**UNIVERSIDAD DE LA REPÚBLICA**

Facultad de Ingeniería

Instituto de Computación

“MDVRP”

Informe Proyecto de Grado

Javier de Prado

Alejandro García

Francisco Güella

Tutores

Sandro Moscatelli

Omar Viera

Contenido

[Abstract 7](#_Toc432313363)

[Capítulo 1 8](#_Toc432313364)

[Introducción 8](#_Toc432313365)

[1.1 Contexto 8](#_Toc432313366)

[1.2 Estructura del documento 9](#_Toc432313367)

[Capítulo 2 10](#_Toc432313368)

[Estado del Arte 10](#_Toc432313369)

[2.1 Variantes en los problemas de ruteo de vehículos 11](#_Toc432313370)

[2.1.1 Ventanas de tiempo 11](#_Toc432313371)

[2.1.2 Tipos de flota disponible 11](#_Toc432313372)

[2.1.3 Periodicidad 12](#_Toc432313373)

[2.1.4 Otras variantes en los problemas de ruteo de vehículos 13](#_Toc432313374)

[2.1.4.1 SDVRP (Split Delivery VRP) 13](#_Toc432313375)

[2.1.4.2 SVRP (Stochastic VRP) 13](#_Toc432313376)

[2.1.4.3 VRPPD (VRP Pickup and Delivery) 14](#_Toc432313377)

[2.1.4.4 VRPB (VRP with Backhauls) 15](#_Toc432313378)

[2.1.4.5 Variantes en la Ruta 15](#_Toc432313379)

[2.1.4.6 Variantes en la función objetivo (funciones Multi-Objetivo) 16](#_Toc432313380)

[2.1.4.7 Variantes en la capacidad de los depósitos en problemas multi-depósito 16](#_Toc432313381)

[2.2 Formulación Matemática 17](#_Toc432313382)

[2.2.1 Formulación Matemática de TSP 18](#_Toc432313383)

[2.2.2 Formulación Matemática de VRP 20](#_Toc432313384)

[2.2.3 Formulación Matemática de MDVRP 22](#_Toc432313385)

[2.3 Revisión de publicaciones de Multi-Depot Vehicle Routing Problem 23](#_Toc432313386)

[2.4 Métodos para la resolución de problemas de ruteo de vehículos 26](#_Toc432313387)

[2.4.1 Métodos Exactos 26](#_Toc432313389)

[2.4.2 Métodos Heurísticos 28](#_Toc432313390)

[2.4.3 Heurísticas para VRP 28](#_Toc432313391)

[2.4.3.1 Algoritmo de Ahorros de Clarke and Wright 29](#_Toc432313392)

[2.4.3.2 Heurísticas de Inserción 30](#_Toc432313393)

[2.4.3.3 Heurísticas de dos fases 31](#_Toc432313394)

[2.4.4 Heurìsticas para MDVRP 32](#_Toc432313395)

[2.4.4.1 Heurísticas de dos Fases para MDVRP y sus variantes 32](#_Toc432313396)

[2.4.5 Meta-Heurísticas para VRP 40](#_Toc432313397)

[2.4.6 Meta–Heurísticas para MDVRP 40](#_Toc432313398)

[2.4.6.1 Particle Swarm Optimization (PSO) 40](#_Toc432313399)

[2.4.6.2 Tabu Serch 41](#_Toc432313400)

[2.5 Post Optimización y mejoras 43](#_Toc432313401)

[2.5.1 Operador λ–Intercambio 43](#_Toc432313405)

[2.5.2 Operador Or–opt 44](#_Toc432313406)

[2.5.3 Operadores de Van Breedam 45](#_Toc432313407)

[2.5.4 GENI y GENIUS 45](#_Toc432313408)

[2.5.4.1 GENI 45](#_Toc432313409)

[2.5.4.2 GENIUS 47](#_Toc432313410)

[2.6 Soluciones de software existentes para MDVRP 47](#_Toc432313411)

[Capítulo 3 50](#_Toc432313412)

[Definición del problema 50](#_Toc432313413)

[3.1 Características buscadas de la aplicación 50](#_Toc432313417)

[3.2 Algoritmos de MDVRP propuestos para la solución. 52](#_Toc432313418)

[3.2.1 Asignacion Por Urgencia sin Capacidad 52](#_Toc432313419)

[3.2.2 Asignación por Urgencia con Capacidades (Fase 1) 53](#_Toc432313420)

[3.2.3 Algoritmo Enajenado Lento (AEL). Fase 2 54](#_Toc432313421)

[3.2.4 Algoritmo Enajenado Rápido (AER). Fase 2 56](#_Toc432313422)

[3.2.5 Ruteo 57](#_Toc432313423)

[3.2.6 Post Optimización 57](#_Toc432313424)

[Capítulo 4 59](#_Toc432313425)

[Implementación 59](#_Toc432313426)

[4.1 Interfaz Gráfica 60](#_Toc432313427)

[4.2 Lectura de Datos y carga de datos. 62](#_Toc432313428)

[4.3 Capa Lógica 63](#_Toc432313429)

[4.3.1 Asignación 63](#_Toc432313430)

[4.3.2 Ruteo 64](#_Toc432313431)

[4.3.3 Flujo del programa 64](#_Toc432313432)

[4.4 Implementación de funcionalidades Misceláneas 65](#_Toc432313433)

[4.4.1 Representación grafica de la matriz de distancias 65](#_Toc432313434)

[4.4.2 Generador de casos de Prueba y su funcionamiento 66](#_Toc432313435)

[Capítulo 5 68](#_Toc432313436)

[Testeos 68](#_Toc432313437)

[5.1 Casos de Prueba 68](#_Toc432313439)

[5.2 Plan de pruebas y resultados esperados 71](#_Toc432313440)

[5.3 Ejecución y Resultados 71](#_Toc432313441)

[5.3.1 Asignación y Clarke & Wright 72](#_Toc432313442)

[5.3.2 Aplicando métodos de Post-Optimización (cambios intra-ruta) 74](#_Toc432313443)

[5.3.3 Aplicando métodos de Post-Optimización (cambios inter-rutas) 76](#_Toc432313444)

[5.4 Análisis de Resultados. 77](#_Toc432313445)

[5.4.1 Discusión de la penalidad. 77](#_Toc432313446)

[5.4.2 Mejoras en los costos 78](#_Toc432313457)

[5.4.3 Casos Misceláneos 80](#_Toc432313458)

[Capítulo 6 82](#_Toc432313459)

[Conclusiones 82](#_Toc432313460)

[Capítulo 7 83](#_Toc432313461)

[Trabajos a futuro 83](#_Toc432313462)

[Anexo 84](#_Toc432313463)

[Comparación de las dos heuristicas 84](#_Toc432313464)

[Manual de Usuario 85](#_Toc432313465)

[Entorno de trabajo 85](#_Toc432313466)

[Procedimientos descriptivos 86](#_Toc432313467)

[Abrir un archivo de prueba 86](#_Toc432313468)

[Configuración de prueba 88](#_Toc432313469)

[Interacción con el Mapa 88](#_Toc432313470)

[Ejecución de Algoritmos 89](#_Toc432313471)

[Algoritmos de Asignación 89](#_Toc432313472)

[Algoritmo de Ruteo 90](#_Toc432313473)

[Algoritmo de Post-Optimización 91](#_Toc432313474)

[Tiempo de ejecución 91](#_Toc432313475)

[Bibliografía 92](#_Toc432313476)

# Abstract

La distribución de mercadería desde los depósitos hacia los clientes es un problema práctico y desafiante en la gestión logística. Mejores decisiones al momento de rutear pueden resultar en un mayor nivel de satisfacción del cliente debido a que más clientes se pueden servir en un tiempo más corto y se pueden lograr ahorros en los costos de distribución. El problema de la distribución se formula como un problema de ruteo de vehículos (VRP). Sin embargo, en los casos cuando una empresa cuenta con más de un depósito, el problema VRP no es adecuado. Para resolver esta limitación, este trabajo se centra en el problema VRP con múltiples depósitos, o multi-depósito de VRP (MDVRP). El problema MDVRP es NP-duro, lo que significa que un algoritmo eficiente para resolver el problema de optimización en forma exacta no es posible debido al elevado costo computacional. Para tratar de resolver estos problemas de manera eficiente, se desarrollan en este documento dos algoritmos los cuales llamamos “Algoritmo Enajenado Rápido” (AER) y “Algoritmo Enajenado Lento” (AEL). La principal diferencia entre los dos algoritmos es que el algoritmo AEL logra mejores resultado pero en mayor tiempo.

# Capítulo 1

# Introducción

Este documento trata sobre el informe final del proyecto de grado de la carrera Ingeniería en Computación de los estudiantes Francisco Güella, Alejandro García y Javier de Prado.

## Contexto

La gestión logística es un elemento clave en la estrategia empresarial, siendo una de sus funciones principales la distribución, y dentro de ella la capacidad para optimizar las rutas de transporte. En este contexto, las empresas deben analizar los factores más relevantes en el diseño de sus rutas vehiculares así como las metodologías más adecuadas para tal optimización. La optimización de una ruta engloba todas las acciones que contribuyen a la mejora de la función de distribución en términos de nivel de servicio, calidad y costos a través de decisiones de carácter estratégico, táctico y operativo. [1]

El estudio de los problemas de optimización combinatoria se remonta a 1784 cuando G. Monge busca la forma óptima de transportar tierra desde un terreno a otro. En su estudio, busca la forma de transportar tierra de forma tal que la distancia total de transporte sea la menor posible. [2].

El Problema a estudiar en este proyecto de grado, el cual está relacionado a la gestión logística y a la optimización combinatoria, es el de Ruteo de Vehículos con múltiples Depósitos (MDVRP, Multi Depot Vehicle Routing Problem). El escenario planteado presenta a un conjunto de clientes a los cuales hay que distribuirles mercadería. Quienes distribuyen la mercadería cuentan con varios depósitos y una flota de vehículos. La mercadería se traslada a través de la flota de vehículos. El problema planteado es el de optimizar la elección de las rutas que deben realizar los vehículos para satisfacer la demanda de los clientes teniendo en cuenta que los vehículos tienen una capacidad limitada para el transporte de la mercadería. Típicamente se plantea que los vehículos comiencen y terminen su ruta en el mismo depósito y además el cliente recibe una única visita de un vehículo de la flota. El mencionado es la versión básica del problema. Bodin et al en [3] formula el problema de Ruteo de Vehículos con múltiples Depósitos. Otras definiciones del problema MDVRP se pueden encontrar en [4] [5] [6]. Se han propuesto distintas variantes las cuales se comentarán más adelante.

El problema de optimizar las rutas para los vehículos se puede entender como el problema de encontrar un valor mínimo para algún criterio como puede ser distancia, tiempo, consumo de combustible, etc., relacionado a la ruta del vehículo. En general a este criterio se lo presenta como costo. En la definición del problema recién expuesta, se planteó que un vehículo distribuye mercadería a un cliente. Un problema equivalente sería el de recoger mercadería de los clientes y llevarlos a los depósitos. Por ejemplo, cuando un camión levanta la leche de los tambos.

## Estructura del documento

En el Capítulo 2 se realiza una reseña de algunos de los problemas de ruteo de vehículos que han sido más estudiados. Se dan formulaciones de estos problemas como problemas de programación entera. Asimismo, se resumen las ideas principales que han sido utilizadas en el diseño de algoritmos para su resolución, tanto a nivel de métodos exactos como de heurísticas y metaheurísticas.

En el Capitulo 3 se presentan información de las soluciones de software existentes para problemas de ruteo de vehículos con múltiples depósitos.

En el Capítulo 4 se describe el problema a resolver.

En el Capítulo 5 se presentan dos algoritmos para resolver el problema.

Los resultados obtenidos por los algoritmos se reportan en el Capítulo 6. El análisis realizado busca cuantificar experimentalmente el desempeño de los algoritmos en términos de la calidad de las soluciones encontradas y el tiempo de ejecución.

Finalmente, en el Capítulo 7 se presentan las conclusiones finales del trabajo y algunas ideas en las cuales se podría profundizar en el futuro

# Capítulo 2

# Estado del Arte

El problema MDVRP es una generalización del problema VRP (Vehicle Routing Problem) [4]. El problema VRP consta de optimizar las rutas en el mismo escenario, con la diferencia que se cuenta con un único depósito. Fue formulado en 1959 por Dantzig y Ramser [7] en donde se presenta “The Truck dispatching Problem” en el cual un camión debe distribuir combustible a un conjunto de estaciones de servicio (clientes). Las estaciones tienen una demanda y los camiones capacidades de carga de combustible. De esta forma se daba comienzo al estudio de lo que luego se conocería como VRP [8].

A su vez, el problema VRP es una generalización del problema TSP. (Travelling Salesman Problem). Así presentaron Dantzig y Ramser  “The Truck dispatching Problem” [7], como una generalización de TSP. Travelling Salesman Problem, en castellano “Problema del Agente Viajero” es el siguiente: Dada una lista de ciudades y las distancias entre cada una de ellas, ¿cuál es la ruta más corta posible que visita cada ciudad exactamente una vez y regresa a la ciudad origen? [2] [9]. Tomando a las ciudades como clientes, y tomando a un único vehículo que no lleva ninguna carga y que solamente debe visitar a los clientes, es que se puede ver al problema VRP como una generalización del problema TSP. O sea que TSP sería un caso particular del problema VRP como a su vez VRP sería un caso particular del problema MDVRP.

El problema TSP es un problema NP-Duro, demostrado por Richard Karp en 1972 [10]. También los problemas VRP y MDVRP son NP-duros [11]. Esta es la razón por la cual el objetivo que se plantea generalmente es encontrar una buena solución y no la que minimiza el costo total (solución óptima).

La complejidad NP-Duro de estos problemas, que aumenta exponencialmente a medida que lo hace el número de clientes, dificulta el desarrollo de métodos que resuelvan el problema de manera óptima en un tiempo razonable. No obstante, y a pesar de su elevado costo computacional, existen métodos exactos aplicados a este tipo de problemas que serán mencionados posteriormente. El enfoque más habitual a la hora de resolver estos problemas es el de aplicar métodos heurísticos o meta heurísticos, capaces de generar soluciones cercanas a la óptima sin incurrir en altos tiempos de ejecución y carga computacional.

## Variantes en los problemas de ruteo de vehículos

La diversidad de aplicaciones donde asuntos de ruteo pueden ser encontrados ha llevado a plantear diferentes variantes de los problemas VRP y MDVRP. Mencionaremos algunas de ellas.

### Ventanas de tiempo

L. Bodin et al, en su estado del arte sobre el ruteo de vehículo [3], presenta la restricción de ventanas de tiempo. La misma restringe en una ventana horaria el tiempo en el cuál un cliente puede ser atendido por un vehículo.

Una variante a esta restricción se puede encontrar en [12]. Donde Lei Wen et al, plantean que la misma no es estricta. Esto quiere decir que se puede dar servicio a un cliente fuera de su ventana de tiempo, pero esto aumentaría el costo de la ruta del vehículo. Distinguen las restricciones de ventana de tiempo entre hard time window y soft time windows. Ellos (Lei Wen et al) utilizan soft time Windows para el abordaje del problema. En hard time Windows el vehículo no podría dar servicio al cliente fuera de la ventana de horario bajo ningún concepto.

### Tipos de flota disponible

L. Bodin et al, en su estado del arte sobre el ruteo de vehículo [3], presenta la siguiente distinción entre los tipos de vehículos que podrían componer la flota.

Flota homogénea: todos los vehículos de la flota son idénticos.

Flota heterogénea: hay varios tipos de vehículos con distintas capacidades.

L. Bodin et al, presenta una situación en el planteo “The Fleet Size and Mix Problem” en la cual la flota heterogénea de los proveedores de servicio se compone por vehículos comprados y vehículos alquilados. En este caso esta situación de flota heterogénea implica en este planteo que los costos de las rutas estén compuestas por un costo fijo (el del alquiler del vehículo) y un costo variable (el costo de transitar la ruta). En esta formulación se asumió que la cantidad de vehículos de cada tipo a disposición es ilimitada.

En [11] Kumar et al, proponen las siguientes dos opciones en la formulación del problema VRP. Dicen que la cantidad de vehículos los cuales dispone la flota, puede ser una cantidad dada a priori, o la misma podría llegar a ser una variable de decisión.

Baldacci et al, presentan en [13] una variante de VRP, en la cuál una flota de vehículos caracterizados por diferentes capacidades y costos está disponible para las actividades de distribución. En este caso clasifica a esta variable de VRP como “Mixed Fleet VRP” o “Heterogeneous Fleet VRP”. Baldacci plantea que hay cierta homogeneidad en llamar a Heterogeneous VRP como la variante que limita el número de vehículos, y Fleet Size and Mix VRP a la variable que no limita la cantidad de vehículos.

Baldacci [13] muestra una clasificación de las variantes VRP según costos fijos y variables de la flota, Fleet Size and Mix y Heterogeneous VRP

### Periodicidad

Tan y Beasley [14] definen a la variante de PVRP (Period VRP) de la siguiente manera:

The Period Vehicle Routing Problema (PVRP) es el problema de designar rutas de vehículos para todos los días de un período dado de días donde no todos los clientes requieren servicio en todos los días del período. Típicamente si un cliente requiere visitas () durante el período entonces estas visitas podrían ocurrir en una manera determinada dado un número de posibles maneras. Por ejemplo si un cliente requiere servicio dos días en un período de 5 días entonces las posibles combinaciones de días para brindar el servicio podrían ser Lunes/Miércoles, Martes/Jueves, o Miércoles/Viernes donde estas serían las únicas combinaciones de días aceptables.

Francis y Smilowitz en [15] proponen una variante al problema recién descrito. El mismo, PVRP-SC (The Period Vehicle Routing Problem with Service Choice) presenta la siguiente característica. Mientras que en PVRP, las frecuencia de visitas al cliente en un período de tiempo está predeterminada, en PVRP-SC la frecuencia es una variable de decisión. En la definición de PVRP recién presentada, las únicas opciones eran que un cliente sea visitado en una de las siguientes combinaciones: Lunes/Miércoles, Martes/Jueves, o Miércoles/Viernes. En PVRP-SC, si bien se plantea una frecuencia mínima (por ejemplo que el cliente tenga que ser visitado por lo menos un día a la semana), dicha frecuencia podría ser mayor y es una variable de decisión. La frecuencia con que se visita a un cliente en este caso influye en el costo de la ruta. O sea que impactaría en la función objetivo del problema. Los beneficios de una mayor frecuencia podrían representar los ahorros en el cliente de costos de almacenamiento, o la disposición del cliente de pagar por más visitas en la semana.

Tan y Beasley en [14] hacen notar que PVRP es una generalización del problema VRP, el cual tendría un período de un solo día.

Mingozzi y Valleta, en [16] relacionan al PVRP con el MDVRP ya mencionado. Dicen que MDVRP puede ser visto como un caso especial de PVRP donde cada día del período de planeamiento corresponde a un depósito diferente y donde cada cliente requiere ser visitado una única vez en el período tomado en cuenta.

### Otras variantes en los problemas de ruteo de vehículos

Algunas otras variantes que se pueden encontrar en los problemas de ruteo de vehículos son las siguientes:

#### SDVRP (Split Delivery VRP)

En [17] se plantea la variante SDVRP la cual es una relajación del VRP, en donde se elimina la restricción de que cada cliente sea visitado una sola vez por un vehículo, admitiéndose que el mismo puede ser visitado cualquier cantidad de veces por distintos vehículos. Es decir que la demanda del cliente puede ser partida de forma tal que sea satisfecha por varios vehículos (Split Delivery). Este problema fue planteado e investigado por Dror y Trudeau entre 1989 y 1990, los cuales mostraron que al permitir entregas partidas en la resolución del VRP se puede lograr una disminución de la cantidad de rutas y por lo tanto una reducción del costo total de la solución.

#### SVRP (Stochastic VRP)

Se trata de un VRP en que uno o varios componentes del problema son aleatorios. Se pueden clasificar en tres diferentes tipos de SVRP:

* Clientes estocásticos: Cada cliente está presente con probabilidad y ausente con probabilidad .
* Demandas estocásticos: La demanda de cada cliente es una variable aleatoria.
* Tiempos estocásticos: Los tiempo de servicio y los tiempos de viajes son variables aleatorias.

En SVRP, dos etapas se realizan para obtener una solución. Una primera solución se determina antes de conocer las realizaciones de las variables aleatorias. En una segunda etapa, un recurso o acción correctiva se pueden tomar cuando se conocen los valores de las variables aleatorias.

El objetivo es reducir al mínimo la flota de vehículos y la suma del tiempo de viaje necesaria para abastecer a todos los clientes con valores aleatorios en cada ejecución para los clientes que se sirven, sus demandas y/o los tiempos de servicio y de viaje [18].

#### VRPPD (VRP Pickup and Delivery)

El VRPPD (VRP Problema con Entregas y Devoluciones) es una variante del VRP donde se contempla la posibilidad de que un cliente que ha recibido un envío disponga además de cierta mercancía que necesita ser recogida. Por lo tanto, se debe tener en cuenta que los productos que los clientes introducen en el vehículo no deben nunca exceder la capacidad del vehículo. Esta restricción dificulta aún más el problema de planificación y puede conllevar una mala utilización de la capacidad de los vehículos, incrementar las distancias reco0rridas o la necesidad de utilizar una flota más amplia. Por consiguiente, es habitual considerar situaciones restrictivas en las que todos los envíos comienzan en el depósito y todas las devoluciones vuelven al depósito central. Así se impide la posibilidad de intercambio de mercancías entre clientes. Una alternativa es la de relajar la restricción de que todos los clientes deben ser visitados al menos una vez. Y otra simplificación habitual es la de considerar que cada vehículo debe entregar todos los productos antes de comenzar con las devoluciones. Así pues, el objetivo es minimizar la flota de vehículos y la suma de los tiempos de transporte con la restricción de que cada vehículo debe tener suficiente capacidad para transportar los productos que van a ser entregados y aquellos que debe recoger en cada cliente para traerlos de vuelta al depósito.

Para que una solución se considere factible, la cantidad total asignada a cada ruta no debe exceder la capacidad del vehículo que sirve esa ruta y dicho vehículo tendrá suficiente capacidad para recoger las devoluciones que cada cliente tenga en su poder [18].

#### VRPB (VRP with Backhauls)

El problema VRPB (VRP con viajes de regreso) es como un VRP clásico en el que los clientes pueden recibir o entregar productos. Así, se necesita un VRPPD para tener en cuenta las mercancías que los clientes devuelven deben caber en el vehículo que les acaba de hacer la entrega. El supuesto fundamental de que todas las entregas se pueden hacer antes de iniciar las devoluciones puede tenerse en cuenta.

