



# UN MÉTODO DE RESOLUCIÓN PARA EL PROBLEMA DE RUTEO DE VEHÍCULOS CON MÚLTIPLES DEPÓSITOS.

**Informe Final**

**Valeria Rocha**

**Javier Salaberry**

***Tutores***

**Sandro Moscatelli**

**Omar Viera**

**Marzo, 2019**

**Instituto de Computación**

**Facultad de Ingeniería – Universidad de la República**

**Montevideo, Uruguay**

# Resumen

En la actualidad las empresas entienden que pueden mejorar los costos de sus operativas utilizando herramientas logísticas para lo que respecta a la distribución de productos. Como valor agregado a la optimización de costos que alcanzan, está la disminución del impacto ambiental asociado con la disminución del uso de combustibles fósiles.

El Problema de Ruteo de Vehículos (VRP) en su forma más simple busca resolver la problemática de la distribución de productos y/o servicios, diseñando rutas que atienden la demanda de los clientes dispersos geográficamente desde un depósito utilizando una flota homogénea de vehículos.

Este proyecto plantea trabajar en una generalización del bien conocido VRP, donde se trabaja con más de un depósito, denominado Problema de Ruteo de Vehículos con Múltiples Depósitos, de ahora en más, MDVRP por su nombre en inglés.

MDVRP es un problema NP-Hard, por lo cual no existen algoritmos exactos eficientes, con un costo computacional razonable para resolver todas sus instancias. Debido a dicha característica del problema, se han desarrollado a lo largo de las últimas décadas una serie de heurísticas y metaheurísticas con el fin de obtener buenas soluciones en menor tiempo que los métodos exactos.

El objetivo de este proyecto es resolver la variante de MDVRP con ventanas de tiempo, capacidades en depósitos y clientes y flota homogénea de vehículos de forma eficiente, implementando algoritmos de asignación, ruteo y post optimización.

Para resolver la problemática objetivo se desarrolló un sistema web con arquitectura cliente-servidor que permite resolver MDVRP utilizando tres tipos de algoritmos de asignación y post optimización para ruteo y asignación. La solución desarrollada permite además poder analizar tanto el problema como el resultado de los algoritmos en un contexto geográfico de forma muy amigable con el usuario.

Los resultados obtenidos con los algoritmos desarrollados fueron los esperados, por lo que el sistema retorna una buena solución para el problema en un tiempo razonable de ejecución.

**Palabras claves**: Ruteo de Vehículos con Múltiples Depósitos, Logística, Asignación, Distribución de Productos, MDVRP, VRP.

# Contenido

# Capítulo 1

## Introducción

En este documento se presenta el Informe Final sobre “Un método de resolución para el problema de ruteo de vehículos con múltiples depósitos” en el contexto del Proyecto de Grado de la carrera de Ingeniería en Computación de los estudiantes Valeria Rocha y Javier Salaberry.

## Contexto

La distribución de productos desde los depósitos o plantas hasta los consumidores finales es un problema clave para la gestión logística, teniendo un rol importante en la misma debido a los costos que implica. [17]

El problema de ruteo de vehículos busca resolver la problemática de la distribución de productos y/o servicios, diseñando rutas que atienden la demanda de los clientes dispersos geográficamente desde un depósito utilizando una flota de vehículos. [8]

La resolución de este tipo de problemas es de suma importancia para los sistemas de distribución y es por este motivo que han sido estudiados ampliamente durante los últimos 60 años. [17]

Distintas necesidades reales y restricciones en el campo de la logística llevaron a la definición de variantes del problema original de ruteo de vehículos generando así una variedad importante de problemas a resolver. Para todas las variantes se mantiene el objetivo principal de optimizar el uso de los recursos de transporte y la satisfacción de los clientes. [17]

El problema que se desarrolla y resuelve en este Proyecto de Grado es el Problema de Ruteo de Vehículos con Múltiples Depósitos, de ahora en más MDVRP por su nombre en inglés. Este problema cuenta con una serie de clientes que demandan ciertos productos, una flota de vehículos con capacidad limitada que realizan la distribución de los productos, y varios depósitos donde se almacenan dichos productos. En el contexto de este proyecto se plantea que los vehículos comienzan y finalizan la ruta en el mismo depósito y que los clientes reciben una única visita de un vehículo. Adicionalmente se plantea que los clientes cuenten con una ventana horaria durante la cual debe efectuarse la entrega. Tomando en cuenta las consideraciones mencionadas anteriormente se busca satisfacer las necesidades de los clientes minimizando la distancia recorrida y realizando las entregas dentro de las ventanas horarias correspondientes.

Para resolver el problema planteado se definieron ciertos requerimientos funcionales y no funcionales como por ejemplo la posibilidad de importar datos y de visualizarlos en un mapa, así como brindarle al usuario la opción de elegir entre distintos algoritmos de asignación y niveles de post optimización entre otros. Entre los requerimientos no funcionales se encuentra por ejemplo la necesidad de desarrollar un producto portable y que permita una alta disponibilidad.

En función de los requerimientos definidos se optó por implementar un sistema web basado en la arquitectura cliente servidor con una componente geográfica importante que permite analizar el problema y su solución de una forma amigable. A su vez, se decidió encapsular los algoritmos en una librería que es fácilmente extensible. El desarrollo web fue realizado con tecnología de vanguardia, donde se eligió HTML5 como lenguaje principal mientras que para el desarrollo de la librería se utilizó el framework de desarrollo .NET en su versión 4.6.

Para realizar los testeos del producto, ante la falta de casos de prueba disponibles se decidió generar una serie de casos propios para evaluar tanto el resultado como el tiempo de ejecución.

En definitiva, se implementó un sistema portable, amigable con el usuario, sencillo de utilizar que permite resolver la problemática definida obteniendo buenos resultados en un tiempo razonable. El producto desarrollado deja el camino abierto para la incorporación tanto de funcionalidades como de algoritmos.

## Estructura del documento

El presente documento se organiza en los capítulos que se presentan a continuación.

En el Capítulo 2 se define específicamente el problema a resolver, y se especifican los algoritmos a desarrollar.

En el Capítulo 3 de este documento se presenta un resumen del documento de Estado del Arte de MDVRP abarcando desde la definición del problema y sus múltiples variantes hasta las soluciones propuestas al mismo.

Luego, en el Capítulo 4 se presenta el diseño, la arquitectura y los detalles de la implementación de la solución propuesta para resolver el problema en cuestión.

En el Capítulo 5 se presenta una descripción de los casos de prueba definidos, así como del plan de pruebas utilizado para la verificación del sistema y de los algoritmos desarrollados. Se incluyen resultados de las ejecuciones de los casos de prueba para las distintas combinaciones de algoritmos.

Luego, en el Capítulo 6 se presentan las conclusiones del proyecto y los posibles trabajos a realizar a futuro.

Finalmente, en la sección de Anexo, se encuentra el Manual de Usuario, donde se explica cómo utilizar el sistema desarrollado.

# Capítulo 2

## Definición del problema

El problema planteado este proyecto de grado es el de MDVRP (Multi Depot Vehicle Routing Problem) donde los vehículos y depósitos cuentan con capacidad limitada y tanto los depósitos como clientes tienen restricciones de ventanas de tiempo.

El problema de MDVRP como se definió en el documento del Estado del Arte consiste básicamente en optimizar el costo de las rutas que deben realizar los vehículos desde 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 que satisface totalmente su necesidad.
- Los vehículos de la flota tienen la misma capacidad.
- Los depósitos tienen capacidad.
- Se cumplen las restricciones de ventanas horarias de depósitos y clientes.

### Algoritmos de MDVRP propuestos para la solución

Durante el proceso donde se realizó el Estado del Arte acerca de MDVRP se detectó la existencia de pocos trabajos donde se tenían en cuenta restricciones de ventanas horarias.

La solución que se propone está basada principalmente en las heurísticas de dos fases para la resolución de MDVRPTW, “Cluster First, Route Second” ya que es la más prometedora para este tipo de problemas.

La solución propuesta tiene en cuenta además que el usuario del programa puede querer darle más importancia al tiempo de ejecución que a la calidad de la solución o vice-versa. Para esto se proponen 3 algoritmos de Asignación para la primera etapa de la heurística, uno clásico basado en urgencias, uno que utiliza particionamiento y uno jerárquico de agrupamiento.

Del mismo modo, estarán disponibles dos técnicas de post optimización que el usuario podrá utilizar de acuerdo a su necesidad.

En todos los algoritmos planteados se utiliza distancia euclidiana y se calcula el tiempo utilizando la velocidad de 20 km/h. Existe de todos modos la posibilidad de incorporar otras formas de calcular la distancia y el tiempo ya que todos los métodos pueden recibir la matriz de tiempos y distancias.

Asignación

- Asignación por Urgencia: Algoritmo Paralelo Simplificado
- Partición Alrededor de Medioids
- UnweightedPairGroupsMethodCentroid

Mejora Asignación

- Algoritmo Revisión Fronteras.

Ruteo

- Clarke & Wright.

Post-optimización.

- λ-intercambio
- OR-opt

## Asignación por Urgencias: Algoritmo Paralelo Simplificado

La urgencia es una de las tantas formas de definir una relación de precedencia entre los clientes; la urgencia de ser asignado es equivalente a una prioridad. Esta relación de precedencia determina el orden en que los clientes se asignan a los depósitos. Los clientes con más urgencia se asignan primero.

El Algoritmo Paralelo Simplificado es una adaptación del algoritmo para VRPTW presentado por Potvin y Rousseau. [21]

Para calcular la urgencia de un cliente en el Algoritmo Paralelo Simplificado se utilizan únicamente dos depósitos,

$$\mu_c = d\,(c, dep'') - d\,(c, dep')$$

Donde $d\,(c, dep'')$es la distancia entre el cliente y su segundo depósito más cercano y $d\,(c, dep')$es la distancia del cliente al depósito más cercano.

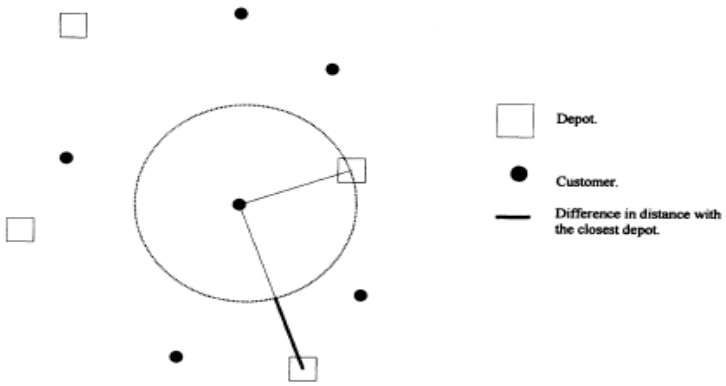



***Figura 1. Asignación por urgencias***

Una vez más, el cliente con el mayor valor de $\mu$ se asigna a su depósito más cercano. Esta heurística compara el costo de asignar un cliente a su depósito más cercano con el costo de asignarlo a su segundo depósito más cercano. El cliente más urgente es aquel para el cual $\mu$ es máximo.

Para incorporar ventanas de tiempo, la urgencia se define como:

$$\mu_c = Cercanía(c, dc'(c)) - Cercanía(c, dc''(c))\ c \in C$$

donde $Cercanía\ (i,j) = d\ (i,j)\ /\ Afinidad\ (i,j)\ \ j \in D\ , i \in C$ y $d\ (i,j)$ es la distancia de $i\ a\ j$

La medida de afinidad que se tiene en cuenta en la definición de cercanía mide el grado de similitud en términos de ventanas de tiempo de un cliente con el grupo de clientes ya asignados al depósito en cuestión.

$$Afinidad\ (i,j) = \left\{\frac{\sum_{j \in C\ (d)\ \cup\ \{d\}} e^{-(DTW(i,j)\ +\ TVij}}{|C|}\right\}\ d \in D\ \ i,j \in C$$

donde $C\ (d)$ es el conjunto de clientes ya asignados al depósito d y $TVij$ es el tiempo de viaje entre i y j.

DTW mide la distancia de las ventanas de tiempo entre clientes o entre un cliente y un depósito de la siguiente manera:

$$DTW\ (i,j) = \begin{cases} e_j - l_i\ si\ l_i < e_j \\ e_i - l_j\ si\ l_j < e_i \\ 0\ en\ otro\ caso \end{cases}$$

donde l y e representan el comienzo y final de las ventanas de tiempo.

La complejidad del algoritmo es O ($3CD + CD^2 + C^2D$) donde C es la cantidad de clientes y D la cantidad de depósitos.

La versión original del algoritmo de urgencias no contempla la restricción de capacidad de los clientes, depósitos y vehículos por lo cual fue adaptado para que incluyera el control de capacidades.

A continuación, se muestran los pasos a seguir para contemplar la restricción de capacidad de tanto de los clientes como depósitos.

1) Inicialización: crear un clúster para cada depósito.
2) Mientras haya clientes sin clúster:
   a) Calcular la urgencia para cada cliente tomando los clústeres de los dos depósitos más cercanos que cuenten con la capacidad para agregar el cliente.
   b) Tomar el cliente con mayor urgencia y agregarlo al clúster correspondiente a dicha urgencia.

## Partition Around Medoids

Partition Around Medoids (PAM) utiliza una búsqueda ávida que puede no encontrar la solución óptima, pero es más rápida que la búsqueda exhaustiva. Los pasos por seguir son:

1. Inicialización: seleccionar k de los n puntos como medoids.
2. Asociar cada punto al medoid más cercano.
3. Mientras el costo de la configuración disminuya:
    1. Para cada medoid m, para cada no medoid o:
        1. Intercambiar m y o, recalcular el costo (suma de la distancia de los puntos a sus medoids).
        2. Si el costo total de la configuración aumentó en el paso anterior, deshacer el intercambio.

En la figura que se muestra a continuación se puede ver el cambio de medoid del clúster claro lo que produce que un cliente del clúster oscuro pase a pertenecer al clúster claro.


***Figura 2. Cambio de medoid del cluster***

Los depósitos pueden ser elegidos como los primeros medoids, y posteriormente se puede elegir tanto depósitos como clientes, mientras que se cumplan todas las restricciones.

La distancia utilizada en los algoritmos de particionamiento incorpora las ventanas de tiempo.

Para dos elementos $e_i$ y $e_j$ la distancia se calcula como:

$$WSUM\ (i,j)\ =\ wxy\ ((x_i\ -\ x_j)^2 + (y_i\ -\ y_j)^2)^{1/2}\ +\ wt\ |t_i\ -\ t_j|$$

donde $x$ e $y$ son coordenadas, $t$ se calcula como $t\ =\ (e+\ l)/2$ donde e y l son el principio y fin de la ventana. $wxy$ y $wt$ son coeficientes tales que $wxy\ >0$ y $wt\ >\ 0$y $wxy\ +wt\ =1$

La complejidad del algoritmo es O($DC^3$) donde C y D son el número de clientes y depósitos respectivamente.

