



UNIVERSIDAD DE LA REPÚBLICA

Facultad de Ingeniería

Instituto de Computación

Departamento de Investigación Operativa

“Problema de Ruteo de Vehículos con Múltiples Depósitos con
Capacidades”

Informe Proyecto de Grado

Javier de Prado

Alejandro García

Francisco Güella

Tutores

Sandro Moscatelli

Omar Viera

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 Multi-Depot 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 se llaman “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.

Contenido

Abstract	3
Capítulo 1	7
Introducción	7
1.1 Contexto	7
1.2 Estructura del documento	8
Capítulo 2	9
Estado del Arte	9
2.1 Introducción al problema de MDVRP	9
2.2 Formulación Matemática de MDVRP	11
2.3 Métodos para la resolución de problemas de ruteo de vehículos	12
2.3.1 Métodos Exactos	13
2.3.2 Métodos Heurísticos	14
2.3.2.1 Heurísticas para VRP	15
2.3.2.2 Heurísticas para MDVRP	16
2.3.2.3 Meta-Heurísticas para VRP	17
2.3.2.4 Meta-Heurísticas para MDVRP	17
2.4 Post Optimización y mejoras	17
Capítulo 3	19
Definición del problema	19
3.1 Características buscadas de la aplicación	20
3.2 Algoritmos de MDVRP propuestos para la solución.	21
3.2.1 Asignación Por Urgencia sin Capacidad	21
3.2.2 Asignación por Urgencia con Capacidades (Fase 1)	22
3.2.3 Algoritmo Enajenado Lento (AEL). Fase 2	23
3.2.4 Algoritmo Enajenado Rápido (AER). Fase 2	25
3.2.5 Justificación de las heurísticas implementadas	26
3.2.6 Comparación AEL y AER	26
3.2.7 Ruteo	26
3.2.8 Post Optimización	27
Capítulo 4	29
Implementación	29
4.1 Interfaz Gráfica	30
4.2 Lectura de Datos y carga de datos	32
4.3 Capa Lógica	33
4.3.1 Asignación	34

4.3.2	Ruteo	36
4.3.3	Flujo del programa	36
4.4	Implementación de funcionalidades Misceláneas	37
4.4.1	Representación gráfica de la matriz de distancias.....	37
4.4.2	Generador de casos de Prueba y su funcionamiento	38
Capítulo 5	40
Testeos.....		40
5.1	Casos de Prueba	40
5.2	Plan de pruebas y resultados esperados.....	43
5.3	Ejecución y Resultados	43
5.3.1	Asignación y Clarke & Wright.....	44
5.3.2	Aplicando métodos de Post-Optimización (cambios intra-ruta)	46
5.3.3	Aplicando métodos de Post-Optimización (cambios inter-rutas).....	48
5.4	Análisis de Resultados.	49
5.4.1	Discusión de la penalidad.....	49
5.4.2	Mejoras en los costos	50
5.4.3	Casos Misceláneos	52
Capítulo 6	54
Conclusiones		54
Capítulo 7	56
Trabajos a futuro		56
Anexo		58
Comparación de las dos heurísticas		58
Manual de Usuario		59
Entorno de trabajo		59
Procedimientos descriptivos.....		60
Abrir un archivo de prueba.....		60
Configuración de prueba		62
Interacción con el Mapa		62
Ejecución de Algoritmos.....		63
Algoritmos de Asignación.....		63
Algoritmo de Ruteo.....		64
Algoritmo de Post-Optimización		65
Tiempo de ejecución		65
Bibliografía		66

Capítulo 1

Introducción

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

1.1 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.

1.2 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 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 capítulo 2 también se presentan las variantes al problema y el modelo matemático del mismo.

En el Capítulo 3 se describe el problema y se presentan dos algoritmos para resolverlo.

En el Capítulo 4 se resume la implementación y sus características principales.

Los resultados obtenidos por los algoritmos se reportan en el Capítulo 5. 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 6 se presentan las conclusiones finales del trabajo y algunas ideas en las cuales se podría profundizar en el futuro se presentan en el Capítulo 7.

Capítulo 2

Estado del Arte

2.1 Introducción al problema de MDVRP

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 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. [11]

La diversidad de aplicaciones donde se pretende solucionar el problema de ruteo de vehículos, ha llevado a plantear diferentes variantes de estos problemas. Dichas variantes están relacionadas a restricciones sobre la realidad, como pueden ser capacidad en los depósitos, horarios de atención en los clientes, distintas capacidades en los vehículos entre otras. Estas variantes se encuentran desarrolladas en el documento de "Estado del Arte" [12] [13] [14] [15] [16].

En el trabajo «A literature review on the vehicle routing problem with multiple depots» [17] 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 Bhawe 1985 [18] *Figura 2.1*. En este mismo trabajo se categorizan las publicaciones según su método de resolución. Pudiendo ser estos métodos: exactos, heurísticos, metaheurísticos o una combinación de estos.

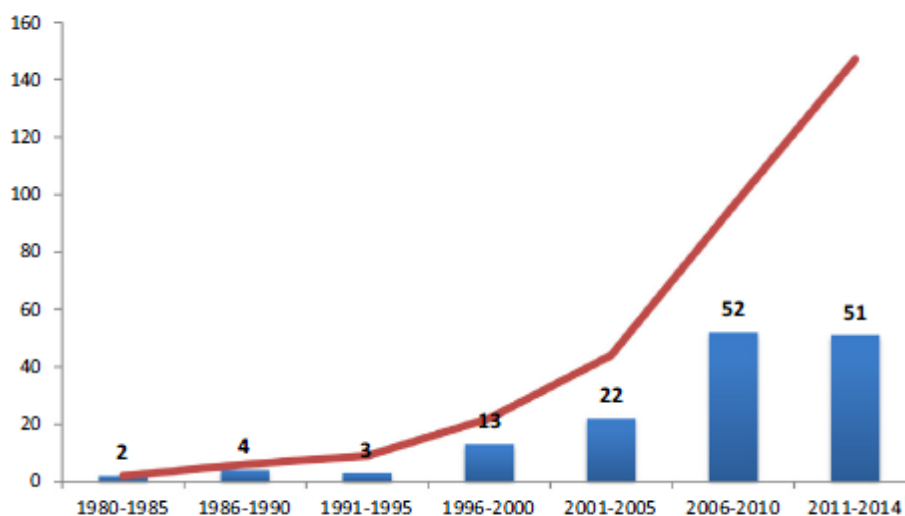


Figura 2.1: Distribución de publicaciones por año de MDVRP

2.2 Formulación Matemática de MDVRP

La formulación del problema de MDVRP se presenta de la siguiente manera siendo $(1 \dots N)$ los N clientes y siendo $(N + 1 \dots N + M)$ los M depósitos: [18].

$$\text{Min} \sum_{i=1}^{N+M} \sum_{j=1}^{N+M} \sum_{k=1}^V c_{ij} x_{ijk} \quad (1.16)$$

s.a.

$$\sum_{i=1}^{N+M} \sum_{k=1}^V x_{ijk} = 1 \quad \forall j = 1, 2, \dots, N-1 \quad (1.17)$$

$$\sum_{j=1}^{N+M} \sum_{k=1}^V x_{ijk} = 1 \quad \forall i = 1, 2, \dots, N-1 \quad (1.18)$$

$$\sum_{i=1}^{N+M} x_{ihk} - \sum_{j=1}^{N+M} x_{hjk} = 0 \quad \forall k = 1, 2, \dots, V; h = 1, 2, \dots, N+M \quad (1.19)$$

$$\sum_{i=1}^{N+M} Q_i \sum_{j=1}^{N+M} x_{ijk} \leq P_k \quad \forall k = 1, 2, \dots, V \quad (1.20)$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^{N+M} c_{ij} x_{ijk} \leq T_k \quad \forall k = 1, 2, \dots, V \quad (1.21)$$

$$\sum_{i=1}^{N+M} \sum_{j=1}^N x_{ijk} \leq 1 \quad \forall k = 1, 2, \dots, V \quad (1.22)$$

$$\sum_{j=N+1}^{N+M} \sum_{i=1}^N x_{ijk} \leq 1 \quad \forall k = 1, 2, \dots, V \quad (1.23)$$

$$x_{ijk} \in \{0,1\} \quad \forall i, j, k \quad (1.24)$$

$$y_i - y_j + (M + N)x_{ijk} \leq N + M - 1 \quad 1 \leq i \neq j \leq N-1, 1 \leq k \leq V \quad (1.25)$$

V = Número de vehículos

P_k = Capacidad del vehículo k

T_k = Costo máximo permitido para la ruta de vehículo k

$Q_i =$ Demanda del nodo i , $Q_N = 0$

$x_{ijh} =$ 1, si el par i, j pertenece a la ruta del vehículo k , 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). Implica que el número de aristas entrantes en un nodo es igual al número de aristas salientes. 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. Dado que según las restricciones presentadas podrían quedar rutas como las que se muestra en la siguiente figura [19]:

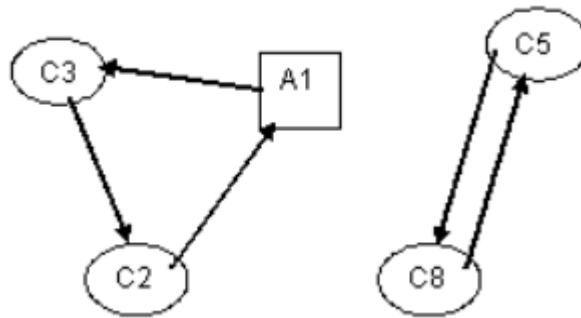


Figura 2.2

2.3 Métodos para la resolución de problemas de ruteo de vehículos

A continuación se analizarán distintos métodos de solución para los problemas MDVRP. A lo largo de la presentación de soluciones también se encontraran los problemas clásicos de VRP. La solución de este tipo de problemas es la base para la solución de problemas más complejos de MDVRP y sus variantes. Las variantes estudiadas tendrán en común las restricciones clásicas de los problemas de ruteo planteadas por Laporte et al 1996 [20].