Las cantidades que se deben entregar y recoger son conocidas de antemano. Un VRPB es similar al VRPPD pero con la restricción estricta de que las entregas para cada ruta deben ser completadas antes de realizar ninguna devolución.

El objetivo es encontrar un conjunto de rutas que minimice la distancia total recorrida [18].

#### Variantes en la Ruta

En algunos planteos de problemas de ruteo se consideran restricciones con respecto a las rutas que deberán transitar los vehículos. Por ejemplo, en la formulación de “The Truck Dispatching Problem” de B. Golden, la cual busca minimizar la distancia recorrida de los camiones, se plantea que el tiempo que le lleva a un camión transitar su ruta no debería superar un máximo dado [8]. La misma podría representar la restricción de que un chofer de camión por las leyes de tránsito no podría manejar más de tantas horas diarias.

O, como se indica en [19] donde se restringe que una ruta no puede superar un costo máximo especificado.

#### Variantes en la función objetivo (funciones Multi-Objetivo)

Al estudiar los problemas de ruteo de vehículos, se encuentran variantes en cuanto a la función objetivo que se plantea. Hasta ahora se mencionó que el problema es minimizar el costo total de las rutas que realizan los vehículos. Pudiendo representar dicho costo, la distancia o el tiempo que implican recorrer la ruta. Pero también se encuentran algunas variantes al respecto. Por ejemplo, García-Najera, en [20] se propone un problema de ruteo de vehículos en donde se proponen minimizar el costo total de las rutas y a su vez minimizar la cantidad de rutas (o vehículos) simultáneamente. Lo presenta como un problema Bi-objetivo.

Otro ejemplo es el estudio realizado por Robert Bowerman en [21] donde presenta una formulación matemática multi-objetivo del problema de ruteo del bus escolar urbano. En el mismo se contempla, además del criterio de eficiencia, que para Bowerman la solución más eficiente es la que utiliza menos vehículos, el criterio de equidad de las rutas para los estudiantes. En el mismo dice que las rutas deben ser equilibradas. Una forma de proveer equidad en las rutas podría ser haciendo un “balance de carga” de los buses, que implica que la cantidad de estudiantes que lleva cada ruta debería ser equivalente. A su vez plantea un “balance de largo” el cual implica que no deberían existir rutas muy distintas en el largo de su recorrido. También se toma en cuenta la distancia que un alumno camina por día desde su casa a la parada del bus. Bowerman plantea 5 funciones matemáticas en la formulación del problema del bus.

*=Distancia total de las rutas de los buses.*

*=Distancia que deben caminar los estudiantes.*

*=Balanceo de carga.*

*=Balanceo del largo.*

Con estas definiciones de funciones, el objetivo es minimizar la función () sujeto a un conjunto de restricciones (por ejemplo que un estudiante no puede caminar más de distancia, o que el largo de una ruta no puede superar ). Teniendo en cuenta que el balance de carga y largo es menor cuanto más parecidas sean las cargas y los largos de las rutas.

#### Variantes en la capacidad de los depósitos en problemas multi-depósito

Es la variante en los problemas de ruteo con múltiples depósitos en los cuales estos están limitados por una capacidad establecida. A diferencia de los clásicos problemas MDVRP, en los cuales se asume que un depósito tiene capacidad ilimitada y las restricciones de capacidad son únicamente para los vehículos. [22]

## Formulación Matemática

Generalmente a estos problemas se los modela a través de grafos ponderados.

Sea un grafo . El conjunto de nodos representa los sitios que participan en el problema, es decir, clientes y depósitos, donde es la cantidad total de clientes y depósitos. [3] [23]

La existencia de un arco indica que es posible transportarse desde el sitio representado por al sitio representado por. Es usual que a cada arco se le asocie un costo que indica la manera más económica de transportarse de a . es la matriz que representa los costos de los arcos.

Cuando se trata de un único depósito, generalmente el mismo se representa con el nodo . Cuando se trata de varios depósitos, se suele dividir el conjunto en dos subconjuntos y donde ={,,…} es el conjunto de los clientes y ={,…., el conjunto de los depósitos [12] [4]. Wen y Meng en [12] utilizan un grafo dirigido en donde representa el costo de transportarse de a . Sin embargo, Wang utiliza un grafo sin dirigir ya que representa la distancia entre el nodo y el nodo . L. Bodin et al utiliza un grafo no dirigido ya que para él representa también la distancia de ir de a y por lo tanto sería igual .

Una ruta es un ciclo simple en que representa la secuencia de visitas realizadas por el vehículo que recorre la ruta. La ruta comienza y finaliza en el mismo depósito en caso del problema VRP y MDVRP, o ciudad origen en el caso de TSP. El costo de una ruta se obtiene sumando los costos de los arcos que forman el ciclo.

En la mayor parte de los casos será un grafo completo, pues en una red de transporte real dados dos sitios cualesquiera existe una manera de transportarse de uno al otro.

Como se muestra a continuación, cada problema particular tendrá sus propias características y restricciones de modelado. Se mostrarán algunos ejemplos de modelado y formulación matemática.

|  |  |
| --- | --- |
| (a) Una instancia de del TSP | (b) Una solución factible |

Figura 1.1: Un TSP de ejemplo y una solución.

### Formulación Matemática de TSP

La siguiente formulación del TSP como problema de programación entera binaria fue propuesta por Dantzig, Fulkerson y Johnson [9] en 1954:

s.a.

(1.1)

(1.2)

(1.3)

(1.4)

(1.5)

|  |
| --- |
|  |

Figura 1.2: Una solución para la instancia de la Figura 1.1(a) formada por 3 sub-tours.

Las variables binarias indican si el arco pertenece a la ruta ( o no . La función objetivo (1.1) establece que el costo total de la solución es la suma de los costos de los arcos utilizados. La restricción (1.2) restringe los valores que puede tomar y las restricciones (1.3) y (1.4) indican que la ruta debe llegar y abandonar cada nodo exactamente una vez. Finalmente en la restricción (1.5) se utiliza el conjunto para prohibir soluciones sub-tour que cumplan las restricciones de asignación (1.2), (1.3) y (1.4). Esta restricción es llamada restricción de eliminación de sub-tours e impone que todo subconjunto de nodos debe ser abandonado al menos una vez. Si no se impusiera esta restricción se estaría admitiendo soluciones que constan de más de un ciclo, como la que se muestra en la *Figura 1.2*. Esta solución está formada por tres sub-tours y viola la restricción (1.5) para los conjuntos .

Se han propuesto varias alternativas para el conjunto . Algunas de estas alternativas son:

### Formulación Matemática de VRP

El problema de ruteo de vehículos (VRP) como ya se mencionó, se consideró por primera vez por Dantzig y Ramser [7], que desarrollaron un enfoque heurístico utilizando las ideas de programación lineal. En este enfoque los vehículos solo tienen restricciones de capacidad y costo máximo de la ruta que recorren.

A continuación la formulación de este problema [8] [19]

s.a.

(1.6)

(1.7)

(1.8)

(1.9)

(1.10)

(1.11)

(1.12)

(1.13)

(1.14)

(1.15)

Donde

= Número de vehículos

= Capacidad del vehículo

= Costo máximo permitido para la ruta de vehículo

= Demanda del nodo ,

= 1, si el par pertenece a la ruta del vehículo , 0 en otro caso.

En la formulación anterior, las restricciones (1.7) y (1.8) aseguran que cada cliente es atendido por uno y sólo un vehículo. La continuidad de la ruta está representada por (1.9). La restricción (1.10) representa las limitaciones de capacidad del vehículo y (1.11) representa las limitaciones de costo de cada ruta. Las restricciones (1.12) y (1.13) aseguran que la disponibilidad de vehículos no sea superada. Finalmente la restricción (1.15) se utiliza para prohibir soluciones sub-tour. Esta última restricción también se puede es escribir como una desigualdad:

Se asume que y ,

La demanda en cada nodo es menor o a lo sumo igual a la capacidad de cada vehículo.

### Formulación Matemática de MDVRP

La formulación del problema de MDVRP se presenta a partir de la formulación vista anteriormente de VRP. Siendo los depósitos. Dicha formulación se encuentra en [19], donde además se presentan distintas formulaciones para el mismo problema.

s.a.

(1.16)

(1.17)

(1.18)

(1.19)

(1.20)

(1.21)

(1.22)

(1.23)

(1.24)

(1.25)

Donde

= Número de vehículos

= Capacidad del vehículo

= Costo máximo permitido para la ruta de vehículo

= Demanda del nodo ,

= 1, si el par pertenece a la ruta del vehículo , 0 en otro caso.

En la formulación anterior, las restricciones (1.17) y (1.18) aseguran que cada cliente es atendido por uno y sólo un vehículo. La continuidad de la ruta está representada por (1.19). La restricción (1.20) representa las limitaciones de capacidad del vehículo y (1.21) representa las limitaciones de costo de cada ruta. Las restricciones (1.22) y (1.23) aseguran que la disponibilidad de vehículos no sea superada. Finalmente la restricción (1.25) se utiliza para prohibir soluciones sub-tour.

## Revisión de publicaciones de Multi-Depot Vehicle Routing Problem

En el trabajo «A literature review on the vehicle routing problem with multiple depots» [24] publicado por Montoya en febrero del 2015, se muestra el aumento en la cantidad de publicaciones de MDVRP y sus variantes desde la publicación inicial de Kulkarni and Bhave 1985 [19]. El grafico se presente en a continuación en la *Figura 1.3.*

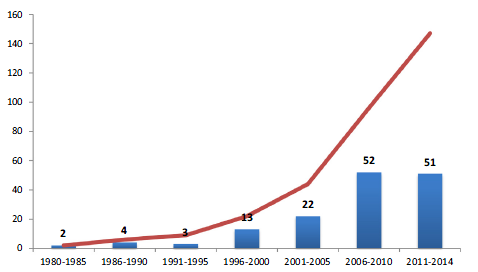


Figura 1.3: Distribución de publicaciones por año de MDVRP

Las variantes de problemas de ruteo de vehículos vistas anteriormente como ventana de tiempo, tipo de flota, periodicidad y recogida y entrega (pick up and deliver) son las que contienen mayor número de publicaciones sobre MDVRP, en cambio otras variantes como variaciones en las rutas Stocastic MDVRP fueron menos estudiadas. Igual así existen numerosas publicaciones de estos tipos de problemas.

Es Importante destacar que al contar con varios depósitos la capacidad de los mismos puede ser una nueva restricción, limitando así el número de clientes asignados a cada depósito. Esta variante en algunos trabajos se referencia como CMDVRP (Capacitated MDVRP) pero en otros simplemente se le llama MDVRP. [25]

Otro aspecto a destacar al contar con más de un depósito; es la variantes de MDVRP en la que los vehículos tengan que pasar por depósitos particulares en su recorrido, o finalizar el recorrido en un depósito distinto al inicial. A esta variante de MDVRP se la conoce como MDVRPI (Multi-Deport Vehicle Routing Problem with Inter-Depot Routes). Existen soluciones para MDVRPI como por ejemplo en [26].

A su vez, el problema de varios depósitos y los recorridos particulares entre los depósitos genera una nueva gama de problemas donde la búsqueda de soluciones está directamente relacionada con MDVRP, como por ejemplo el Track and Trailer Vehicule Routing Problem (TTVRP) estudiado en [27], donde se presente su semejanza con MDVRP [24]. El problema de TTVRP presentado en [27] optimiza el recorrido para las exportaciones e importaciones así como el plan de rutas para camiones y tráiler dado un conjunto de clientes. Siendo necesario visitar un depósito, puerto y el cliente. En el caso de las exportaciones se inicia el recorrido desde el depósito pasando por el cliente a cagar la mercadería y luego al puerto. Para el caso de las importaciones, se parte del puerto, se descarga el contenedor en el cliente y se lleva el contenedor vacío al depósito. Estos problemas tienen dos particularidades implícitas, una es el manejo de ventanas de tiempo para los clientes y otra es que el objetivo final requiere encontrar la mejor ruta, así como la cantidad mínimas de camiones y los distintos tipos de camiones necesarios.

Otro aspecto a considerar en las publicaciones de MDVRP son las relacionadas a los objetivos finales de la solución. Como se vio anteriormente en las variantes de los problemas de ruteo, el objetivo final puede ser una variable o una función multi-objetivo. En la revisión de MDVRP de 2015 [24] se analizan a grandes rasgos las soluciones multi-objetivo de este problema, en donde se presentan múltiples variables de decisión para la solución final. Muchas veces estos objetivos pueden ser contradictorios como por ejemplo minimizar el número de vehículos y maximizar el nivel de servicios. Existen numerosas publicaciones sobre MDVRP multi-objetivo (MO-MDVRP) aunque su número es mucho menor a las publicaciones de Single Objetive MDVRP.

Según el análisis de 147 publicaciones de MDVRP publicado en [24], aproximadamente 12% Corresponden a MO-MDVRP y entre dichas publicaciones las funciones objetivos varían entre demanda, balanceo de carga de vehículos, número de vehículos, costo/distancia y otras. Centrándose en la mayoría de las publicaciones únicamente en el costo/distancia (un 80%).

Otro aspecto del artículo «A literature review on the vehicle routing problem with multiple depots» [24] es la descripcion de las distintos tipos de soluciones y los metodos de resolucion del problema de MDVRP.

## Métodos para la resolución de problemas de ruteo de vehículos

A continuación se analizaran distintos métodos de solución para los problemas MDVRP y sus variantes. A lo largo de la presentación de soluciones también se encontraran los problemas clásicos como TSP y VRP a modo de ejemplos de métodos de solución. La solución de estos dos tipos de problemas es la base para la solución de problemas más complejos de MDVRP y sus variantes. La variantes estudiadas tendrán en común las restricciones clásicas de los problemas de ruteo plateadas por Laporte et al 1996 [28].

1. Cada ruta comienza y termina en el mismo depósito.
2. Cada cliente es atendido exactamente una vez.
3. La demanda total de cada ruta no excede la capacidad del vehículo.
4. El costo total de la distribución es minimizado.

Cómo ya se mencionó, se ha demostrado que los problemas de ruteo de vehículos aquí descritos son problemas NP-Duros. Y por lo tanto, el enfoque más habitual para solucionar estos problemas es a través de heurísticas y meta-heurísticas ya que para números grandes de clientes el costo computacional de los métodos exactos es demasiado elevado.

A continuación se hace mención a los métodos desarrollados para encontrar la solución óptima (métodos exactos) en los problemas de ruteos de vehículos para luego pasar a revisar los métodos heurísticos.



### Métodos Exactos

G. Dantzig, R. Fulkerson, y S. Johonson [9] en el año 1954 abordan el problema de encontrar una solución óptima para TSP (Traveling Salesman Problem). Muestran que la cantidad de posibilidades para encontrar la solución óptima es finita. Para n ciudades las posibilidades son . En este estudio se delinea una manera de aproximarse al problema que, al menos, algunas veces permite encontrar el camino óptimo y además probar que el camino encontrado es el óptimo. Se concluye que el método mostrado es factible para encontrar soluciones óptimas, pero únicamente para un número moderado de ciudades. El problema plantado como ejemplo en dicho estudio consta de 49 ciudades.

En [19] R.V. Kulkarni plantea a los problemas de ruteo de vehículos como problemas de Programación Entera Lineal, se concluye que los métodos de solución de estos, aún no han sido suficientemente desarrollados como para abarcarlos en tiempos razonables de cómputo para cantidades grandes de clientes (o ciudades).

En el libro “Survey Combinatorial Optimization” [28], G. Laporte presenta un capítulo sobre algoritmos exactos para el problema de ruteo de vehículos. En el mismo realizan un sondeo sobre los algoritmos exactos existentes hasta el momento y nos muestran que al parecer todos caen en una de las siguientes categorías de tipos de algoritmo:

1. Búsqueda arborescente
2. Programación dinámica (DP)
3. Programación entera linean (ILP)

La última categoría es muy extensa y cuenta con el mayor esfuerzo de investigación en los últimos años. Se subdivide en tres subcategorías:

1. Formulación de particionamiento del conjunto
2. Formulación de flujo de vehículos
3. Formulación del flujo de mercancía

Entrar en detalle en cada uno de los métodos de solución exacta se aleja del propósito de este estado del arte. Por lo cual se limita únicamente a mencionar la existencia sobre el trabajo realizado al respecto. Haciendo énfasis que para grandes cantidades de clientes (o ciudades, puntos, etc.), los métodos exactos requieren demasiado procesamiento de cómputo, por lo cual el enfoque utilizado para encarar este tipo de problemas (sobre todo con grandes cantidades de clientes) es heurístico. Dicho enfoque aplica para todas las variantes de ruteo de vehículos. Tanto para TSP, VRP, MDVRP, y cualquiera de sus variantes dado que la complejidad de los problemas no parece disminuir en ningún caso.

En [29] se resolvió el problema de MDVRP con ventanas de tiempo y flota heterogénea para un conjunto de 20 clientes. En dicha publicación se resuelve el problema a nivel óptimo a través de programación lineal entera en un tiempo de 0.5 a 3 segundos, sirviendo estos como punto de comparación para futuras investigaciones en el campo de las heuristicas y meta-heuristicas.

### Métodos Heurísticos

A continuación se presenta una breve referencia a lo que se entiende por heurística y meta-heurística.

En [23] dice que las heurísticas (para soluciones de VRP) son procedimientos simples que realizan una exploración limitada del espacio de búsqueda y dan soluciones de calidad aceptable en tiempos de cálculo generalmente moderados. A su vez, se presentan a las meta heurísticas como procedimientos genéricos de exploración del espacio de soluciones para problemas de optimización y búsqueda.

En general las meta heurísticas obtienen mejores resultados que las heurísticas clásicas, pero incurriendo en mayores tiempos de ejecución.

En [30] los autores F. Glover y G. A. Kochenberg, introducen al lector en el libro indicando que:

*“Las Meta heurísticas son métodos de solución que orquestan una interacción entre procedimientos de mejora local y estrategias a un nivel superior para crear procesos capaces de escapar de óptimos locales y de esta forma realizar una búsqueda más robusta del espacio de soluciones.”*

### Heurísticas para VRP

Si bien el objetivo de este “estado del arte” es el problema MDVRP (Multi Depot Vehicle Routing Problem), sería imposible abarcar el problema del mismo sin antes hacer un sondeo de los métodos de solución para problemas con un único depósito (VRP). Por lo cual, a continuación, sin entrar en demasiado detalle, se presentan las heurísticas que se han desarrollado para VRP. La cantidad de heurísticas existentes para este problema es bastante extensa, por lo cual nos vamos a centrar en las clásicas.

#### Algoritmo de Ahorros de Clarke and Wright

A continuación se muestra la idea general de dicho algoritmo basándose en los apuntes de J. Lysgaard [31] el cuál proporciona además un ejemplo sobre el mismo. La idea es la siguiente:

El concepto básico del “ahorro” expresa el costo ahorrado obtenido por juntar dos rutas en una misma ruta como se ilustra en la imagen a continuación:

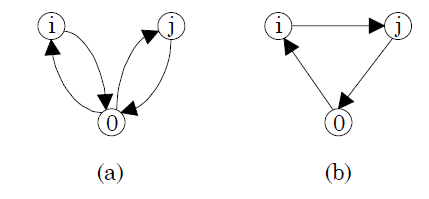


Figura 1.4

Inicialmente los consumidores y son visitados en rutas separadas (a). Una alternativa a esto es visitar a los dos clientes en la misma ruta, por ejemplo como lo ilustrado en (b). El ahorro por hacer esto puede ser calculado. Denotando el costo de transporte entre y con , el costo total de transporte , en la *figura 1.4 (a)*, es:

De la misma forma, el costo de transporte en la *figura 1.4 (b)* es:

Al combinar las dos rutas, se obtiene el siguiente ahorro :

Los mayores valores indicarán que la unión de esas rutas es más atractiva en comparación con otras de menor ahorro. También se deberán verificar las restricciones del problema como la capacidad del vehículo.

Con esta idea básica sobre el ahorro al unir rutas es que se forma el algoritmo.

Existen distintas variantes y extensiones a la versión básica del algoritmo de ahorros. Como por ejemplo, se puede distinguir entre la versión secuencial y la versión paralela. En la secuencial se construye de a una ruta a la vez, en cambio en la versión paralela se van construyendo varias ruta al mismo tiempo. Además se encuentra la versión del algoritmo basada en matching. Por las mismas y otras extensiones del algoritmo se sugiere consultar [23].

#### Heurísticas de Inserción

Como se indica en [23], las heurísticas de inserción son métodos constructivos en los cuales se crea una solución mediante sucesivas inserciones de clientes en las rutas. En cada iteración se tiene una solución parcial cuyas rutas sólo visitan un subconjunto de los clientes y se selecciona un cliente no visitado para insertar en dicha solución. En las heurísticas de inserción secuencial sólo se considera insertar clientes en la última ruta creada. La principal desventaja de este enfoque es que los últimos clientes no visitados tienden a estar dispersos y por lo tanto las últimas rutas construidas son de costo muy elevado. Las heurísticas de inserción en paralelo surgen para remediar esta deficiencia, permitiendo insertar un cliente en cualquiera de las rutas de la solución. Esta distinción en similar a la hecha para las dos versiones del Algoritmo de Ahorros de Clarke and Wright.

Existen varias heurísticas del tipo de Inserción. A continuación se muestra a modo de ejemplo la idea general de las heurística de **Inserción Secuencial de Mole & Jameson** [32]

##### Inserción Secuencial de Mole & Jameson

El algoritmo puede ser pensado en términos de repetir una secuencia de tres pasos. En el primer paso se determina el lugar más ventajoso en la ruta emergente para cada cliente que aún no está asignado. Se utiliza el criterio del costo que se agrega al insertar al cliente entre dos clientes ya pertenecientes a la ruta. Cuando el costo es el mínimo se determina que esa es la posición más ventajosa. Se entiende como ruta emergente a la ruta que se está construyendo en esta etapa del algoritmo.

En el segundo paso se identifica al siguiente cliente no asignado el cuál se va a asignar a la ruta emergente. Aquí se podría utilizar el que implica un costo mínimo también, pero además, generando incentivos adicionales para los clientes que se encuentran más dispersos. De esta forma se evita dejar para el final los clientes lejanos al depósito lo cual podría resultar en rutas ineficientes.

En el tercer paso se explora si alguna reasignación del orden de los clientes en la ruta es más eficiente. Luego se verá algo de optimización en este estado del arte.