Para la variante de MDVRP que se pretende resolver en este proyecto es necesario ajustar el algoritmo para considerar la capacidad de los depósitos. A continuación, se muestran los pasos a llevar a cabo con el algoritmo resultante.

3) Inicialización: seleccionar los depósitos como los primeros medoids y generar un clúster para cada uno.
4) Asociar cada cliente al clúster del medoid más cercano si la capacidad de clúster lo permite.
   a) Si la capacidad no es suficiente probar con el siguiente más cercano. Repetir el procedimiento mientras no encuentre un clúster adecuado.
5) Mientras el costo de la configuración disminuya:
   a) Para cada medoid m, para cada no medoid o:

i) Intercambiar m y o, recalcular el costo (suma de la distancia de los puntos a sus medoids).
ii) Si el costo total de la configuración aumentó en el paso anterior, deshacer el intercambio.

b) Si el medoid cambió, reasignar cada cliente teniendo en cuenta los nuevos medoids.

## Unweighted Pair Groups Method Centroid

El agrupamiento jerárquico es un método de agrupamiento, el cual busca construir una jerarquía de grupos. Las estrategias para agrupamiento jerárquico generalmente se pueden clasificar en dos tipos:

- **Aglomerativas**: Este es un acercamiento ascendente: cada punto comienza en su propio grupo, y los pares de grupos son mezclados mientras uno sube en la jerarquía.
- **Divisivas**: Este es un acercamiento descendente: todos los puntos comienzan en un grupo, y se realizan divisiones mientras uno baja en la jerarquía.

En general, las mezclas y divisiones son determinadas de forma ávida.

El UPGMC es un método que calcula el centroide o la media de los elementos que se unen desde los clústeres.

El algoritmo comienza con un clúster para cada elemento y en cada paso se agrupan los clústeres más cercanos hasta que el número de clústeres no excede el número de depósitos y las restricciones se cumplen.

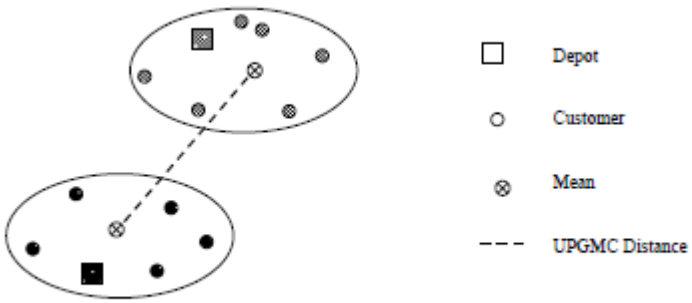



***Figura 3. UPGMC, clústeres***

En UPGMC la distancia que se utiliza para agrupar los clústeres se mide desde la media de cada clúster.

En particular, UPGMC presenta dificultades en encontrar clústeres compactos cuando tienen diferente forma y tamaño.

La complejidad de estos algoritmos es $O((D+C)^2 + (D+C)^2 \log(D+C))$ donde D es el número de depósitos y C es el número de clientes.

Para la variante de MDVRP que se pretende resolver en este proyecto es necesario ajustar el algoritmo para considerar la capacidad de los depósitos. A continuación, se muestran los pasos a llevar a cabo con el algoritmo resultante.

1) Inicialización: generar un clúster para cada elemento (clientes y depósitos) y asignar el elemento como centroide del clúster.
2) Calcular la distancia entre todos los clústeres y ordenarlas de menor a mayor.
3) Mientras la cantidad de clústeres sea mayor que la cantidad de depósitos:
    a) Tomar el par de clústeres más cercanos y si aún no fueron unidos a ningún otro:
        i) Si solo uno de ellos tiene un depósito y la carga de los clientes se puede satisfacer con el depósito en cuestión, unir los clústeres, recalcular el centroide y calcular las nuevas distancias.
        ii) Si ninguno tiene un depósito, unir los clústeres, recalcular el centroide y calcular las nuevas distancias.
        iii) Si los dos tienen depósitos, ignorar.

## Comparación de las tres técnicas de asignación

Según Tansini et al. el Algoritmo Simplificado Paralelo es el que tiene el mejor balance calidad/tiempo de ejecución. [24]

Mientras que UPGMC brinda mejores soluciones, como en general lo hacen los algoritmos de clustering respecto a los clásicos de asignación, es más lento por lo que podría ser más interesante para aplicar en casos pequeños donde el tiempo de ejecución no sea tan grande. [23]

PAM es más rápido que UPGMC, pero brinda una solución de peor calidad.

## Algoritmo de Revisión de Fronteras

Dado que se tuvo que modificar cada algoritmo de asignación para considerar las restricciones de capacidad, es probable que los mismos no retornen la solución óptima que darían en el caso sin restricciones. Es por esto que se consideró la idea de plantear un algoritmo de post optimización para la parte de asignación.

La experiencia en el área de optimización de uno de los integrantes de este proyecto fue clave para definir el nuevo algoritmo.

La idea en un principio es utilizar un algoritmo de intercambios al igual que para la post optimización del ruteo.

Lo que se plantea es una heurística que, a partir de una solución inicial de la etapa de asignación busca mejoras para los clústeres.

La estrategia de esta heurística es detectar clientes para los cuales el cambio de clúster podría implicar una mejora en el costo total del problema. Los clientes por analizar serían los que tienen a sus dos clientes vecinos más cercanos en otro clúster.

Supongamos que se tiene un cliente C1, asignado al clúster del depósito D1, cuyos dos vecinos más cercanos C2 y C3 están asignados al clúster del depósito D2. En esta heurística vamos a suponer que cerca de nuestro cliente C1, pasa la ruta a la cual pertenecen C2 y C3, por lo cual, a esta ruta quizás no le implicaría mucho esfuerzo pasar también por C1. Por lo tanto, si el depósito D2 tiene capacidad para atender a C1, se realiza el cambio y se ejecuta el algoritmo de ruteo para saber si implica una mejora en el costo total. De esta forma se van buscando clientes candidatos para probar el cambio de clúster. 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 D1 y D2.

A continuación, se observa un ejemplo de un cliente asignado a un clúster (negro) que tiene los dos vecinos en otro clúster (rojo).

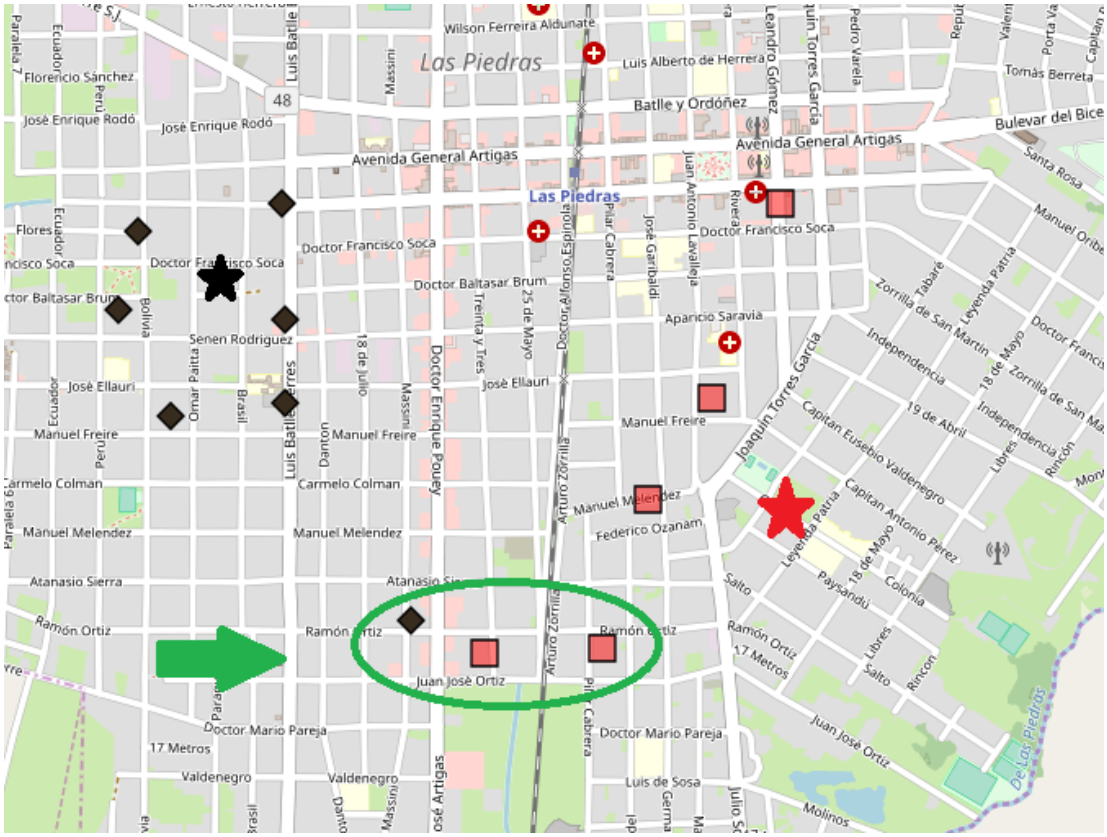



***Figura 4. Revisión de fronteras***

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 clúster del depósito D2 (de C2 y C3) en caso de que no tenga lugar para atender al cliente C1. Se busca entonces a algún cliente que esté en la misma situación que C1 (o sea que sus dos vecinos más cercanos pertenezcan a otro clúster) y que el depósito que lo atendería tenga capacidad para hacerlo. Podría ser el depósito D1 y en este caso sería un intercambio de clientes entre dos clústeres, muy similar a lo que es el intercambio de clientes entre rutas. Se recalculan los costos de los clústeres implicados, y si hay mejora se hacen los cambios.

Es de esperar que al cambiar el cliente de clúster se mejore el costo del clúster original ya que no se desvía tanto para atenderlo y que el sobre costo que genera en el nuevo clúster no sea significativo.

Al presentarle la idea a los tutores, comentaron que ya existía un proyecto de grado con una idea muy similar, pero de todos modos permitieron que se continuara por esa línea.

## Ruteo

La segunda fase de Cluster First, Route Second es realizar el cálculo de rutas de los vehículos.

En la solución se utiliza el algoritmo de Clarke & Wright. El Algoritmo de Clarke & Wright fue explicado en el Estado del Arte. La versión por implementar 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. Posteriormente, se recorre la lista de ahorros, y a medida que sea factible (por la capacidad de los vehículos y las ventanas de tiempo de depósitos y clientes), se van uniendo las rutas.

El algoritmo de Clarke y Wright es muy utilizado en la literatura, es fácil de implementar y de él se obtienen buenos resultados con bajos costos computacionales.

## Post Optimización

La solución del problema tendrá la opción de realizar una post optimización mediante dos métodos. Un método Intra-ruta que optimiza el orden en el cual los clientes son visitados por el vehículo y otro método Inter-ruta que busca intercambiar clientes entre dos rutas distintas para buscar mejores resultados.

Ambos métodos serán opcionales para el usuario, quien determinará en función de sus necesidades que nivel de post optimización requiere el problema.

La post optimización Inter-ruta se basa en el método λ-intercambio. Consiste, básicamente, en intercambiar $x$ clientes de la ruta 1 con $y$ clientes de la ruta 2, donde $x \leq \lambda$ e $y \leq \lambda$. Se prueban todas las combinaciones de $x$ e $y$ que cumplan $x \leq \lambda$ e $y \leq \lambda$.

Ejemplo:

Con $\lambda = 2$ se hacen los siguientes intercambios:

(0,2) 0 clientes de la ruta I y 2 clientes de la ruta II
(0,1) 0 clientes de la ruta I y 1 cliente de la ruta II
(1,0) 1 cliente de la ruta I y 0 clientes de la ruta II
(1,1) 1 cliente de la ruta I y 1 cliente de la ruta II
(1,2) 1 cliente de la ruta I y 2 clientes de la ruta II
(2,0) 2 clientes de la ruta I y 0 clientes de la ruta II
(2,1) 2 clientes de la ruta I y 1 cliente de la ruta II
(2,2) 2 clientes de la ruta I y 2 clientes de la ruta II

La post optimización Intra-ruta está basada en el método Or-Opt descrito en el Estado del Arte. Consiste en eliminar una secuencia de clientes consecutivos de la ruta y colocarlos en otra posición de esta, de modo que permanezcan consecutivos y en el mismo orden.

En la versión 3-opt se realizan las movidas con k = 3, luego con k = 2 y finalmente con k = 1.

# Capítulo 3

## Estado del arte

## Introducción

El problema de VRP, definido en 1958 por Dantzig y Ramset, consiste en, dado un conjunto de clientes dispersos geográficamente, un depósito y una flota de vehículos, determinar un conjunto de rutas de costo mínimo que comiencen y terminen en el depósito, para que los vehículos visiten a los clientes. [8]

Debido a la característica de NP-Hard del problema, se han desarrollado a lo largo de las últimas décadas una serie de heurísticas y metaheurísticas con el fin de obtener buenas soluciones en menor tiempo que los métodos exactos. [17]

La variante con más relevancia para el presente estado del arte es la del Problema de Ruteo de Vehículos con Múltiples Depósitos, MDVRP por su nombre en inglés, *Multi-Depot Vehicle Routing Problem.* En este caso, el problema cuenta con la posibilidad de utilizar múltiples depósitos para atender la demanda de sus clientes. [6]

La literatura clasifica esta variante como un problema NP-Hard al igual que lo es el problema original. [17]

De acuerdo con Montoya Torres [17], la tasa de artículos por año sobre MDVRP en el período 1984-2000 era de 1.12, mientras que para el período de 2011-2014 había aumentado a 12.75. Esto se debe a que diversas aplicaciones en la vida real, como por ejemplo la distribución de gas, de productos químicos, de alimentos y de bebidas entre otras han motivado la investigación acerca de métodos para resolver el problema de MDVRP debido a la considerable importancia económica que tiene en esos casos. [10][22]

# Problema de Ruteo de Vehículos con Múltiples Depósitos

Como se menciona previamente MDVRP es una generalización de VRP donde se considera que existe más de un depósito. La solución al problema con múltiples depósitos es un conjunto de rutas que cumple con las siguientes condiciones [17]:

a) cada ruta comienza y finaliza en el mismo depósito.
b) cada cliente es atendido una sola vez por un solo vehículo.
c) los productos entregados en una ruta no exceden la capacidad del vehículo.
d) el costo de la solución es el mínimo posible.