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.

2.3.1 Métodos Exactos

G. Dantzig, R. Fulkerson, y S. Johnson [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 $((n - 1)!)/2$. 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 planteado como ejemplo en dicho estudio consta de 49 ciudades.

En [18] 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” [20], 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:

- i) Búsqueda arborescente
- ii) Programación dinámica (DP)
- iii) Programación entera lineal (ILP)

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

- i) Formulación de particionamiento del conjunto
- ii) Formulación de flujo de vehículos
- iii) Formulación del flujo de mercancía

Entrar en detalle en cada uno de los métodos de solución exacta se aleja del propósito de este documento. 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 [21].

En [22] 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 heurísticas y meta-heurísticas.

2.3.2 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 [24] los autores F. Glover y G. A. Kochenberger, 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.”

2.3.2.1 Heurísticas para VRP

Algoritmo de Ahorros de Clarke and Wright:

A continuación se muestra la idea general de dicho algoritmo basándose en los apuntes de J. Lysgaard [25] 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:

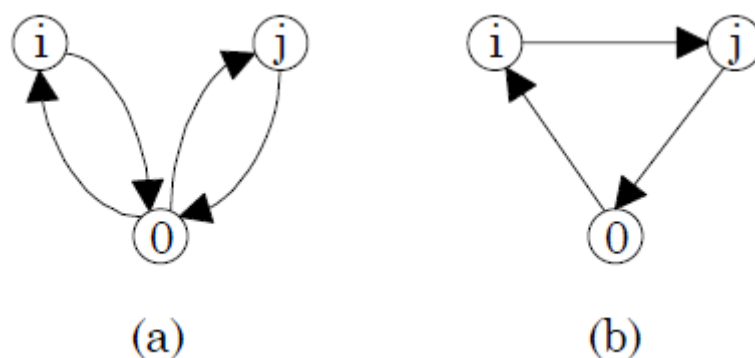


Figura 2.3

Inicialmente los clientes i y j 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 i y j con c_{ij} , el costo total de transporte D_a , en la *figura 2.3 (a)*, es:

$$D_a = c_{0i} + c_{i0} + c_{0j} + c_{j0}$$

De la misma forma, el costo de transporte D_b en la *figura 2.3 (b)* es:

$$D_b = c_{0i} + c_{ij} + c_{j0}$$

Al combinar las dos rutas, se obtiene el siguiente ahorro S_{ij} :

$$S_{ij} = D_a - D_b = c_{i0} + c_{0j} - c_{ij}$$

Los mayores valores S_{ij} 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].

2.3.2.2 Heurísticas para MDVRP

Se han propuesto numerosas heurísticas para el problema de ruteo de vehículos con múltiples depósitos en los últimos años [17], 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.

Un enfoque estudiado en [27] por L. Tansini y O. Viera y en [26] junto a D. Giosa se 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.

2.3.2.3 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, Tabú Search 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 Tabú Search 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.

2.3.2.4 Meta-Heurísticas para MDVRP

Particle Swarm Optimization (PSO) y Tabú Search son dos ejemplos de metaheurísticas para MDVRP de las cuales se puede encontrar más información en el Documento de “Estado del Arte” [28].

2.4 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 s se define un conjunto de soluciones vecinas $N(s)$. Un procedimiento de Búsqueda Local parte de una solución s , la reemplaza por una solución $s^* \in N(s)$ 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 s^* puede buscarse la mejor solución de $N(s)$ o tomar la primera solución de $N(s)$ que mejore el costo.

En el documento de “Estado del Arte” se pueden encontrar otras formas de post-optimización para los problemas de ruteo de vehículos como son lambda intercambio y Or-opt.

Capítulo 3

Definición del problema

El problema planteado para este trabajo es el de MDVRP (Multi 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 el costo 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 de este documento 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 distintos algoritmos que permitan mejorar la solución. De esta forma se implementa una solución al problema clásico de MDVRP, incluyendo la variante de contar con capacidades en los depósitos.

3.1 Características buscadas de la aplicación

Las características buscadas para la aplicación fueron mayoritariamente sugeridas por los tutores mientras que otras fueron agregadas por los integrantes del proyecto.

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

- La entrada de datos corresponde con los formatos de TSPLib [29], 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 implementaciones 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).

A continuación las características visuales requeridas:

- Es sugerido contar con 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 gráfica 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.

3.2 Algoritmos de MDVRP propuestos para la solución.

Analizando el problema con los tutores y basado principalmente en las heurísticas de dos fases para la resolución de MDVRP se llegó a identificar 4 posibles etapas en la resolución de este tipo de problemas: Asignación, Mejora de la asignación, Ruteo, Post-optimización. En la sección 5.4.1 del documento de Estado del Arte se describe las heurísticas de dos fases para la resolución de MDVRP.

Para la resolución de MDVRP se implementaron algoritmos basados en conocidos algoritmos de asignación, ruteo y post-optimización los cuales se encuentran explicados en el documento de Estado del Arte. También se crean dos algoritmos de mejora de asignación (AER y AEL).

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

A continuación se detallan los algoritmos utilizados en las distintas etapas.

3.2.1 Asignación Por Urgencia sin Capacidad

Este algoritmo asigna clientes 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.

3.2.2 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 5.4.1.1 - 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:

$$\mu_c = \left(\sum_{dep \in D} d(c, dep) \right) - d(c, dep')$$

Donde $d(c, dep)$ es la distancia entre el cliente c y el depósito dep , D es el conjunto de depósitos y $d(c, dep')$ es la distancia entre el cliente c y el depósito más cercano dep' .

El cliente con mayor valor de μ_c 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 μ_c para todo los clientes luego de que se asigna un cliente a un depósito. En este caso, al recalcular el μ_c 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 capacidad 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 depósito- demanda total de los clientes), ya que cuando no se aplica la mejora, se puede ahorrar el recalcular 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 ejecución podría llegar a crecer exponencialmente. Ver anexo Comparación de dos Heurísticas.

3.2.3 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 los clientes donde sus dos clientes vecinos más cercanos están asignados a otro depósito. Estos se llamaran “enajenados”. Supongamos que se tiene un cliente C_1 , asignado al depósito A , cuyos dos vecinos más cercanos C_2 y C_3 están asignados al depósito B . En esta heurística vamos a suponer que cerca de nuestro cliente C_1 , pasa una ruta la cual pertenece al depósito B , por lo cual, a esta ruta quizás no le implicaría mucho esfuerzo pasar también por C_1 . Y quizás, sería un ahorro para la ruta original de C_1 . Por lo tanto, si el depósito B tiene capacidad para atender a C_1 , 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 depósitos A y B .

