

CordPolar(i) = Round(180 + ((Atn((diry(i) / dirx(i)))) \* 57.2957795131), 3) '(180/Pi)=57.29577

ElseIf ((dirx(i) < 0) And (diry(i) < 0)) Then

CordPolar(i) = Round(180 + ((Atn((diry(i) / dirx(i)))) \* 57.2957795131), 3) '(180/Pi)=57.29577

End If

Next i

End Sub

Public Sub BSort()

Dim BSort As Boolean, iCta As Integer, Last As Integer, k As Integer, iAux As Single, iAux1 As Integer: Last = TamVect

Do While Not BSort

For k = 1 To Last - 1

If (CordPolarAux(k) > CordPolarAux(k + 1)) Then

iAux = CordPolarAux(k): CordPolarAux(k) = CordPolarAux(k + 1): CordPolarAux(k + 1) = iAux

iAux1 = NodosAux(k): NodosAux(k) = NodosAux(k + 1): NodosAux(k + 1) = iAux1

iCta = iCta + 1

End If

Next k

If iCta = 0 Then

BSort = True

Else

Last = Last - 1: iCta = 0

End If

Loop

End Sub

Public Sub BSortDRAW()

Dim BSortDRAW As Boolean, iCta As Integer, Last As Integer, k As Integer, iAux As Single, iAux1 As Integer: Last = TamVect

Do While Not BSortDRAW

For k = 1 To Last - 1

If (CordPolarAux(k) = CordPolarAux(k + 1)) Then

If (RadioAux(NodosAux(k)) > RadioAux(NodosAux(k + 1))) Then

iAux = CordPolarAux(k): CordPolarAux(k) = CordPolarAux(k + 1): CordPolarAux(k + 1) = iAux

iAux1 = NodosAux(k): NodosAux(k) = NodosAux(k + 1): NodosAux(k + 1) = iAux1

iCta = iCta + 1

End If

End If

Next k

If iCta = 0 Then

BSortDRAW = True

Else

Last = Last - 1: iCta = 0

End If

Loop

End Sub

Public Sub NewCalculos()

Dim i As Integer

For i = 1 To TamVect

ReDim Preserve CordPolarAux(i): CordPolarAux(i) = CordPolar(i)

ReDim Preserve RadioAux(i): RadioAux(i) = Radio(i)

ReDim Preserve NodosAux(i): NodosAux(i) = Nodos(i)

ReDim Preserve NodosNew(i): NodosNew(i) = NodosAux(i)

Next i

End Sub

Private TruckAsing() As Integer

Private Truck As Integer

Private ClientesAsing() As Integer

Private PesoCarga() As Integer

Private Rotation As Integer

Private MVecinos As Integer

Private i As Integer

Private j As Integer

Private k As Integer

Private RegisNodo As Integer

Private Sec() As Integer

Private ContSec() As Integer

Private T As Single

Private FO As Single

Private DistRuta As Single

Private TimeRuta As Single

Public Sub Clustering()

Dim i As Integer: TamDeFlota = Infi

ReDim TruckAsing(1 To TamVect + 1, 0 To TamVect + 1)

For Rotation = 0 To TamVect - 1

Call RotacionCartesiana

Call Barrido

If (Truck < TamDeFlota) Then

TamDeFlota = Truck

ReDim TruckAsignado(1 To TamVect + 1, 0 To TamVect + 1)

For Truck = 1 To Truck

ReDim Preserve ClientesAsignados(Truck): ClientesAsignados(Truck) = ClientesAsing(Truck)

ReDim Preserve PesoDeLaCarga(Truck): PesoDeLaCarga(Truck) = PesoCarga(Truck)

For i = 1 To ClientesAsing(Truck)

TruckAsignado(Truck, i) = TruckAsing(Truck, i)

Max = Maximo(ClientesAsing(Truck), Max)

Next i

frmConstruccion.MSFlexGrid1.Cols = (Max + 1)

frmConstruccion.MSFlexGrid1.Rows = (Truck + 1)

frmConstruccion.MSFlexGrid1.TextMatrix(Truck, 0) = "Gluster > " & Truck

For i = 1 To Max

frmConstruccion.MSFlexGrid1.TextMatrix(0, i) = "Nodo > " & i

Next i

For i = 1 To ClientesAsing(Truck)

frmConstruccion.MSFlexGrid1.TextMatrix(Truck, i) = TruckAsignado(Truck, i) 'Mostrar Clustering's...