##### Heurísticas de Inserción para VRPTW (Solomon)

Solomon en [33] publica un diseño y análisis de heurísticas para problemas de ruteo con restricciones de ventanas de tiempo. Ahí se pueden encontrar varias adaptaciones de heurísticas para VRP para contemplar las restricciones de ventanas de tiempo. Una de ellas es una adaptación de la versión paralela del algoritmo de ahorros de Clarke and Wright. No se va a entrar en detalle sobre estos tipos de heurísticas. Por más información de heurísticas para VRPTW consultar [33]

#### Heurísticas de dos fases

En [23] se pueden encontrar las siguientes familias de estrategias para heurísticas para VRP. Asignar Primero – Rutear Después por un lado y Rutear Primero – Asignar Después por otro. El termina “Asignar” refiere al hecho de asignar un cliente a una ruta. Y “Rutear” como el nombre lo indica es establecer la ruta.

Como se indica en [23], en el caso Asignar Primero – Rutear Después se busca generar grupos de clientes, también llamados clusters, que estarán en una misma ruta en la solución final. Luego, para cada cluster se crea una ruta que visite todos los clientes. Las restricciones de capacidad son consideradas en la primera etapa, asegurando que la demanda total de cada cluster no supera la capacidad del vehículo. Por lo tanto, construir las rutas para cada cluster es un TSP. Un ejemplo de esta estrategia es la heurística de Barrido o Sweep, en la cual los clusters se forman girando una semirrecta con origen en el depósito e incorporando los clientes “barridos” por dicha semirrecta hasta que se viole la restricción de capacidad. Luego cada cluster es ruteado resolviendo un TSP.

En este caso solo se tienen en cuenta restricciones de capacidad de los vehículos. Sería distinto el algoritmo con más restricciones, como por ejemplo con la restricción de que el largo total de una ruta no puede ser mayor a un valor predeterminado.

En las heurísticas de Rutear Primero-Asignar Después, lo primero que se hace, como lo dice el nombre de la estrategia, es calcular una ruta que visite a todos los clientes. Para hacer esto basta con resolver un problema TSP. En general esta ruta no va a respetar las restricciones del problema y se debe partir en varias rutas, cada una de las cuales, respetaría las restricciones [23].

### Heurìsticas para MDVRP

Se han propuesto numerosas heurísticas para el problema de enrutamiento de vehículos con múltiples depósitos en los últimos años [24], este tema se abordó a inicios de la década del 90´ y fueron estudiados MDVRP así como sus variantes con flotas heterogéneas, periodicidad, ventanas de tiempo, etc. A continuación se detalla una heurística de dos fases aplicada a MDVRP; existiendo además otras heurísticas como se describe en la revisión de publicaciones de Montoya [24].

#### Heurísticas de dos Fases para MDVRP y sus variantes

Un enfoque estudiado en [34] por Libertad Tansini y Omar Viera, utiliza una heurística de dos fases donde la primera fase es de asignación de clientes a depósitos (zonificación, asignación, clusterización) y la segunda es la resolución de rutas para cada depósito. Es necesario destacar que aunque la heurística es del tipo de dos fases no es igual a la estudiada anteriormente para el caso de VRP ya que la etapa de asignación es de clientes a clústeres (zonas) y no de clientes a rutas.

##### Etapa de asignación

Para la asignación de clientes a zonas, en la publicación [34] se comparan los resultados obtenidos por los distintos algoritmos. A continuación se describen algoritmos de asignación por urgencia, barrido, cíclica y por zona. Luego se incluirá la variación de ventanas de tiempo [35] para el caso de asignación por urgencias y asignación por zonas.

##### Asignación a través de urgencias

La urgencia o prioridad que tienen los clientes determina la forma de asignarlos. Un cliente con más urgencia se asigna primero. La urgencia es una manera de definir una relación de precedencia entre los clientes.

A continuación se mencionan dos formas de asignar a través de urgencias, asignación en paralelo y asignación simplificada.

##### Asignación en paralelo

En esta asignación la urgencia para cada cliente se calcula teniendo en cuenta todos los depósitos al mismo tiempo.

Se calcula como:

Donde es la distancia entre el cliente y el depósito , es el conjunto de depósitos y es la distancia entre el cliente y el depósito más cercano . El cliente con mayor valor de será asignado al depósito más cercano.

Esta heurística compara el costo de la asignación de un cliente para su depósito más cercano con el costo de asignar el cliente a cualquier otro depósito.

##### Asignación simplificada

En esta heurística sólo dos depósitos están implicados en la evaluación de la urgencia de cada cliente mediante el cálculo:

Donde es la distancia entre el cliente y el segundo depósito más cercano y es la distancia entre el cliente y el depósito más cercano . El cliente con mayor valor de será asignado al depósito más cercano.

Esta heurística compara el costo de asignar un cliente al depósito más cercano con el costo de asignar a un cliente con el segundo depósito más cercano.

##### Barrido de asignación

En esta heurística, los clientes son “barridos” en la dirección del depósito con mayor demanda insatisfecha.

En primer lugar, es necesario determinar un depósito \* con mayor demanda insatisfecha. La evaluación de la urgencia se realiza con el siguiente cálculo:

La urgencia se mide como la diferencia entre la asignación de un cliente para su depósito más cercano y el depósito con mayor demanda insatisfecha, por lo tanto un valor elevado en la urgencia significa que debería ser asignado al depósito más cercano.

##### Asignación Cíclica

El procedimiento consiste en asignar de una manera cíclica, un cliente a la vez. En primer lugar, el algoritmo asigna a cada depósito el cliente más cercano. Luego se asigna a cada depósito, el cliente más cercano al último cliente asignado a dicho depósito. En general, la asignación es muy pobre para los últimos clientes asignados.

##### Asignación por Zona

Una zona está definida por un depósito y los clientes asignados a él. El algoritmo trata de construir un grupo compacto de clientes para cada depósito. Se analizan dos algoritmos para la resolución de dicho problema. [23]

##### Propagación de coeficientes

Para esta forma se definen coeficientes de atracción para los depósitos y para los clientes ya asignados a depósitos. La asignación se hace en función de una distancia escalada (definida a continuación) con los clientes no asignados. El siguiente cliente a asignar es el que minimiza la distancia escalada, siendo la fórmula de la distancia escalada la siguiente:

El cálculo del coeficiente de atracción de *c* que fue asignado por su cercanía al cliente *c´* es el siguiente:

El coeficiente de degradación es valorado arbitrariamente para todos los depósitos y clientes (ejemplo 0.5). Los nuevos clientes asignados tienen coeficientes más altos por lo cual tienen menos atracción.

##### Zonificación por 3 criterios

El procedimiento utilizado por este algoritmo para incluir clientes en un clúster es en base a la distancias promedio a los clústeres, la varianza de la distancia a los clientes en los clústeres y distancia al cliente más cercano en cada clúster. Se aplican cotas que para cada uno de estos 3 criterios, aplicando el segundo si el primero no llega a la cota y el tercero si el segundo no llega a la cota. Por ejemplo, para un cliente y un clúster, dichas cotas se establecen comparando el porcentaje de mejora con respecto a este cliente con los otros clúster.

##### Métodos de Asignación con ventanas de tiempo.

En [35] se modifican los métodos de Asignación por zona para considerar ventanas de tiempo. De este modo los clientes con ventanas de tiempo similares son agrupados en la misma zona (clúster).

En dicha publicación se definen los conceptos de *Afinidad* y *Cercanía* para luego presentar el criterio de asignación paralela simplificada (dentro de la categoría Asignación por urgencia). Finalmente se presenta la asignación por zonificación de 3 criterios (dentro de la categoría Asignación por Zona) la cual se calcula en función de la distancia *Angulo*.

Definiciones previas.

***Afinidad (i,d)* =**

Donde C son los Clientes y D son los Depósitos

*DTW(i,j)* mide la distancia en la ventana de tiempo entre i y j (siendo i y j clientes o depósitos). Para el cliente *i*, y representan el inicio y fin de la ventana de tiempo.

*DTW(i,j) =*

*TV(i,j)* es el tiempo de viajar de i a j.

***Cercania(i,j) = d(i,j) / Afinidad (i,d).***

*d(i,j)* es la distancia entre i y j.

***Angulo (i,j)*** =

Donde *x* e *y* son las coordenadas, *l* y *e* es el inicio y fin de la ventana de tiempo.

##### Asignación por urgencia con ventanas de tiempo, Asignación paralela simplificada.

Ecuación de Urgencias, solo dos depósitos son incluidos en esta ecuación.

La variable c es un cliente, es el depósito más cercano a c y es el segundo depósito más cercano.

##### Asignación por Zona con ventanas de tiempo, Zonificación por 3 criterios.

Para este caso la distancia es en base a distancia *Angulo* definida anteriormente. La misma parte de la comparación de vectores al explorar los algoritmos de “clustering” [35].

Los tres criterios son el distancia *Angulo* promedio al clúster, varianza de la distancia *Angulo* promedio al clúster y distancia *Angulo* al cliente más cercano del clúster.

En la publicación antes mencionada se ejecutaron distintas corridas de este algoritmo en donde se observa que el cálculo en base a distancia *Angulo* es mejor que el cálculo en base a la distancia en un mapa o euclidiana.

##### Heurística de asignación Hibrida para depósitos con capacidad limitada.

En [25] se presenta una heurística hibrida entre la asignación Simplificada (Heurística de Urgencia) y zonificación por tres criterios (Asignación por zona).

Para esto se parte de que cada cliente pertenece a uno de estos conjuntos.

1) CSA, clientes no asignados a ningún depósito.

2) , conjunto de clientes asignados al depósito .

Por otro lado la asignación de un cliente a un depósito es factible si:

1) El depósito puede cumplir la demanda del cliente.

2) El cliente pertenece CSA

3) Si cumple la condición de urgencia calculada para cada cliente *CEj*.

Para esta heurística, es la minimización de una función que toma como parámetros el costo de asignar el cliente al depósito más cercano en comparación con el próximo más cercano. El parámetro es calculado para todos los clientes *CEj*. Como muestra la siguiente ecuación.

Si después de la evaluación de la función es positivo se asigna al deposito , sino se asigna al que haga mínima .

##### Etapa de Construcción de rutas para cada depósito

Una vez que los clientes han sido asignados al depósito, fin de la etapa de asignación, se procede a realizar el enrutamiento. Para la resolución se utiliza cualquier algoritmo de ruteo, los clientes están asignados a los depósitos, ahora el problema de ruteo de MDVRP se puede ver como problemas de ruteo de VRP, donde se utiliza un algoritmo de resolución de VRP para cada uno de los depósitos. En [34] se utilizó el algoritmo de Clarke and Wright.

En otras publicaciones se crearon nuevas formas de resolver este problema a través de meta-heurísticas las cuales explicaremos en la sección 5.5. La heurística de asignación hibrida vista en la sección de zonificación fue utilizada para generar la solución inicial, luego de esto en [25] se presenta la Meta-heurística MOSS (Multi Objetive Scatter Search) en donde a partir de un conjunto de soluciones aleatorias, y a través de selecciones sistemáticas y estratégicas se obtiene una mejor solución creando las rutas y optimizando la asignación de clientes a depósitos de la etapa de zonificación. La construcción del conjunto de referencia para el MOSS y metodologías aplicadas se encuentra detallada en [22].

### Meta-Heurísticas para VRP

Siguiendo con la referencia de [23], se presenta a continuación meta heurísticas para la resolución de problemas VRP. Algoritmos de hormigas, Búsquedas Tabú y Algoritmos Genéticos son meta heurísticas representantes de tres paradigmas diferentes. Los Algoritmos de Hormigas son procedimientos basados en agentes que utilizan métodos constructivos aleatorizados y cooperan entre si compartiendo información. Los algoritmos de búsqueda Tabú son métodos de búsqueda local que aceptan empeorar las soluciones para escapar de los óptimos locales. Los Algoritmos Genéticos se basan en mantener un conjunto de soluciones lo suficientemente diverso como para cubrir gran parte del espacio de soluciones.

### Meta–Heurísticas para MDVRP

A continuación se muestran algunas meta-heurísticas desarrolladas para resolver problemas de ruteos de vehículos con varios depósitos.

#### Particle Swarm Optimization (PSO)

Lei Wen y Fanhua Meng, publicaron en el 2008 el artículo “An Improved PSO for the Multi-Depot Vehicle Routing Problem with Time Windows”. [12]

En el mismo explican el algoritmo PSO utilizado. Particle Swarm Optimization. En castellano, Optimización por Enjambre de Partículas es un algoritmo de optimización basado en la teoría de partículas. La idea principal de PSO es modelar el vuelo de un enjambre de pájaros (partículas) alrededor de una cumbre. El estado de una partícula en el espacio de búsqueda está definido por dos factores: posición y velocidad. La posición y velocidad de la partícula i en un espacio de búsqueda de d-dimensiones puede ser representado como y respectivamente. Cada partícula conoce su mejor posición que obtuvo en su camino recorrido. Y también conoce la mejor posición global, denotado como que es la mejor posición que obtuvo alguna partícula del enjambre. En cada iteración se calcula una nueva velocidad para cada partícula.

En donde y son constantes llamadas coeficientes de aceleración, es llamada peso del factor de inercia, y son dos números aleatorios distribuidos uniformemente en el rango [0,1]. Por lo tanto, la posición de cada partícula es actualizada en cada iteración de acuerdo a la siguiente ecuación:

En el caso de utilizar la meta-heurística PSO en el problema MDVRPTW, las posiciones de las partículas son las soluciones candidatas para el ruteo. En cada posición se calcula el costo total para poder compararlas y de esta forma saber cuál es la mejor posición conocida de cada partícula y la mejor posición global.

El primer paso en la solución planteada es encontrar una solución factible inicial. Para esto se asignan los clientes al depósito más cercano. Luego para cada depósito se busca una solución para el ruteo utilizando al algoritmo de ahorros de Clarke and Wright. Luego se establece la posición inicial para un conjunto de partículas variando aleatoriamente la solución factible recién encontrada. Para esto se cambian aleatoriamente algunos clientes de depósito y se asignan a nuevas rutas. Luego de esta etapa inicial comienza una etapa iterativa en donde las partículas van buscando el óptimo en el espacio de soluciones.

Para esto es necesario codificar las soluciones de ruteo en posiciones de partículas en un espacio multidimensional. También es necesario tener en cuentas que el algoritmo PSO aplica en espacios continuos, cuando las soluciones de ruteo están en un espacio discreto. No se entra en detalle en este documento de cómo se codifican las partículas y de cómo se calcula la actualización de la posición de la partícula en cada iteración.

El objetivo es mostrar cómo es que la meta-heurística PSO explora el espacio de soluciones.

#### Tabu Serch

En [36] “A Tabú Search Heuristic For The Multi-Depot Vehicle Routing Problem, la estrategia que utiliza la meta-heurística Tabú Search también comienza a partir de una solución inicial. Para esto se asignan los clientes al depósito más cercano y luego se rutean los vehículos utilizando una heurística para VRP.

Luego que se tiene la solución inicial, Tabú Search se basa en un algoritmo llamado FIND. El mismo consiste en tres fases: Fast Improvement, INtensification, y Diversification.

Cada una de estas fases se basa en alguno, o todos, de los siguientes procedimientos básicos:

1-route: Se utiliza para post-optimizar una ruta simple de vehículos. Se basa en aplicar el algoritmo 4-opt [37]. Este algoritmo de mejora parte de una ruta inicial y prueba todas las combinaciones posibles que surgen de cambiar 4 aristas del grafo a la vez (viendo a la ruta como un grafo con vértices y aristas donde el vértice representa al cliente y la arista el camino entre dos clientes). De esta forma se busca el óptimo local en el espacio de soluciones.

2-route: Se utiliza para buscar mejoras en la solución moviendo clientes provenientes de dos rutas distintas. Podrían ser rutas asignadas a depósitos distintos.

3-route: Similar a 2-route pero se intercambian clientes provenientes de 3 rutas.

Al aplicar FIND, la mejor solución encontrada y su valor es recordado, pero como se permite deteriorar la función objetivo, la solución tenida en cuenta en el algoritmo en cada iteración no es necesariamente la mejor solución. Cuando un cliente es movido de su ruta actual, moverlo nuevamente a su ruta original es declarado un movimiento tabú por un número aleatorio de iteraciones.

A continuación se explican las tres fases de FIND:

Fast Improvement: En esta fase, el algoritmo intenta mejorar aplicando repetidamente los siguientes pasos:

* Inter-depot: aplicar intercambios 2-route entre rutas de dos diferentes depósitos.
* Intra-depot: aplicar intercambios 2-route entre rutas del mismo depósito.
* 3-route: intercambiar clientes entre tres rutas.

La secuencia es repetida hasta que no hay mejoras por iteraciones consecutivas. En cada uno de estos tres pasos, cualquier movida que encuentra una mejora, es implementada inmediatamente como la solución actual a tomar en cuenta. En otro caso, la mejor solución no-tabú que deteriora la función objetivo, es implementada. Cuando se implementa un movimiento de clientes entre rutas, se aplica 1-route a todas las rutas involucradas en el movimiento.

Intensification:

Esta fase intensifica la búsqueda de rutas mejores, comenzando con la mejor solución conocida y trabajando con un único depósito a la vez. Aplica iteraciones de intra-depot a las rutas de cada depósito.

Diversification:

Esta fase aplica una combinación Inter-depot e Intra-depot, moviendo clientes entre distintos depósitos, con el objetivo de realizar una exploración más extensa del espacio de soluciones.

## Post Optimización y mejoras

Una vez que se tiene una solución para el problema, se puede intentar mejorarla mediante algún procedimiento de búsqueda local. Para cada solución se define un conjunto de soluciones vecinas . Un procedimiento de Búsqueda Local parte de una solución , la reemplaza por una solución de menor costo y repite el procedimiento hasta que la solución no pueda ser mejorada. Al terminar, se obtiene una solución localmente óptima respecto a la definición de la vecindad. Para obtener puede buscarse la mejor solución de o tomar la primera solución de que mejore el costo.

Usualmente se define como las soluciones que pueden obtenerse aplicando a alguna regla o procedimiento sencillo llamado movida. Las movidas para el VRP pueden clasificarse en movidas de una ruta y movidas multi-ruta. En las movidas de una ruta los clientes que se visitan en una ruta no varían luego de la aplicación del operador, lo que varía es el orden en que se realizan las visitas. En las movidas multi-ruta, además de cambios en el orden de las visitas suele modificarse el conjunto de clientes visitados en cada ruta.








### Operador λ–Intercambio

El λ-intercambio definido por Lin [38] consiste en eliminar λ arcos (camino entre dos clientes consecutivos) de la solución (λ > 1) y reconectar los λ segmentos restantes. Una solución se dice λ-óptima si no puede ser mejorada utilizando λ-intercambios. Se llama λ-opt a un algoritmo de búsqueda local que utiliza λ-intercambios hasta alcanzar una solución λ-óptima.

El algoritmo λ–opt divide la ruta en λ segmentos y los reordena de todas las formas posibles para obtener nuevas soluciones que sean mejores. La división en segmentos también se hace de todas las formas posibles. La combinación de todas las posibles divisiones en segmentos con todas las posibles reordenaciones de los segmentos da lugar a una gran cantidad de nuevas soluciones. En cada paso del algoritmo, se procede a generar todas las nuevas soluciones de forma sistemática. Durante la generación, el algoritmo puede presentar dos tipos de comportamiento: cuando encuentre una solución mejor que la actual, la devuelve; o espera hasta haber explorado todas las nuevas soluciones y devuelve la mejor de ellas [39].

Este algoritmo es en el peor caso, siendo el número de clientes. Usualmente se implementan 2-intercambios y 3-intercambios. Posibles movidas de este tipo se muestra en la Figura 1.5

|  |  |
| --- | --- |
| 1. Ejemplo de 2-intercambios | 1. Ejemplo de 3-intercambios |

Figura 1.5 Ejemplo de 2-intercambios

### Operador Or–opt

Una versión reducida del algoritmo 3-opt es el algoritmo Or-opt [40], que consiste en eliminar una secuencia de clientes consecutivos de la ruta y colocarlos en otra posición de la ruta, de modo que permanezcan consecutivos y en el mismo orden. Primero se realizan las movidas con , luego con y finalmente con . Si una ruta visita clientes existen de estas movidas. En la Figura 1.6 se muestra un ejemplo del algoritmo Or-opt.

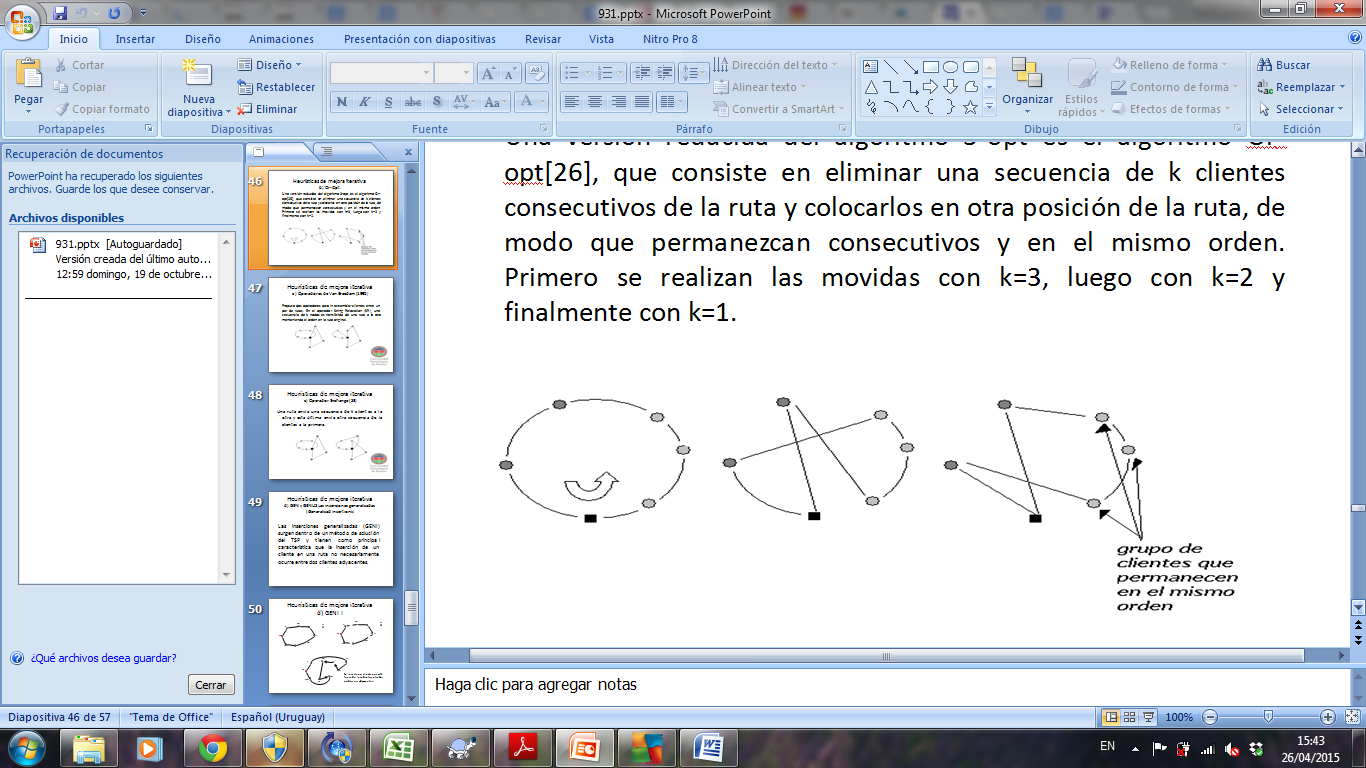


Figura 1.6 Ejemplo de algoritmo Or-opt

### Operadores de Van Breedam

Van Breedam [41] propuso dos operadores para intercambiar clientes entre un par de rutas:

* Operador String Relocation (SR): Una secuencia de clientes es transferida de una ruta a la otra manteniendo el orden en la ruta original.
* Operador Exchange (SE): Una ruta envía una secuencia de clientes a la otra y esta última envía otra secuencia de clientes a la primera.