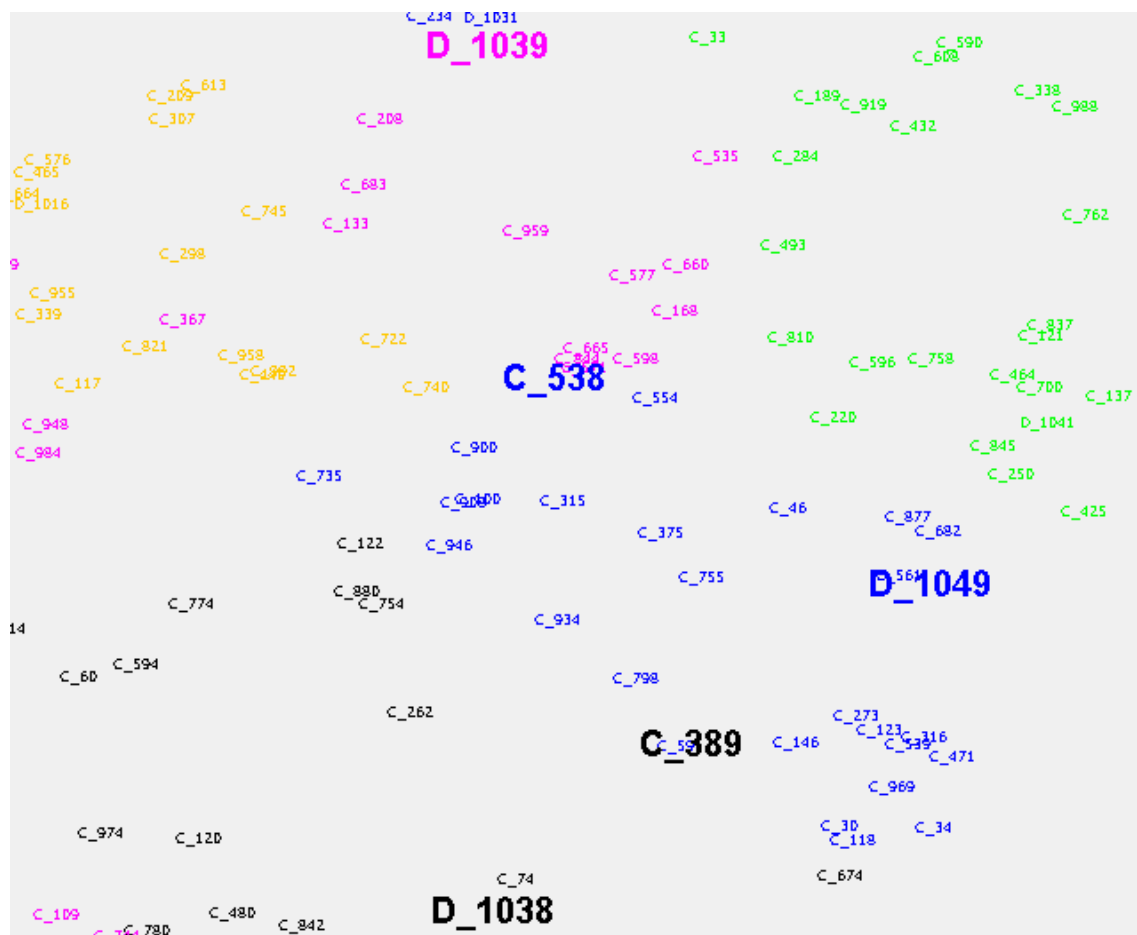
Si el depósito B no tiene capacidad para atender al cliente C_1 , se hace un intento de colocar al cliente C_1 en el depósito más cercano que tenga capacidad de atenderlo. Se recalculan los costos de los depósitos implicados y si hay mejora se aplica.

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 B (de C_2 y C_3) en caso que no tenga lugar para albergar al cliente C_1 . Pero ¿Qué cliente se saca del depósito B ? Se busca a algún cliente que esté en la misma situación que C_1 (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 A y en este caso

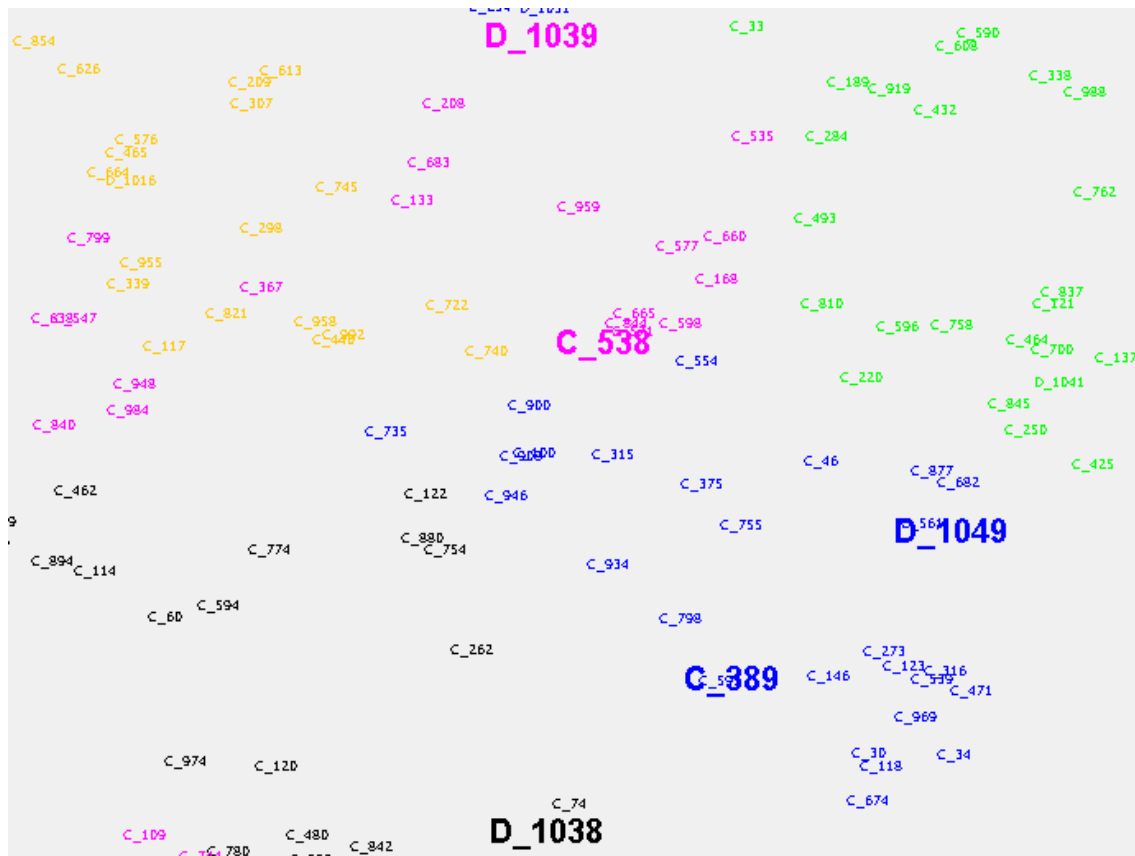
sería un intercambio de clientes entre dos depósitos. Se recalculan los costos en los depósitos implicados, y si hay mejora se hacen los cambios.

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:



3.2.4 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 se tiene un cliente C_1 , asignado al depósito A , cuyos dos vecinos más cercanos C_2 y C_3 están asignados al depósito B . Por lo tanto, se puede decir con seguridad de que hay una ruta en B que podría visitar al cliente C_1 reduciendo el costo de distancia. Dado esta afirmación, el cliente C_1 se pasa al depósito B . 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.

3.2.5 Justificación de las heurísticas implementadas

Con los algoritmos AER y AEL, se realizan búsquedas en el espacio de soluciones. Al encontrar alguna mejora, se toma la misma y se realizan nuevas iteraciones. La estrategia utilizada se basa en la suposición que al cambiar un cliente “enajenado” de un depósito, la probabilidad de encontrar una mejora es alta. Es de esperar que el costo de las rutas en el depósito origen (el depósito del cliente enajenado antes de realizar el cambio) mejoren al sacar al enajenado, y también es de esperar que el sobre costo en el depósito destino (el depósito al cual se cambia al enajenado) no sea demasiado.

3.2.6 Comparación AEL y AER

La principal diferencia entre los algoritmos AEL y AER, es que en el AER, no se recalculan los costos de las rutas en cada iteración para saber si hay una mejora del costo total. En el AEL, se recalculan los costos de las rutas únicamente en los depósitos implicados en el movimiento del cliente enajenado. Los nombres “R” (rápido) y “L” (lento) provienen del tiempo computacional en obtener los resultados en cada caso. En el algoritmo rápido, se parte de la suposición de que la mayoría de las veces en que se cambia un cliente enajenado, hay una mejora. El algoritmo lento obtiene mejores resultados que el rápido, pero el tiempo de cómputo es más elevado. En la sección de resultados se confirman estas afirmaciones para todos los casos de prueba realizados.

3.2.7 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 2.3.2.1. 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.

La decisión de optar por este algoritmo se basa principalmente por su reputación en toda la literatura encontrada. Esto implica que hay mayor información de su ejecución. Entre sus características principales es fácil de implementar y se obtienen buenos resultados con bajos costos computacionales.

3.2.8 Post Optimización

En la solución implementada, se agregaron dos métodos de post-optimización 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 post-optimización Intra-ruta e Inter-ruta. En la solución implementada, la post-optimización Inter-ruta incluye la post-optimización Intra-ruta. Por lo cual las opciones a utilizar serían: C & W, o C & W con post-optimización intra-ruta, o C & W con post-optimización intra-ruta e inter-ruta

La post-optimización **Intra-ruta** se basa en el método λ -intercambio descrito en 6.1 del estado del arte.