El problema se puede modelar mediante un grafo $G = (V, A)$ completo, donde $V$ es el conjunto de vértices y $A$ es el conjunto de arcos. El conjunto $V$ se puede particionar en dos subconjuntos donde $VC = \{1 \ldots N\}$ es el conjunto de los clientes y $VD = \{N + 1 \ldots N + M\}$ es el conjunto de los depósitos. Dichos conjuntos son disjuntos y la unión de estos es $V$.

Los arcos de $A$ tienen costo mayor o igual que cero (sea $(i, j) \in A, c_{ij} \geq 0$) que puede estar asociado a la distancia, tiempo de viaje o cualquier costo definido por la operativa.

Cada cliente tiene una demanda asociada $d_i$que debe ser atendida por alguno de los $K$ vehículos que tiene la flota con capacidad $P_k$ cada uno. Para simplificar el problema se asume que la demanda de cada cliente es menor o igual que la capacidad de cada vehículo.

El tiempo de servicio del cliente $i$ es $t_i$ y la duración máxima de la ruta es $T$.

A los efectos de simplificar la formulación se asume que el costo, la distancia y el tiempo entre dos nodos tienen el mismo valor.

Montoya Torres explica en su trabajo que el modelo matemático de MDVRP requiere la definición de una variable binaria$x_{ijk}$ que indica si los clientes $i$ y $j$ se encuentran en la ruta del vehículo $k$. Del mismo modo son necesarias variables auxiliares $y_i$para eliminar la posibilidad de ciclos en las rutas. [17]

De acuerdo con lo anterior, el modelo que se define es el siguiente:

Minimizar $\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}\sum_{k=1}^{K}(c_{ij}.x_{ijk})$ (1)

s.a.

$\sum_{i=1}^{N+M}\sum_{k=1}^{K}x_{ijk} = 1\ con\ j = 1,..,N$ (2)

$\sum_{j=1}^{N+M}\sum_{k=1}^{K}x_{ijk} = 1\ con\ i = 1,..,N$ (3)

$\sum_{i=1}^{N+M}x_{ihk} - \sum_{j=1}^{N+M}x_{hjk} = 0\ con\ k = 1,..,K;\ h = 1,..,N+M$ (4)

$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}d_i.x_{ijk} \leq P_k\ con\ k = 1,..,K$ (5)

$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}(c_{ij} + t_i)x_{ijk} \leq T_k\ con\ k = 1,..,K$ (6)

$\sum_{j=N+1}^{N+M}\sum_{i=1}^{N}x_{ijk} \leq 1\ con\ k = 1,..,K$ (7)

$\sum_{i=N+1}^{N+M}\sum_{j=1}^{N}x_{ijk} \leq 1\ con\ k = 1,..,K$ (8)

$y_i - y_j + (M+N)\,x_{ijk} \leq N + M - 1\ Para\ 1 \leq i \neq j \leq N\ y\ 1 \leq k \leq K$ (9)

$x_{ijk} \in \{0,1\}\ \forall i,j,k$ (10)

La función objetivo es el costo total de la solución. Las restricciones (2) y (3) aseguran que cada cliente es atendido una sola vez por un único vehículo. La restricción (4) se encarga de asegurar la continuidad de la ruta mientras que (5) y (6) aseguran que se cumpla con las restricciones de demanda y tiempo.

La disponibilidad de los vehículos se asegura mediante (7) y (8) mientras que la restricción (9) impide que la solución contenga ciclos. [17]

# Métodos exactos para VRP y MDVRP

El problema de ruteo de vehículos, con sus distintas variantes, ha sido estudiado extensamente en la literatura. Desde que fue presentado en 1959 por Dantzig se han desarrollado diversos métodos de solución, en su mayoría heurísticos debido a la complejidad computacional del problema. [8]

Los métodos de solución exactos son algoritmos capaces de solucionar de forma óptima los sistemas de ecuaciones lineales derivados de la formulación de los problemas de ruteo de vehículos [17].

Laporte y Norbert, en su publicación de 1987 presentan una clasificación de los métodos exactos, donde definen tres grandes grupos [13]:

- Direct Tree Search Methods
- Dynamic Programming
- Integer Linear Programming

La literatura sobre los enfoques exactos específicamente para el MDVRP es escasa. De hecho, la mayoría de los autores se han centrado en el desarrollo de métodos heurísticos y metaheurísticos para encontrar soluciones de buena calidad rápidamente [6].

Baldacci y Mingozzi [1] proponen un método basado en el procedimiento de generación sistemática de límites basado en la propuesta de Christofides et al. [3]. El algoritmo exacto utiliza tres tipos de procedimientos para generar límites inferiores, basados en relajación lineal y en relajación lagrangeana, realizadas a la formulación matemática. Los procedimientos propuestos reducen el número de variables del modelo matemático, lo que permite resolver el problema resultante de forma exacta utilizando un software comercial de programación lineal entera.

En el trabajo de Baldacci et al. [2] se describe una formulación de programación entera del problema de ruteo vehículos con visita periódicas, Periodic Vehicle Routing Problem (PVRP), que se utiliza para obtener diferentes límites inferiores. El MDVRP se puede formular como un PVRP debido a que los depósitos pueden modelarse como los múltiples períodos en el contexto de un PVRP. Por lo tanto, cualquier algoritmo que resuelve el PVRP también puede resolver el MDVRP.

Contardo y Martinelli [6] en su trabajo de 2014 presentan un algoritmo exacto considerando restricciones de capacidad y largo de ruta, el cual utiliza dos formulaciones para el MDVRP, vehicle-flow y set-partitioning, utilizadas en diferentes etapas del algoritmo. El límite inferior calculado con la formulación vehicle-flow se utiliza para eliminar los bordes no prometedores, reduciendo así la

complejidad del subproblema utilizado para resolver la formulación de set-partitioning. Los resultados computacionales muestran que el algoritmo propuesto es competitivo frente a los métodos más avanzados para el problema de ruteo de vehículos con múltiples depósitos, y es capaz de resolver de manera óptima algunas instancias no resueltas anteriormente. Además, para las instancias que no pueden ser resueltas por el algoritmo propuesto por los autores, los límites inferiores finales resultan más fuertes que los obtenidos por métodos anteriores.

Existen otros trabajos donde se propone un modelo de programación lineal entera mixta (MILP por su nombre en inglés, Mixed Integer Lineal Programming) para modelar MDVRP teniendo en cuenta las ventanas de tiempo.

Dondo, Méndez y Cerdá en 2003 [9] proponen un marco MILP para el problema de ruteo de vehículos con múltiples depósitos, flota heterogénea y ventanas de tiempo (MDVRPHFTW). Su procedimiento proporciona un plan óptimo y también el tamaño óptimo de la flota para cada ruta. [6].

Isaza, Franco,y Herazo-Padilla [18] proponen un modelo basado en una formulación matemática de programación lineal entera mixta MILP para la resolución de un problema de ruteo de vehículos con múltiples depósitos, flota heterogénea y ventanas de tiempo para la minimización del costo total de la operación basada en la formulación desarrollada por Dondo & Cerdá con una modificación en las variables de carga por cada vehículo que logra una versión un poco más compacta..

Li, Li y Pardalos [14] resuelven el problema de ruteo de vehículos con múltiples depósitos, ventanas de tiempo y una elección flexible del depósito de llegada. Ellos formulan el problema como un modelo de programación lineal entera utilizando como costo el tiempo total de viaje, considerando las limitaciones de la capacidad, las ventanas de tiempo, la duración de la ruta, el número de vehículos de cada depósito y el número de espacios de estacionamiento de cada depósito.

Profundizar en estos métodos se aleja del propósito de este documento ya que para grandes cantidades de nodos (cliente y depósitos, entre otros), los métodos exactos requieren demasiado tiempo de cómputo, por lo cual el enfoque utilizado para encarar este tipo de problemas es esencialmente heurístico. Dicho enfoque aplica para todas las variantes de ruteo de vehículos.

## Heurísticas para VRP y MDVRP

En este capítulo se describen técnicas, métodos, algoritmos y estrategias que permiten ser aplicados para la búsqueda de soluciones al problema VRP.

Dado que el problema MDVRP es una generalización del problema VRP las heurísticas descritas para VRP en este capítulo son de suma importancia dado que las mismas serán utilizadas y combinadas en los métodos metaheurísticos aplicados para MDVRP. [19]

## Algoritmo de ahorros

El algoritmo de ahorros propuesto por Clarke y Wright en 1964 combina rutas diferentes de una solución, para conformar nuevas rutas que produzcan un ahorro en el total de la distancia recorrida, cumpliendo con la restricción de capacidad del vehículo que realiza la ruta [5].

A continuación, se observa una combinación entre dos rutas diferentes.

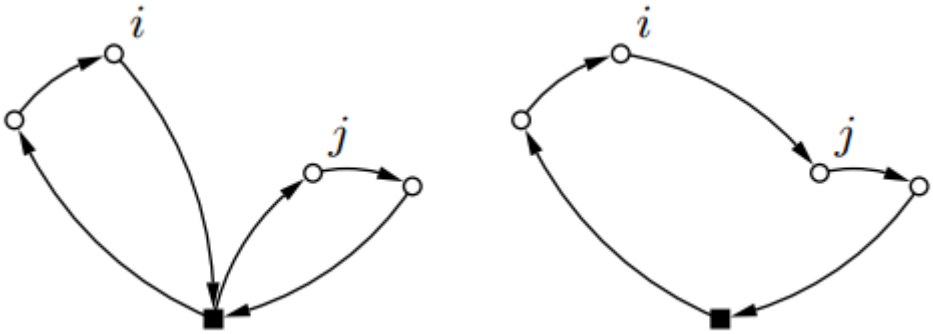



***Figura 5. Dos rutas antes y después de ser unidas***

Analizando la imagen anterior, al combinar las siguientes rutas utilizando el algoritmo de ahorros y definiendo las rutas como:

- ruta 1 $(Depósito, \ldots., vértice\ i, , Depósito)$
- ruta 2 $(Depósito, vértice\ j, \ldots, Depósito)$

Se puede producir un ahorro en la distancia recorrida si la suma de distancia de los arcos $(vértice\ i, Depósito)$, $(Depósito, vértice\ j)$ del grafo, que son eliminados en la combinación, supera a la distancia del arco $(vértice\ i, vértice\ j)$ agregado en la ruta producto de la combinación.

El ahorro ganado en distancia al combinar las rutas anteriores es:

$$s_{ij} = (c(i, depósito) + c(depósito, j)) - c(i, j)$$

Para aplicar la heurística de algoritmo de ahorros es necesario partir de una solución inicial. Mediante la aplicación de reiteradas combinaciones a la solución

inicial, se busca alcanzar mejores soluciones, verificando en cada una de las mismas que se sigue cumpliendo con las restricciones de capacidad. [19]

En muchas aplicaciones del algoritmo de ahorros se ha observado que pueden producirse rutas circulares, lo que puede ser no deseado para la resolución del problema. [19]

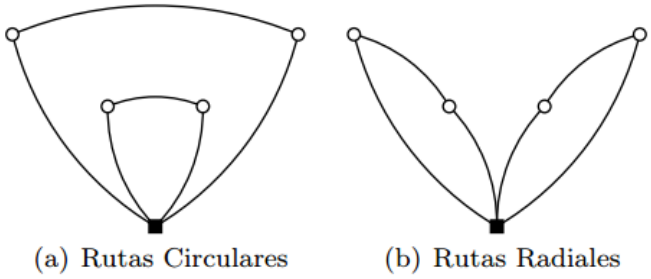



***Figura 6. Un ejemplo de rutas circulares y radiales.***

Para evitar este problema, se puede definir la función de ahorro como se observa a continuación, con la inclusión de un parámetro $\lambda$ que penaliza la unión de rutas con clientes que se encuentran demasiado lejos.

$$s_{ij} = (c(i, depósito) + c(depósito, j)) - \lambda c(i, j)$$

El parámetro $\lambda$ es también utilizado para generar diferentes conjuntos de soluciones mediante la ejecución del algoritmo con diferentes valores de $\lambda$.

El algoritmo de Clarke y Wright es muy utilizado en la práctica debido a la facilidad con el que se aplica, retornando soluciones aceptables en un corto tiempo.

A continuación, se presenta el pseudocódigo para el algoritmo de ahorros tanto para la versión paralela como para la versión secuencial.

Algoritmo de Ahorros (Versión paralela)

- Paso 1 (inicialización). Para cada cliente $i$ construir la ruta (0, $i$, 0), donde 0 es el depósito.
- Paso 2 (cálculo de ahorros). Calcular $s_{ij}$ para cada par de clientes $i$ y $j$.
- Paso 3 (mejor unión). Sea $s_{i*j*} = máx\, s_{ij}$, donde el máximo se toma entre los ahorros que no han sido considerados aún. Sean $r_i^*$ y $r_j^*$las rutas que contienen a los clientes $i^*$ y $j^*$ respectivamente. Si $i^*$ es el último cliente de $r_i^*$ y $j^*$ es el primer cliente de $r_j^*$ y la combinación de $r_i^*$ y $r_j^*$ es factible, combinarlas. Eliminar $s_{i*j*}$ de futuras consideraciones. Si quedan ahorros por examinar ir a 3, si no terminar.

Algoritmo de Ahorros (Versión secuencial)

- Paso 1 (inicialización). Para cada cliente $i$ construir la ruta (0, $i$, 0), donde 0 es el depósito.
- Paso 2 (cálculo de ahorros). Calcular $s_{ij}$ para cada par de clientes $i$ y $j$.
- Paso 3 (selección). Si todas las rutas fueron consideradas, terminar. Si no, seleccionar una ruta que aún no haya sido considerada.
- Paso 4 (extensión). Sea $(0, i, \ldots, j, 0)$la ruta actual. Si no existe ningún ahorro conteniendo a $i$ (o a $j$), ir a 3. Sea $s_{k*i}$ (o $s_{jl*}$ ) el máximo ahorro conteniendo a $i$ (o a $j$). Si $k^*$(o $l^*$) es el último (o primer) cliente de su ruta y la combinación de dicha ruta con la actual es factible, realizar dicha combinación. Eliminar $s_{k*i}$ (o $s_{jl*}$) de futuras consideraciones. Ir a 4.

## Heurísticas de Inserción

Las heurísticas de inserción son métodos que permiten la creación de una solución, mediante múltiples y sucesivas inserciones de clientes en las rutas. En cada iteración se tiene una solución parcial del problema con rutas que se encuentran formadas por un subconjunto de clientes, donde se elige un nuevo cliente no visitado para ser parte de la solución del problema. [16]

Algunas de las heurísticas de inserción son “Heurística de inserción de Mole y Jameson [16] y el método de inserción en paralelo de Christofides, Mingozzi y Toth [4].