Next i

Next Truck

End If

Next Rotation

Call InsercionProxima: Call frmConstruccion.InfoAdi: frmConstruccion.Show: MDIForm1.mnuMejoramiento.Enabled = True

End Sub

Private Sub InsercionProxima()

ReDim Secuencias(1 To TamDeFlota, 0 To (ClientesAsignados(TamDeFlota) + 1)): ReDim SecuenciasAux(1 To TamDeFlota, 0 To (ClientesAsignados(TamDeFlota) + 1))

For Truck = 1 To TamDeFlota

RegisNodo = TruckAsignado(Truck, 0)

ReDim ContSec(0 To TamVect): ContSec((TruckAsignado(Truck, 0))) = 1

For j = 1 To (ClientesAsignados(Truck))

ContSec(TruckAsignado(Truck, j)) = 0

Next j

For i = 0 To ClientesAsignados(Truck)

T = 10000

ReDim Preserve Sec(i): Sec(i) = RegisNodo

For j = 0 To ClientesAsignados(Truck)

If ((MtzDist(Sec(i), TruckAsignado(Truck, j)) < T) And (ContSec((TruckAsignado(Truck, j))) = 0)) Then

T = (MtzDist(Sec(i), TruckAsignado(Truck, j))): RegisNodo = TruckAsignado(Truck, j)

End If

Next j: ContSec(RegisNodo) = 1

Next i

If (TypeOpt = 4) Then

For k = 1 To 3

If (k = 1) Then Opt = 1

If (k = 2) Then Opt = 2

If (k = 3) Then Opt = 3

Call Intercambios

Next k

Else

Call Intercambios

End If

Next Truck

End Sub

Private Sub RotacionCartesiana()

Dim i As Integer

Dim Temporal As Integer

For i = 1 To TamVect

ReDim Preserve NodosNew(i): NodosNew(i) = NodosAux(i)

Next i

If (Rotation <> 0) Then

Temporal = NodosAux(1)

NodosNew(1) = NodosAux((Rotation + 1))

NodosNew((Rotation + 1)) = Temporal

End If

End Sub

Private Sub Barrido()

Dim i As Integer

Dim k As Integer

ReDim Preserve ClientesAsing(Truck): ClientesAsing(Truck) = 0

ReDim Preserve PesoCarga(Truck): PesoCarga(Truck) = 0

For i = 1 To TamVect

TruckAsing(Truck, k) = NodosNew(i)

ReDim Preserve ClientesAsing(Truck): ClientesAsing(Truck) = ClientesAsing(Truck) + 1

ReDim Preserve PesoCarga(Truck): PesoCarga(Truck) = PesoCarga(Truck) + PesoKg(NodosNew(i))

If (PesoCarga(Truck) > CapacityTruck) Then

ClientesAsing(Truck) = ClientesAsing(Truck) - 1

PesoCarga(Truck) = PesoCarga(Truck) - PesoKg(NodosNew(i))

Truck = Truck + 1

i = i - 1

End If: k = k + 1

Next i

End Sub

Private Sub OneOpt()

Dim Temporal As Integer

Temporal = SecuenciasAux(Truck, (MVecinos + 1))

SecuenciasAux(Truck, (MVecinos + 1)) = SecuenciasAux(Truck, MVecinos) 'SIGUIENTE

SecuenciasAux(Truck, MVecinos) = Temporal 'PRIMERO

End Sub

Private Sub TwoOpt()

Dim Temporal As Integer

Temporal = SecuenciasAux(Truck, (MVecinos + 2))