Consiste en eliminar λ arcos (camino entre dos clientes consecutivos) de la solución ($\lambda > 1$) 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 [30].

En el mismo se propone 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 rutas $(v_{ih}, v_{jh}, v_{kh}, v_{lh})$, $h = 1, 2$. Entonces se prueban los siguientes seis movimientos para buscar mejoras:

- 1) Insertar v_{j1} entre v_{i2} y v_{j2}
- 2) Insertar v_{j2} entre v_{i1} y v_{j1}
- 3) Intercambiar v_{j1} y v_{j2}
- 4) Insertar (v_{j1}, v_{k1}) entre v_{i2} y v_{j2}
- 5) Insertar (v_{j2}, v_{k2}) entre v_{i1} y v_{j1}
- 6) Intercambiar (v_{j1}, v_{k1}) con (v_{j2}, v_{k2})

La combinación que presenta el costo menor es la elegida.

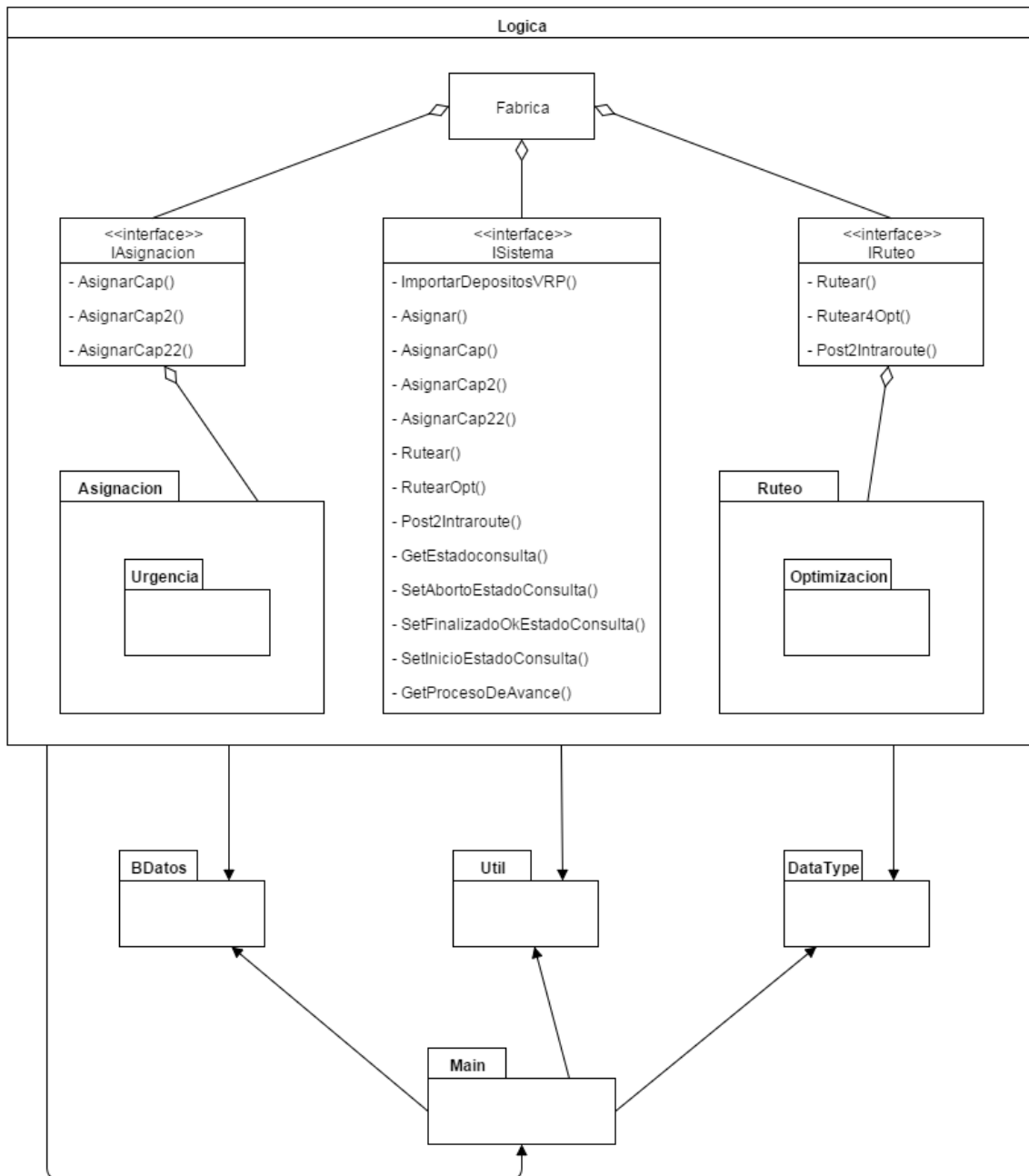
Capítulo 4

Implementación

Para la implementación de la aplicación descrita en la sección anterior se eligió el lenguaje de programación Java. La versión utilizada fue 1.8.0_45 con el entorno de desarrollo Eclipse Luna. La decisión de usar este entorno de desarrollo se debió al conocimiento y dominio sobre el mismo de los integrantes del proyecto. 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 [29].

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



4.1 Interfaz Gráfica

Para la interfaz gráfica se utilizaron los componentes de swing de java. 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 gráfica 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 la asignación de valores de parámetros para la ejecución de los algoritmos. Estos parámetros pueden ser holgura de los depósitos, tiempos de cálculo, mejoras que corresponden a los criterios de parada de los algoritmos implementados. Para los algoritmos de optimización se cuenta con el parámetro lambda.

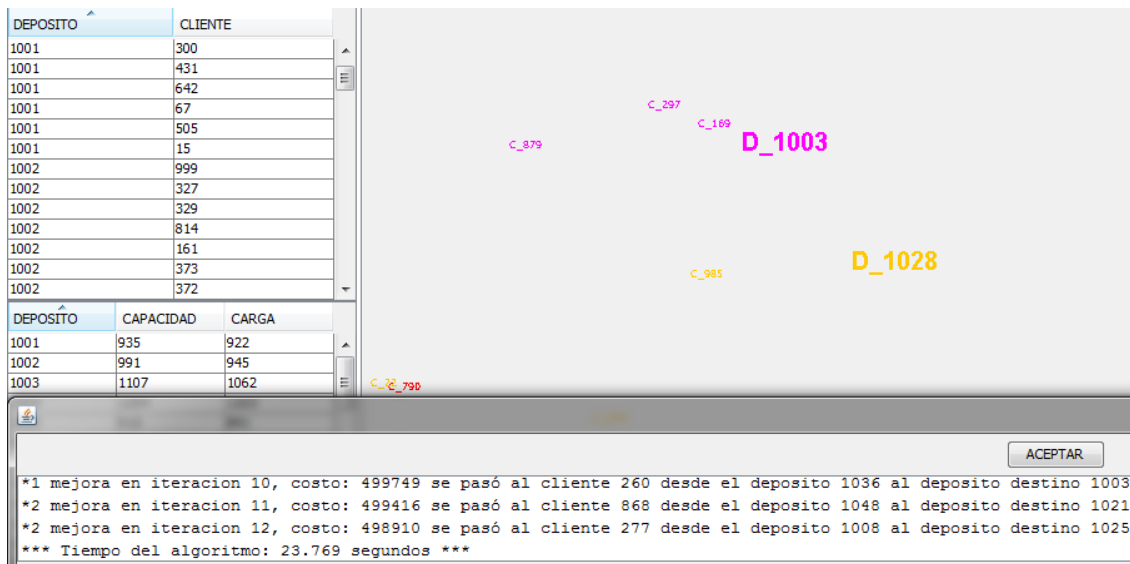
Por otro lado la capa grafica se comunica con el paquete “bdatos”, que es un controlador del archivo de datos. El archivo de datos contiene la información de los depósitos y clientes según el formato TSPLib como se comentó anteriormente. Durante la carga del archivo de datos los mismos se cargan en memoria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

Como es de esperar la comunicación principal de la capa gráfica es con la capa lógica donde se invocan 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 las distintas etapas de ejecución del algoritmo en pantalla, se implementó un sistema de “logs” de mensajes así como otro sistema para resaltar un nodo en particular en la interfaz gráfica.