En la figura 1.7 se muestran la aplicación de los operadores String Relocation y String Exchange.

|  |  |
| --- | --- |
| 1. Operador String Relocation | 1. Operador String Exchange |

Figura 1.7 Operadores String Relocation y String Exchange

### GENI y GENIUS

#### GENI

Las inserciones generalizadas [42] (GENI) tienen como principal característica que la inserción de un cliente en una ruta no necesariamente ocurre entre dos clientes adyacentes. Sin embargo, después de la inserción estos dos clientes se convierten en adyacentes a . Si es un cliente de la ruta, denotamos a su sucesor y a su predecesor. La inserción de un cliente entre los clientes y (no necesariamente consecutivos) se puede realizar de dos formas, se ilustran en la Figura 1.8

|  |  |
| --- | --- |
| 1. Inserción GENI de Tipo I | 1. Inserción GENI de Tipo I |

Figura 1.8 Inserción GENI (GENeralizad Insertions)

##### Inserción Tipo I

Se considera un cliente en el camino de a ( y ). La inserción consiste en eliminar los arcos , y y agregar los arcos , , y . Es necesario además, invertir el sentido de los caminos y . Una inserción de este tipo se muestra en la Figura 1.8 (a).

##### Inserción Tipo II

Se debe seleccionar en el camino de a de modo que y y además en el camino de a tal que y . La inserción consiste en eliminar los arcos , , y , y agregar los arcos , , , y . Además, debe revertirse el orden de los caminos y . Una inserción de este tipo se muestra en la Figura 1.8 (b).

#### GENIUS

El algoritmo de post-optimización GENIUS [42] comienza considerando el primer cliente de la ruta: se le elimina mediante Unstringing y se le vuelve a insertar utilizando Stringing.

El procedimiento Stringing consiste en realizar una inserción de GENI de Tipo I o Tipo II con *p-vecindades* y el Unstringing es el inverso de una inserción GENI de Tipo I o Tipo II. La *p-vecindad* del cliente se define como el conjunto de los clientes de la ruta más cercanos a según los costos del problema.

El proceso de eliminar mediante Unstringing y volver a insertar utilizando Stringing puede incrementar el costo de la solución. Si la solución es mejorada, se repite el proceso con el segundo cliente de la nueva ruta. El proceso termina luego de eliminar e insertar el último cliente de la ruta.

## Soluciones de software existentes para MDVRP

En Febrero de 2014 el sitio “OR/MS Today” público un informe donde se provee la información de los programas existentes para el problema de enrutamiento de vehículos [43]. El mismo es extraído de un cuestionario a los distintos proveedores. La encuesta está dividida en 17 secciones que abarcan las especificaciones técnicas de dichos programas. En la tabla 1.1 se puede observar las distintas soluciones de software existentes para los problemas de MDVRP:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Producto | Vendedor | Año | Plataformas que soporta | | | | |
| Windows | iOS | Android | Software basado en Web como servicio | Otros |
| ArcGIS for TransportationAnalytics | Esri | 2012 | Si | Si | Si | Si |  |
| Descartes Routing, Mobile & Telematics | Descartes | 1995 | Si | No | Si | Si |  |
| DISC | MJC2 | 1990 | Si | Si | Si | Si | UNIX & Linux |
| eRoute Logistics | WM Logistics | 2002 | Si | No | Si | Si |  |
| JOpt.SDK | DNA Evolutions GmbH | 2006 | Si | No | No | Si |  |
| Logistics Optimizer | Runzheimer International | 2005 | No | No | No | Si |  |
| Logvrp | Netakil | 2010 | Si | Si | Si | Si | REST + SOAP Web Service API |
| Optrak vehicle routing software | Optrak Distribution Software Ltd | 1992 | Si | No | No | Si |  |
| ORTEC Routing and Dispatch | ORTECÂ | 2003 | Si | Si | Si | Si |  |
| Paragon Routing and Scheduling Optimizer | Paragon Software Systems, Inc | 1997 | Si | No | No | Si |  |
| Roadnet Transportation Suite | Roadnet Technologies | 1983 | Si | Si | Si | Si |  |
| Routist.com | KKT srl | 2013 | No | No | No | Si |  |
| StreetSync Basic | RouteSolutions | 2008 | Si | No | No | Si |  |
| StreetSync Desktop | RouteSolutions | 2005 | Si | No | No | No |  |
| StreetSync Pro | RouteSolutions | 2011 | Si | No | No | Si |  |
| TMW Appian DirectRoute | TMW Systems | 1996 | Si | No | No | Si |  |
| TruckStops | MapMechanics | 1983 | Si | No | No | No | Server or PC |

Tabla 1.1 Distintas soluciones de software existentes para los problemas de MDVRP

En la tabla 1.1 se puede observar que la gran mayoría del software relevado se desarrolló en los últimos años, eso reafirma el aumento en la cantidad de publicaciones sobre los problemas de ruteo de vehículos.

Las distintas plataformas que soporta el software relevado se puede ver que casi el 100% es basado en Web services. Las ventajas de contar con este tipo de servicio es que no se encuentra atado a ningún sistema operativo ni ningún lenguaje de programación y se encuentra disponible a través de Internet o Intranet.

La gran mayoría del software puede ejecutarse en Windows (en sus diferentes versiones) y actualmente han surgido software para Android e iOS como por ejemplo ArcGIS for TransportationAnalytics de ESRI. Otros sistemas que ya tienen sus años se han actualizado para poder soportar los nuevos sistemas operativos como por ejemplo Roadnet Transportation Suite de Roadnet Technologies.

Dentro de los nuevos sistemas operativos, Android es el que soporta más software en comparación con iOS.

También hay otras plataformas que soportan software de ruteo de vehículos como UNIX & Linux se puede ejecutar el software DISC. El software logvrp soporta REST + SOAP Web Service API, donde son los servicios SOAP son Web services que estandarizan su información a través de mensajes XML, en formato SOAP. Los servicios REST son Web services que estandarizan su información a través de mensajes XML, en formato SOAP.

# Capítulo 3

# Definición del problema

El problema planteado para este trabajo es el de MDVRP (Mullti Depot Vehicle Routing Problem) para un número elevado de clientes y depósitos con capacidad limitada. El problema de MDVRP como se definió en la sección anterior consiste básicamente en optimizar la elección de las rutas que deben realizar los vehículos para los depósitos de tal forma que satisfagan la demanda de los clientes. Cumpliendo con las siguientes características.

* Cada cliente es atendido por un único depósito.
* Los vehículos tienen una capacidad limitada para el transporte de la mercadería.
* Los vehículos comienzan y terminan su ruta en el mismo depósito.
* El cliente recibe una única visita de un vehículo de la flota.
* Los vehículos de la flota tienen la misma capacidad.
* Los depósitos tienen capacidad.

En la sección de Estado del Arte se encuentra el modelo del problema de MDVRP.

En el proyecto de grado se planteó la investigación del problema de MDVRP. Luego de la etapa de elaboración del Estado del Arte, se inició la implementación de una aplicación y algoritmos que solucionen el problema a partir de la información analizada en la etapa de investigación. Finalmente se sugirió crear distintas algoritmos que permitan mejorar la solución.



## Características buscadas de la aplicación

Las características buscadas para la aplicación fueron mayoritariamente sugeridas por los tutores del proyecto mientras que otras fueron agregadas para brindar un producto final lo más sólidos posible.

Entre los requerimientos y sugerencias podemos destacar los siguientes puntos a tener en cuenta:

* La entrada de datos corresponde con los formatos de TSPLib, considerando todo los formatos presentes en la implementación de la misma.
* Es de sumo interés permitir generar un entorno amigable para agregar o quitar distintas implementación de los algoritmos.
* Los algoritmos y soluciones deben ser parametrizables permitiendo restringir el número de iteraciones, tiempo de corrida o rango de mejoras.
* Se debe manejar una holgura en los depósitos que permita cierta flexibilidad en la capacidad máxima del depósito (oferta del depósito).
* Es sugerido contar un entorno grafico para la visualización de los clientes, depósitos y rutas luego de ejecutar un algoritmo. Fue recomendación de los tutores que los distintos depósitos y los clientes abastecidos por el deposito fueran destacados en una zona; delimitando su frontera y resaltados en un color particular. Se puede seleccionar un depósito y sus clientes, solo un cliente o solo un depósito y esta selección será resaltada en forma gráfica.
* La representación grafica de la solución permite visualizar los cambios en cada mejora o iteración y comparar distintas corridas de los algoritmos o de distintos algoritmos. Por otro lado, dado el gran número de clientes y depósitos es deseable poder hacer un zoom sobre una sección de la pantalla.

## Algoritmos de MDVRP propuestos para la solución.

Analizando el problema con los tutores y basado principalmente en las heuristicas de dos fases para la resoulucion de MDVRP se llegó a identificar 4 posibles etapas en la resolucion de este tipo de problemas: Asignación, Mejora de la asignación, Ruteo, Post-optimización.

Para la resolución de MDVRP se implementaron algoritmos basados en conocidos algoritmos de asignación, ruteo y post-optimización los cuales se encuentras explicados en la sección Estado del Arte. También se crean algoritmos de mejora de asignación.

Los algoritmos utilizados son:

* Asignación
  + Asignación por Urgencia (no se considera la capacidad de los depósitos)
  + Asignación por Urgencia (Fase 1, se considera la capacidad de los depósitos)
* Mejora de la asignación (Fase 2)
  + Algoritmo Enajenado Rápido (AER)
  + Algoritmo Enajenado Lento (AEL).
* Ruteo
  + Clarke & Wright.
* Post-optimización.
  + λ-intercambio
  + R-iopt

### Asignacion Por Urgencia sin Capacidad

Este algorimto asinga cliente a los depósitos más cercanos sin tener en cuenta la capacidad de los depósitos. Por lo cual estos pueden quedar sobrecargados. Se implementó para tener una referencia y poder comparar con los algoritmos que sí tienen en cuenta la capacidad de los depósitos.

### Asignación por Urgencia con Capacidades (Fase 1)

Esta es la implementación de la heurística *Asignación a través de Urgencias (en paralelo)* vista en el Estado del Arte en el título 2.4.4.1 - 1.1.1 con una pequeña mejora agregada. (comprobado según los resultados los cuales en la mayoría de las pruebas demostraron que la mejora agregada obtiene una mejora en los costos al rutear)

Según el estado del arte, la asignación a través de urgencias en paralelo propone calcular la urgencia para cada cliente según:

Donde es la distancia entre el cliente y el depósito , es el conjunto de depósitos y es la distancia entre el cliente y el depósito más cercano .

El cliente con mayor valor de será asignado al depósito más cercano. A medida que los depósitos se van quedando sin lugar, los clientes se irían agregando al depósito más cercano con lugar disponible.

La mejora agregada implica en recalcular el para todo los clientes luego de que se asigna un cliente a un depósito. En este caso, al recalcular el para un cliente, solo se toman en cuenta únicamente los depósitos con capacidad suficiente como para servirlo. Tanto como para encontrar al depósito más cercano, como para calcular la sumatoria de distancias a los demás depósitos. O sea, que si los depósitos no tienen suficiente espacio como para servir al cliente, no se toman en cuenta.

Se ha podido comprobar que esto implica una mejora en la heurística en una gran cantidad de casos. Como desventaja el tiempo de ejecución podría ser mayor. Depende de la holgura (capacidad total de los depóstido- demanda total de los clientes), ya que cuando no se aplica la mejora, se puede ahorrar el recalculo del depósito más cercano cuando la holgura total es mayor. Por lo tanto, con grandes cantidades de clientes, al aplicar la mejora, el tiempo de ejcución podría llegar a crecer exponencialmente. Ver anexo Comparación de dos Heurísticas.

### Algoritmo Enajenado Lento (AEL). Fase 2

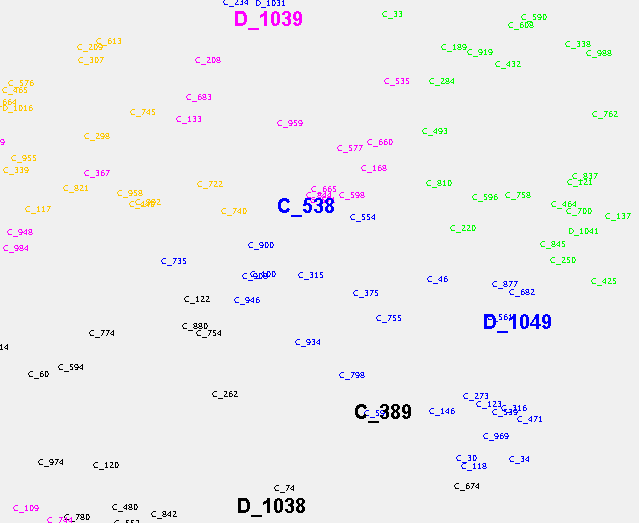
Esta es una heurística que, a partir de una solución inicial (Fase 1), busca mejoras en la asignación de clientes a depósitos. En comparación al algoritmo Fase 2 -Rápido (AER), este algoritmo en cada iteración realiza un cálculo del costo de las rutas aplicando C&W para saber si se logró una mejora en la asignación de clientes a depósitos. La estrategia de esta heurística es detectar clientes los cuales el cambio de los mismos a otro depósito, podría implicar una mejora en el costo total del problema. Serían aquellos los cuales sus dos clientes vecinos más cercanos, están asignados a otro depósito. Estos se etiquetan como “enajenados”. Supongamos que tenemos a un cliente , asignado al depósito , cuyos dos vecinos más cercanos y están asignados al depósito . En esta heurística vamos a suponer que cerca de nuestro cliente , pasa una ruta cercana la cual pertenece al depósito , por lo cual, a esta ruta quizás no le implicaría mucho esfuerzo pasar también por . Y quizás, sería una ahorro para la ruta original de . Por lo tanto, si el depósito tiene capacidad para atender a , se realiza el cambio y se rutea para saber si implica una mejora en el costo total. De esta forma se van buscando clientes candidatos para probar el cambio de depósito. Notar que no es necesario recalcular todo el ruteo de todos los depósitos para saber si existe una mejora. Basta recalcular únicamente las rutas de los depósito y .

Si el depósito no tiene capacidad para atender al cliente , se hace un intento de colocar al cliente en el depósito más cercando que tenga capacidad de atenderlo. Se recalculan los costos de los depósitos implicados y si hay mejora se aplica.

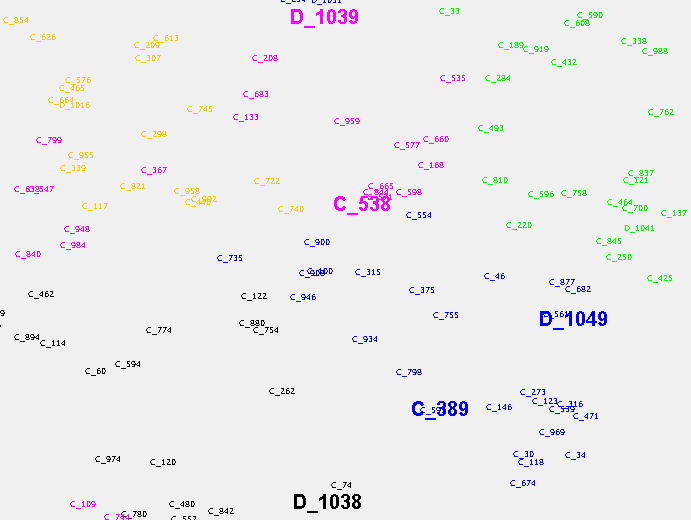
Luego de que con la estrategia recién mencionada no se encuentran más mejoras, se hace un último intento de encontrar mejoras tratando de hacer lugar en el depósito (de y ) en caso que no tenga lugar para albergar al cliente . Pero ¿Qué cliente sacamos del depósito ? Buscamos a algún cliente que esté en la misma situación que (o sea que sus dos vecinos más cercanos pertenezcan a otro depósito) y que el depósito que lo albergaría tenga capacidad para atenderlo. Podría ser el depósito y en este caso sería un intercambio de clientes entre dos depósitos. Se recalculan los costos en los depósitos implicados, y si hay mejora se hacen los cambios.

En el siguiente ejemplo, el cliente 389 inicialmente estaba asignado al depósito 1038 y el cliente 538 estaba asignado al depósito 1049. En ese inicio, ambos clientes cumplían con la condición de que sus dos vecinos más cercanos estaban asignados a otro depósito. Luego de la aplicación del algoritmo, el resultado es el que muestra la imagen a continuación. Se pasa al cliente 389 desde el depósito 1038 al depósito 1049 y para hacer lugar se pasa al cliente 538 del depósito 1049 al depósito 1039

Antes:



Después:



### Algoritmo Enajenado Rápido (AER). Fase 2

Como una mejora del algoritmo de urgencia con capacidades se planteó el algoritmo AER. El mismo, inicialmente corre el algoritmo de urgencias con capacidades para luego cambiar los clientes donde sus dos vecinos más cercanos pertenecen a otro depósito.

Supongamos que tenemos a un cliente , asignado al depósito , cuyos dos vecinos más cercanos y están asignados al depósito . Por lo tanto, se puede decir con seguridad de hay una ruta en B que podría visitar al cliente reduciendo el costo de distancia. Dado esta afirmación, el cliente se pasa al depósito . El algoritmo finaliza cuando todos los clientes son pasados al mismo depósito que sus dos clientes más cercanos. Notar que en caso que sus clientes más cercanos pertenezcan a depósitos distintos no se realiza ningún cambio.

Un aspecto importante a tener en cuenta es que este algoritmo considera las capacidades de los depósitos más una holgura definida en la ejecución del algoritmo. De modo que si el depósito de destino está lleno, no se realiza el cambio.

### Ruteo

Luego de la asignación de clientes a depósitos se procede a calcular las rutas de los vehículos. En la solución se utiliza el algoritmo de Clarke & Wright. El Algoritmo de Clarke & Wright fue explicado en el título 2.4.3.1. La versión implementada es la paralela.

Se parte de una solución inicial que implica una ruta por cada cliente. O sea que la ruta va del depósito al cliente y luego vuelve al depósito. Luego se calcula una lista de ahorros, para todas las combinaciones de pares de clientes, y se ordena en orden descendente. Y luego, se recorre la lista de ahorros, y a medida que sea factible (por la capacidad de los vehículos), se van uniendo las rutas.

### Post Optimización

En la solución implementada, se agregaron dos métodos de postoptimización para el cálculo de las rutas. Uno que solo optimiza el orden el cual los clientes son visitados por el vehículo. Y otro que busca intercambiar clientes entre dos rutas distintas para buscar mejores resultados. Los llamamos postoptimización Intra-ruta e Inter-ruta. En la solución implementada, la postoptimización Inter-ruta incluye la postoptimización Intra-ruta. Por lo cual las opciones a utilizar serían: C & W, o C & W con postoptimización intra-ruta, o C & W con postoptimización intra-ruta e inter-ruta

La postoptimización **Intra-ruta** se basa en el método λ-intercambio descrito en 2.5.1.

Consiste en eliminar λ arcos (camino entre dos clientes consecutivos) de la solución (λ > 1) y reconectar los λ segmentos restantes. Una solución se dice λ-óptima si no puede ser mejorada utilizando λ-intercambios. La solución permite configurar el parámetro “Lambda-Opt” que por defecto viene con el valor 3.

La postoptimización **Inter-ruta** está basada en el método 2-route descrito en [29].

En el mismo se propones probar con 6 movimientos determinados entre dos rutas para encontrar una mejora. Probar con todas las posibilidades no sería factible. Tomando cuatro clientes consecutivos en dos ruta , . Entonces se prueban los siguientes seis movimientos para buscar mejoras:

1. Insertar entre y
2. Insertar entre y
3. Intercambiar y
4. Insertar (,) entre y
5. Insertar (,) entre y
6. Intercambiar (,) con (,)

La combinación que presenta el costo menor es la elegida.

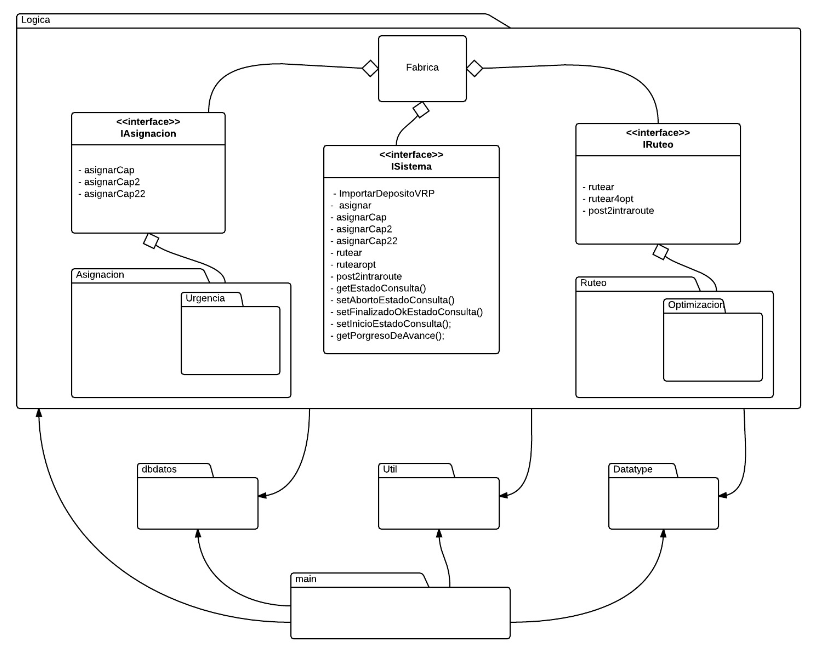
# Capítulo 4

# Implementación

Para la implementación de la aplicación descrita en la sección anterior se eligió el lenguaje de programación Java pues los integrantes del proyecto estábamos familiarizados con el mismo. La versión utilizada fue 1.8.0\_45 con el entorno de desarrollo Eclipse Luna. La solución implementada consiste en una aplicación de escritorio donde se permite la ejecución de los distintos algoritmos y visualización en pantalla del resultado.