SecuenciasAux(Truck, (MVecinos + 2)) = SecuenciasAux(Truck, MVecinos) 'SIGUIENTE

SecuenciasAux(Truck, MVecinos) = Temporal 'PRIMERO

End Sub

Private Sub ThreeOpt()

Dim Temporal As Integer

Temporal = SecuenciasAux(Truck, (MVecinos + 3))

SecuenciasAux(Truck, (MVecinos + 3)) = SecuenciasAux(Truck, MVecinos) 'SIGUIENTE

SecuenciasAux(Truck, MVecinos) = Temporal 'PRIMERO

End Sub

Private Function Maximo(numero1 As Integer, numero2 As Integer) As Integer

If numero1 > numero2 Then

Maximo = numero1

Else

Maximo = numero2

End If

End Function

Private Sub Intercambios()

For MVecinos = 0 To (ClientesAsignados(Truck) - Opt) 'LIMITE IGUAL AL METODO OR-OPT ELEJIDO...

If (MVecinos <> 0) Then

If (TypeOpt = 1) Then Call OneOpt

If (TypeOpt = 2) Then Call TwoOpt

If (TypeOpt = 3) Then Call ThreeOpt

If (k = 1) Then Call OneOpt

If (k = 2) Then Call TwoOpt

If (k = 3) Then Call ThreeOpt

For i = 0 To ClientesAsignados(Truck)

ReDim Preserve Sec(i): Sec(i) = SecuenciasAux(Truck, i)

Next i

End If

ReDim Preserve Sec((ClientesAsignados(Truck) + 1)): Sec((ClientesAsignados(Truck) + 1)) = Sec(0)

For j = 0 To (ClientesAsignados(Truck))

DistRuta = DistRuta + MtzDist(Sec(j), Sec(j + 1))

TimeRuta = TimeRuta + MtzTime(Sec(j), Sec(j + 1))

Next j

If (DistRuta < FO) Then

frmConstruccion.MSFlexGrid2.Cols = (Max + 3): frmConstruccion.MSFlexGrid2.Rows = (Truck + 1)

frmConstruccion.MSFlexGrid2.TextMatrix(Truck, 0) = "Ruta > " & Truck

FO = DistRuta

ReDim SecuenciasAux(1 To TamDeFlota, 0 To (ClientesAsignados(Truck) + 1))

ReDim Preserve SecuenciasLocales(0 To TamDeFlota, 0 To (Max + 2)) 'preserve

For j = 0 To (ClientesAsignados(Truck) + 1)

ReDim Preserve MejorSec(j): MejorSec(j) = Sec(j)

SecuenciasAux(Truck, j) = MejorSec(j) 'SE UTILIZA PARA REALIZAR MVECINOS OR-OPT...

SecuenciasLocales(Truck, j) = MejorSec(j)

frmConstruccion.MSFlexGrid2.TextMatrix(Truck, j + 1) = SecuenciasLocales(Truck, j) 'Mostrar Secuencias Iniciales...

frmConstruccion.MSFlexGrid2.TextMatrix(0, j + 1) = "Nodo > " & j + 1

Next j

ReDim Preserve DistRutaTotal(1 To TamDeFlota): DistRutaTotal(Truck) = FO

ReDim Preserve TimeRutaTotal(0 To TamDeFlota): TimeRutaTotal(0) = 0: TimeRutaTotal(Truck) = TimeRuta

DistTotalRuteo = DistTotalRuteo + FO

TimeTotalRuteo = TimeTotalRuteo + TimeRuta

PesoTotal = PesoTotal + PesoDeLaCarga(Truck)

End If

Next MVecinos

End Sub

Private Tabu() As Integer

Private MtzEventF() As Integer

Private Truck As Integer

Public Fobject() As Single

Public Sub BusquedaTabu()