Para desplegar mensajes en el sistema de logs, 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 mapa a través de la función *public void setResaltados (DataType Nodo)*;

A continuación podemos observar una captura de pantalla luego de la ejecución del algoritmo AEL. En la misma se pueden observar los mensajes en la ventana de logs 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 JPaneletiquetas.java, a través de un listener del botón correspondiente al algoritmo elegido. La acción de apretar un botón genera un evento de llamada a la función correspondiente. Por ejemplo, para el caso de llamada al algoritmo de asignación por urgencia se utiliza:

```
Fabrica.getInstance().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).

4.2 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 presentación, interactuando con el controlador "bdatos" se encarga de la manipulación del archivo de datos. 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 en la documentación de la librería TSPLib [29]. 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.
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 particular de la máquina de estados:

Si el token `EDGE_WEIGHT_TYPE` tiene el valor `"EXPLICIT"` se tienen dos opciones para la lectura de datos, y se tiene que leer 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.

4.3 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.

Como se dijo en la sección 4.1 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: Asignación y Ruteo. El paquete Asignación se encarga de los algoritmos de asignación y mejoras de asignación y el paquete Ruteo, el cual contiene los algoritmos de post-optimización y ruteo.

4.3.1 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: Colección de DTAsignacion. DTAsignacion 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: Clase heredada de clientes que 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 enajenado va a ser: el cliente (de tipo *clienteCap2* pues es de interés la información de sus dos vecinos más cercanos), el depósito al cual el cliente está asignado y el depósito al que pertenecen sus dos vecinos más cercanos. Finalmente tengo el valor de “mu” para el cliente.

4.3.2 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)`

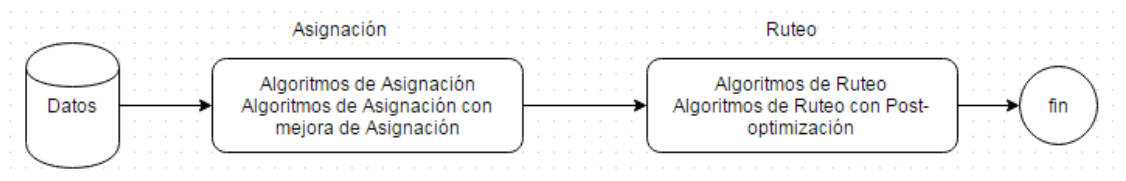
Entrada: DTAsignacion es el resultado de la asignación para un depósito.

Capacidad de los vehículos.

Salida: Colección 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)

4.3.3 Flujo del programa



4.4 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.

4.4.1 Representación gráfica 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 generan puntos aleatoriamente para buscar una solución a un problema, se genera una representación gráfica y se despliega en pantalla. Así mismo se transforma esta representación a coordenadas cartesianas a modo de poder representarla. Esta transformación no es exacta sino que aproximada. El algoritmo de transformación consiste en seleccionar un nodo inicial, y luego a través de la generación de números aleatorios, seleccionar los que cumplen más cercanamente con la matriz de distancias. De esta forma se agregan uno a uno los clientes hasta obtener el grafo completo.

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.

```

TYPE : CVRP
DIMENSION : 19
EDGE_WEIGHT_TYPE : EXPLICIT
EDGE_WEIGHT_FORMAT : LOWER_COL
DISPLAY_DATA_TYPE : NO_DISPLAY
CAPACITY : 140
EDGE_WEIGHT_SECTION
555
611      145
418      137      218
503      52      178      85
435      268      207      201      286
748      155      301      298      209      422
392      378      507      418      329      622      325
435      237      176      170      255      31      391      590
711      131      276      273      184      395      75      276

```

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 más 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.

4.4.2 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 // número de clientes  
d=50 // número de depósitos  
max_x=10000 // coordenada máxima de x.  
max_y=10000 // coordenada máxima de y  
RANDOM=1234 // semilla para el generador de números 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 depósito:

```
Demanda_total = sum (demanda(ci))
```

Donde n representa al número 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

5.1 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 distintos formatos de matrices (euc_2D, Geo, ATT, Matrix) los cuales se detallan en la documentación de TSPLib [31].

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. Tampoco se encontraron resultados oficiales para los casos de MDVRP sin capacidades en los depósitos a modo de poder comparar con los resultados de los casos de prueba propuestos. Esto motivo a generar diversos casos de prueba para comparar los resultados entre sí.

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 55; 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	50 depositos de capacidad de oferta 889	50 depositos de capacidad de oferta 969
1000 clientes con demanda variable entre [25,55]	1000 clientes con demanda variable entre [25,55]	1000 clientes con demanda variable entre [25,55]
Representacion en Coordenadas Euclideanas.	Representacion en Coordenadas Euclideanas	Representacion en Coordenadas Euclideanas.

Luego se generó 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 se cuenta 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]	50 depositos de capacidad de oferta variable entre [444,1333]	50 depositos de capacidad de oferta variable entre [484,1454]
1000 clientes con demanda variable entre [25,55]	1000 clientes con demanda variable entre [25,55]	1000 clientes con demanda variable entre [25,55]
Representacion en Coordenadas Euclideanas.	Representacion en Coordenadas Euclideanas	Representacion en Coordenadas Euclideanas.

Casos de prueba Misceláneos.

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 y 10.000 nodos.

Para representar 5.000 clientes se utilizó el caso de estudio test5250.txt. En la sección de conclusiones se detallarán los resultados de esta prueba sin entrar mucho en detalles pues el resultado 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 gráfica “similar” y que cumpla las propiedades de distancia descritas en la matriz.

5.2 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. Se encontraron publicados resultados del caso de prueba (sin capacidades en los depósitos) [31], en donde no se especifican los métodos utilizados para la resolución por lo tanto no tiene sentido hacer una comparación directa.

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 deberían alterar significativamente el resultado final, comparando los casos de prueba B y C.

5.3 Ejecución y Resultados

A modo de poder comparar las distintas ejecuciones se tomaron 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 descrito en el manual de usuario condicionan dicho resultado. Por ejemplo, en el 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:

$$Penalidad = 100 * \frac{\sum_{i=1}^n cap(di) - oferta(di) \text{ si } cap(di) < oferta(di)}{\sum_{i=1}^n oferta(di)}$$

$cap(di)$ Es la demanda de todos los clientes asignados al depósito i .

$oferta(di)$ 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.

5.3.1 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 depósito con el algoritmo 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%

5.3.2 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 $\lambda=3$. Se obtuvieron los siguientes resultados. Notar que la columna de penalidad es igual a las pruebas anteriores ya que no hay cambios inter-depósitos. 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

5.3.3 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

5.4 Análisis de Resultados.

5.4.1 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 determinar la penalidad. Esto sucede por la implementación del algoritmo de urgencia, donde los últimos clientes en asignar se asignan directamente al depósito más 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 resultado 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)

5.4.2 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 depósito manualmente sin contar con una referencia.

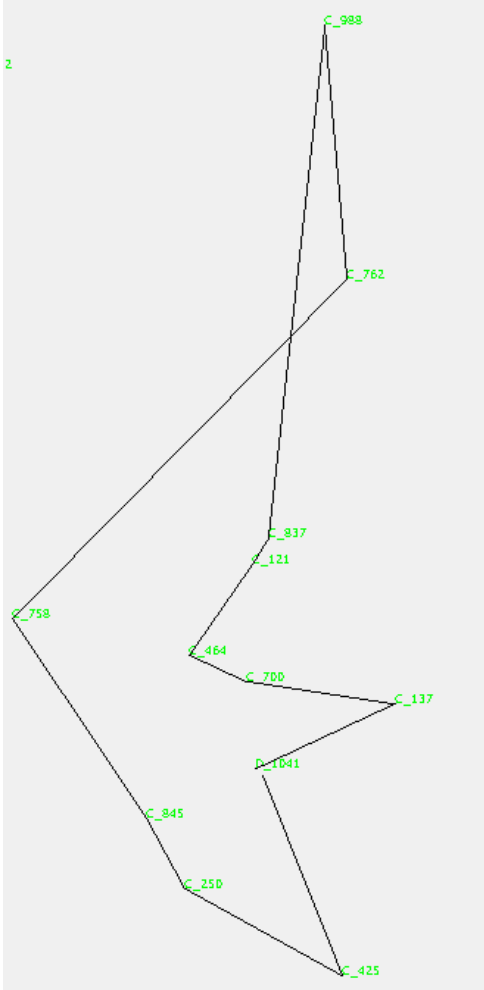
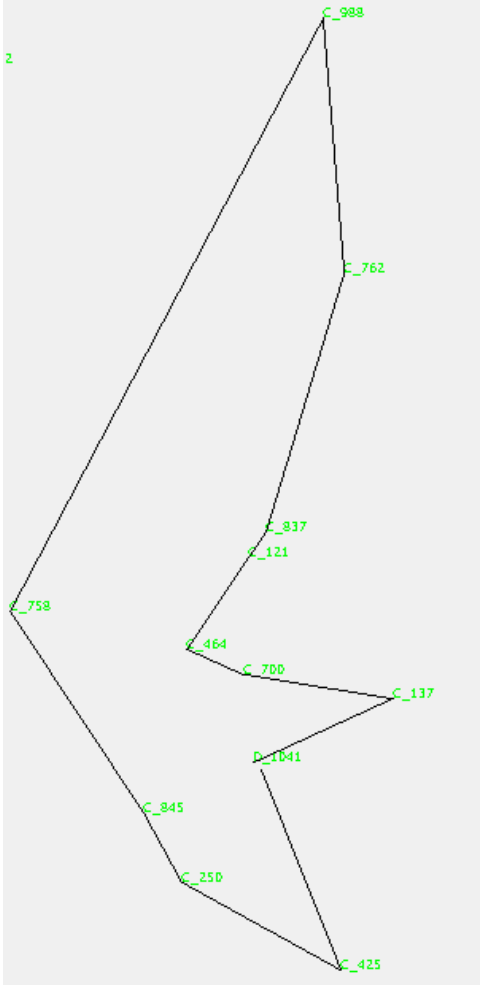
Para los otros algoritmos implementados, dado que TSPLib no considera las capacidades de los depósitos, no se puede hacer una comparación directa de los valores con los resultados de la librería. El tiempo de demora en la ejecución fue menor en el algoritmo de asignación.

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 resolver el problema de los clientes enajenados, ya sea en su forma rápida o lenta 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 continuación se muestra un ejemplo de salida del algoritmo de post optimización inter-opt. El mismo, como se muestra en la siguiente figura mejoró el resultado para el caso de estudio B1.

	
Clarke & Wright	Clarke & Wright + Intra-Route

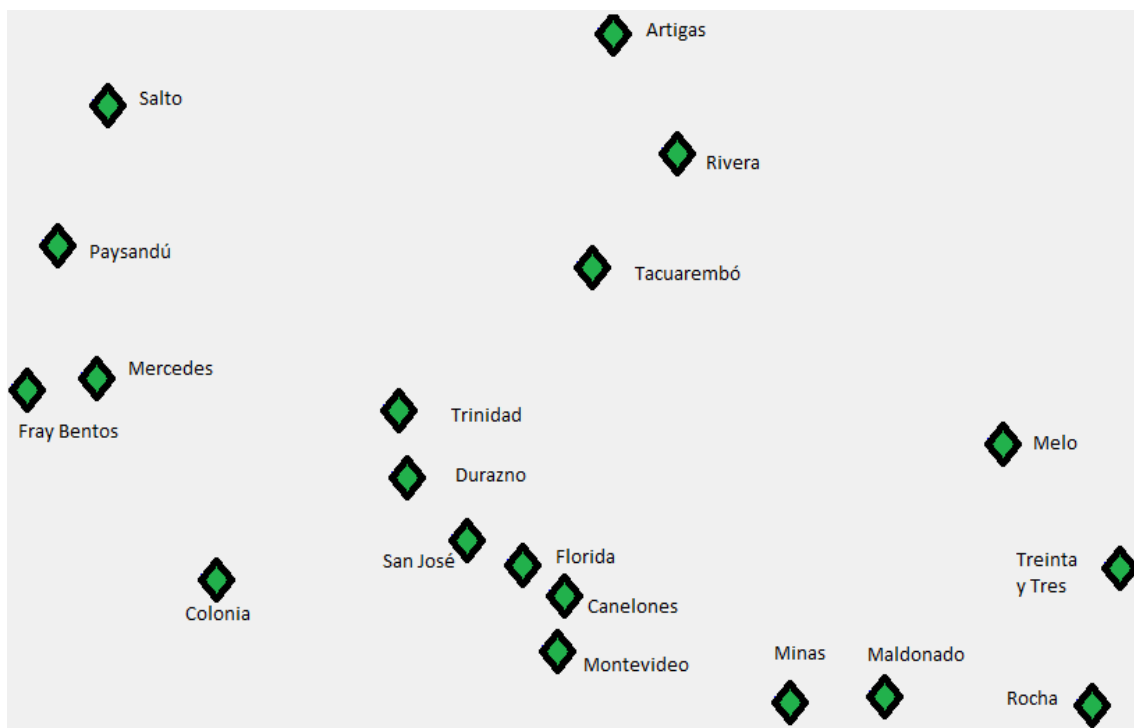
5.4.3 Casos Misceláneos

Para la ejecución del caso de prueba con 5000 clientes, el algoritmo de Enajenados Rápido corrió en 61 segundos mientras que el algoritmo de Ruteo finalizó su ejecución unos pocos segundos más tarde. Los algoritmos de post optimización, al ser locales en un depósito también fueron ejecutados en segundos. Por otro lado el algoritmo de asignación Enajenados Lento demora 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 gráfica del resultado del caso javier2.txt. 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 búsqueda de mejores algoritmos de solución de MDVRP, es un problema bajo investigación y continuo crecimiento, donde todos los meses se pueden encontrar nuevas publicaciones académicas. 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.

A través de la representación gráfica de las rutas y los depósitos, se pudo observar en forma amigable el resultado y evolución de los distintos algoritmos. La implementación del resaltado de nodos, permitió visualizar los cambios ocurridos en cada iteración. A su vez, una consola muestra mensajes sobre los costos y los cambios realizados en la asignación. Con estas utilidades resultó sumamente práctico a la hora de analizar y experimentar con algoritmos.

Resultó de gran utilidad la utilización del formato estándar especificado por TSPLib para el planteo de problemas de ruteos de vehículos. La solución implementada cumple con todas las variantes de este estándar lo cual brinda compatibilidad y permite ser utilizada para cualquier problema que cumpla con el formato. A su vez, el ejecutable, al estar desarrollado en Java se puede correr prácticamente en cualquier sistema operativo.