Como librería auxiliar se utilizó únicamente package.collections4 de apache para el cálculo de permutaciones, las mismas fueron utilizadas en los algoritmos de post optimización. Los archivos de datos de entrada respetaron los formatos de entrada de TSPLib cumpliendo sus estándares. Dichos formatos se encuentran detallados en la documentación de la librería (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/DOC.PS , s.f.).

A continuación se muestra un diagrama de la solución realizada, a nivel de paquetes (java packages) implementados y la interacción entre los mismos. La especificacion de cada clase y las funcionalidades de la msma se encuentran documentadas en los javadocs de la aplicacion, adjuntos en la entrega del código.



## Interfaz Gráfica

Para la interfaz gráfica se utilizaron los componentes de swing de java, y para el mapa y los componentes gráficos del mismo se utilizaron awt. La implementación de los componentes de la interfaz gráfica utilizados está bajo el paquete ¨main¨. Este paquete es el punto de entrada e interactúa con las interfaces del sistema.

Como decisión y buenas prácticas de diseño de aplicaciones, la comunicación de la capa de interfaz grafica con la capa lógica se realiza únicamente con tipos de datos strings o datatypes (los mismos están en el paquete de ¨DataType¨).

La comunicación de la interfaz gráfica con el paquete ¨utils¨ corresponde a asignar valores de parámetros para le ejecución de algoritmos. Estos parámetros pueden ser holgura de los depósitos tiempo de cálculo, mejoras que corresponden a los criterios de parada de los algoritmos implementados. Para los algoritmos de optimización contamos con el parámetro lambda.

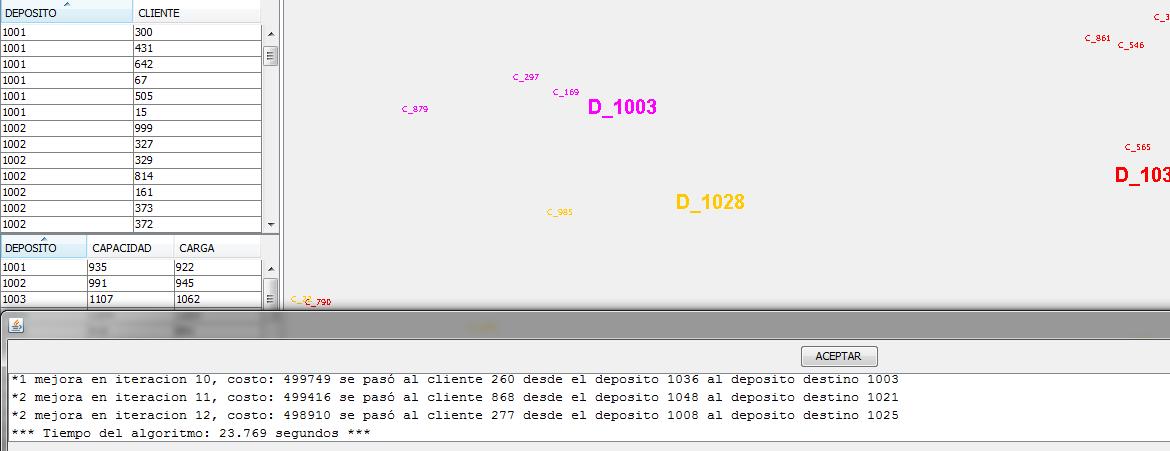
Por otro lado la capa grafica se comunica con el paquete ¨dbdatos¨, que es un controlador del archivo de datos. El archivo de datos contiene la información de los depósitos y clientes según el formato TSPLib como se comento anterirmente. Durante la carga del archivo de datos los mismos se cargan en memoria para realizar los posteriores cálculos.

Como es de esperar la comunicación principal le la capa grafica es con la capa lógica donde se llaman a las funciones de las mismas a través de la fábrica (patrón de diseño Factory) luego de esto se manejan las llamadas a las distintas interfaces.

A modo de permitir la visualización de la distinta etapa de ejecución del algoritmo en pantalla, es implemento un sistema de logeo de mensajes así como otro sistema para resaltar un nodo en particular en la interfaz gráfica.

Para desplegar mensajes en el sistema de logeo, se interactúa desde el algoritmo con la clase sistema donde se actualiza el mensaje a desplegar a través de la función *public void setMensaje(String m).* Este mismo proceso se realiza para resaltar nodos en el mapaa través de la función *public void setResaltados (DataType Nodo);*

A continuación podemos observar una captura de pantalla luego de la ejecución de algoritmo AEL, en la misma se pueden observar los mensajes en la ventana de logeo así como los nodos resaltados en la sección del mapa.



Finalmente la llamada de un algoritmo de la capa lógica se realiza en la clase JPaneletiquetaz.java, a través de un listener del botón correspondiente al algoritmo elegido. La accion de apretar un boton genera un evento de llamada a la función correspondiente. Por ejemplo, para el caso de llamada al algoritmo de asignación por urgencia utilizamos:

Fabrica.getInstancia().getSistema().asignarCap(vrp).

Antes de la llamada al algoritmo es necesario iniciar un temporizador para calcular el tiempo de demora de su ejecución (el temporizador se accede a través del paquete ¨util¨, ya sea desde la capa grafica como desde la capa lógica).

## Lectura de Datos y carga de datos.

La lectura de datos se realiza principalmente en la clase sistema dentro de la capa lógica. Pero Inicialmente, la capa de presentacion, interactuando con el controlador ¨dbDatos¨ se encarga de la manipulación del archivo de dato. Luego este controlador realiza un pre-procesamiento de los datos y carga en memoria la información del archivo. Esta información finalmente es procesada y cargada en memoria en la clase sistema dentro de la capa lógica, estos datos tienen el formato necesario para la ejecución de los algoritmos.

El formato de los datos de entrada de la aplicación se define la documentación de la librería TSPLib. Para la implementación de la lectura de datos se creó un algoritmo que realiza los siguientes pasos.

1. Leer el archivo de entrada y transformarlo en una colección de strings en memoria.
2. Procesar las líneas buscando los tokens que especifiquen la tarea a realizar (sección en la que estoy y tipo de representación de los datos en TSPLib).
3. Crear los datos dtNodo para clientes y depósitos.
4. Diferenciar los que son depósitos y nodos (con la variable esDepositos = true)
5. Retornar un elemento DTDepositoVRP, que contiene datos del archivo y una colección de DTNodo. Esta es la entrada para los algoritmos de asignación.

Dado que existen numerosos tipos de tokens distintos para representar distintas formas de entrada de datos, fue necesario construir una máquina de estados para la carga de los mismos en el sistema.

Estados:

* + - * estado="NODE\_COORD\_SECTION";
      * estado="DEMAND\_SECTION";
      * estado="DEPOT\_SECTION";
      * estado="EDGE\_WEIGHT\_SECTION";

Caso partícula de la máquina de estados:

Si el tokenEDGE\_WEIGHT\_TYPE tiene el valor "EXPLICIT" tengo dos opciones para la lectura de datos, y tengo que leerla como una matriz de distancias.

EDGE\_WEIGHT\_FORMAT = "LOWER\_COL" // columnas por debajo.

EDGE\_WEIGHT\_FORMAT = "LOWER\_ROW" // filas por debajo.

En otro caso, leo los clientes y depósitos como coordenadas euclidianas.

## Capa Lógica

La capa lógica contiene la implementación de todos los algoritmos de ruteo, asignación, post-optimización y mejoras de asignación. Estos algoritmos se comunican con distintas clases del paquete ¨util¨ para definir su forma de ejecución, cálculos de distancia, manejo de holgura en los depósitos y cualquier parámetro o variable que no es propia de los algoritmos y permite su parametrización.

Como se dijo en la sección de interfaz gráfica. Durante la ejecución de un algoritmo e interactuando con la clase sistema se permiten enviar mensajes para ser desplegados en pantalla y también se permite resaltar nodos particulares. Esto es sumamente importante al momento de verificar el correcto funcionamiento del algoritmo.

Existen dos paquetes fundamentales en la capa lógica, uno es Asignación. El cual se encarga de los algoritmos de asignación y mejoras de asignación y el otro paquete es ruteo, el cual contiene los algoritmos de post-optimización y ruteo.

### Asignación

Los algoritmos de asignación tienen un formato de entrada y un formato de salida que es fundamental para que esto sea compatible con las siguientes etapas y también sea desplegado en pantalla correctamente. A modo de Ejemplo

**Ejemplo: public** Collection<DTAsignacion> asignar(DTDepositoVRP d)

Entrada: DTDepositoVRP es la salida del algoritmo de lectura de datos.

Salida: Coleccion de DTAsignacion. DTAdignacion es un DTNodo (deposito) y una colección DTNodo (colección de clientes asignados a él).

Los algoritmos implementados y sus respectivas clases.

* Asignación
  + Asignación por Urgencia: Urgencias.java
  + Asignación por Urgencia con Capacidades: UrgenciasCap.java (Fase 1)
* Mejora de la asignación (Fase 2)
  + Algoritmo Enajenado Rápido: UrgenciasCap2.java
  + Algoritmo Enajenado Lento: UrgenciasCap22.java

Cliente.java: Define las características de un cliente y guarda los datos necesarios para los algoritmos de asignación. Estos datos son el depósito más cercano, un DTnodo con toda la información del cliente y mu, utilizado para la asignación de algoritmo de un cliente a un depósito.

ClienteCap2.java: heredada de clientes y contiene atributos necesarios para los algoritmos de mejora de asignación (Fase 2). Estos atributos son los dos clientes más cercanos.

Depositos.java: Define las características de los depósitos para los algoritmos de asignación. Los datos incluyen un DTnodo con la información del depósito, una colección de clientes asignados y la capacidad libre del depósito.

Enajenados.java: Contiene la información de un cliente enajenado. Recordemos que definimos un cliente enajenado, como un cliente (c1) que pertenece a un depósito (d1) y sus dos vecinos más cercanos (c2 y c3) pertenecen a otro depósito (d2). Por lo tanto la información de un nodo enajenados va a ser: el cliente (de tipo clienteCap2 pues es de interés la información de sus dos vecinos más cercanos), el depósitos al cual el cliente está asignado y el depósitos al que pertenecen mis dos vecinos más cercanos. Finalmente tengo el valor de mu para el cliente.

### Ruteo

El formato de entrada y de salida para los algoritmos de ruteo y de post-optimización utilizado es el siguiente:

**Ejemplo: Public** Collection<DTRuteo> rutear(DTAsignacion dt,**int** capacidad)

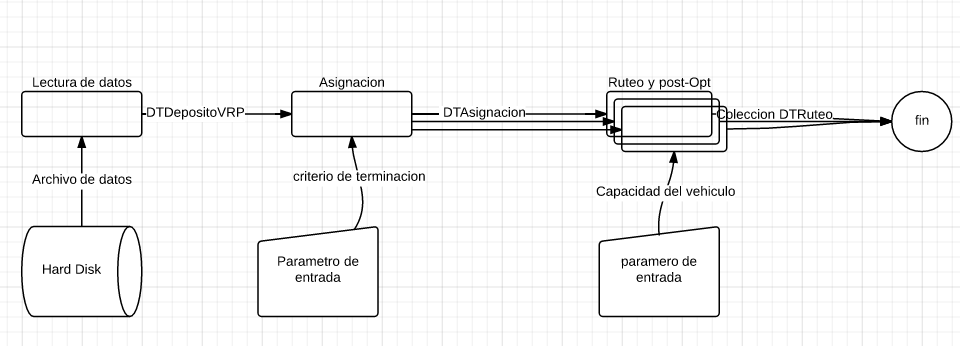
Entrada: DTAsignacion es el resultado de la asignación para un depósito.

Capacidad de los vehículos.

Salida: Coleccion de DTRuteo, un DTRuteo contiene un DTNodo (un deposito), una lista de DTNodos (clientes por donde pasa una ruta) y finalmente un costo.

* Ruteo
  + Clarke & Wright (ClarkWright.java)
* Post-optimización.
  + λ-intercambio (Landainterchange.java)
  + R-iopt (Route14opt.java)

### Flujo del programa



## Implementación de funcionalidades Misceláneas

En esta sección incluiremos programas o funcionalidades de la aplicación principal que fueron necesarias implementar pero no fueron incluidas en los requerimientos. Las mismas no son obligatorias pero dan valor agregado al producto final. La documentación de las mismas permite su extensión en caso de que sean necesarios trabajos posteriores sobre dichas funcionalidades.

### Representación grafica de la matriz de distancias

A partir de los datos de entrada en formato de matriz de distancias para TSPLib e inspirado en el Método de Monte Carlo, se genera una representación gráfica y se despliega en pantalla. Así mismo se transforma esta representación a coordenadas cartesianas que son las utilizadas por los algoritmos descritos anteriormente. Esta transformación no es exacta sino que aproximada.

El formato de entrada de datos para TSPLib para la representación de los clientes y los depósitos en función de la distancia corresponde con una matriz triangular en donde se guardan las distancias de cada nodo al resto de los nodos.

En la siguiente figura se puede observar un ejemplo de este formato donde se distinguen los siguientes parámetros: las Dimensiones (cantidad de nodos), la medición del peso de las aristas (la misma es representada explícitamente), y el formato de representación para el peso de las aristas de la matriz triangular (este formato condiciona la forma de lectura de la misma). Al final en la figura se puede observar una sección con un extracto de la matriz de distancias.

|  |
| --- |
|  |

Al igual que el formato del resto de los archivos de entrada descritos en la sección de lectura de datos, luego de la matriz triangular se encuentran los datos de la demanda de los nodos y finalmente cuales de estos nodos son depósitos.

El algoritmo de transformación consiste en seleccionar un nodo inicial, y luego a través de la generación de números aleatorios, seleccionar los que cumplen mas cercanamente con la matriz de distancias. De esta forma se agregan uno a uno los clientes hasta obtener el grafo completo y guardar las coordenadas cartesianas en el formato de datatype DTDepositoVRP para su futuro procesamiento.

### Generador de casos de Prueba y su funcionamiento

La generación de casos de prueba aleatorios se implementó en linux (Ubuntu) a través de un script de Shell. Para la misma se utilizó bash y la función de generación de números aleatorios RANDOM. El código de generación de casos de prueba cuenta con un parámetro de entrada, que representa el porcentaje de holgura para agregar a cada deposito (ej 10, 20 que representan 10% y 20% respectivamente).

Existen dos script de generación de casos de prueba:

*gen.sh:* Permite generar casos de prueba donde todos los depósitos tienen la misma oferta.

*gen\_diff.sh:* Permite generar casos de prueba donde los depósitos tienen distinta oferta.

Los casos de prueba generados corresponde al formato EDGE\_WEIGHT\_TYPE : EUC\_2D de TCPLIB, cumpliendo con su sintaxis y especificaciones.

Parámetros editables en el script.

*c=1000 // numero de clientes*

*d=50 // numero de depositos*

*max\_x=10000 // coordenada maxima de x.*

*max\_y=10000 //coordenada maxima de y*

*RANDOM=1234 // semilla para el generador de numeros aleatorios.*

*echo "NAME : xxxxx" // Nombre del caso de prueba*

*echo "COMMENT : xxxxx" // comentarios.*

Ejemplo de ejecución para una holgura de los depósitos del 10%: gen.sh 10 > test.txt

Expresión del cálculo de coordenadas para cada cliente:

*Coordx = $((RANDOM%max\_x))*

*Coordy = $((RANDOM%max\_y))*

Expresión para el cálculo de las capacidades de cada cliente:

*temp= $((RANDOM%32+25));*

Expresión para el cálculo de las capacidades de cada depósitos:

*Demanda\_total = sum (demanda(ci))*

Donde *n* representa al numero de clientes y *ci* el cliente *i*.

Para el script *gen.sh:* *Cap\_Dep = Demanda\_total \* (100+holgura)/n*

Para el script *gen\_diff.sh*: Cap\_Dep\_RAND es una variable aleatoria uniformemente distribuida en el intervalo [*Cap\_Dep/2*, *Cap\_Dep/2 + Cap\_Dep*]

Estas relaciones fueron realizadas en base a pruebas y pueden ser editados en el script de generación de casos de prueba.

# Capítulo 5

# Testeos



## Casos de Prueba

A partir de lo implementado en el capítulo anterior, se realizaron pruebas del comportamiento de los distintos algoritmos. Para esto fue necesario crear casos de prueba de MDVRP que permitan la comparación y análisis de los resultados.

Para la carga de datos en el sistema se utilizó el mismo formato de TSPLib. Como se dijo en el estado del arte, esta es una biblioteca de casos de prueba para TSP y VPR principalmente. La misma provee distintos formatos de datos, los cuales se basan en representar los clientes por una matriz de distancias o cada cliente por un par de coordenadas. A su vez existen distintas formatos de matrices (euc\_2D, Geo, ATT, Marix) los cuales se detallan en la documentación de TSPLib [30].

Los casos de prueba utilizados en esta sección se basaron en la representación de los clientes y depósitos en coordenadas cartesianas. Se crearon escenarios con los clientes y sus demandas y los depósitos y sus ofertas.

Es importante resaltar que la librería de problemas de TSPLib no incluye problemas de MDVRP con limitaciones de capacidad de los depósitos. Por lo tanto fue necesario modificar el caso de prueba gil262 para incluir las capacidades en los mismos.

A modo de tener un escenario documentado el cual sirve para su comparación con otras publicaciones [30] se construyó el siguiente caso de estudio a partir del ejemplo gil262 de TSPLib.

**Caso de estudio A)** gil262 con la capacidad de los depósitos modificada.

|  |
| --- |
| A (gil262modif.txt) |
| 12 depositos de capacidad de oferta diferente.  250 clientes con demanda variable.  Representacion en Coordenadas Euclideanas. |

**Caso de estudio B)** Otra limitante de TSPLIB es el número de clientes y de depósitos ya que se necesitaba realizar pruebas con un gran número de estos. Por lo tanto fue necesario generar de forma aleatoria 1000 clientes y 50 depósitos, donde las capacidades de los clientes varían aleatoriamente entre 25 y 50; y las capacidades de todos los depósitos son iguales.

Este escenario fue propuesto por los tutores del proyecto. La documentación de cómo se generan este tipo de casos de prueba se encuentra en la sección implementación.

A partir del caso B, se crearon 3 casos distintos donde los datos de los clientes son iguales y varían las capacidades de los depósitos. Estos 3 casos varían según la capacidad de los depósitos de satisfacer la demanda de los clientes, y fueron generados de esta forma para permitir comparar los algoritmos.

**B.1)** Puedo cumplir apenas la demanda de los clientes si sumo la capacidad de todos los depósitos.

**B.2)** Los depósitos tienen un 10% más de capacidad que la demanda de los clientes.

**B.3)** Los depósitos tienen un 20% más de capacidad que la demanda de los clientes.

|  |  |  |
| --- | --- | --- |
| B.1 (test0por) | B.2 (test10por) | B.3 (test20por) |
| 50 depositos con capacidad de oferta 808  1000 clientes con demanda variable entre [25,55]  Representacion en Coordenadas Euclideanas. | 50 depositos de capacidad de oferta 889  1000 clientes con demanda variable entre [25,55]  Representacion en Coordenadas Euclideanas | 50 depositos de capacidad de oferta 969  1000 clientes con demanda variable entre [25,55]  Representacion en Coordenadas Euclideanas. |

Luego generamos el **Caso de estudio C,** el mismo es similar al caso de estudio B diferenciándose en que la capacidad de cada depósito es variable. Para este escenario se utilizaron los mismos clientes y depósitos que el caso de estudio B, siendo la capacidad de los depósitos una variable aleatoria uniformemente distribuida en un intervalo fijo. Dicho intervalo depende de la demanda total de los clientes. Más información se puede encontrar en la sección implementación del generador aleatorio de casos de prueba.

Al igual que en el escenario anterior contamos con 3 casos distintos con las siguientes características.

**C.1)** Puedo cumplir apenas la demanda de los clientes si sumo la capacidad de todos los depósitos.

**C.2)** Los depósitos tienen un 10% más de capacidad que la demanda de los clientes.

**C.3)** Los depósitos tienen un 20% más de capacidad que la demanda de los clientes.

|  |  |  |
| --- | --- | --- |
| C.1 (test\_c1\_00.txt) | C.2 (test\_c2\_10.txt) | C.3 (test\_c3\_20.txt) |
| 50 depositos con capacidad de oferta variable entre [404,1212]  1000 clientes con demanda variable entre [25,55]  Representacion en Coordenadas Euclideanas. | 50 depositos de capacidad de oferta variable entre [444,1333]  1000 clientes con demanda variable entre [25,55]  Representacion en Coordenadas Euclideanas | 50 depositos de capacidad de oferta variable entre [484,1454]  1000 clientes con demanda variable entre [25,55]  Representacion en Coordenadas Euclideanas. |

**Casos de prueba Miscelaneos.**

Para verificar el correcto funcionamiento de la aplicación se crearon varios juegos de datos con las distintas representaciones de datos de TSPLib así como casos de prueba con 5.000, 10000 nodos.

Para representar 5.000 clientes se utilizó el caso de estudio test5250.txt. En la sección de conclusiones se detallaran los resultados de esta prueba sin entrar mucho en detalles pues el resultados fue el esperado y el interés de esta prueba fue verificar el correcto funcionamiento con gran número de nodos.

Para verificar el formato de entrada de matriz de distancia de TSPLib, se utilizó el caso de estudios javier2.txt. El mismo desplegó en la ventana del mapa una representación gráfica acertada de la matriz de distancia. Para generar dicha representación, se aplicó una metodología inspirada en el Método Monte Carlo generando el mapa a partir de puntos aleatorios que cumplan aproximadamente las restricciones de distancia de la matriz. La implementación de dicho métodos se puede ver en la sección implementación dentro de misceláneos. Posteriormente sobre este ejemplo se pueden aplicar los distintos algoritmos implementados. El resultado esperado para este caso es obtener una representación grafica “similar¨ y que cumpla las propiedades de distancia descritas en la matriz.

## Plan de pruebas y resultados esperados

Sobre los casos A, B y C procederemos a aplicar los algoritmos de MDVRP implementados.

Luego de aplicar los algoritmos AEL y AER se realiza una comparación de los resultados obtenidos. Para el caso de estudio A, el resultado esperado es obtener valores semejantes a los de TSPLib, ya que este caso parte inicialmente de un ejemplo de la librería. A partir de los resultados del caso B se comparan entre B1, B2 y B3 en base a la holgura de los depósitos para determinar la incidencia de estos cambios en la mejora total de las rutas.