## Búsqueda Local

La heurística de búsqueda local no se preocupa tanto por la búsqueda de nuevas rutas, si no que se enfoca en el estado actual de la solución y los intercambios de clientes que llevan a la solución a otro estado mejor. [19]

Los algoritmos de búsqueda local presentan las siguientes ventajas:

- Son ahorrativos: usan poca memoria en donde no se guarda la secuencia de estados.
- Razonables: Ofrecen buenas soluciones, cuando el espacio de estados posibles de solución es infinito.

Los algoritmos de búsqueda local son propensos a encontrar óptimos locales que no son la mejor solución, muchas veces encontrar un óptimo global en tiempo limitado es generalmente imposible por el tamaño del espacio de soluciones. [15]

Muchas metaheurísticas utilizan la heurística de búsqueda local en la búsqueda de una mejor solución en el espacio de soluciones. [19]

Mediante el procedimiento de búsqueda local se puede obtener una mejor solución al problema partiendo de una solución inicial $s$. El algoritmo de búsqueda local define un conjunto de soluciones vecinas $N(s)$ a $s$, del cual se selecciona una solución $s^*$ de menor costo y se reemplaza $s$ por $s^*$ y se repite el procedimiento hasta que la solución $s$ no se pueda mejorar más.

En esta sección se detalla el funcionamiento del Operador $\lambda$ y del Operador Or-Opt, los cuales fueron seleccionados para realizar la post optimización de las rutas calculadas mediante el algoritmo de Clarke & Wright.

### Operador $\lambda$ intercambio de Lin

El operador $\lambda$ intercambio de Lin [15], utilizado en el procedimiento de búsqueda local aplicado para una ruta consiste en eliminar $\lambda$ arcos de esta y reconectar los $\lambda$ segmentos restantes de la ruta solución.

Se define $\lambda - opt$ al algoritmo de búsqueda local en el cual se utiliza el operador de $\lambda$ intercambio para obtener la solución óptima. La solución de óptima obtenida por el operador de $\lambda$ intercambio se llama solución $\lambda - óptima$.

La cantidad de $\lambda$ intercambios aplicados a una ruta que atiende a $n$ clientes está dado por la ecuación:

$$2^{\lambda-1}(\lambda-1)!\binom{n+1}{\lambda}$$

Donde$\binom{n+1}{\lambda}$son las posibles formas de eliminar $\lambda$ arcos de la ruta y $2^{\lambda-1}(\lambda-1)!$ es la cantidad de formas que hay de reconectar la ruta.

Verificar si una solución es $\lambda - óptima$ puede realizarse en tiempo O ($n^{\lambda}$) en el peor caso. [15]

Generalmente los operadores de $\lambda$ intercambios más utilizados son “2-intercambios” y “3-intercambios”.

### Operador Or-Opt

El algoritmo Or-Opt [20] consiste en eliminar una secuencia de $k$ clientes consecutivos de la ruta y colocarlos en otra posición de la ruta, de modo que permanezcan consecutivos y en el mismo orden.

En la versión 3-opt se realizan las movidas con $k = 3$, luego con $k = 2$ y finalmentecon $k = 1$.

En la siguiente figura se muestra una ruta y todas las posibles maneras de reubicar los 3 primeros clientes a la manera de Or-Opt. Si una ruta visita $n$ clientes existen O ($n^2$) de estas movidas.

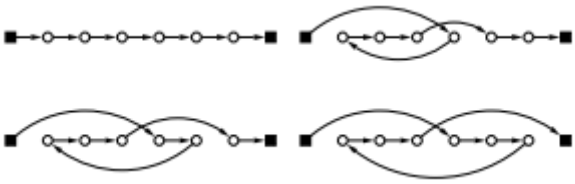


***Figura 7. Movidas para reubicar los 3 primeros clientes de una ruta.***

## Asignar primero y luego rutear

Un enfoque alternativo para resolver el problema del ruteo de vehículos es asignar primero lugar y luego rutear (*cluster first, route second*). Como es de esperar este tipo de método tiene dos fases. [19]

La primera fase consiste en generar clústeres de clientes, donde cada grupo de clientes (clúster) pertenece a una ruta de la solución final. En esta primera etapa de clusterización se debe contemplar las restricciones capacidad donde la demanda total de la ruta no puede exceder la capacidad del vehículo que atiende a la ruta de clientes perteneciente al clúster. [19]

La segunda fase consiste en crear una ruta para cada clúster, en el cual, dependiendo de la cantidad de clientes de este, se puede aplicar métodos exactos o aproximados para la creación de la ruta solución. [19]

Giosa et al. [11] implementan la estrategia de “asignar primero y rutear después”, aplicada a una variante del MDVRP que considera ventanas de tiempo, “MDVRP with Time Windows” (MDVRPTW). En la etapa de clusterización se asignan los clientes a los depósitos, luego cada clúster se resuelve usando el algoritmo de ahorros. El artículo realiza un estudio computacional con seis diferentes técnicas de clusterización.

La cantidad de clústeres generados es igual a la cantidad de depósitos del problema.

Una mala decisión al momento de la generación de clúster puede afectar el posterior ruteo generando rutas de alto costo. Por eso la generación de clústeres y la posterior generación de rutas no son independientes.

### Algoritmos de asignación para MDVRPTW

En esta sección se analizarán algoritmos de generación de clústeres de clientes y la generación de las rutas para cada clúster generado como forma de resolver el problema de Multi Depot Vehicle Routing Problem Time Windows [11].

#### Algoritmo de Asignación:

Un pseudocódigo de general para todos los algoritmos de asignación es el siguiente:

Hasta que todos los clientes sean asignados a un depósito

Determinar el próximo mejor cliente a ser asignado y a cuál depósito, tomando en cuenta la demanda de los clientes, la capacidad de los depósitos y las ventanas de tiempo de los clientes y depósitos.

Los próximos algoritmos de asignación que se describen a continuación presentan diferentes tipos de medidas para asignar un vehículo a un depósito. Estas medidas son medidas de urgencia, de asignación cíclica y asignación vía clúster. [11]

*Asignación a través de urgencia:*

La urgencia es la manera de expresar la relación de precedencia entre clientes para ser asignados a los depósitos, siendo una forma de representar la prioridad de cada cliente al momento de ser asignado a un depósito. Esta medida de urgencia nos indica que el cliente con mayor urgencia entre los demás, va a ser el primero por seleccionarse para ser asignado a un depósito. [11]

Entre los algoritmos de urgencia se encuentran, “Parallel Algorithm” y “Simplified Algorithm” que varían entre sí, sólo en la forma en que calculan la medida de urgencia y algoritmo de asignación de intercambio. [11]

*Parallel Algorithm:*

La idea de este algoritmo es que la medida de urgencia para cada cliente es calculada considerando todos los depósitos al mismo tiempo. [11]

La fórmula de urgencia para este algoritmo es la siguiente:

$$uc = ( \sum_{dep \, \epsilon \, Depositos} d(c, dep)) - d\,(c, dep')$$

Donde $d(c, dep)$ representa a la distancia entre el cliente y un depósito del conjunto de Depósitos.

La función $d(c, dep')$ representa la distancia entre el cliente y su depósito más cercano.

Está heurística compara los costos de asignar un determinado cliente a su depósito más cercano, con el costo de asignar el cliente a los otros depósitos. El cliente donde la función $uc$ sea máxima, indica que es el mismo presenta la mayor prioridad para ser asignado a un depósito. [11]

El costo del cómputo de este algoritmo en el peor caso es de orden $\theta(3CD + CD^2 + C^2D)$, siendo $C\ y\ D$ la cantidad total de clientes y depósitos del problema respectivamente. [11]

### *Simplified Algorithm:*

Este algoritmo de asignación es similar al "Parallel Algorithm", variando solamente en la forma de calcular la urgencia, la cual se encuentra definida por la siguiente ecuación [11]:

$$uc = d(c, dep'') - d(c, dep')$$

Siendo $dep''$ el segundo depósito más cercano en distancia al cliente $c$ y $dep'$ el depósito más cercano en distancia al mismo cliente $c$ .

La función $d$ hace referencia a la distancia entre el cliente y un depósito. [11]

De forma similar el cliente con mayor $uc$ va a tener la mayor prioridad para ser asignado a su depósito más cercano. [11]

El orden del cómputo de este algoritmo en el peor caso es $\theta(3CD + CD^2 + C^2D)$ [11].

### *Algoritmo de asignación de intercambio:*

Está heurística calcula la urgencia dependiendo si cliente se siente atraído al depósito con mayor demanda insatisfecha. [11]

La función de urgencia del algoritmo de asignación de intercambio es la siguiente [11]:

$$uc = d(c, dep^*) - d(c, dep')$$

Siendo $dep^*$ y $dep'$el depósito con la mayor demanda insatisfecha y el depósito más cercano en distancia al cliente "$c$" respectivamente [11].

Un alto valor de la medida de urgencia significa que es más conveniente asignar el cliente $c$ al su depósito más cercano que al depósito de mayor demanda insatisfecha [11].

El costo de cómputo de este algoritmo en el peor caso es de orden $\theta(3CD + C^2D + D(D^2 + DC + C))$ siendo $C\ y\ D$ la cantidad de clientes y depósitos como se ha comentado en los algoritmos anteriores [11].

*Asignación cíclica:*

La heurística de asignación cíclica consiste en primera instancia en asignar a cada uno de los depósitos del problema su cliente más cercano. Luego de forma cíclica y procediendo de a un cliente a la vez, se selecciona el cliente más cercano en distancia al cliente anterior ya asignado a un depósito y se le asigna el mismo depósito. [11]

El costo de cómputo de este algoritmo en su peor caso es de orden $\theta(CD^2 + C)$ [11].

*Asignación vía clúster:*

Cada clúster se encuentra formado por un depósito y un conjunto de clientes asignado a él. Los algoritmos de esta clase construyen un clúster compacto de clientes para cada depósito. [11]

*Coeficiente de Atracción*

En este procedimiento la forma en que los clientes son asignados a un clúster es definida por coeficientes de atracción sobre los depósitos y sobre los clientes ya asignados a un clúster. [11]

En la heurística de asignación por clúster, en cada paso se selecciona el cliente que minimiza la escala de distancia a un depósito o a un cliente ya asignado, para ser vinculado a un clúster. [11]

La fórmula de la escala de distancia es la siguiente:

$$dscaled(c, c') = d(c, c') * coefc'$$

Si un cliente o depósito presenta un coeficiente de atracción menor que a otro, esa corta distancia con respecto a los demás clientes hace que se sienta atraído a ser vinculado al clúster que forman ellos. En cambio, si el coeficiente de atracción es mayor el cliente se siente rechazado. [11]

Como los clientes que no fueron asignados no presentan valor de atracción, cuando el mismo es asignado obtiene su coeficiente de atracción a través de una función que puede variar. Para calcular el coeficiente de atracción de un cliente $c$ con su cliente más cercano $c'$ se utiliza un coeficiente de degradación. [11]

La fórmula para el cálculo del coeficiente de atracción para un cliente es la siguiente [11]:

$$coefc = min(1.coefc' + (coefc' * degc'))$$

Este algoritmo presenta en costo de cómputo en el peor caso $\theta(C^3+C^2D)$ [11]

#### *Triple Criterio de Agrupamiento*

Este algoritmo es una adaptación del algoritmo que asigna clientes a días de la semana para la recolección de basura. Los criterios que se utilizan para incluir un cliente en un clúster son [11]:

1- Distancia promedio a los clientes del clúster.
2- Varianza de la distancia promedio a los clientes del clúster.
3- Distancia a los clientes más cercanos de cada clúster.

Si hay un cliente con una distancia promedio a su grupo más cercano suficientemente más pequeña que la distancia promedio a su segundo grupo más cercano (mejora del 10% o más) entonces puede ser asignado; el que maximiza la diferencia de las distancias promedio se asigna a su clúster más cercano. De lo contrario, se tiene en cuenta la varianza de la distancia promedio y si hay un cliente con una varianza suficientemente pequeña de la distancia promedio a sus clústeres más cercanos (40%) se puede asignar; de nuevo se asigna el que maximiza la diferencia. Finalmente, si las dos primeras medidas fallan, la decisión se toma en función de la distancia al cliente más cercano en su clúster más cercano, y ahora se asigna el cliente que minimiza esta distancia. [11]

La complejidad de todo el algoritmo es $O = 3C^2D + 3C^2D^2 + CD^2)$ siendo C la cantidad de clientes y D la cantidad de depósitos. [11]

### MDVRPTW

En el trabajo de Tansini & Viera se presentan dos grandes clases de algoritmos de clustering para utilizar en la variante de MDVRP con ventanas de tiempo: Particionamiento y Jerarquía. Los algoritmos de particionamiento que se

incluyen en el trabajo son KNN, K-Means y PAM. Los algoritmos de jerarquía presentados son Aglomeration y Rock. [23]

*KNN*

KNN, K vecinos más cercanos por su nombre en inglés, utiliza los K vecinos más cercanos a un elemento para decidir a qué clúster va a pertenecer. El elemento formará parte del clúster que cuente con mayor cantidad de elementos dentro de los K vecinos. [23]

La complejidad de todo el algoritmo es $O = CD)$ siendo C la cantidad de clientes y D la cantidad de depósitos. [23]

*K-Means*

Este algoritmo clasifica a cada elemento teniendo en cuenta su distancia al centroide. Esta distancia considera tanto las coordenadas geográficas como las ventanas de tiempo. [23]

El orden del algoritmo es $O = CD^2)$ siendo C la cantidad de clientes y D la cantidad de depósitos. [23]

*Partition Around Medoids*

Partition Around Medoids (PAM) utiliza una búsqueda ávida que puede no encontrar la solución óptima, pero es más rápida que la búsqueda exhaustiva. [23]

Los pasos por seguir son:

1. Inicialización: seleccionar *k* de los *n* puntos como “medoids”.
2. Asociar cada punto al “medoid” más cercano.
3. Mientras el costo de la configuración disminuya:
    1. Para cada “medoid” *m*, para cada no “medoid” *o*:
        1. Intercambiar *m* y *o*, recalcular el costo (suma de la distancia de los puntos a sus “medoids”).
        2. Si el costo total de la configuración aumentó en el paso anterior, deshacer el intercambio.