Dim i As Integer

Dim FOVecinaTabu As Single

Dim Cont As Integer

Dim Temp As Integer

Dim j As Integer

Dim foSec As Single

Dim foSecTabu As Single

Dim isTabu As Boolean

Dim Secuencia() As Integer

Dim Msvecina() As Integer

Dim MsvecinaTabu() As Integer

Dim TotalIter As Long

Dim FOVecina As Single

Dim LimIter As Integer

Dim TabuList() As Integer

Dim Penalizar() As Single

Dim Tabu2() As Integer

ReDim MtzEventF(0 To TamVect, 0 To TamVect + 1)

For Truck = 1 To TamDeFlota

ReDim Secuencia(1 To TamDeFlota, 0 To (ClientesAsignados(Truck) + 1))

ReDim Msvecina(1 To TamDeFlota, 0 To (ClientesAsignados(Truck) + 1))

ReDim MsvecinaTabu(1 To TamDeFlota, 0 To (ClientesAsignados(Truck) + 1))

ReDim TabuList(0 To TListTabu, 1 To 2)

ReDim Penalizar(0 To TListTabu)

For i = 0 To (ClientesAsignados(Truck) + 1)

Secuencia(Truck, i) = SecuenciasLocales(Truck, i)

Next i

While (LimIter < Iteraciones) CInt(((ClientesAsignados(Truck)) ^ 2) / 2))

If ((TotalIter > 0) And (Cont > 0)) Then

For i = 0 To (ClientesAsignados(Truck) + 1)

Secuencia(Truck, i) = Msvecina(Truck, i)

Next i

ElseIf ((TotalIter > 0) And (Cont = 0)) Then

For i = 0 To (ClientesAsignados(Truck) + 1)

Secuencia(Truck, i) = MsvecinaTabu(Truck, i)

Next i

End If

For i = 1 To (ClientesAsignados(Truck) - 1)

Temp = Secuencia(Truck, i + 1)

Secuencia(Truck, i + 1) = Secuencia(Truck, i)

Secuencia(Truck, i) = Temp

isTabu = False

For j = 0 To (ClientesAsignados(Truck))

foSec = foSec + MtzDist((Secuencia(Truck, j)), (Secuencia(Truck, j + 1)))

foSecTabu = foSecTabu + MtzPenalDist((Secuencia(Truck, j)), (Secuencia(Truck, j + 1)))

Next j: If (foSecTabu > Infi) Then isTabu = True

If ((isTabu = False) And (foSec < FOVecina)) Then 'CRITERIO DE ASPIRACION POR OBJETIVO FORMA GLOBAL...

ReDim Preserve Tabu(1 To 2): Tabu(1) = Secuencia(Truck, i + 1): Tabu(2) = Secuencia(Truck, i)

FOVecina = foSec

For j = 0 To (ClientesAsignados(Truck) + 1)

Msvecina(Truck, j) = Secuencia(Truck, j)

Next j

ElseIf ((isTabu = True) And (foSec < DistRutaTotal(Truck))) Then 'CRITERIO DE ASPIRACION POR DEFECTO...

ReDim Preserve Tabu(1 To 2): Tabu(1) = Secuencia(Truck, i + 1): Tabu(2) = Secuencia(Truck, i)

FOVecina = foSec

For j = 0 To (ClientesAsignados(Truck) + 1)

Msvecina(Truck, j) = Secuencia(Truck, j)

Next j

ElseIf ((isTabu = True) And (foSec < FOVecinaTabu) And (Cont = 0)) Then 'SI TODAS SON MALAS POSICIONES...

ReDim Preserve Tabu2(1 To 2): Tabu2(1) = Secuencia(Truck, i + 1): Tabu2(2) = Secuencia(Truck, i)

FOVecinaTabu = foSec

For j = 0 To (ClientesAsignados(Truck) + 1)

MsvecinaTabu(Truck, j) = Secuencia(Truck, j)

Next j

End If

Temp = Secuencia(Truck, i + 1)

Secuencia(Truck, i + 1) = Secuencia(Truck, i)

Secuencia(Truck, i) = Temp

Next i

If (Cont = 0) And (ClientesAsignados(Truck) > 1) Then 'SE ESCOGIO UNA SECUENCIA TABU COMO LA MEJOR...