Para el caso C se comparan los resultados entre C1, C2 y C3 para analizar la holgura como se realiza en el caso B. También se verifica que las capacidades iguales o variables en todos los depósitos no debería alterar significativamente el resultado final, comparando los casos de prueba B y C.

## Ejecución y Resultados

A modo de poder compara de las distintas ejecuciones se tomaros las siguientes métricas:

Tiempo de procesamiento (Tiempo Total). El mismo consiste en el tiempo que demora en correr el algoritmo completamente (Asignación + Ruteo). Los parámetros de entrada sobre los criterios de terminación del algoritmo que se encuentra descritos en el manual de usuario condicionan dicho resultado. Por ejemplo, en el en caso de que el criterio de terminación sea por tiempo, la métrica tiempo total es sumamente dependiente del parámetro de entrada.

Distancia recorrida (Costo Total). Es la suma de los recorridos de todos los vehículos para todos los depósitos.

Capacidad extra en los depósitos (Penalidad). Es el porcentaje que se excede de las capacidades de todos los depósitos. O lo que es lo mismo:

es la demanda de todos los clientes asignados al depósito *i.*

es la capacidad de oferta del depósito *i*.

Notar que esto depende de la holgura del depósito; siendo la holgura una variable que se define antes de correr el algoritmo. Ver manual de usuario parámetro de holgura.

### Asignación y Clarke & Wright

A continuación se despliega el resultado de aplicar los distintos algoritmos de asignación implementados. Luego de la etapa de asignación se calculan las rutas para cada deposito con el algoritmos de Clarke & Wright. Los tiempos totales se desglosan en algoritmos de asignación más el algoritmo de ruteo.

Asignación por Urgencia (no se considera la capacidad de los depósitos)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tiempo Total | Costo Total | Penalidad |
| A | 0,011+0,060 | 3388 | 49% |
| B.1 | 0,011+0,779 | 341831 | 20% |
| B.2 | 0,011+0,789 | 341831 | 15% |
| B.3 | 0,011+0,774 | 341831 | 12% |
| C.1 | 0,011+0,752 | 341831 | 25% |
| C.2 | 0,011+0,782 | 341831 | 20% |
| C.3 | 0,011+0,784 | 341831 | 16% |

Asignación por Urgencia (se considera la capacidad de los depósitos)

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tiempo Total | Costo Total | Penalidad |
| A | 0,011+1,807 | 7317 | 0% |
| B.1 | 0,478+0,191 | 514711 | 1% |
| B.2 | 0,437+0,210 | 415336 | 0% |
| B.3 | 0,458+0,231 | 397145 | 0% |
| C.1 | 0,472+0,271 | 505591 | 0% |
| C.2 | 0,485+0,291 | 455269 | 0% |
| C.3 | 0,504+0,331 | 420624 | 0% |

Algoritmo Enajenado Rápido (AER).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tiempo Total | Costo Total | Penalidad |
| A | 1,878+2,183 | 7084 | 0% |
| B.1 | 0,786+0,193 | 513184 | 1% |
| B.2 | 0,701+0,204 | 412787 | 0% |
| B.3 | 0,772+0,240 | 390138 | 0% |
| C.1 | 0,882+0,271 | 505979 | 0% |
| C.2 | 0,882+0,290 | 450559 | 0% |
| C.3 | 0,858+0,301 | 415721 | 0% |

Algoritmo Enajenado Lento (AEL).

|  |  |  |  |
| --- | --- | --- | --- |
|  | Tiempo Total | Costo Total | Penalidad |
| A | 131,940+2,022 | 7064 | 0% |
| B.1 | 9,274+0,191 | 513343 | 1% |
| B.2 | 26,060+0,201 | 408309 | 0% |
| B.3 | 47.512+0,250 | 380452 | 0% |
| C.1 | 18,444+0,261 | 498910 | 0% |
| C.2 | 47.591+0,295 | 446723 | 0% |
| C.3 | 75,481+0,295 | 404348 | 0% |

### Aplicando métodos de Post-Optimización (cambios intra-ruta)

A partir del resultado anterior se ejecutó el método de post-optimización λ-intercambios para cada uno de los depósitos donde λ=3. Se obtuvieron los siguientes resultados. Notar que la columna de penalidad ya es igual a las pruebas anteriores ya que no hay cambios inter-depositos. Los Tiempos Totales corresponden al algoritmo de ruteo y Post-Optimización.

Asignación por Urgencia (no se considera la capacidad de los depósitos)

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 0,062 | 3324 |
| B1 | 0,771 | 337198 |
| B2 | 0,772 | 337198 |
| B3 | 0,801 | 337198 |
| C1 | 0,801 | 337198 |
| C2 | 0,782 | 337198 |
| C3 | 0,801 | 337198 |

Asignación por Urgencia (se considera la capacidad de los depósitos)

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 1,750 | 7277 |
| B1 | 0,222 | 509186 |
| B2 | 0,231 | 410262 |
| B3 | 0,272 | 391836 |
| C1 | 0,312 | 498901 |
| C2 | 0,343 | 449676 |
| C3 | 0,342 | 414886 |

Algoritmo Enajenado Rápido (AER).

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 2.030 | 7052 |
| B1 | 0,235 | 507941 |
| B2 | 0,250 | 407894 |
| B3 | 0,272 | 385358 |
| C1 | 0,310 | 500038 |
| C2 | 0,332 | 445316 |
| C3 | 0,361 | 411511 |

Algoritmo Enajenado Lento (AEL).

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 1,952 | 7039 |
| B1 | 0,222 | 508397 |
| B2 | 0,231 | 404169 |
| B3 | 0,71 | 376089 |
| C1 | 0,31 | 492925 |
| C2 | 0,342 | 442242 |
| C3 | 0,350 | 400280 |

### Aplicando métodos de Post-Optimización (cambios inter-rutas)

A partir del resultado de la etapa de asignación, se ejecutó el método de post-optimización entre rutas del mismo depósito de R-iopt. Se obtuvieron los siguientes resultados.

Asignación por Urgencia (no se considera la capacidad de los depósitos)

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 0,071 | 3324 |
| B1 | 0,966 | 337171 |
| B2 | 0,789 | 341831 |
| B3 | 0,796 | 337171 |
| C1 | 0,796 | 337171 |
| C2 | 0,796 | 337171 |
| C3 | 0,796 | 337171 |

Asignación por Urgencia (se considera la capacidad de los depósitos)

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 1,928 | 7277 |
| B1 | 0,251 | 509012 |
| B2 | 0,318 | 409912 |
| B3 | 0,337 | 391836 |
| C1 | 0,331 | 498901 |
| C2 | 0,343 | 449676 |
| C3 | 0,363 | 414873 |

Algoritmo Enajenado Rápido (AER).

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 2,076 | 7052 |
| B1 | 0,241 | 507767 |
| B2 | 0,253 | 407544 |
| B3 | 0,282 | 385358 |
| C1 | 0,579 | 500021 |
| C2 | 0,352 | 445316 |
| C3 | 0,360 | 411511 |

Algoritmo Enajenado Lento (AEL).

|  |  |  |
| --- | --- | --- |
|  | Tiempo Total | Costo Total |
| A | 2,049 | 7052 |
| B1 | 0,242 | 508288 |
| B2 | 0,242 | 403819 |
| B3 | 0,282 | 376089 |
| C1 | 0.382 | 492925 |
| C2 | 0,331 | 442242 |
| C3 | 0,361 | 400280 |

## Análisis de Resultados.

### Discusión de la penalidad.

Al ver las ejecuciones del primer algoritmo de asignación, el Algoritmo de asignación por urgencia, notamos una elevada penalidad. La asignación por urgencia no considera las capacidades por lo que es de esperar que la penalidad sea elevada. Para el los casos B y C, al ir aumentando la holgura en los depósitos la penalidad es cada vez menor como es de esperar.

En referencia a la información sobre el algoritmo de asignación por urgencia considerando las capacidades, el único caso que merece la pena analizar es B1 (penalidad de 1%), En este caso los depósitos apenas cubren la demanda de los clientes. Esto en la práctica genera el problema que cada depósito, luego de tener clientes asignados, tiene una oferta pequeña residual la cual no será utilizada por ningún cliente. Como esto sucede para cada depósito, la suma de este valor residual va a ser igual a lo que se va a saturar un depósito. Esto sucede por la implementación del algoritmo de urgencia, donde los últimos clientes en asignar se asignan directamente al deposito mas cercano si no hay depósitos que pueda cubrir sus demandas. Por lo tanto genera que uno o varios depósitos se saturen.

En el algoritmo AER para el caso A se observa que la penalidad aumentó. Dicho algoritmo realiza el cambio sin considerar la capacidad disponible del depósito. Por lo que el resultado de una penalidad es el esperado. El caso B1 presenta el mismo problema de falta de holgura en los depósitos y el error residual descrito anteriormente.

Finalmente para el caso del algoritmo AEL, se observa que la penalidad se encuentra únicamente para B1 donde la explicación es el problema del error residual cuando la demanda de los clientes es apenas satisfecha por la oferta de los depósitos.

Como es de esperar dado la holgura de los depósitos, los casos B2 y B3 no presentan problemas de penalidad.

La ejecución de los algoritmos cuando la penalidad es muy grande implica una demora de hasta el doble, esto se debe a la implementación del método de asignación.

Los resultados de penalidad de los casos C1, C2 y C3 son iguales a los de B1, B2 y B3 siendo la penalidad independiente de si la capacidad de los depósitos es la misma (caso B) o si es distinta (Caso C)



### Mejoras en los costos

El costo del caso A es mayor que el resultado de TSPLib para el algoritmo de asignación básico, pero recordemos que TSPLib no considera las capacidades, y las mismas fueron agregadas para cada depósitos manualmente sin contar con una referencia. También es importante tener presente que esta asignación (por urgencia) así como el método de ruteo (Clarke & Wright) son básicos dentro de la bibliografía de MDVRP.

Para los otros algoritmos implementados, dado que TSPLib no considera las capacidades de los depósitos, no se puede hacer una comparación directa de los valores con los resultados de la librería. El tiempo de demora en la ejecución fue menor en el algoritmo de asignación.

Notar que en los métodos de post-optimización, para el caso A no se encontraron mejoras en las rutas.

En todas las pruebas se encontró que el resolver el problema de los clientes enajenados, ya sea en su forma rápida o lenta (con o sin intercambios) plantea una mejora en el algoritmo y por lo tanto en el costo de las rutas. Representaremos el porcentaje de mejora a través de la siguiente tabla, se toma como base el algoritmo de urgencia con capacidades.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Caso de  Estudio | AEL | | AER | |
| % Incremento del Tiempo | % Mejora | % Incremento de Tiempo | % Mejora |
| A | 7268,65 | 3,46 | 123,38 | 3,18 |
| B.1 | 1314,80 | 0,27 | 46,34 | 0,30 |
| B.2 | 3958,89 | 1,69 | 39,88 | 0,61 |
| B.3 | 6832,08 | 4,20 | 46,88 | 1,76 |
| C.1 | 2417,50 | 1,32 | 55,18 | 0,08 |
| C.2 | 6070,88 | 1,88 | 51,03 | 1,03 |
| C.3 | 8974,97 | 3,87 | 38,80 | 1,17 |

A continuacion mpostramos un ejemplo de salida del algoritmo de post optimizacion inter-opt. El mismo, como se muestra en la siguiente figura mejoro el resultado para el caso de estudio B1.

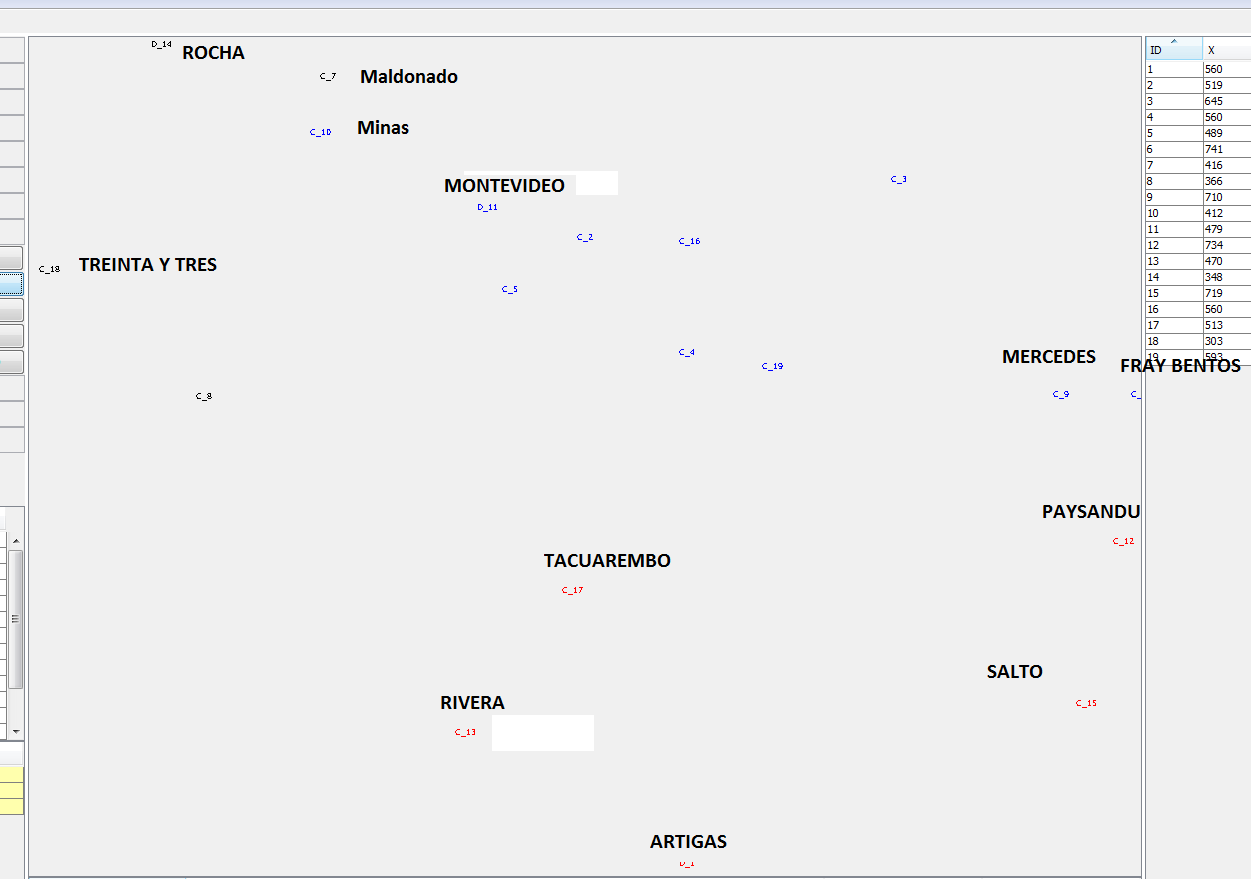
|  |  |
| --- | --- |
|  |  |
| Clarke & Wright | Intra-Route |

### Casos Misceláneos

Para el ejecución del caso de prueba con 5000 clientes, el algoritmo de Enajenados Rápido corrió en 61 segundos mientras que el algoritmo de Ruteo finalizo su ejecución unos pocos segundos más tarde. Los algoritmos de post optimización, al ser locales en un deposito también fueron ejecutados en segundos. Por otro lado el algoritmo de asignación Enajenados Lento (con intercambio entre depósitos) demoro más de 1000 segundos obteniendo una solución considerablemente mejor a la del algoritmo enajenados rápidos.

El caso de prueba javier2.txt, que corresponde a la representación en pantalla de una matriz de distancias, describe las distancias entre las 19 capitales departamentales de Uruguay. En el archivo javier2vrp.xls se puede observar la matriz completa con los nombres de las distintas capitales departamentales, por lo que utilizaremos esta información para interpretar el resultado.

A continuación se presenta la representación grafica del resultado del caso javier2.txt. El primer punto a notar es que no hay un orden determinado, casualmente el mapa esta con los puntos cardinales completamente opuestos. En la figura se puede notar tendencias que corroboran el resultado, por ejemplo que el litoral está bastante bien representado así como las ciudades del Este del País. No entraremos más en detalle en el análisis de estos resultados pero se puede notar que se aproximan a la realidad.



# Capítulo 6

# Conclusiones

La implementación y mejoras de algoritmos de solución de MDVRP, es un problema abierto y de continuo crecimiento, donde todos los meses se pueden encontrar nuevas publicaciones académicas con algoritmos y mejoras. Por lo que una de las metas del proyecto es la de brindar una solución informática para la implementación de nuevos algoritmos y de esta forma proveer un ambiente amigable para ejecutar, validar y comparar distintos algoritmos de resolución del problema de MDVRP.

Haciendo referencia a los algoritmos implementados para la etapa de asignación de MDVRP. En base a las pruebas realizadas y analizando los resultados de los algoritmos implementados llegamos a la conclusión que el algoritmo de enajenados rápido corrió en un tiempo relativamente corto y mejoro la solución respecto a los métodos de asignación por urgencia. El algoritmo de Ruteo finalizo su ejecución en pocos segundos, y los algoritmos de post optimización, al ser locales para cada deposito también fueron ejecutados en segundos. El algoritmo de asignación Enajenados Lento (con intercambio entre depósitos) demoro un tiempo considerable pero llego a una solución mejor que todas las otras formas de asignación. Finalmente el problema de la mejora de los clientes enajenados produjo mejores rutas en todos los casos que se probaron.

# Capítulo 7

# Trabajos a futuro

El área con mayor trabajo a futuro es la implementación y mejoras de algoritmos de solución de MDVRP. Con este proyecto de grado se establece un ambiente de trabajo el cual permite implementar nuevos algoritmos, lo cual de forma sencilla permite agregar nuevos algoritmos en la aplicación.

Por otro lado notamos que el problema de los enajenados es solamente una de las particularidades encontradas. Así como el problema de los enajenados pueden existir muchas otras particularidades en los clientes y depósitos que su análisis puede incidir en mejorar las rutas, como por ejemplo analizar los cúmulos de clientes o la dispersión de los mismos.

Como se dijo anteriormente en este documento, los algoritmos implementados corresponden a etapas, asignación, mejora de la asignación, ruteo y post-optimización. Por ejemplo el algoritmo de enajenados rápido mejora la asignación por urgencia (considerando capacidades) trasladando un cliente posiblemente problemático de un deposito a otro. Por otro lado el algoritmo de enajenados lento genera una mejora aun mayor a la asignación por urgencia (considerando capacidades); intercambiando clientes entre depósitos y recalculando las rutas para medir y comparar las mejoras. Otras formas de retroalimentación entre etapas pueden ser analizadas generando mejores soluciones.

# Anexo

## Comparación de las dos heuristicas

Se adjunta una tabla con un pequeño resumen sobre la comparación de las dos heurísticas, poniendo en la columna resultado, el resultado de rutear con C&W luego de haber realizado la asignación:

|  |  |  |  |
| --- | --- | --- | --- |
| Ejemplo | Resultado:  Asignación sin Mejora | Resultado:  Asignación con Mejora | Mejora % |
| test\_1000c | 413368 | 404346 | 2,2% |
| test0por | 541489 | 509012 | 6,3% |
| test10por | 439985 | 409912 | 7,3% |
| test20por | 381478 | 391836 | -2,6% |
| test50por | 349301 | 351861 | -0.7% |
| test5250 | 1416609 | 1328265 | 6,6% |
| test10500 | 1322852 | 1401071 | -5,5% |
| gil262 | 3469 | 3512 | -1,2% |
| gil262modif | 7337 | 7277 | 0,8% |

## Manual de Usuario

### Entorno de trabajo



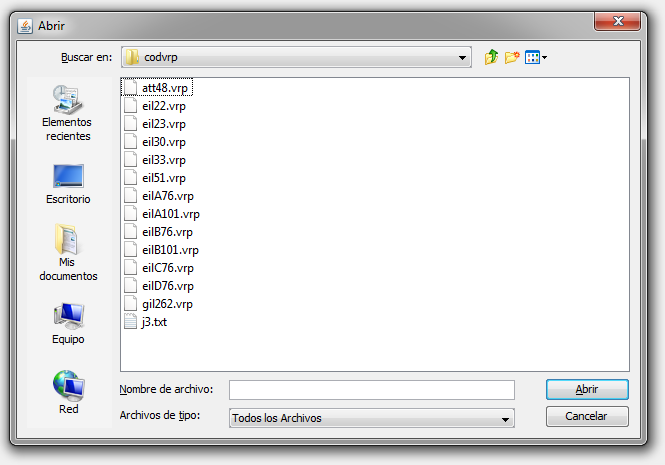
A continuación se podrá reconocer cada uno de los elementos con que cuenta el sistema. Según la señalización se podrá identificar los siguientes elementos:

1. **Menú**: Al hacer clic sobre este botón se desplegará la opción de Configuración.
2. **Abrir Archivo**: Para abrir un archivo de prueba.
3. **Información del Archivo de Prueba**: Esta espacio indica la información del archivo de prueba utilizado. Contiene la información de Nombre, Formato de Entrada, Dimensión, Tipo, entre otras.
4. **Botones de Algoritmos**: Contiene los distintos botones para ejecutar los distintos algoritmos de Asignación, Ruteo y Post-Optimización.
5. **Información General**: En este espacio se podrá ver la información de Capacidad Total de Depósitos, Carga Total, Costo Total y poder Visualizar en el mapa el contorno y sombreado de las distintas rutas.
6. **Listado de Asignación Deposito-Cliente**: Contiene la información de la asignación entre los depósitos y los clientes. El listado permite ordenar de forma ascendente o descendente según el identificador del deposito o del cliente.
7. **Listado de Depósitos**: Contiene la información de capacidad de cada depósito y su carga. Se señaliza en caso de que la carga del depósito sea mayor a la capacidad. El listado permite ordenar de forma ascendente o descendente según el identificador del depósito, capacidad y carga.
8. **Mapa**: Espacio donde se pueden observar los cliente y depósitos situados en el mapa, como tambien las rutas que componen la solución. El mapa permite aplicar **zoom** para poder visualizar mejor las distintas rutas.
9. **Listado con Rutas Generadas**: En este espacio se muestran las distintas rutas generadas luego de la ejecución de los distintos algoritmos de asignación y luego ruteo. Contiene la información del costo de cada uno y la carga que contiene. El listado permite ordenar de forma ascendente o descendente según el identificador del depósito, ruta, costo y carga.
10. **Listado de Clientes-Depósitos**: En este espacio se puede observar la información de las coordenadas X e Y y la demanda en caso de los clientes o la Capacidad en caso de los Depósitos. El listado permite ordenar en forma ascendente o descendente según el identificador del depósito o cliente, las coordenasa X e Y, la demanda/capacidad o el tipo.