En la figura que se muestra a continuación se puede ver el cambio de “medoid” del clúster claro lo que produce que un cliente del clúster oscuro pase a pertenecer al clúster claro. [23]


***Figura 8. Cambio de "medoid".***

Los depósitos pueden ser elegidos como los primeros "medoids", y posteriormente se puede elegir tanto depósitos como clientes, mientras que se cumplan todas las restricciones. [23]

La distancia utilizada en los algoritmos de particionamiento incorpora las ventanas de tiempo. [23]

Para dos elementos $e_i$ y $e_j$ la distancia se calcula como:

$$WSUM\ (i,j)\ =\ wxy\ ((x_i\ -\ x_j)^2 + (y_i\ -\ y_j)^2)^{1/2}\ +\ wt\ |t_i\ -\ t_j|$$

donde $x$ e $y$ son coordenadas, $t$ se calcula como $t\ =\ (e+\ l)/2$ donde e y l son el principio y fin de la ventana. $wxy$ y $wt$ son coeficientes tales que $wxy\ >0$ y $wt\ >\ 0$y $wxy\ +wt\ =1$. [23]

La complejidad del algoritmo es O($DC^3$) donde C y D son el número de clientes y depósitos respectivamente. [23]

*Agglomeration*

Este es un acercamiento ascendente: cada punto comienza en su propio clúster, y en cada paso los clústeres más cercanos se fusionan en un nuevo clúster mientras que el número de clústeres no exceda el número de depósitos y se cumplan todas las restricciones. [23]

La distancia entre clústeres se puede calcular de distintas formas:

1- Single Linkeage: la distancia entre los clústeres se calcula como la mínima distancia entre un par de elementos pertenecientes a distintos clústeres. Es sensible a elementos que se alejan mucho de la mediana. [23]

2- Complete Linkeage: la distancia entre los clústeres se calcula como la máxima distancia entre un par de elementos pertenecientes a distintos clústeres. Es sensible a elementos que se alejan mucho de la mediana. [23]
3- Average Linkeage (Unweighted Pair Groups Method Centroid): es un método que calcula la distancia utilizando los centroides de los clústeres. Tiene dificultades para encontrar clústeres compactos cuando difieren mucho en los tamaños. [23]
4- Weighted Averrage Linkeage (WPGMC, Weighted Pair-groups Method Centroid): Difiere del anterior dando peso al miembro más reciente de un clúster para eliminar de esta forma la influencia del tamaño del clúster. [23]
5- Centroid (Unweighted Pair Groups Method Average): los dos clústeres con la menor distancia promedio se unen para formar un nuevo clúster. [23]
6- Median (Weighted Pair Groups Method Average): difiere del anterior dándole peso a los miembros más recientes del clúster. [23]
7- Within Grups (WG):es similar a UPGMA, pero los clústeres se fusionan con el objetivo de minimizar la varianza interna. Esto produce clústeres más compactos. [23]
8- Ward: es similar a WG y la pertenencia a un clúster se determina calculando la suma de las desviaciones de la media al cuadrado. El criterio de fusión determina que debe producirse el menor incremento posible en el error de la suma de los cuadrados. [23]

La distancia entre dos elementos cualesquiera se calcula de la siguiente manera:

$$Angle(i,j) = Cos^{-1}\left(\frac{x_i . x_j y_i . y_j . t_i . t_j}{(x_i^2 + y_i^2 + t_i^2)^{1/2}(x_j^2 + y_j^2 + t_j^2)^{1/2}}\right)$$

donde $x$ e $y$ son coordenadas y $t$ se calcula como $t = (e + l)/2$ donde $e$ y $l$ son el comienzo y fin de la ventana de tiempo. [23]

La complejidad de estos algoritmos es $O((D+C)^2 + (D+C)^2 \log(D+C))$ donde D es el número de depósitos y C es el número de clientes. [23]

Para evaluar esta nueva heurística de asignación para la búsqueda de soluciones al problema MDVRP, debemos generar las rutas para cada clúster de clientes [23]

Para la generación de rutas se utiliza el algoritmo de ahorros de Clarke and Wright, considerando que la flota de vehículos es ilimitada y que una ruta se considera completa cuando no se puede agregar un cliente a la misma, porque se violan las restricciones de capacidad del vehículo que la realiza. [23]

*Rock*

Este algoritmo de agrupamiento se basa en el concepto de enlace entre elementos para determinar si dos clústeres deben o no ser combinados. El enlace entre elementos se define como la cantidad de vecinos que tienen en común. Los clústeres con mayor cantidad de vecinos en común serán agrupados. [23]

Para determinar si dos elementos son vecinos se utiliza una función de similitud (Sim) y un valor límite (T). Dos elementos, i y j son considerados vecinos si:

$$Sim(i,j) \leq T$$

En este caso la función Sim es la siguiente:

$$Sim(i,j) = |distancia(i) - distancia(j)| \;\; para\ los\ elementos\ i,j.$$

La complejidad de este algoritmo es

$$O = (C+D)^2 + (D+C) + ((D+C)-1)^{an} + \; (D+C)^2 \log(D+C)$$

donde $an$ es el número promedio de vecinos, D es la cantidad de depósitos y C es la cantidad de clientes. [23]

# Metaheurísticas para VRP y MDVRP

Las metaheurísticas [1] son procedimientos genéricos de exploración del espacio de soluciones para los problemas de optimización. Para encontrar la solución a problemas de optimización, las metaheurísticas recurren a técnicas para realizar una mejor exploración del espacio de soluciones consiguiendo así mejores resultados que los obtenidos por heurísticas. [12]

Las metaheurísticas que se disponibles para MDVRP abarcan desde algoritmos de colonias de hormigas y búsqueda tabú hasta algoritmos genéticos.

En vista de que no se utilizan metaheurísticas para la resolución del problema planteado en el contexto de este proyecto no tiene sentido profundizar sobre las mismas. Por más información sobre las metaheurísticas existentes para MDVRP se puede consultar el Estado del Arte.

[1] Glover las define en 1986 como “Métodos de solución que orquestan una interacción entre procedimientos de mejoramiento local y estrategias de más alto nivel para crear un proceso capaz de escapar de los óptimos locales y realizar una búsqueda robusta del espacio de la solución”

# Capítulo 4

En este capítulo se describe la arquitectura y el diseño a grandes rasgos de la solución propuesta.

## Diseño de la Solución

En esta sección se explica el diseño a alto nivel y la arquitectura del sistema que se implementará para solucionar el problema planteado en el Proyecto de Grado.

La arquitectura está orientada a entornos Web. Bajo este diseño las tareas se ejecutan por el lado del servidor, evitando delegar tales responsabilidades hacia las máquinas clientes desde sus navegadores. Esto se debe a que los algoritmos que se van a desarrollar son complejos y los navegadores de los clientes pueden no contar con los recursos necesarios.

Asimismo, esta arquitectura asegura la disponibilidad a tiempo completo desde un equipo con conexión a Internet.

La arquitectura respetará el paradigma de programación orientado a objetos. Esta característica generalmente depende del lenguaje de programación utilizado, pero en esta solución se asegurará la manipulación de los datos y operaciones de manera encapsulada a través de clases y objetos interrelacionados entre sí por invocaciones a los métodos respectivos.

### Patrón de Diseño

En esta sección se describe el patrón de diseño propuesto para la solución, el cual se basa en el patrón de diseño Model View Controller (MVC).

El patrón de diseño MVC maneja en forma separada los datos de la aplicación, la interfaz de usuario, y la lógica de negocio en tres componentes distintos, cumpliendo con el desarrollo de software en capas.

A continuación, se observa un diagrama del patrón propuesto y la descripción de cada componente.

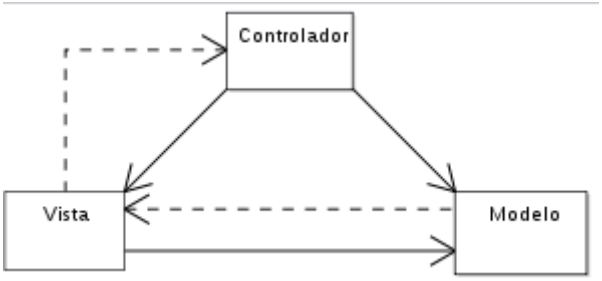



***Figura 9. Patrón modelo vista controlador***

*Modelo*: en el patrón MVC, el modelo representa a las clases de información que se manipulan en la aplicación para ser enviados a la “Vista” en forma de información para responder a las solicitudes.

*Controlador*: es el encargado de responder a los eventos generados por las solicitudes del usuario de la aplicación. El controlador será encargado de invocar a la componente de optimización y cálculos de rutas mediante la tecnología de Web Services (middleware) y generar los modelos de la información necesaria a ser presentada en las “Vista”. Para la creación de los modelos el controlador tendrá acceso a un contexto que encapsula el acceso a los datos del repositorio.

*Vista*: presenta la información en forma adecuada para ser presentada en la interfaz de usuario de la aplicación.

## Definición del sistema

En esta sección se presentan los paquetes de funcionalidades que hacen más fácil el establecimiento y definición de la arquitectura final junto con las clases de diseño necesarias para su construcción.

**Paquete Seguridad**: El paquete de seguridad es el encargado de encapsular las funcionalidades de restricción y acceso a los usuarios habilitados para hacer uso del sistema.

**Paquete Vista Mapa:** Este paquete es el encargado de la interacción del sistema con la visualización de los resultados en el componente Mapa de la aplicación.

**Paquete Algoritmos:** Este paquete es el encargado de resolver los algoritmos de optimización y ruteo del sistema.

**Paquete de Reportes:** Este paquete es el encargado de la generación de reportes con los resultados obtenidos por parte del sistema a partir del uso del paquete de algoritmos.

**Paquete de Datos:** Este paquete es el encargado de encapsular lo necesario para la importación de los datos requeridos por parte del sistema, de los que hará uso el paquete de algoritmos.

## Diseño de la interfaz del sistema

En esta sección se describen lineamientos generales para el diseño de la interfaz del sistema.

En la resolución de problemas donde la solución tiene componentes de clustering y rutas que son mejor apreciados gráficamente, es importante contar con una interfaz gráfica que permita interpretar los resultados de la mejor manera posible.

A continuación, se observa un diagrama de cómo se va a presentar a la interfaz de la solución propuesta.

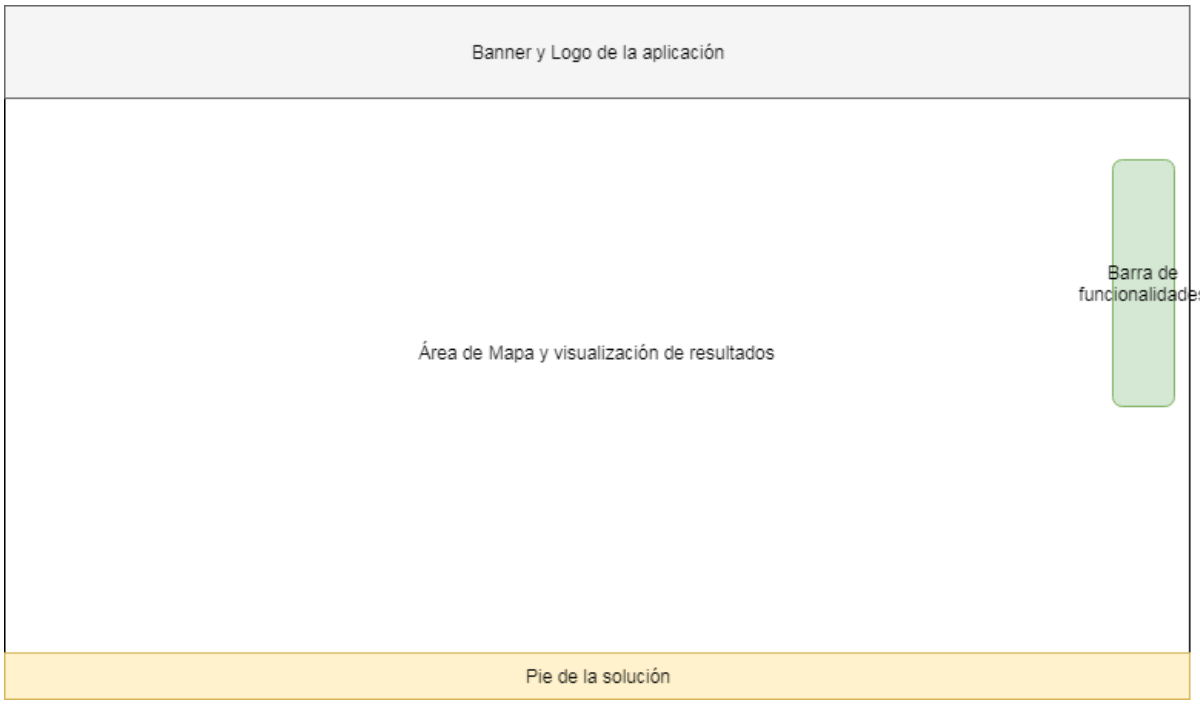



***Figura 10. Diseño de interfaz***

A continuación, se describen las distintas secciones de la interfaz del sistema.

La sección del banner y logo de la aplicación contendrá lo referido al nombre del aplicativo a modo de presentación.

En el “Área de Mapa y Visualización de Resultados” se observarán los resultados obtenidos de la ejecución de los algoritmos de optimización y ruteo para el problema planteado. Se dispondrá de un mapa donde observarán los clústeres, las rutas de los vehículos e información relevante para la lógica del negocio.

La “Barra de Funcionalidades” contendrá accesos para la carga de información de entrada para ser procesada, invocación de algoritmos de optimización y solicitud de visualización de reportes de información de escenarios y resultados obtenidos.

Por último, en la sección de “Pie de la solución” se observará el número de versión en el que se encuentra liberado el producto para su uso, de manera que nos permita tener un seguimiento del estado y los controles de cambios de la solución.

## Arquitectura General

A continuación, se observa un diagrama completo de arquitectura que se adoptará en la solución. La arquitectura propuesta es cliente-servidor como modelo de diseño de software, donde el cliente realiza invocaciones a una aplicación del servidor quien le responde.