La solución permite definir parámetros que controlan la ejecución de los algoritmos. Con esto se logra mayor control sobre la ejecución y permite una experimentación más en profundidad sobre las posibles soluciones. También se brinda la posibilidad de indicar una holgura que permita la sobrecarga de depósitos. De esta forma se pueden correr el algoritmo con diferentes holguras y comparar los resultados obtenidos.

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 se llega a la conclusión que el algoritmo de Enajenados

Rápido corrió en un tiempo relativamente corto y mejoró la solución respecto a los métodos de asignación por urgencia. El algoritmo de Ruteo finalizó su ejecución en pocos segundos, y los algoritmos de post optimización, al ser locales para cada depósito también fueron ejecutados en segundos. El algoritmo de asignación Enajenados Lento demoró un tiempo considerable pero llegó 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 y sus variantes. Con este proyecto de grado se establece un ambiente de trabajo el cual permite implementar nuevos algoritmos, lo cual de forma sencilla los permite agregar en la aplicación.

Por otro lado notamos que la característica de los clientes a los cuales llamamos enajenados es solamente una de las particularidades encontradas. Así como esta característica, pueden existir muchas otras particularidades en los clientes y depósitos cuyo 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 las etapas de 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.

El proyecto se enfocó en MDVRP con capacidades, quedando pendiente a futuro considerar otras variantes en la naturaleza de los problemas como ser ventadas de tiempo, periodicidad, flota heterogénea, entre otros.

También se podría agregar, dado que la tecnología lo permite fácilmente, la utilización de coordenadas geográficas para la visualización de los problemas MDVRP reales. Para esto, habría que superar la limitación existente en TSPLib la cual consiste en que las distancias cuando se representan con matrices, carecen de información para la

representación geográfica real. También se podrían integrar muchas utilidades de sistemas de información geográfica, para las representaciones y cálculos de distancia.

Anexo

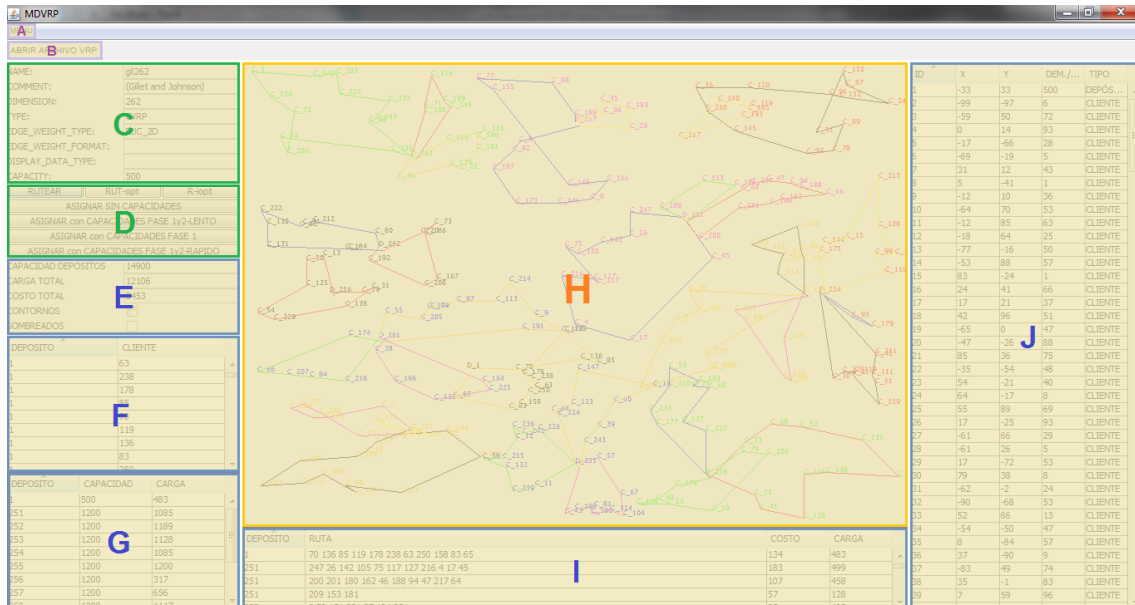
Comparación de las dos heurísticas

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:	Resultado:	Mejora %
	Asignación sin Mejora	Asignación con 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:

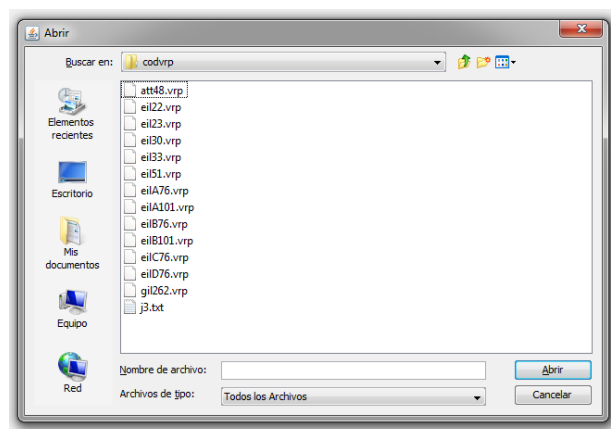
- A. **Menú:** Al hacer clic sobre este botón se desplegará la opción de Configuración.
- B. **Abrir Archivo:** Para abrir un archivo de prueba.
- C. **Información del Archivo de Prueba:** Esta espacio indica la información del archivo de prueba utilizado. Contiene la información de Nombre, Formato de Entrada, Dimensión, Tipo, entre otras.
- D. **Botones de Algoritmos:** Contiene los distintos botones para ejecutar los distintos algoritmos de Asignación, Ruteo y Post-Optimización.
- E. **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.
- F. **Listado de Asignación Depósito-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 depósito o del cliente.

- G. **Listado de Depósitos:** Contiene la información de capacidad de cada depósito y su carga. Se señala 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.
- H. **Mapa:** Espacio donde se pueden observar los cliente y depósitos situados en el mapa, como también las rutas que componen la solución. El mapa permite aplicar **zoom** para poder visualizar mejor las distintas rutas.
- I. **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.
- J. **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 coordenada X e Y, la demanda/capacidad o el tipo.

Procedimientos descriptivos

Abrir un archivo de prueba

- a. Para abrir un archivo de prueba hacer clic en “Abrir Archivo VRP” se abrirá una ventana:



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

- b. 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.

NAME:	gil262
COMMENT:	(Gillet and Johnson)
DIMENSION:	262
TYPE:	CVRP
EDGE_WEIGHT_TYPE:	EUC_2D
EDGE_WEIGHT_FORMAT:	
DISPLAY_DATA_TYPE:	
CAPACITY:	500
<input type="button" value="RUTEAR"/> <input type="button" value="RUT-opt"/> <input type="button" value="R-ipt"/>	
<input type="button" value="ASIGNAR SIN CAPACIDADES"/>	
<input type="button" value="ASIGNAR con CAPACIDADES FASE 1y2-LENTO"/>	
<input type="button" value="ASIGNAR con CAPACIDADES FASE 1"/>	
<input type="button" value="ASIGNAR con CAPACIDADES FASE 1y2-RAPIDO"/>	
CAPACIDAD DEPOSITOS	14900
CARGA TOTAL	12106
COSTO TOTAL	-
CONTORNOS	<input type="checkbox"/>
SOMBREADOS	<input type="checkbox"/>

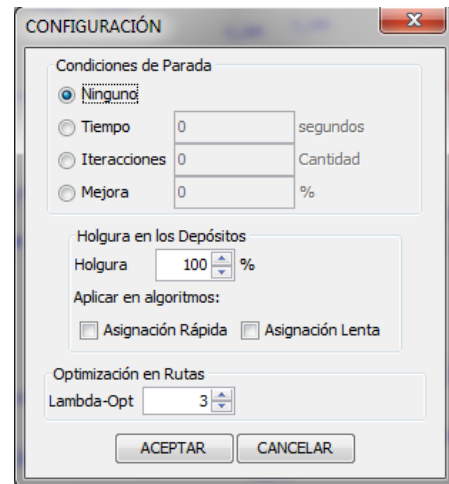
En el espacio de ”Listado de Clientes-Depósitos” se cargará la información de los mismos, al igual que su ubicación en el mapa.

ID	X	Y	DEP./...	TIPO
1	-33	33	500	DEPÓS...
2	-99	-97	6	CLIENTE
3	-59	50	72	CLIENTE
4	0	14	93	CLIENTE
5	-17	-66	28	CLIENTE
6	-69	-19	5	CLIENTE
7	31	12	43	CLIENTE
8	5	-41	1	CLIENTE
9	-12	10	36	CLIENTE
10	-64	70	53	CLIENTE
11	-12	85	63	CLIENTE
12	-18	64	25	CLIENTE
13	-77	-16	30	CLIENTE
14	-53	80	57	CLIENTE
15	83	-24	1	CLIENTE
16	24	41	66	CLIENTE
17	17	21	37	CLIENTE
18	42	96	51	CLIENTE
19	-65	0	47	CLIENTE
20	-47	-26	88	CLIENTE
21	85	36	75	CLIENTE
22	-35	-54	48	CLIENTE
23	54	-21	40	CLIENTE
24	64	-17	8	CLIENTE
25	55	89	69	CLIENTE
26	17	-25	93	CLIENTE
27	-61	66	29	CLIENTE
28	-61	26	5	CLIENTE
29	17	-72	53	CLIENTE
30	79	38	8	CLIENTE
31	-62	-2	24	CLIENTE
32	-90	-68	53	CLIENTE
33	52	66	13	CLIENTE

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 iteraciones o por porcentaje de mejora. En caso de no querer configurar ninguna condición de parada los algoritmos continúan la ejecución hasta encontrar una solución óptima.

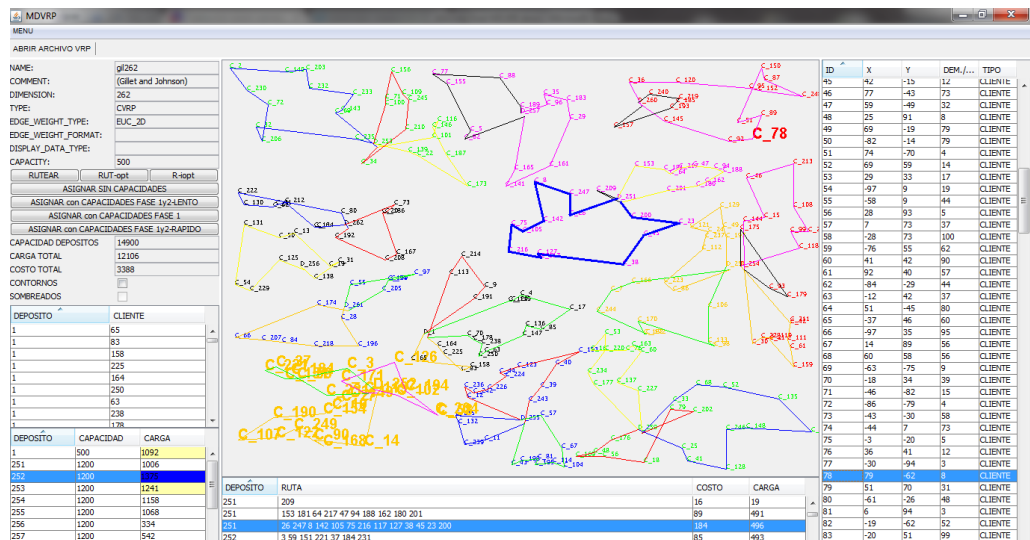


Se podrá también configurar el porcentaje de holgura de los depósitos y también para cuál de los algoritmos de enajenado aplica (AER o AEL).

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

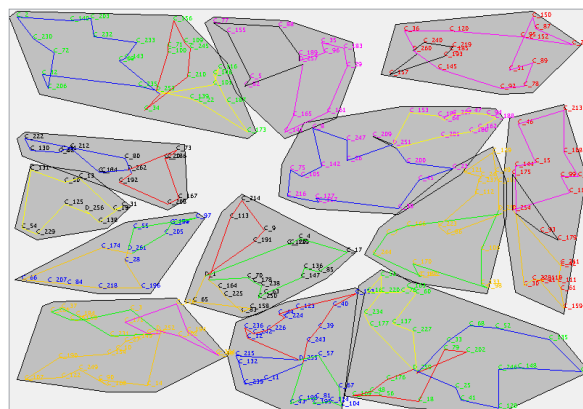
Interacción con el Mapa

El mapa cuenta con la funcionalidad de poder resaltar Depósitos, Cliente y Rutas. Al hacer clic en un Cliente/Depósito de la “Lista de Clientes-Depósitos” se resaltarán en el mapa dicho Cliente/Depósito pudiendo así poder ubicar más 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 más específico y un “Zoom Out” que se pasa a un plano más 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”.