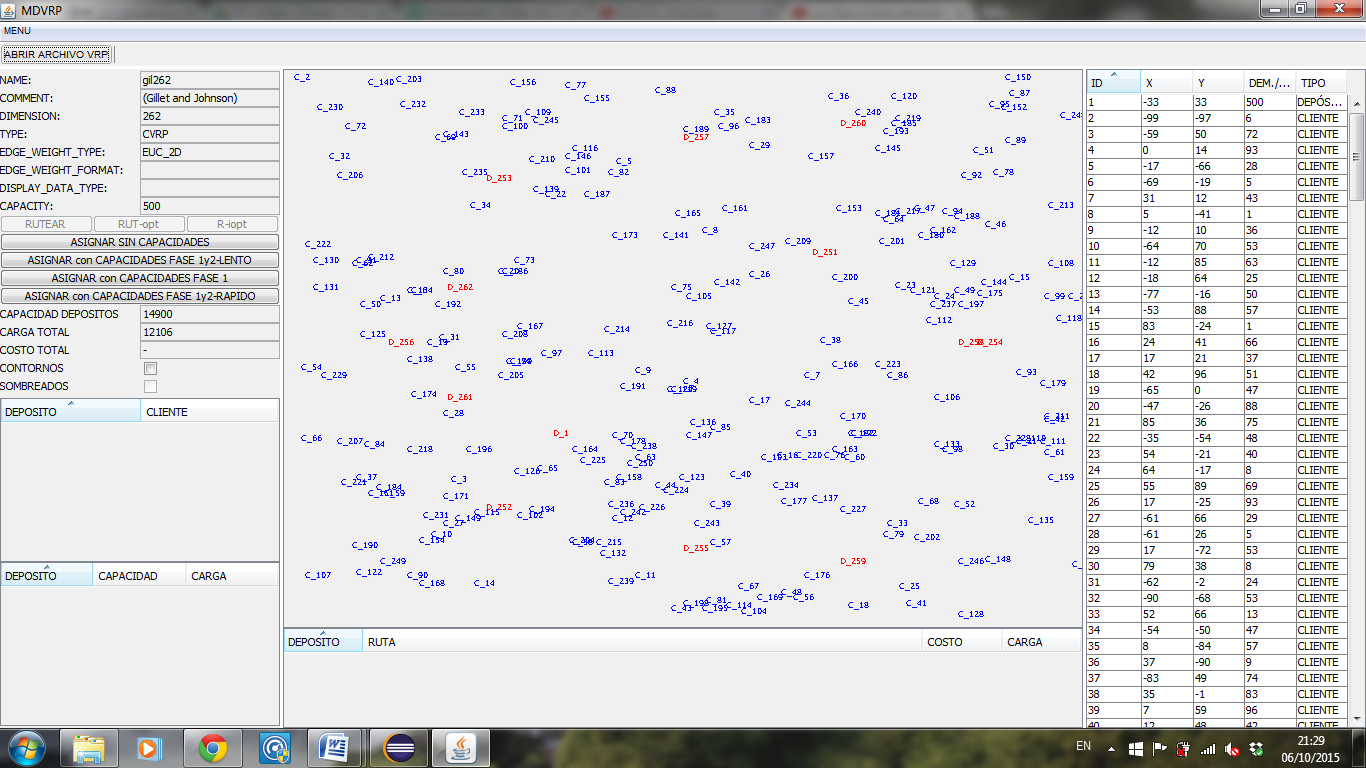
### Procedimientos descriptivos

#### Abrir un archivo de prueba

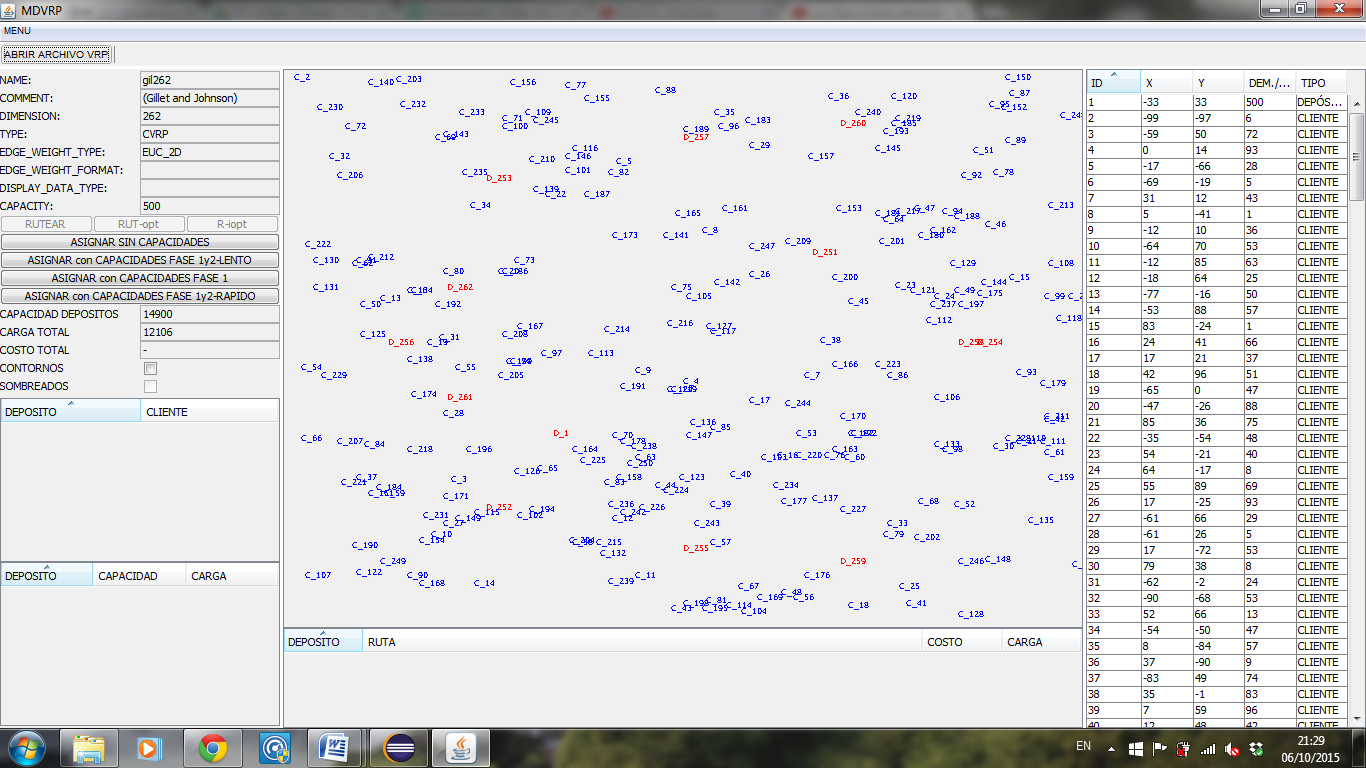
* 1. Para abrir un archivo de prueba hacer clic en “Abrir Archivo VRP” se abrira una ventana:



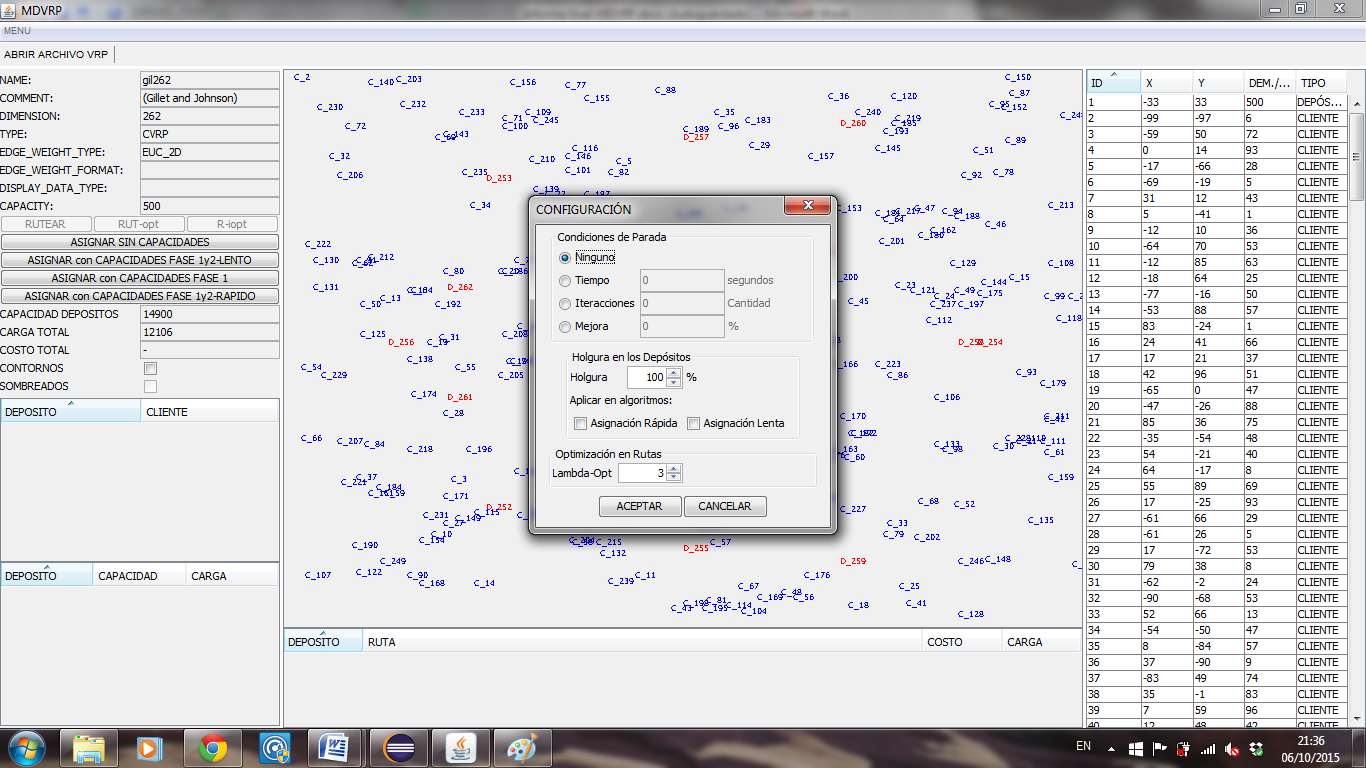
Allí se tiene que buscar la carpeta y el nombre del archivo de prueba a utilizar.

* 1. Al hacer clic en el botón “Abrir” se cargara el archivo de prueba seleccionado y se mostrara su información en el espacio de “Información del archivo de prueba”. Se calculara la capacidad de los depósitos y la carga total y en el espacio de “Botones de Algoritmos” se habilitaran los botones de los algoritmos de Asignación mientras que los botones de los algoritmos de Ruteo seguirán inhabilitados.

En el espacio de ”Listado de Clientes-Depósitos” se cargará la información de los mismos, al igual que su ubicación en el mapa.



#### Configuración de prueba

Al hacer clic en “Menú”, luego en “Configuración” allí se podrá configurar las Condiciones de Parada de los algoritmos, Holgura de los Depósitos y la Optimización de Rutas.

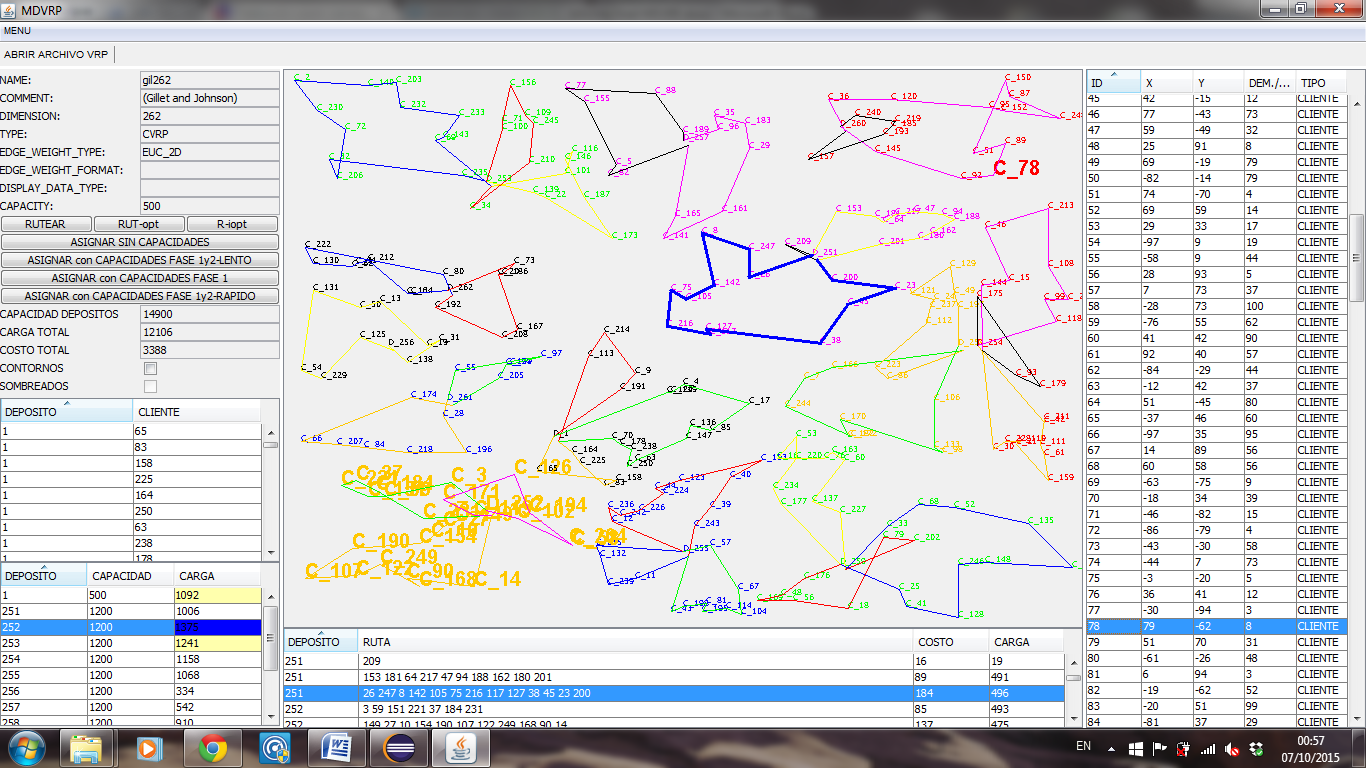
Las Condiciones de Parada que se pueden configurar son: por tiempo transcurrido, por cantidad de iteracciones o por porcentaje de mejora. En caso de no querer configurar ninguna condición de parada los algoritmos continúan la ejecución hasta encontrar una solución optima.

Se podrá también configurar el porcentaje de holgura de los depósitos y también para cual de los algoritmos de enajenado aplica (AER o AEL).

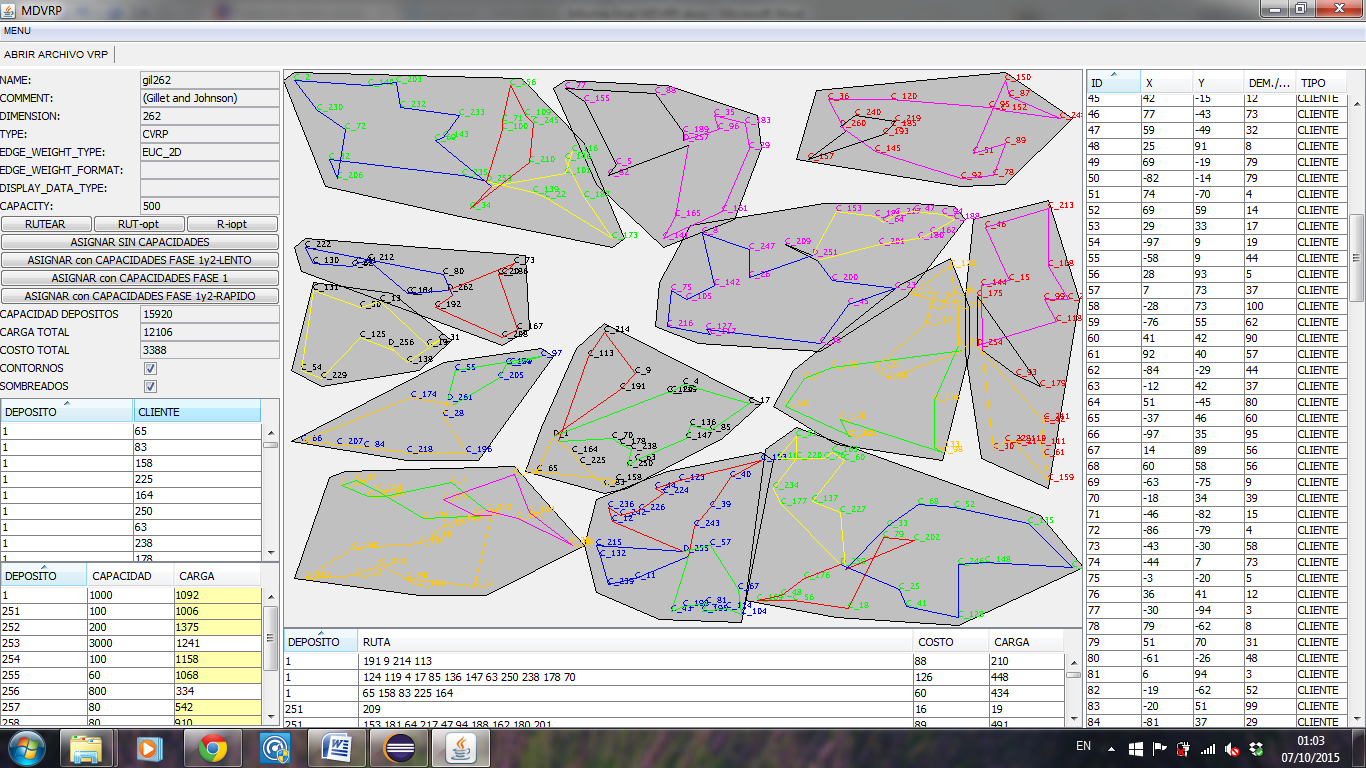
La optimización en rutas se podrá configurar a través del -opt.

#### Interacción con el Mapa

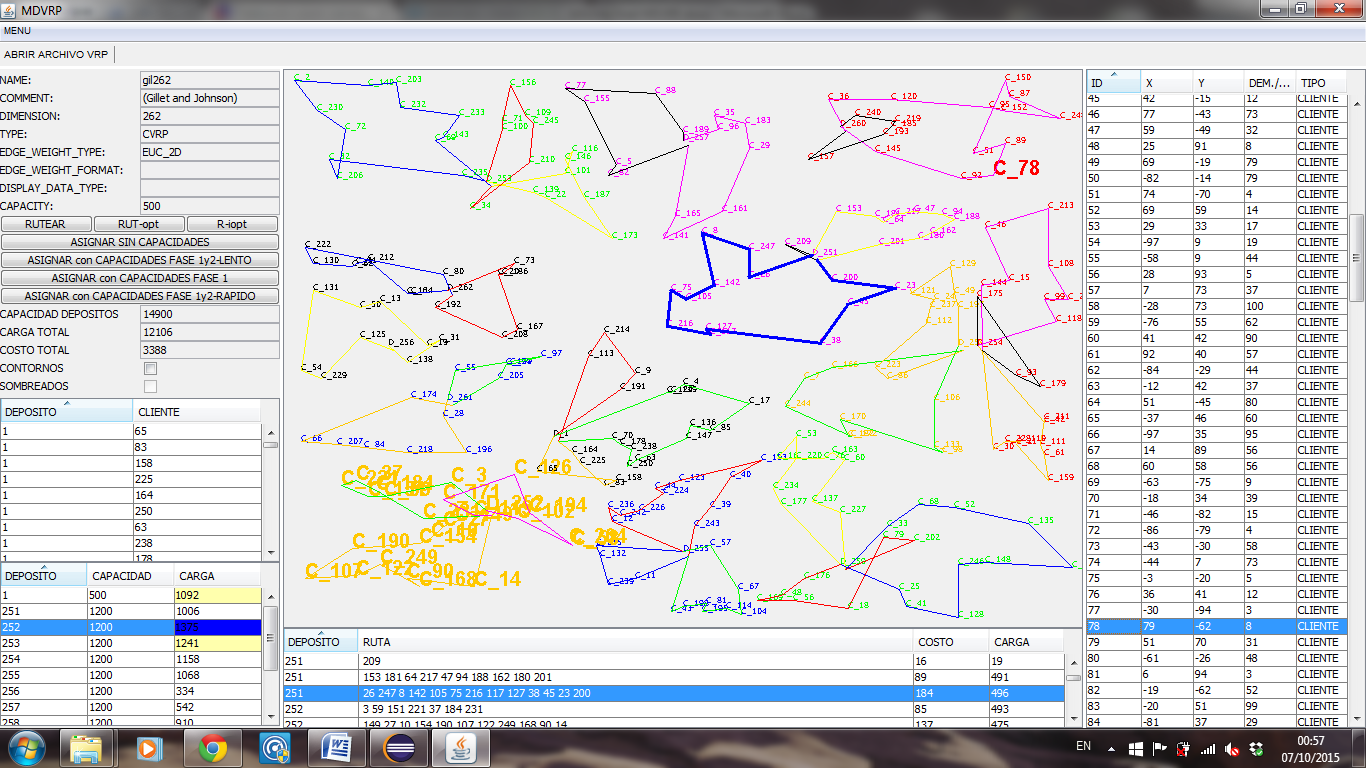
El mapa cuenta con la funcionalidad de poder resaltear Depósitos, Cliente y Rutas. Al hacer clic en un Cliente/Depósito de la “Lista de Clientes-Depósitos” se resaltará en el mapa dicho Cliente/Depósito pudiendo asi poder ubicar mas fácilmente en el mapa. Al seleccionar una ruta del “Listado de Rutas Generadas” se podrá resaltar esta en el mapa. De igual forma al hacer click en un Depósito del “Listado de Depósitos” podemos observar como en el mapa se resalta el depósito seleccionado y todos los clientes que están asignados a él.



Para una mejor observación de los clientes, depósitos y rutas en el mapa se cuenta con un zoom. Se puede realizar un “Zoom In” que permite un plano mas especifico y un “Zoom Out” que se pasa a un plano mas general.

Para tener una mejor visualización de las asignaciones de los clientes a los depósitos el mapa permite marcar los contornos y poder sombrearlos. De esta forma se puede visualizar mejor la “zonas” que quedan definidas.

### Ejecución de Algoritmos

La ejecución de los distintos algoritmos de Asignación, Ruteo y Post-Optimización se realiza en el espacio de “Botones de Algoritmos”.