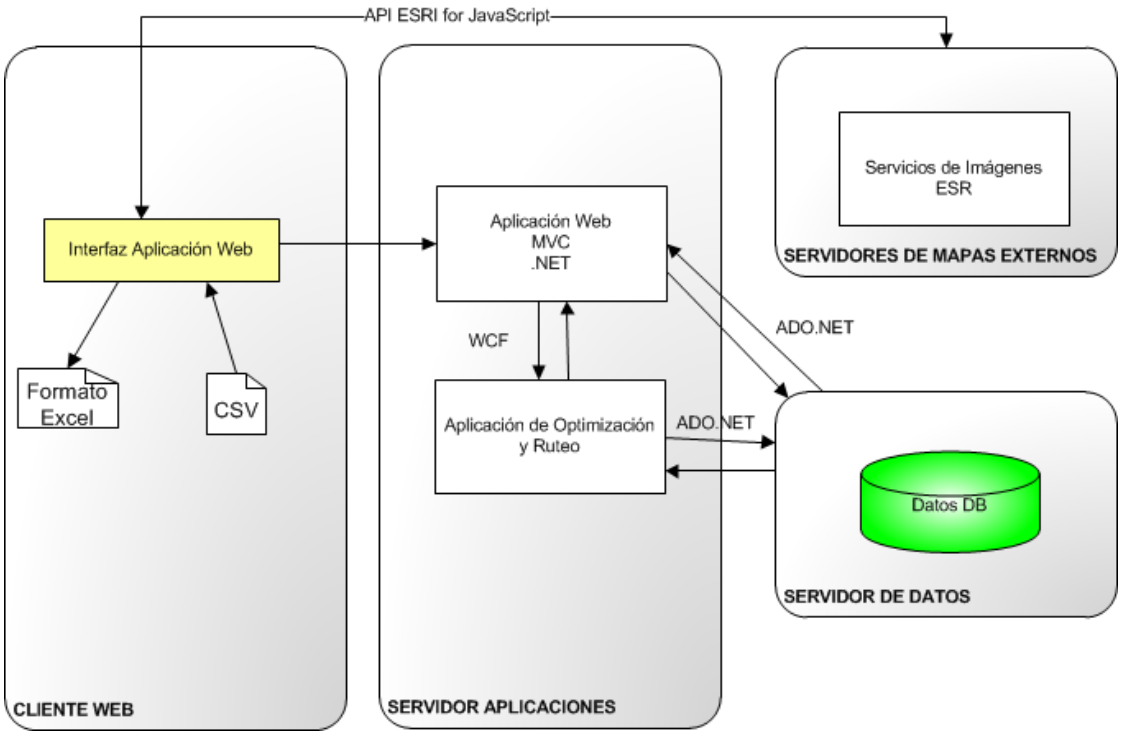



***Figura 11. Diagrama de arquitectura del Sistema Web***

Se prevé que la aplicación permita leer datos de un archivo Excel con determinado formato. Esto le da al usuario cierta flexibilidad a la hora de ingresar los datos del problema.

Para mejorar la visualización de la solución que devuelve el ruteo se cuenta con un mapa de fondo provisto por un servicio de Esri que le permitirá al usuario tener una idea gráfica, más real, de la solución.

## Tecnologías utilizadas en la solución

Los lenguajes de programación, frameworks y API que se utilizaran en la solución se describen a continuación.

En lo que respecta a los lenguajes de programación, se utilizarán C# y JavaScript.

- C# es un lenguaje de programación orientado a objetos desarrollado y estandarizado por Microsoft como parte de su plataforma .NET.
- JavaScript es un lenguaje de programación interpretado, dialecto del estándar ECMAScript. Se define como orientado a objetos, basado en prototipos, imperativo, débilmente tipado y dinámico.

Para el desarrollo de la solución se prevé el uso de frameworks. Los frameworks permiten agilizar el desarrollo y mantener el código ordenado. En este caso en particular se utilizará .Net 4.6 y Entity Framework.

- .NET es un framework de Microsoft que hace énfasis en la transparencia de redes, con independencia de plataforma de hardware y que permite un rápido desarrollo de aplicaciones.
- Entity Framework es un conjunto de tecnologías de ADO.NET que permiten el desarrollo de aplicaciones de software orientadas a datos.

Las API, al igual que los frameworks permiten agilizar el desarrollo y reutilizar código. Para lograr una mejor visualización de la solución se utilizarán mapas, por lo que es necesario contar con una API que haga manejo de operaciones sobre los mismos. En este caso en particular se va a utilizar la API Esri for JavaScript 3.22 o posterior.

- API Esri for JavaScript 3.22. o posterior es una interfaz de programación de aplicaciones diseñada para maximizar la productividad y la construcción de aplicaciones Web de mapas.

Es importante destacar que todos son de conocimiento de los integrantes del grupo y en dicha experiencia se basa su elección.

# Funcionalidades del producto

## Requerimientos Funcionales:

En lo que respecta a requerimientos funcionales del sistema, se definen los siguientes:

**Requerimiento 1:** El sistema permitirá el registro de usuarios, para permitir el ingreso de usuarios al sistema.

**Requerimiento 2:** Para la utilización del sistema el usuario se deberá estar autenticado por el sistema.

**Requerimiento 3:** El sistema permitirá importar datos para la resolución del ruteo mediante un archivo Excel. Este requerimiento pretende darle flexibilidad al usuario a la hora de ingresar los datos para resolver el problema.

**Requerimiento 4:** El sistema permitirá agregar vehículos, pedidos y depósitos en forma manual. Para el ingreso de pedidos y depósitos de forma manual se deberá localizar su ubicación en el mapa.

**Requerimiento 5:** El sistema permitirá la visualización de los depósitos, clientes y rutas en un mapa de la aplicación Web.

**Requerimiento 6:** El sistema permitirá obtener la información de los clientes y depósitos del mapa haciendo clic sobre ellos.

**Requerimiento 7:** El sistema permitirá configurar los algoritmos a utilizar para el MDVRP. El usuario podrá elegir tanto el algoritmo de asignación, como otros parámetros que utilicen los algoritmos.

**Requerimiento 8:** El sistema permitirá definir los niveles de post optimización a utilizar en la solución del MDVRP. El usuario podrá elegir si desea realizar una post optimización y en caso afirmativo podrá seleccionar el nivel deseado según sus necesidades.

**Requerimiento 9:** El sistema mostrará en pantalla una vez resuelto el problema el tiempo de ejecución que implicó encontrar dicha solución.

**Requerimiento 10**: El sistema permitirá la visualización de los clústeres de clientes producto de la solución de resolver la asignación del problema MDVRP.

**Requerimiento 11**: El sistema permitirá la visualización de las rutas resultado de la ejecución de la optimización.

## Requerimientos No Funcionales:

En lo que respecta a requerimientos no funcionales del sistema, se definen los siguientes:

**Requerimiento 1:** El sistema será desarrollado con una interfaz gráfica de usuario basada en controles Web para permitir acceder a la solución desde distintos clientes.

**Requerimiento 2:** El sistema será accesible desde cualquier ambiente de trabajo que cuente con Internet Explorer (9.0 o superior), Mozilla Firefox (3.5 o superior) y Google Chrome (4 o superior).

**Requerimiento 3:** El sistema trabajará con un administrador de bases de datos SQLServer.

**Requerimiento 4:** El sistema contará con manuales de usuario para su entendimiento.

## Consideraciones sobre el sistema

Como alcance de la propuesta se respetarán las siguientes restricciones:

**Seguridad:** Acceso al sistema a personas mediante cuentas de usuario y contraseña. En función a los perfiles y accesos se controlará el nivel de visibilidad de la información.

**Escalabilidad:** La arquitectura posibilitará la incorporación de nuevas funcionalidades y módulos flexiblemente sin procedimientos drásticos para el desarrollador.

**Usabilidad:** Para la familiarización del usuario con el software se requiere una interfaz gráfica ligera e intuitiva sumada a una correcta emisión de avisos de error y advertencia. El usuario iniciará todas las operaciones requeridas.

## Actores involucrados en el sistema

**Usuario**: Toda persona que presenta acceso para poder ejecutar un determinado de conjunto de funcionalidades.

A continuación, se observa el diagrama de interacción con el sistema.

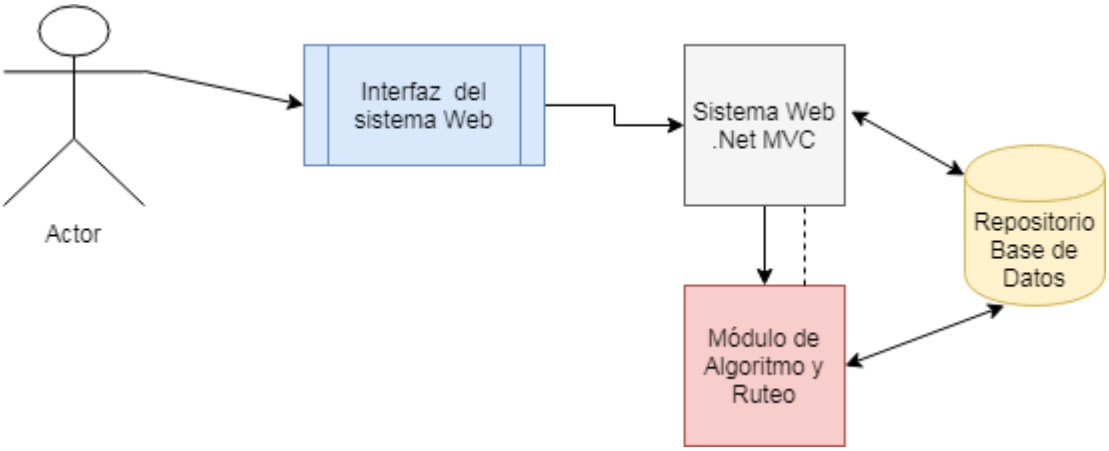



***Figura 12. Diagrama de interacción del sistema con el Actor***

## Restricciones de Hardware y Software

Las restricciones identificadas son las siguientes:

- Disponibilidad equipo servidor para computo con pool de aplicaciones .NET (4.0 o posterior) e Internet Information Services instalado para alojamiento de la aplicación Web y Módulo de algoritmos Optimización.

- Servidor de base de datos con capacidad de almacenamiento para poder almacenar la información generada y cargada mediante la aplicación.

- Disponibilidad de equipos de cómputos para la realización de pruebas, análisis, diseño y construcción.

# Diagrama de base de datos

A continuación, se observan las tablas de la base de datos de la solución Web que se encuentran alojadas en un servidor de datos Sql Server Express.

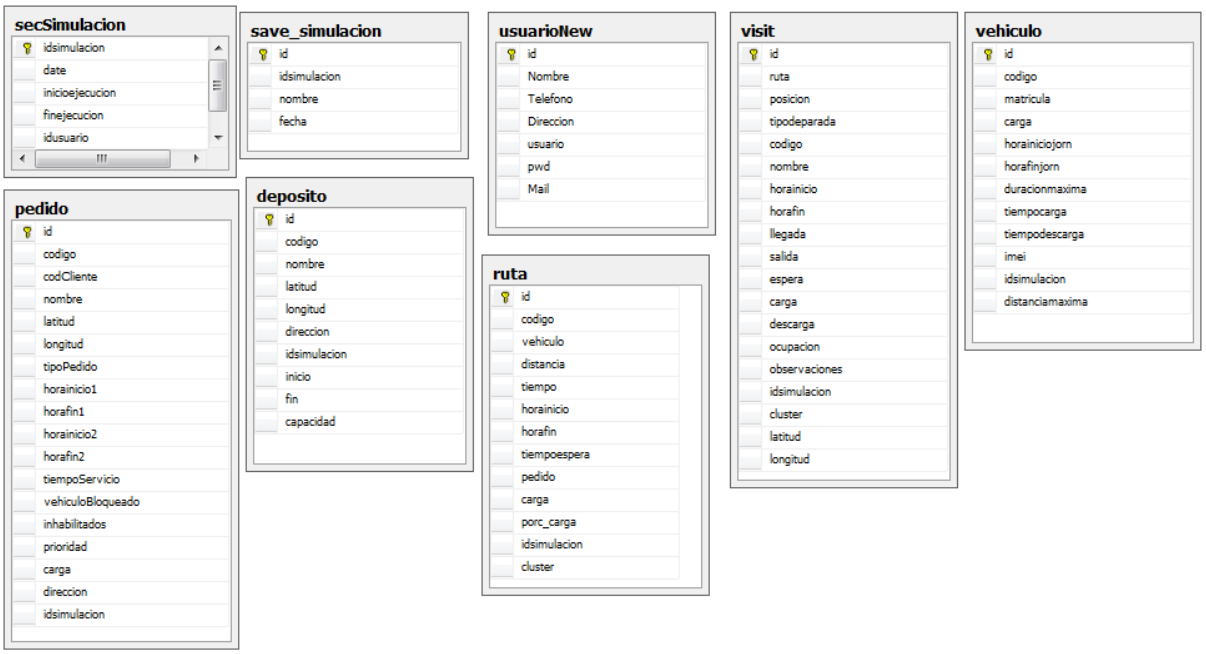



***Figura 13. Diagrama de tabla de base de datos***

Las tablas de la base de datos se pueden categorizar en tres categorías:

- Tablas donde se almacenan datos generales del problema escenario que se desea simular.

| Tabla | Descripción |
|---|---|
| secSimulacion | Tabla donde se generan los id únicos para cada simulación |
| saveSimulación | Tabla donde si el usuario desea guardar el problema que está ejecutando para su posterior uso, se almacenan la fecha y nombre del problema para su posterior identificación. |
| usuarioNew | Tabla donde se almacenan los usuarios registrados en el sistema |

- Tablas de entrada de datos del problema, donde se almacenan los datos leídos por la aplicación al procesar el archivo Excel.

| Tabla | Descripción |
|---|---|
| deposito | Tabla donde se almacenan los datos de los depósitos. |
| Pedido | Tabla donde se almacenan los datos de los pedidos. |
| Vehiculo | Tabla donde se almacenan los datos de los vehículos |

- Tablas de resultados del algoritmo MDVRP, donde se almacenan las rutas y las visitas que se realizan en las mismas, como consecuencia de la ejecución del algoritmo.

| Tabla | Descripción |
|---|---|
| Ruta | Tabla donde se almacenan los datos de las rutas generadas por la ejecución del algoritmo. |
| Visit | Tabla donde se almacenan los datos de los pedidos asignados a las rutas resultantes. |

## Generador de Casos de Prueba

La generación de los casos de prueba se implementará en el lenguaje de programación C# como parte del proyecto de la librería de algoritmos. El mismo exportará cada caso generado a un archivo Excel para luego ejecutarlo desde la interfaz web y poder hacer uso de las bondades de ésta.

# Capítulo 5

# Testeos

## Introducción

En este capítulo se detalla la elaboración de los casos de prueba utilizados para evaluar los distintos algoritmos desarrollados en el contexto de este Proyecto de Grado, así como la justificación y los distintos resultados obtenidos.

## Casos de estudio

Los casos de prueba presentados en esta sección se basan en la representación de los clientes y depósitos en coordenadas geográficas. Se crearon distintos escenarios con los clientes con su demanda y ventana horaria y los depósitos con su capacidad de almacenamiento y ventana horaria.

Fue sugerido por los tutores en un principio utilizar problemas de la bien conocida librería de problemas TSPLib, pero la misma no incluye problemas de MDVRP con las restricciones que se manejan en este proyecto, por lo cual fue descartada para realizar las pruebas. Tampoco se encontraron resultados oficiales para los casos de MDVRP con las restricciones lo cual motivó la generación de diversos casos de prueba para comparar los resultados entre sí.

- **Caso 1:**
  - 10 depósitos con igual capacidad y ventana horaria amplia (12 horas). La capacidad de los depósitos cubre la demanda de los clientes.
  - 100 clientes con igual demanda y ventana horaria amplia (12 horas).
  - 10 vehículos con capacidad igual a los depósitos.