Tabu(1) = Tabu2(1): Tabu(2) = Tabu2(2): FOVecina = FOVecinaTabu

For j = 0 To (ClientesAsignados(Truck) + 1)

Msvecina(Truck, j) = MsvecinaTabu(Truck, j)

Next j

End If

If (FOVecina < DistRutaTotal(Truck)) Then 'SE ENCONTRO UNA SECUENCIA NO TABU COMO LA MEJOR...

DistRutaTotal(Truck) = FOVecina: LimIter = 0: TimeRutaTotal(Truck) = (DistRutaTotal(Truck) / VelTruck)

frmMejoramiento.MSFlexGrid2.Cols = (Max + 3): frmMejoramiento.MSFlexGrid2.Rows = (Truck + 1)

Dim z As Integer

For z = 1 To Truck

DistTotalRuteo = DistTotalRuteo + DistRutaTotal(z)

Next z

frmMejoramiento.List3.Clear

frmMejoramiento.List1.AddItem "Recorrido (" & Truck & ") >> " & DistRutaTotal(Truck) & " Uad - Tiempo >> " & TimeRutaTotal(Truck) & " Uat"

frmMejoramiento.List3.AddItem "Recorrido Total >> " & DistTotalRuteo & " Uad - Tiempo Total >> " & Round((DistTotalRuteo / VelTruck), 3) & " Uat", 0

frmMejoramiento.List3.AddItem "Flota Total >> " & TamDeFlota & " Camiones - Costo Total >> " & ((CosteTruck \* 3000) \* TamDeFlota) & " Pesos/24 Hrs", 1

For j = 0 To (ClientesAsignados(Truck) + 1)

SecuenciasLocales(Truck, j) = Msvecina(Truck, j) 'SE ESCOGIO UNA SECUENCIA NO TABU COMO LA MEJOR...

frmMejoramiento.MSFlexGrid2.TextMatrix(Truck, j + 1) = SecuenciasLocales(Truck, j) 'Mostrar Mejor Secuencia Vecina...

frmMejoramiento.MSFlexGrid2.TextMatrix(0, j + 1) = "Nodo > " & j + 1

frmMejoramiento.MSFlexGrid2.TextMatrix(Truck, 0) = "Ruta > " & Truck

Next j

End If

If (((LimIter) > (TListTabu - 1)) And (ClientesAsignados(Truck) > 1)) Then 'SACAR ATRIBUTOS TABUS DE LA LISTA...

MtzPenalDist(TabuList(0, 1), TabuList(0, 2)) = Penalizar(0)

For i = 1 To TListTabu

TabuList(i - 1, 1) = TabuList(i, 1)

TabuList(i - 1, 2) = TabuList(i, 2)

Penalizar(i - 1) = Penalizar(i)

Next i

TabuList(TListTabu - 1, 1) = Tabu(1)

TabuList(TListTabu - 1, 2) = Tabu(2)

Penalizar(TListTabu - 1) = MtzDist(Tabu(1), Tabu(2))

Call Matriz\_Eventos\_Frecuentes

ElseIf (((LimIter) <= (TListTabu - 1)) And (ClientesAsignados(Truck) > 1)) Then

TabuList(LimIter, 1) = Tabu(1)

TabuList(LimIter, 2) = Tabu(2)

Penalizar(LimIter) = MtzDist(Tabu(1), Tabu(2))

Call Matriz\_Eventos\_Frecuentes

End If

Call Matriz\_Eventos\_Frecuentes

If (ClientesAsignados(Truck) > 1) Then: MtzPenalDist(Tabu(1), Tabu(2)) = Infi

LimIter = LimIter + 1: TotalIter = TotalIter + 1

ReDim Preserve Fobject(1 To TamDeFlota, 0 To Iteraciones + 1): Fobject(Truck, LimIter) = FOVecina