RUTEAR	RUT-opt	R-ipt
ASIGNAR SIN CAPACIDADES		
ASIGNAR con CAPACIDADES FASE 1y2-LENTO		
ASIGNAR con CAPACIDADES FASE 1		
ASIGNAR con CAPACIDADES FASE 1y2-RAPIDO		

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

DEPOSITO	CLIENTE
1	65
1	83
1	158
1	225
1	164
1	250
1	63
1	238
1	177R

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 señala). También se actualiza el mapa donde se muestra ahora como quedaron asignados los distintos clientes a los distintos depósitos.

DEPÓSITO	CAPACIDAD	CARGA
1	500	1092
251	1200	1006
252	1200	1375
253	1200	1241
254	1200	1158
255	1200	1068
256	1200	334
257	1200	542
258	1200	910

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.

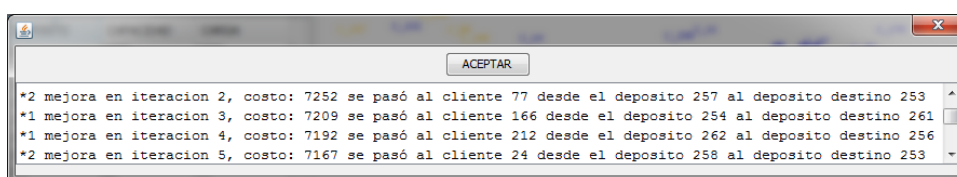
Asignar con Capacidades Fase 1 y 2 – Rápido

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 algoritmo a diferencia de los demás cuenta con una consola de progreso que nos va indicando los movimientos 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 Ruteo. 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.

DEPÓSITO	RUTA	COSTO	CARGA
1	191 9 214 113	88	210
1	124 119 4 17 85 136 147 63 250 238 178 70	126	448
1	65 158 83 225 164	60	434
251	209	16	19
251	153 181 64 217 47 94 188 162 180 201	89	491

También 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

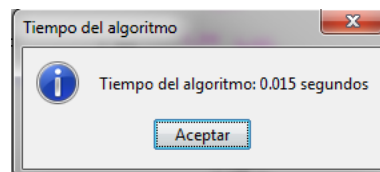
Este botón invoca al Algoritmo de Post-Optimización Intra-Ruta.

Rut-iopt

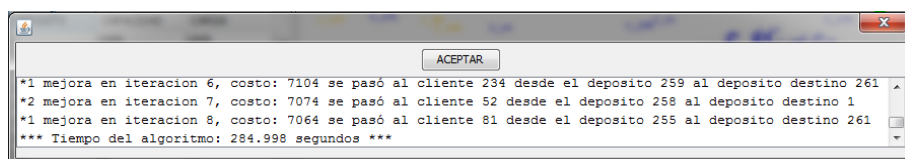
Este botón invoca al Algoritmo de Post-Optimización Inter-Ruta.

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» n° 5, pp. 45-64, Diciembre 2013.
- [2] A. Schrijver, «On the History of Combinatorial Optimization (Till 1960)» de *Handbooks in Operations Research and Management Science*. ISBN: 978-0-444-51507-0, K. Aardal, G.L. Nemhauser and R. Weismantel, 2005, pp. 1-68.
- [3] L. Bodin, «Routing and scheduling of vehicles and crew: The State of the Art» *Comput. & Ops Res.*, vol. 10, n° 2, pp. 63-211, 1983.
- [4] Y. Wang, «Research of Multi-Depot Vehicle Routing Problem by Cellular Ant Algorithm» *Journal of Computers*, vol. 8, n° 7, pp. 1722-1727, Julio 2013.
- [5] P. Surekha y D. S. Sumathi, «Solution to Multi-Depot Vehicle Routing Problem Using Genetic Algorithms» *World Applied Programming*, vol. 1, n° 3, pp. 118-131, Agosto 2011.
- [6] J. Carlsson, D. Ge, A. Subramaniam, A. Wu y Y. Ye, «An ant colony optimization technique for solving min–max Multi-Depot Vehicle Routing Problem» de *Swarm and Evolutionary Computation*, vol. 13, 2013, pp. 63-73.
- [7] G. B. Dantzig y J. H. Ramser, «The Truck Dispatching Problem» *Management Science*, vol. 6, n° 1, pp. 80-91, Octubre 1959.
- [8] B. L. Golden, «Vehicle Routing Problems: Formulations and Heuristic Solution Techniques» *Technical Reports*, n° 113, Agosto 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, n° 4, pp. 393-410, Noviembre 1954.
- [10] R. M. Karp, «Reducibility Among Combinatorial Problemas» de *Complexity of Computer Computations*, New York, R. E. Miller and J. W. Thatcher, 1972, pp. 85-103.
- [11] S. N. Kumar y R. Panneerselvam, «A Survey on the Vehicle Routing Problem and Its Variants» *Intelligent Information Management*, n° 4, pp. 66-74, 2012.
- [12] R. Baldacci, M. Battarra y D. Vigo, «Routing a Heterogeneous Fleet of Vehicles» *Technical Report DEIS OR.INGCE 2007/1*, Enero 2007.
- [13] C. Tan y J. Beasley, «A Heuristic Algorithm for the Period Vehicle Routing Problem» *OMEGA Int. J. of Mgmt Sci.*, vol. 12, n° 5, pp. 497-504, 1984.
- [14] P. Francis y K. Smilowitz, «The Period Vehicle Routing Problem with Service Choice» *Transportation Science*, vol. 40, n° 4, pp. 439-454, 2006.
- [15] R. Bowerman, B. Hall y P. Calamai, «A Multiobjective Optimization Approach to Urban School Bus Routing: Formulation and Solution Method» Departament of Systems Design Engineering, University of Waterloo, 1995.

- [16] 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.
- [17] 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.
- [18] R. V. Kulkarni y P. R. Bhawe, «Integer programming formulations of vehicle Routing Problems» *Euroean Journal of Operational Research*, vol. 20, pp. 58-67, 1985.
- [19] J. H. Restrepo y P. D. Medina, «Un problema logístico de Ruteo de vehículos y una solución con la heurística R: Un caso de estudio» *Scientia et Technica*, nº 37, pp. 407-411, 2007.
- [20] S. Martello, G. Laporte, M. Minoux y C. Ribeiro, *Survey Combinatorial Optimization*, North Holland-Amsterdam, New York - Oxford, Tokyo: North Holland. ISBN: 0 444 70136 2, 1987.
- [21] A. Mingozzi y A. Valletta, «An exact algorithm for period an multi-depot vehicle routing problems» 2003.
- [22] S. Nieto Isaza, J. Lopez Franco y N. Herazo Padilla, «Desarrollo y codificación de un Modelo Matemático para la Optimización del MDVRP» de *10th Latin American and Caribbean Conference for Engineering and Technology*, Panama, 2012.
- [23] A. Olivera, «Heurísticas para Problemas de Ruteo de Vehículos» Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Montevideo, Uruguay., 2004.
- [24] F. Glover y G. A. Kochenberger, *Handbook of Metaheuristics*, New York, Boston, Dordrecht, London, Moscow: Kluwer Academic Publishers, 2003.
- [25] J. Lysgaard, «Clarke & Wright's Savings Algorithm» Department of Management Science and Logistics, The Aarhus School of Business, Dinamarca, 1997.
- [26] D. Giosa, L. Tansini y O. Viera, «New assignment algorithms for the multi-depot vehicle routing problem» *Journal of the Operational Research Society*, vol. 53, nº 9, pp. 977-984, 2002.
- [27] L. Tansani y O. Viera, «New measures of proximity for the assignment algorithms in the MDVRPTW» *Journal of Operational Reserch Society*, vol. 57, nº 3, pp. 241-249, 2006.
- [28] L. Wen y F. Meng, «An Improved PSO for the Multi-Depot Vehicle Routing Problem with Time Windows» *Pacific-Asia Workshop on Computational Intelligence and Industrial Application*, pp. 852-856, 2008.
- [29] Documentación Librería TSPLib, «<http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/DOC.PS>».
- [30] 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.
- [31] R. -. K. -. U. Heidelberg, «Discrete and Combinatorial Optimization» <http://comopt.ifl.uni-heidelberg.de/software/TSPLIB95/>.