  Este caso es pequeño, sin problemas para satisfacer la demanda de los clientes ni restricciones horarias muy fuertes. Se quiere probar en este caso el peso de la geografía para armar los clústeres y rutas.

- **Caso 2:**
  - 50 depósitos con igual capacidad.
  - 1000 clientes con demanda variable.
  - 50 vehículos.

Este caso es grande y de él derivan 5 casos distintos que juegan con la capacidad del depósito y las ventanas horarias.

- ❖ Caso 2.1: La suma de la demanda de los clientes es exactamente igual a la suma de las capacidades de los depósitos. Las ventanas horarias son amplias tanto para depósitos como para clientes.
- ❖ Caso 2.2: La suma de las capacidades de los depósitos es 20% mayor a la suma de las demandas de los clientes. Las ventanas horarias son amplias tanto para depósitos como para clientes.
- ❖ Caso 2.3: La suma de las capacidades de los depósitos es 40% mayor a la suma de las demandas de los clientes. Las ventanas horarias son amplias para los depósitos, pero no así para los clientes.
- ❖ Caso 2.4: La suma de las capacidades de los depósitos es 40% mayor a la suma de las demandas de los clientes. Las ventanas horarias son acotadas para los depósitos, siendo que algunos se limitan a la mañana o a la tarde, pero no así para los clientes.
- ❖ Caso 2.5: La suma de las capacidades de los depósitos es 40% mayor a la suma de las demandas de los clientes. Las ventanas horarias son acotadas para los depósitos, siendo que algunos se limitan a la mañana o a la tarde, del mismo modo que para los clientes.

- **Caso 3:**
  - 50 depósitos con igual capacidad.
  - 1000 clientes con demanda variable.
  - 50 vehículos.

Este caso es grande y de él derivan 2 casos distintos que juegan con las ventanas horarias y la ubicación geográfica.

- ❖ Caso 3.1: La suma de las capacidades de los depósitos es 100% mayor a la suma de las demandas de los clientes. Las ventanas horarias son acotadas para los depósitos y para los clientes. Adicionalmente se asignan ventanas no muy compatibles para clientes cercanos geográficamente.
- ❖ Caso 3.2: La suma de las capacidades de los depósitos es 100% mayor a la suma de las demandas de los clientes. A diferencia del caso anterior, las ventanas horarias son amplias para los depósitos y para los clientes.

- **Caso 4:** Se generan casos aleatorios con gran número de clientes y depósitos (entre 5000 y 10000) con el fin de verificar el correcto comportamiento de los algoritmos y el sistema.

## Plan de Pruebas

Sobre los casos 1, 2, 3 y 4 planteados en la sección anterior se realizarán las pruebas de los algoritmos de MDVRP implementados.

El caso 1 de estudio tiene como propósito poder comparar las distintas soluciones que resultan de ejecutar los distintos algoritmos de asignación, así como medir el impacto que tiene el algoritmo de revisión de fronteras y por último la mejora de resultados con la post optimización.

El caso 2 de estudio, plantea un escenario de pruebas donde se medirá el impacto que tienen las restricciones de capacidad y de tiempo en la problemática a resolver. De todos modos, se realizarán ejecuciones con los distintos algoritmos para determinar el correcto funcionamiento de estos y poder comparar sus tiempos de ejecución y resultados.

El caso 3 por su parte busca comprobar que ventanas horarias no compatibles en clientes muy cercanos puede afectar fuertemente la construcción de clústeres y rutas a priori intuitivas para el usuario. Al igual que para el caso anterior, se realizarán ejecuciones con los distintos algoritmos para determinar el correcto funcionamiento de los mismos y poder comparar tiempos de ejecución y resultados.

Por último, el caso 4 busca no solo comparar los tiempos de ejecución de los distintos algoritmos y sus resultados, sino que también mide el impacto de la cantidad de nodos los tiempos de ejecución.

## Ejecución y Resultados

A continuación, se presentan los resultados obtenidos al aplicar las distintas combinaciones de algoritmos implementados. Para este primer cuadro comparativo se tienen en cuenta las ejecuciones de los algoritmos de asignación seguidos por el algoritmo de Clarke & Wright sin ningún tipo de post optimización ni mejora de asignación. Los valores de tiempo están expresados en segundos y el costo es la distancia en metros.

Algoritmo Paralelo Simplificado (PAS) + Clarke & Wright

| | **Tiempo Ejecución (s)** | | **Costo** |
|---|---|---|---|
| | **PAS** | **C&W** | |
| **Caso 1** | 0.073632 | 0.115168 | 1152424.4 |
| **Caso 2.a** | 33.917204 | 55.338596 | 6924705.8 |
| **Caso 2.b** | 37.18103861 | 66.09962419 | 7156212.8 |
| **Caso 2.c** | 39.7722546 | 62.2078854 | 7252280.9 |
| **Caso 2.d** | 41.5040756 | 62.2561134 | 7186613.6 |
| **Caso 2.e** | 44.2754359 | 58.6906941 | 7283411.7 |
| **Caso 3.a** | 33.1936892 | 61.6454228 | 4873717.5 |
| **Caso 3.b** | 38.3498304 | 52.9592896 | 4836461.9 |

Algoritmo PAM + Clarke & Wright

| | **Tiempo Ejecución (s)** | | **Costo (m)** |
|---|---|---|---|
| | **PAM** | **C&W** | |
| **Caso 1** | 0.075988224 | 0.118853376 | 1165101.1 |
| **Caso 2.a** | 40.01873049 | 50.93292971 | 7146296.3 |
| **Caso 2.b** | 47.03401384 | 57.48601691 | 7313649.5 |
| **Caso 2.c** | 38.71370075 | 65.91792289 | 7484353.9 |
| **Caso 2.d** | 43.30742768 | 62.32044472 | 7236919.9 |
| **Caso 2.e** | 43.39816447 | 62.45101717 | 7370812.6 |
| **Caso 3.a** | 43.93137346 | 51.57161232 | 5024802.7 |
| **Caso 3.b** | 34.22357127 | 58.27256729 | 4913845.3 |

Algoritmo UPGMC + Clarke & Wright

| | Tiempo Ejecución (s) | | Costo (s) |
|---|---|---|---|
| | **UPGMC** | **C&W** | |
| **Caso 1** | 0.15812108 | 0.23718162 | 1135138.1 |
| **Caso 2.a** | 78.411996 | 95.836884 | 6848534 |
| **Caso 2.b** | 85.64999382 | 118.2785629 | 6998776.1 |
| **Caso 2.c** | 77.75753849 | 126.8675628 | 7128992.1 |
| **Caso 2.d** | 95.78206348 | 112.4398136 | 7193800.2 |
| **Caso 2.e** | 75.23127793 | 128.0965003 | 7261561.4 |
| **Caso 3.a** | 83.48732674 | 102.040066 | 4834727.7 |
| **Caso 3.b** | 71.04846978 | 120.9744215 | 4802606.7 |

En el siguiente cuadro, se incluyen los resultados de la ejecución de los casos de prueba utilizando en esta oportunidad el Algoritmo de Revisión de Fronteras.

Algoritmo PAS + ARF + Clarke & Wright

| | Tiempo ejecución (s) | | Costo (m) |
|---|---|---|---|
| | **PAS + ARF** | **C&W** | |
| **Caso 1** | 0.1320278 | 0.16136736 | 1129375.941 |
| **Caso 2.a** | 59.243537 | 72.40876775 | 6723889.293 |
| **Caso 2.b** | 56.515179 | 92.20897575 | 6991619.876 |
| **Caso 2.c** | 60.18256 | 98.1925976 | 7158001.26 |
| **Caso 2.d** | 64.828328 | 85.93522613 | 7129120.681 |
| **Caso 2.e** | 59.989097 | 86.32577373 | 7086759.545 |
| **Caso 3.a** | 55.993012 | 68.43590322 | 4824980.276 |
| **Caso 3.b** | 44.899434 | 76.4503869 | 4705877.429 |

Algoritmo PAM + ARF + Clarke & Wright

| | Tiempo ejecución (s) | | Costo (m) |
|---|---|---|---|
| | **PAM + ARF** | **C&W** | |
| **Caso 1** | 0.1518228 | 0.155047152 | 1144129.279 |
| **Caso 2.a** | 59.986672 | 72.52849191 | 6974785.232 |
| **Caso 2.b** | 66.715686 | 103.3807624 | 7182003.761 |
| **Caso 2.c** | 69.698196 | 77.12396979 | 7342151.177 |
| **Caso 2.d** | 58.949267 | 81.9513848 | 7106655.327 |
| **Caso 2.e** | 54.109003 | 89.08267127 | 7333958.53 |
| **Caso 3.a** | 64.092221 | 82.30829327 | 4864009.005 |
| **Caso 3.b** | 62.291641 | 78.00754342 | 4835223.766 |

Algoritmo UPGMC + ARF + Clarke & Wright

| | Tiempo Ejecución (s) | | Costo (m) |
|---|---|---|---|
| | **UPGMC + ARF** | **C&W** | |
| **Caso 1** | 0.308818375 | 0.273857805 | 1114705.578 |
| **Caso 2.a** | 143.2761416 | 117.225934 | 6629380.909 |
| **Caso 2.b** | 198.1206714 | 111.4428777 | 6900793.224 |
| **Caso 2.c** | 162.6032905 | 127.7597282 | 6957896.325 |
| **Caso 2.d** | 162.1382113 | 138.1177355 | 7049924.2 |
| **Caso 2.e** | 174.5101322 | 111.5720517 | 7029191.459 |
| **Caso 3.a** | 164.4885864 | 100.8155852 | 4689685.879 |
| **Caso 3.b** | 170.8158832 | 100.3204393 | 4687344.107 |

Post Optimización

A continuación, se presentan los resultados de la ejecución de los casos de prueba de la sección anterior, pero utilizando métodos de post optimización. A modo de poder analizar la relación entre la mejora de los costos y el incremento del tiempo solo se presentarán los resultados para el caso donde tanto Or Opt como $\lambda$ toman el valor 3.

Algoritmo PAS + ARF + Clarke & Wright + Post Optimización

| | Tiempo Ejecución (s) | Costo (m) |
|---|---|---|
| **Caso 1** | 3.64256 | 910415.2997 |
| **Caso 2.a** | 303.46972 | 5262776.378 |
| **Caso 2.b** | 774.604971 | 5581845.961 |
| **Caso 2.c** | 887.227218 | 6019393.157 |
| **Caso 2.d** | 883.9126993 | 5749290.872 |
| **Caso 2.e** | 998.601777 | 6118065.794 |
| **Caso 3.a** | 788.8403592 | 4142659.833 |
| **Caso 3.b** | 493.069248 | 3627346.425 |

Algoritmo PAM + ARF + Clarke & Wright + Post Optimización

| | Tiempo ejecución (s) | Costo (m) |
|---|---|---|
| **Caso 1** | 3.993463378 | 972509.8871 |
| **Caso 2.a** | 556.9152057 | 5300836.776 |
| **Caso 2.b** | 836.8918862 | 5314682.783 |
| **Caso 2.c** | 956.908514 | 5653456.406 |
| **Caso 2.d** | 888.9641741 | 6040657.028 |
| **Caso 2.e** | 695.9583693 | 5573808.488 |
| **Caso 3.a** | 1155.204116 | 3307526.123 |
| **Caso 3.b** | 975.0942927 | 3578065.587 |

Algoritmo UPGMC + ARF + Clarke & Wright + Post Optimización

| | Tiempo Ejecución (s) | Costo (m) |
|---|---|---|
| **Caso 1** | 5.651958944 | 813735.0722 |
| **Caso 2.a** | 2526.870133 | 5104623.3 |
| **Caso 2.b** | 2476.508393 | 5313610.782 |
| **Caso 2.c** | 1945.432226 | 5705474.987 |
| **Caso 2.d** | 840.7166511 | 5146444.666 |
| **Caso 2.e** | 2088.399943 | 4779850.192 |
| **Caso 3.a** | 2493.859213 | 3235883.257 |
| **Caso 3.b** | 2521.567799 | 3843622.167 |

## Análisis de los Resultados

Al analizar las ejecuciones donde se realizan asignación y ruteos puramente se desprende que el Algoritmo Paralelo Simplificado es el que presenta la mejor relación costo/tiempo de ejecución. En este algoritmo particularmente se pueden ver los inconvenientes que pueden generar las ventanas de tiempo sobre todo en clientes que se encuentran ubicados muy cercanos entre ellos.

UPGMC tiene mejores resultados en algunos casos, pero siempre implica un mayor tiempo de cálculo. Esto hace que a medida que crecen las instancias a resolver sea realmente menos conveniente utilizarlo.

Por último, en lo que respecta a los algoritmos de asignación, PAM obtiene buenos resultados, pero no destaca por sobre los demás ni en tiempo de ejecución ni en el costo de la solución. Sí obtiene clústeres más balanceados en cantidad de clientes respecto a la solución obtenida con PAS, pero este no era un requerimiento de este proyecto.

El Algoritmo de Revisión de Fronteras obtiene mejoras en las soluciones de la asignación del orden del 3% pero incrementa considerablemente el tiempo de ejecución. En los casos donde existe un buen porcentaje de holgura, es decir la capacidad de los depósitos es mucho mayor que la carga de los clientes la cantidad de iteraciones de este algoritmo aumenta considerablemente. Parecería que no vale la pena incorporarlo a medida que aumenta la cantidad de nodos a procesar y en los casos donde los depósitos están subutilizados.

En lo que respecta a la post optimización, la incorporación de los algoritmos de Or Opt e Intercambio Lambda mejora los resultados considerablemente. Las soluciones obtenidas tienen rutas con menos cruzamiento y un orden a simple vista más intuitivo para la realización de las visitas. El gran inconveniente de estos algoritmos es el alto tiempo de ejecución que conllevan. Para planificaciones reales donde el tiempo de ejecución es un factor sensible puede no ser la alternativa ideal.

# Capítulo 6

## Conclusiones

Como se ha podido ver a lo largo de los capítulos de este informe, MDVRP, es un problema bajo investigación, importante para el sector logístico, donde todos los meses se pueden encontrar nuevas publicaciones académicas al respecto.

El aspecto más importante de este trabajo es la realización de un relevamiento de métodos para resolver VRP y MDVRP, generando conocimiento que permitió la construcción de una aplicación web destinada a resolver problemas MDVRP con capacidades y ventanas de tiempo.