Wend

Next Truck

MDIForm1.mnuPareto.Enabled = True

MDIForm1.mnuPlanificacion.Enabled = True

MDIForm1.mnuWIP.Enabled = True

End Sub

Private Sub Matriz\_Eventos\_Frecuentes()

MtzEventF(Tabu(1), Tabu(2)) = ((MtzEventF(Tabu(1), Tabu(2))) + 1)

frmEventFrec.MSFlexGrid1.TextMatrix(0, Tabu(2) + 1) = "Nodo >> " & Tabu(2)

frmEventFrec.MSFlexGrid1.TextMatrix(Tabu(1) + 1, 0) = "Nodo >> " & Tabu(1)

frmEventFrec.MSFlexGrid1.TextMatrix(Tabu(1) + 1, Tabu(2) + 1) = MtzEventF(Tabu(1), Tabu(2)) & " (" & Tabu(1) & "," & Tabu(2) & ")" & " R" & Truck

End Sub

Private Truck As Integer

Private Rotation As Integer

Public Sub Scheduler()

Dim F As Integer

Dim F1 As Integer

ReDim Times(0 To TamDeFlota)

For Rotation = 0 To (TamDeFlota - 1)

Call RotacionCartesiana

Call Programar

If ((Ruta <= F) And (TiempoTotal < F1)) Then

F = Ruta: F1 = TiempoTotal

Call FrmDiagramaGantt.Parametros

End If

Next Rotation

MDIForm1.mnuGants.Enabled = True

MDIForm1.mnuControlRutero.Enabled = True

End Sub

Private Sub Programar()

Dim Posicion As Integer

ReDim Gantt(1 To TamDeFlota, 0 To TamDeFlota)

ReDim Preserve TimeXRuta(0 To TamDeFlota)

For Truck = 1 To TamDeFlota

Gantt(Ruta, Posicion) = PosTimes(Truck)

TimeXRuta(Ruta) = TimeXRuta(Ruta) + Times(Truck)

If ((Times(Truck) > HH) And (Posicion = 1)) Then

Posicion = 0 'Posicion Inicial

ElseIf ((TimeXRuta(Ruta) > 480) And (Posicion > 1)) Then

Gantt(Ruta, Posicion) = 0

TimeXRuta(Ruta) = TimeXRuta(Ruta) - Times(Truck)

Tciclo = Maximo(Tciclo, TimeXRuta(Ruta))

Gantt(Ruta, 0) = 0

Ruta = Ruta + 1 'Ruta Siguiente

Posicion = 0 'Posicion Inicial

Else

Tciclo = Maximo(Tciclo, TimeXRuta(Ruta))

End If

Posicion = Posicion + 1

Next Truck

For Truck = 1 To Ruta

TiempoTotal = TiempoTotal + TimeXRuta(Truck)

Next Truck

End Sub

Private Sub RotacionCartesiana()

Dim i As Integer

Dim Temporal As Single

Dim Temporal2 As Single

For i = 1 To TamDeFlota

ReDim Preserve Times(i): Times(i) = TimeRutaTotal(i)

ReDim Preserve PosTimes(i): PosTimes(i) = i

ReDim TimeXRuta(i): TimeXRuta(i) = 0

Next i

If (Rotation <> 0) Then

Temporal = Times(1)

Times(1) = Times((Rotation + 1))

Times((Rotation + 1)) = Temporal

Temporal2 = PosTimes(1)

PosTimes(1) = PosTimes((Rotation + 1))

PosTimes((Rotation + 1)) = Temporal2

End If

End Sub

**4. VALIDACIÓN DEL SOFTWARE RV-SMARTCITIES**

La validación se realizó comprobando que el sistema cumpliera con los requisitos que se exponen en el proyecto de investigación ALTERNATIVA ALGORÍTMICA PARA PROBLEMAS DE ENRUTAMIENTO DE VEHÍCULOS EN SMART CITIES. Para esto se realizaron pruebas de aceptación en instancias conocidas y aleatorias, y en todos los casos mostrando buenos resultados.