#### Algoritmos de Asignación

Al final de la ejecución de cada uno de los algoritmos de asignación se carga el listado de Asignación Depósito-Cliente (donde se muestra la asignación entre los depósitos y los clientes), el Listado de Depósitos (donde se muestra la capacidad de cada depósito y su carga, en caso que la carga sea mayor a la capacidad se señaliza). Tambien se actualiza el mapa donde se muestra ahora como quedaron asignados los distintos clientes a los distintos depósitos.

##### Asignar sin Capacidades

Este botón invoca al Algoritmo de Asignación por Urgencia sin Capacidad.

##### Asignar con Capacidades Fase 1

Este botón invoca al algoritmo de Asignación por Urgencia con Capacidades Fase 1.

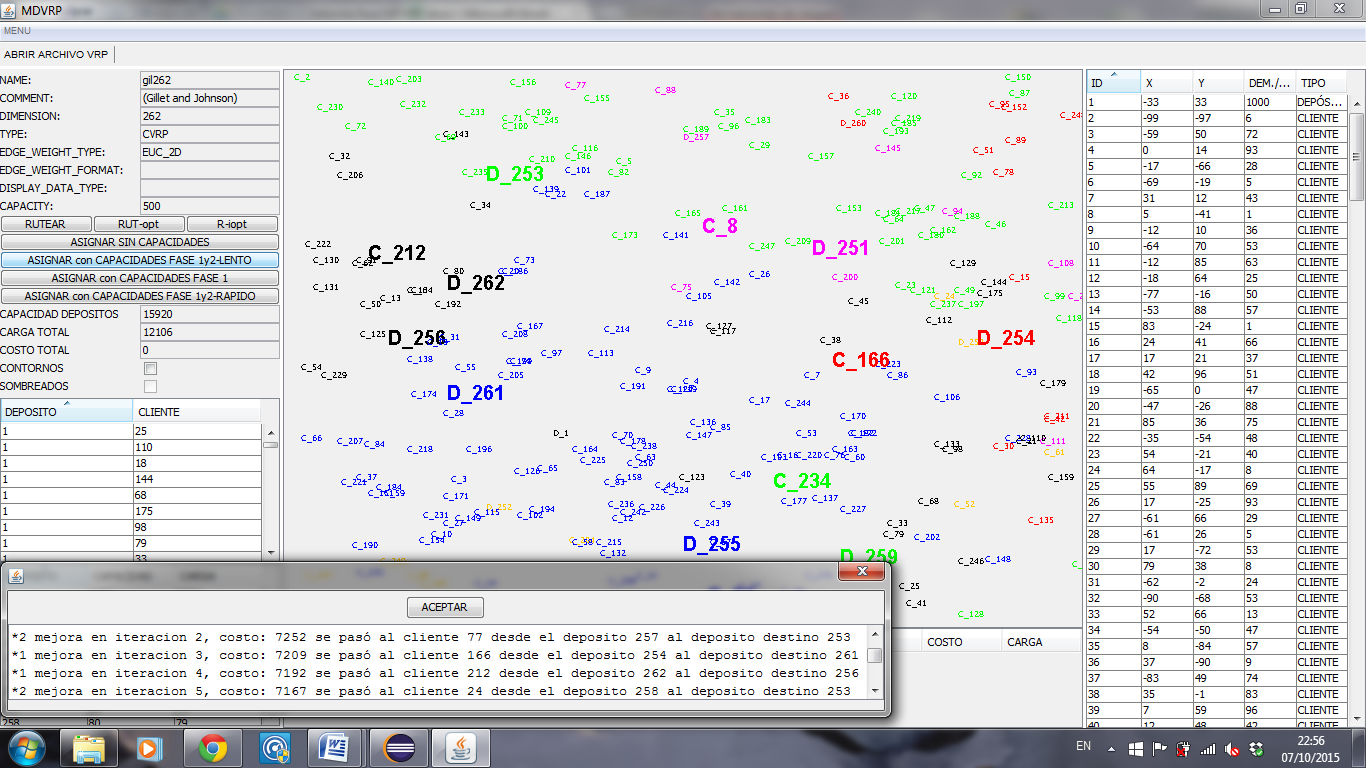
##### Asignar con Capacidades Fase 1 y 2 – Rápido

Este botón invoca al Algoritmo Enajenado Rápido (AER) Fase 2.

##### Asignar con Capacidades Fase 1 y 2 – Lento

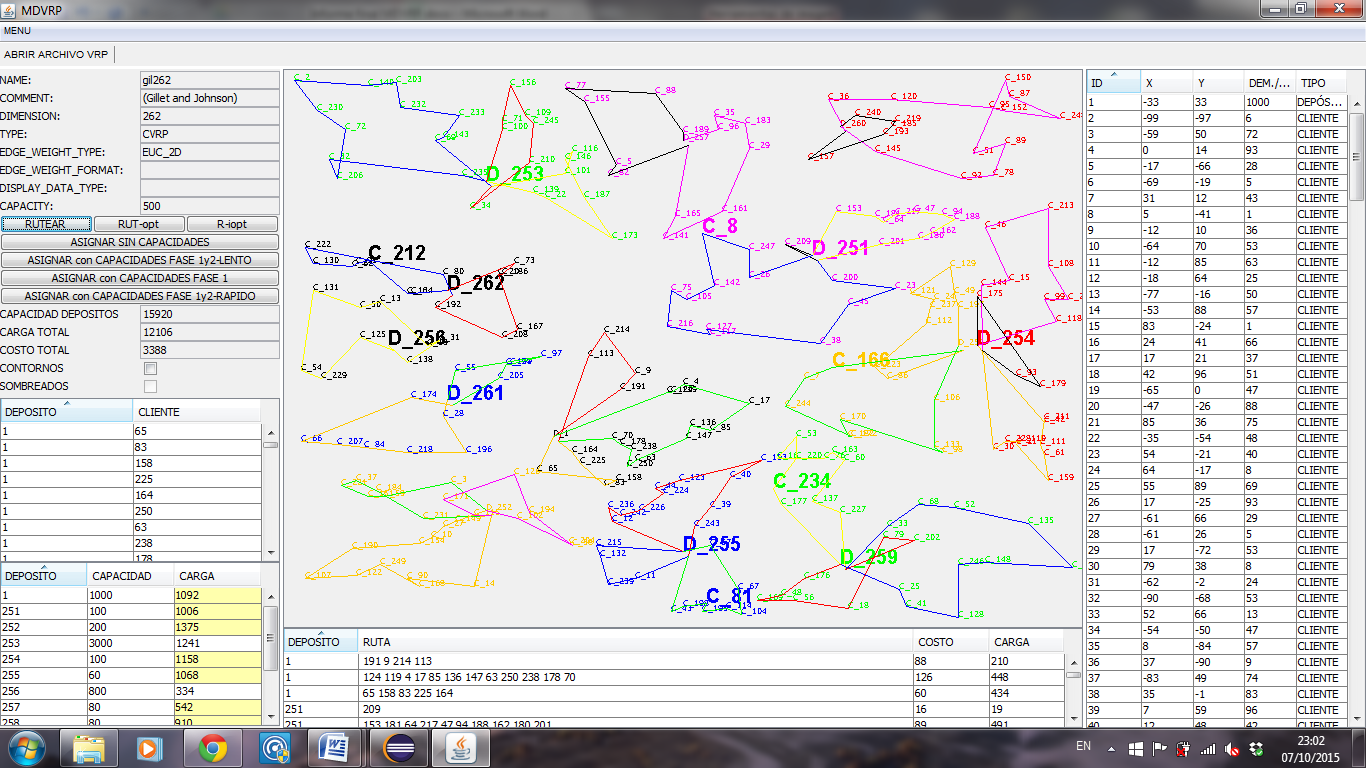
Este botón invoca al Algoritmo Enajenado Lento (AEL) Fase 2.

Este algortimo a diferencia de los demas cuenta con una consola de progreso que nos va indicando los movimiento de los clientes. Nos indica en que iteración hubo una mejora, el nuevo costo de la solución y cual fue el movimiento del cliente, desde que depósito se pasó hasta que depósito.



#### Algoritmo de Ruteo

Este botón invoca al Algoritmo de Rutear. Al final de la ejecución del algoritmo se carga el Listado con Rutas Generadas donde se muestran las distintas rutas generadas por el algoritmo de ruteo, se visualiza en la zona de Información General el costo total.



Tambien se actualiza el mapa donde se muestra ahora las rutas generadas.

#### Algoritmo de Post-Optimización

Al final de la ejecución de cada uno de los algoritmos de Post-Optimización se actualiza el Listado con Rutas Generadas, se despliegan las rutas en el mapa y se visualiza el nuevo costo total.

##### Rut-opt

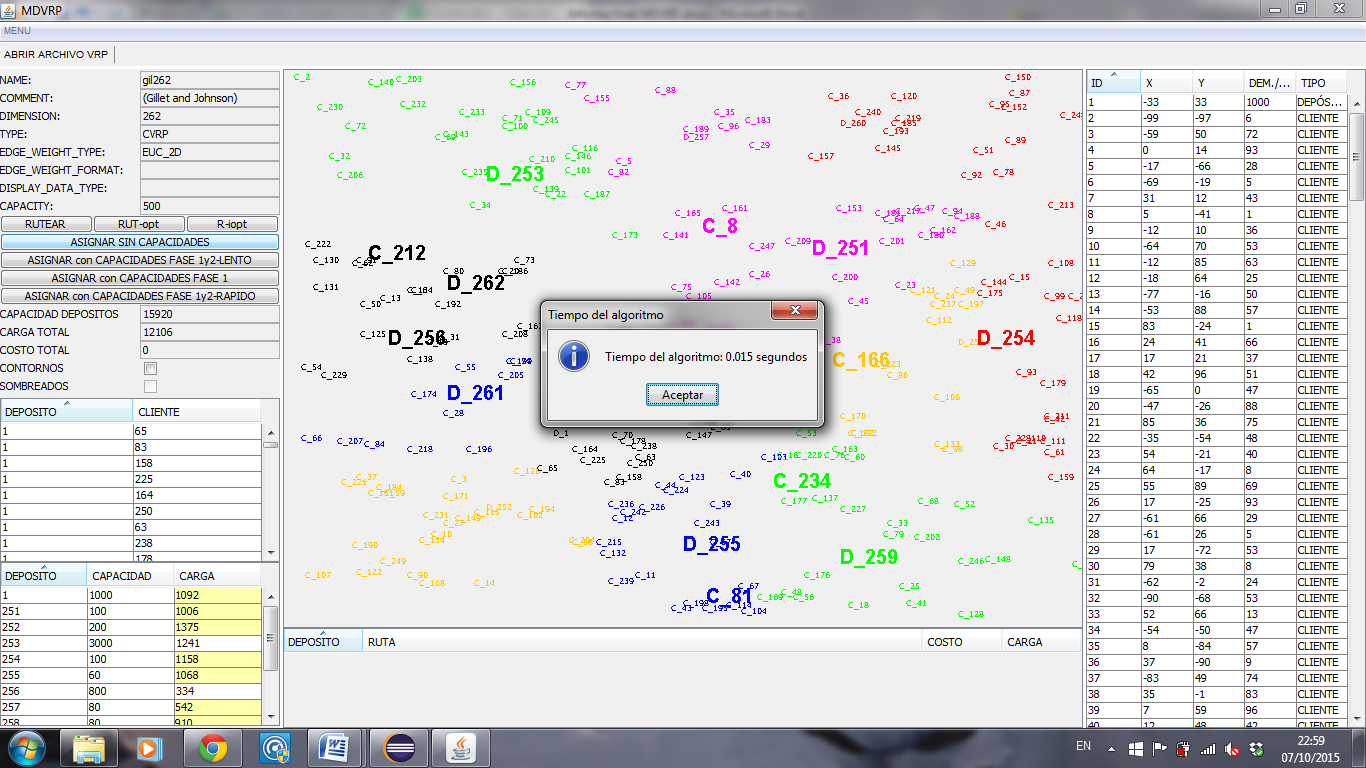
Este botón invoca al Algoritmo de Post-Optimización Intra-Ruta.

##### Rut-iopt

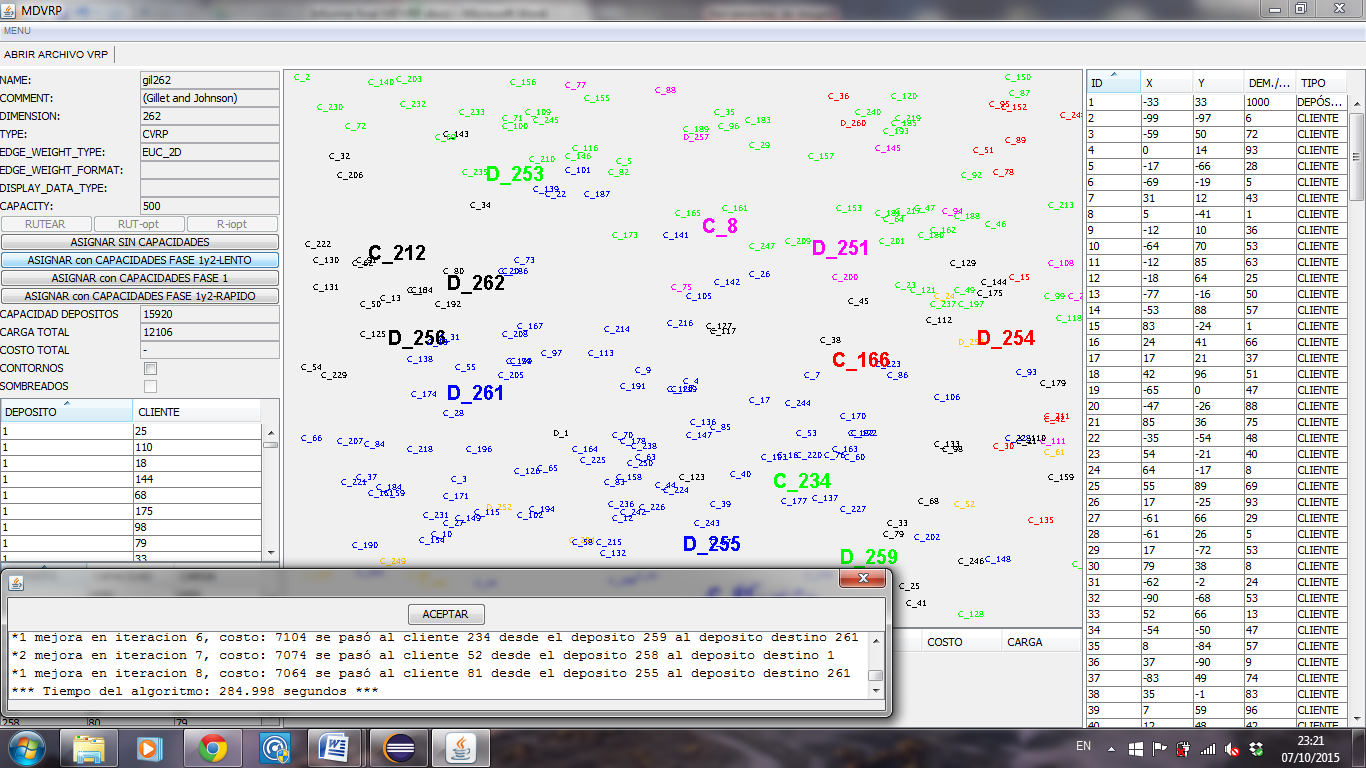
Este botón invoca al Algoritmo de Post-Optimización Inter-Ruta.

#### Tiempo de ejecución

Cada vez que se ejecuta un algoritmo se muestra en pantalla el tiempo total que insumió la ejecución. Para todos algoritmos con excepción del algoritmo AEL se muestra un mensaje con el tiempo total.



Para el algoritmo AEL se muestra el tiempo total en la consola de progreso.



# Bibliografía

|  |  |
| --- | --- |
| [1] | I. Gallegos Mateos, A. Gómez Gómez y D. Arguelles Martino, «A hybrid method for the resolution of the MDVRP,» pp. 45-64, 2013. |
| [2] | A. Schrijver, «On the History of Combinatorial Optimization,» 1960. |
| [3] | L. Bodin, «Routing and scheduling of vehicles and crew: The State of the Art,» *Comput. & Ops Res.,* vol. 10, nº 2, pp. 63-211, 1983. |
| [4] | Y. Wang, «Research of Multi-Depot Vehicle Routing Problem by Cellular Ant Algorithm,» *Journal of Computers,* vol. 8, nº 7, pp. 1722-1727, 2013. |
| [5] | Surekha y S. Sumathi, «Solution to Multi-Depot Vehicle Routing Problem Using Genetic Algorithms,» *World Applied Programming,* vol. 1, nº 3, pp. 118-131, 2011. |
| [6] | J. Carlsson, D. Ge, A. Subramaniam, A. Wu y Y. Ye, «Solvin Min-Max Multi-Depot Vehicle Routing Problem,» 2006. |
| [7] | G. B. Dantzig y J. H. Ramser, «The Truck Dispatching Problem,» *Management Science,* vol. 6, nº 1, pp. 80-91, 1959. |
| [8] | B. L. Golden, «Vehicle Routing Problems: Formulations and Heuristic Solution Techniques,» *Technical Reports,* nº 113, 1975. |
| [9] | G. Dantzig, D. Fulkerson y S. Johnson, «Solution of a Large-Scale Traveling-Salesman Problem,» *Journal of the Operations Research Society of America,* vol. 2, nº 4, pp. 393-410, 1954. |
| [10] | R. M. Karp, «Reducibility Among Combinatorial Problemas,» 1971. |
| [11] | S. N. Kumar y R. Panneerselvam, «A Survey on the Vehicle Routing Problem and Its Variants,» *Intelligent Information Management,* nº 4, pp. 66-74, 2012. |
| [12] | L. Wen y F. Meng, «An Improved PSO for the Multi-Depot Vehicle Routing Problem with Time Windows». |
| [13] | R. Baldacci, M. Battarra y D. Vigo, «Routing a Heterogeneous Fleet of Vehicles,» *Technical Report DEIS OR.INGCE 2007/1,* 2007. |
| [14] | C. Tan y J. Beasley, «A Heuristic Algorithm for the Period Vehicle Routing Problem,» *OMEGA Int. J. of Mgmt Sci.,* vol. 12, nº 5, pp. 497-504, 1984. |
| [15] | P. Francis y K. Smilowitz, «The Period Vehicle Routing Problem with Service Choice,» *Teansportation Science,* vol. 40, nº 4, pp. 439-454, 2006. |
| [16] | A. Mingozzi y A. Valleta, «An exact algorithm for period an multi-depot vehicle routing problems». |
| [17] | R. V. Kulkarni y P. R. Bhave, «Integer programming formulations of vehicle Routing Problems,» *Eurorean Journal of Operational Research,* vol. 20, pp. 58-67, 1985. |
| [18] | A. Garcia-Najera y J. A. Bullinaria, «Bi-objective Optimization for the Vehicle Routing Problem with Time Windows,» School of Computer Science, University of Birmingham. |
| [19] | R. Bowerman, B. Hall y P. Calamai, «A Multiobjetive Optimization Approach to Urban School Bus Routing: Formulation and Solution Method,» 1995. |
| [20] | A. Olivera, «Heurísticas para Problemas de Ruteo de Vehículos,» Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay., 2004. |
| [21] | J. R. Montoya-Torres, J. López Franco, S. Nieto Isaza, H. Felizzola Jiménez y N. Herazo-Padilla, «A literature review on the vehicle routing problem with multiple depots,» *Computers & Industrial Engineering,* vol. 79, pp. 115-129, 2015. |
| [22] | G. Laporte y Y. Nobert, Survey Combinatorial Optimization. |
| [23] | J. L. F. N. H. P. Santiago Nieto Isaza, «Desarrollo y codificacion de un Modelo Matematico para la Optimizacion del MDVRP,» de *LACCEI*, Panama, 2012. |
| [24] | F. Glover y G. A. Kochenberger, Handbook of Metaheuritics, Kluwer Academic Publishers, 2003. |
| [25] | J. Lysgaard, «Clarke & Wright's Savings Algorithm». |
| [26] | Mole y Jameson, «A Sequential Route-Building Algorithm Employing a Generalised Savings Criterion». |
| [27] | L. T. 0. V. D. Giosa, «New assignment algorithms for the multi-depot vehicle routing problem,» *Journal of the Operational Research Society,* vol. 53, nº 9, pp. 977-984, 2002. |
| [28] | O. V. L. Tansini, «New measures of proximity for the assignment algorithms in the MDVRPTW,» *Journal of Operational Reserch Society,* vol. 57, nº 3, pp. 241-249, 2006. |
| [29] | J. Renaudl, G. Laporte y F. F. Boctor, «A tabu search heuristics for the multi-depot vehicle routing problem,» *Computers & Operations Research,* vol. 23, nº 3, pp. 229-235, 1996. |
| [30] | R.-K.-U. Heidelberg, «Discrete and Combinatorial Optimization,» [En línea]. Available: http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/. |
| [31] | P. Toth y D. Vigo, The Vehicule Routing Problem. |
| [32] | G. González Vargas y F. González Aristizabal, «Metaheurísticas aplicadas al ruteo de vehículos. Un caso de estudio. Parte 1: formulación del problema,» *Revista Ingeniería e Investigación,* vol. 26, nº 3, pp. 149-156, 2006. |
| [33] | K. Jansen, «Bounds for the general capacitated routing problem.,» vol. 23, pp. 165-173, 1993. |
| [34] | P. Toth y A. Tramontani, «An Integer Linear Programming Local Search for Capacitated Vehicle Routing Problems,» *The vehicle routing problem: Latest advances and new challenges,* vol. 2, pp. 275-295, 2008. |
| [35] | T.-H. Wu, C. Low y J.-W. Bai, «Heuristic solutions to multi-depot location-routing problems,» *Computers & Operations Research 29,* pp. 1393-1415, 2002. |
| [36] | B. Crevier, J.-F. Cordeau y G. Laporte, «The multi-depot vehicle routing problem with inter-depot routes,» *European Journal of Operational Research 176,* pp. 756-773, 2007. |
| [37] | W. Ho, G. T. Ho, P. Ji y H. C. Lau, «A hybrid genetic algorithm for the multi-depot vehicle routing problem,» *Engineering Applications of Artificial Intelligence 21,* pp. 548-557, 2008. |
| [38] | T. Vidal, T. G. Crainic, M. Gendreau y C. Prins, «Heuristics for Multi-Attribute Vehicle Routing Problems: A Survey and Synthesis,» *CIRRELT,* 2012. |
| [39] | C. Prins, «A simple and effective evolutionary algorithm for the vehicle routing problem,» *Computers & Operations Research,* vol. 31, pp. 1985-2002, 2004. |
| [40] | A. Olivera, «Memorias adaptativas para el problema de ruteo de vehículos con múltiples viajes,» Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay., 2005. |