El relevamiento cumplió con el objetivo de facilitar el acercamiento al problema y a sus variantes, a pesar de la dificultad que plantea la inmensa cantidad de propuestas de solución. Particularmente, durante la elaboración del Estado del Arte se estudiaron las heurísticas y metaheurísticas más difundidas y utilizadas, así como nuevas propuestas más puntuales. La elaboración del Estado del Arte se extendió por un mayor período de tiempo del que fue planificado, atrasando las etapas posteriores del proyecto. A esto se le sumó que uno de los integrantes del proyecto pasó por problemas personales y se debió realizar una pausa en el mismo.

La construcción de una solución web, permitió crear un ambiente de fácil acceso, sin necesidad de que los usuarios que deseen resolver sus problemas deban instalar algún software en particular. Con solo contar con un navegador web se logra, ejecutar los diferentes problemas de cada usuario, permitiendo una alta disponibilidad del sistema.

El sistema se desarrolló con la tecnología HTML5 lo que permite, que el sistema pueda ser ejecutado y compatible con la totalidad de navegadores web que existen actualmente. Otra característica importante es la portabilidad de la solución diseñada que al ser una página web se puede ejecutar en los diferentes dispositivos móviles eliminando la necesidad de descargar aplicativos nativos de cada sistema operativo.

La representación de los depósitos, pedidos y rutas en el mapa de la aplicación web permitió observar de forma amigable tanto el problema como la solución lo cual es muy práctico para el análisis de cada caso y los resultados obtenidos.

La solución permite configurar con qué parámetros se ejecuta el algoritmo, definiendo entre otros, qué algoritmo de asignación utilizar y el nivel de post

optimización deseado. Con esto se logra mayor control sobre las ejecuciones y permite una experimentación más en profundidad sobre las posibles soluciones.

En lo que respecta a los algoritmos implementados para la fase de asignación, se obtienen mejores resultados con el Parallel Simple Algorithm, lo cual coincide con los resultados presentados previamente por la academia. El algoritmo desarrollado para mejorar la asignación cumple su objetivo y disminuye los costos posteriores de las rutas asociadas a los clústeres.

El algoritmo de Clarke & Wrigth cumplió con las expectativas y resultó no solo ser sencillo de implementar, sino que obtiene soluciones razonables en poco tiempo.

La post optimización mejora considerablemente la solución a medida que aumenta el nivel, pero tiene como desventaja que insume prolongados tiempos de cálculo.

En definitiva, se implementó un sistema que se puede utilizar desde prácticamente cualquier dispositivo, donde se encapsularon los algoritmos en una librería que obtiene buenas soluciones y que además ofrece la posibilidad de extenderse con nuevos métodos.

## Trabajos futuros

Entre los posibles trabajos a futuros se destaca la implementación de nuevos algoritmos que resuelvan MDVRP y mejoras para los ya existentes. Este proyecto de grado brinda una interfaz web y una librería de algoritmos que permite que la incorporación tanto de algoritmos, mejoras y nuevas configuraciones para resolver MDVRP sea sencilla.

Una rama que se encuentra creciendo actualmente es la de Inteligencia Artificial que se podría incorporar en un futuro para las mejorar tanto la asignación de pedidos a clústeres como su posterior ruteo.

El desarrollo se basa en una aplicación web desarrollada en HTML5, que ejecuta una librería alojada en el servidor de aplicaciones web, responsable de realizar todos los cálculos de los algoritmos. Se podría entonces en un futuro realizar una arquitectura de clúster para mejorar los tiempos de respuesta en un uso masivo de la aplicación web.

Los datos del problema y los resultados en este proyecto son visualizados mediante un mapa donde se localizan los pedidos, depósitos, rutas y clústeres,

permitiendo así la posibilidad de en un futuro poder incorporar la amplia gama de funcionalidades que exponen las librerías de Sistemas de Información Geográfica que existen actualmente, enriqueciendo la aplicación tanto en visualización como en información del entorno geográfico.

Al tratarse de una aplicación web basada en una arquitectura cliente servidor, en un futuro se puede dejar disponible la aplicación en la nube donde los usuarios puedan contratar un servicio que les permita la ejecución de diferentes escenarios de problemas de MDVRP. De acuerdo a las necesidades del cliente podrían definirse diferentes tipos de precios para utilizar el servicio. Del mismo modo, se podría otorgar un servicio similar pero gratuito a distintos Estados y organizaciones que traten problemas de toma de decisiones para casos de desastres naturales y logística humanitaria. Esto les permitiría obtener una solución rápida para distribuir de forma más eficiente los medicamentos y otro tipo de asistencia.

Otro aspecto en el que se puede trabajar es en la migración del servidor de base de datos SQL Server Express a un SQL Server Enterprise lo cual permite mayor capacidad de almacenamiento de escenarios y mayor disponibilidad.

Este proyecto en particular se enfocó en la resolución de la variante de MDVRP donde se consideran ventanas de tiempo y capacidades. Queda pendiente para un futuro considerar otras variantes donde por ejemplo se cuente con flota heterogénea de vehículos, con pedidos de recolección, puntos de carga específicos para ciertos pedidos, entre otros.

En resumen, existe una gran variedad de opciones que se podrían considerar para enriquecer la solución propuesta.

# Anexo

# Manual de Usuario

## Introducción

En este documento se describe el manual de usuario para la interacción con el sistema Web que implementa la solución al problema de MDVRP definido para este proyecto de grado.

## Propósito

Guiar al usuario en el uso del sistema para su correcto uso y presentar una descripción de las funcionalidades que están permitidas realizar.

A continuación, se observa la página de inicio del sistema:

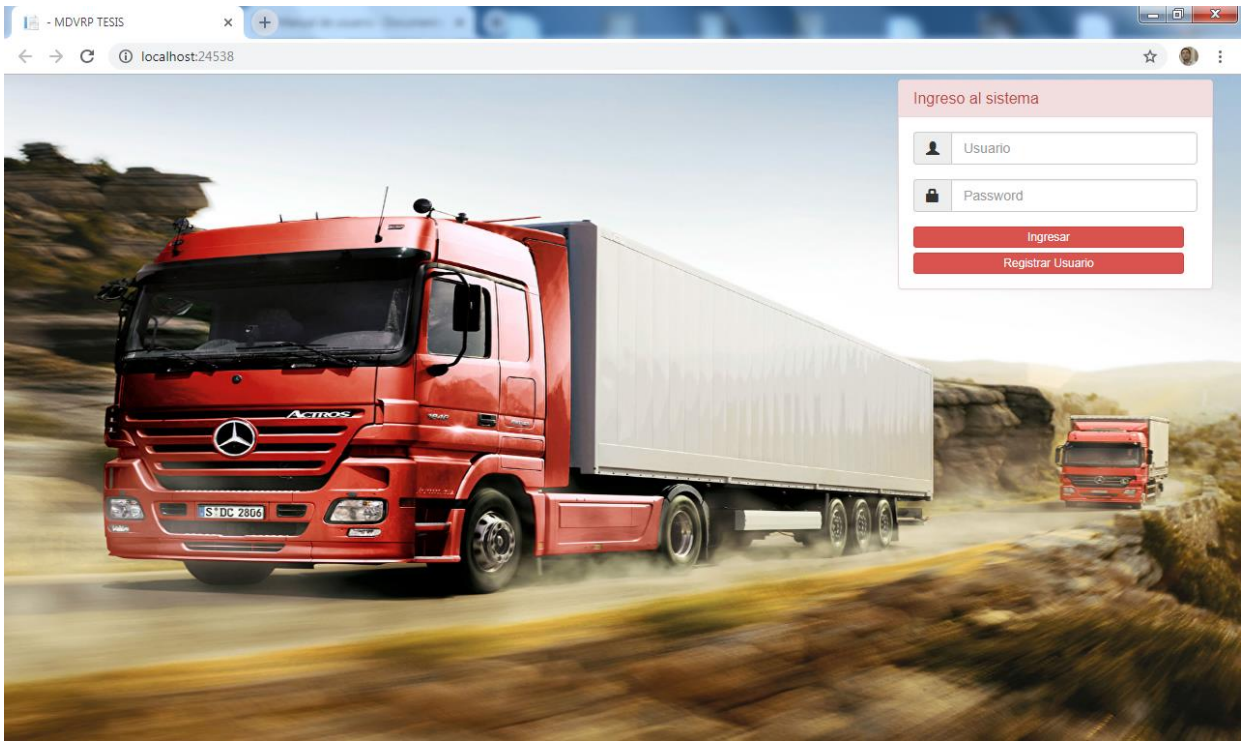



***Figura 14. Página de inicio del Sistema Web***

# Funcionalidades

## Registro de Usuario

Para ingresar al sistema, el usuario se debe registrarse ingresando los datos que se muestra en la imagen de a continuación.

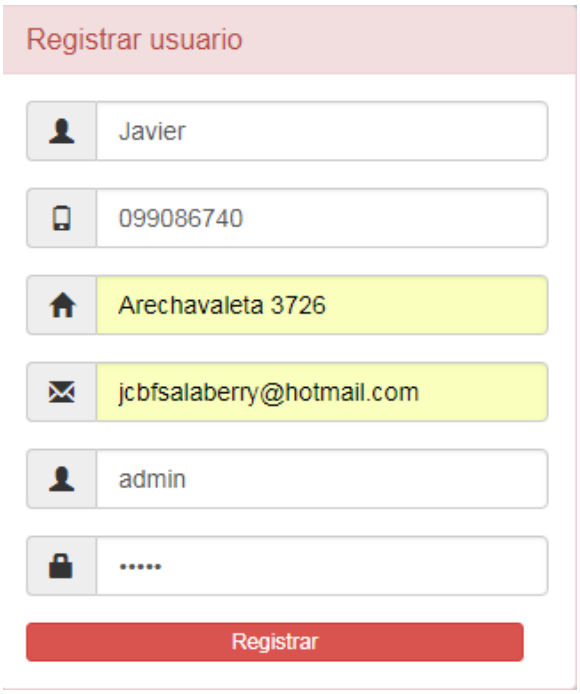



***Figura 15. Registro de Usuario***

La necesidad de registrar usuarios surge para poder separar los distintos escenarios de los usuarios que utilizan el sistema. No es conveniente que un usuario acceda a escenario de otro.

## Ingreso de Usuario

Para ingresar al sistema el usuario registrado debe contar con un nombre de usuario y una clave de acceso al sistema.

A continuación, se observa el formulario de ingreso al sistema.

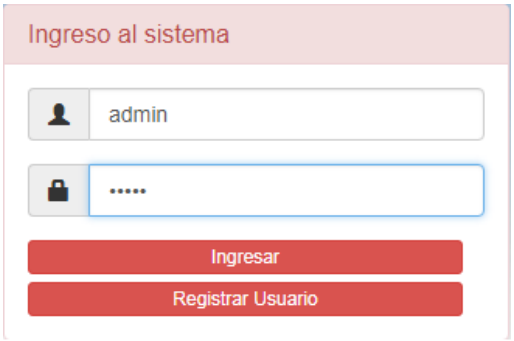



***Figura 16. Ingreso al sistema***

En caso de no estar registrado, el sistema responde con el mensaje genérico de *"Usuario o contraseña incorrecta"*. Luego de ingresar al sistema se observa un mapa de fondo y un menú de funcionalidades a la derecha.

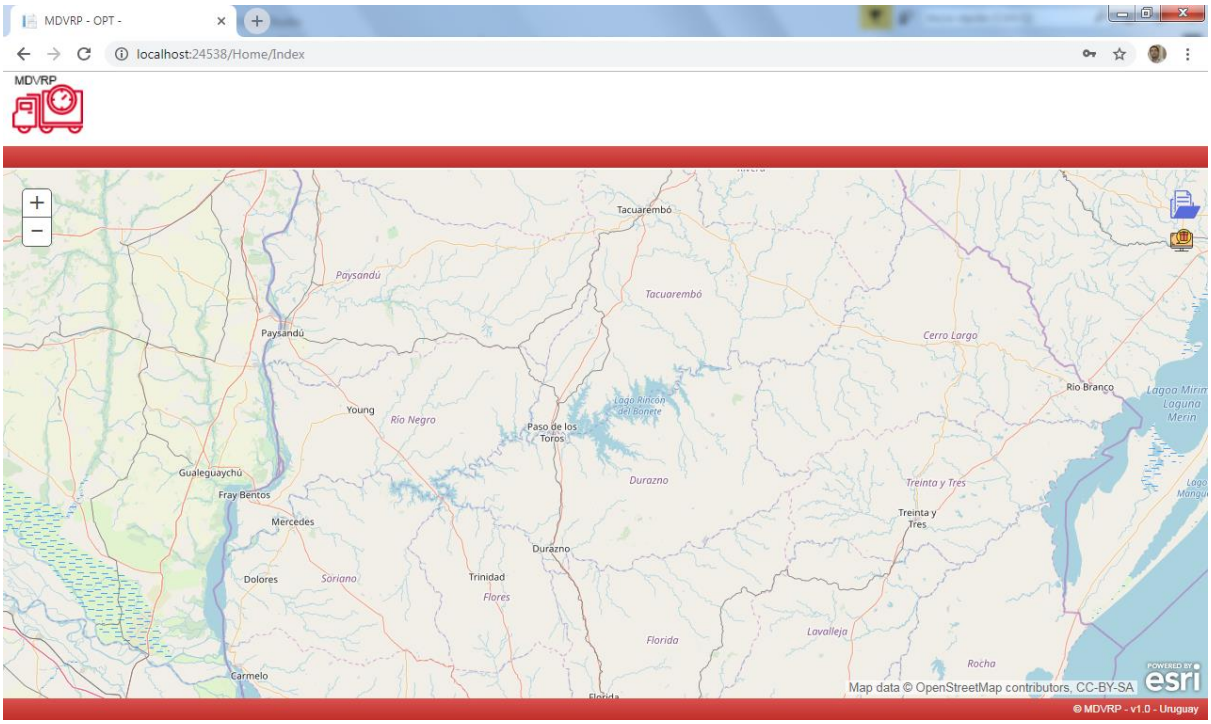



***Figura 17. Sistema Web, mapa base y barra de herramientas***

## Acercar y alejar Mapa

El sistema cuenta con una barra de funcionalidades en la cual se encuentran las herramientas de acercar y alejar la vista en el mapa  . Haciendo clic sobre el símbolo de más y de menos, queda habilitada la posibilidad de acercarnos o alejarnos a una determinada extensión del mapa.

## Cargar pedidos, vehículos y depósitos

Esta funcionalidad permite cargar los datos de los vehículos, depósitos y pedidos en el sistema. Para cargar los datos se debe hacer clic sobre la herramienta *"Cargar pedidos, depósitos y vehículos"*  de la barra de funcionalidades.

Como resultado de esta interacción con el sistema se abre una ventana para seleccionar el archivo Excel con los datos de pedidos, vehículos y depósitos a cargar en el sistema.

Luego de seleccionar el archivo se hace clic en el botón "Subir" y los datos son importados hacia el sistema.

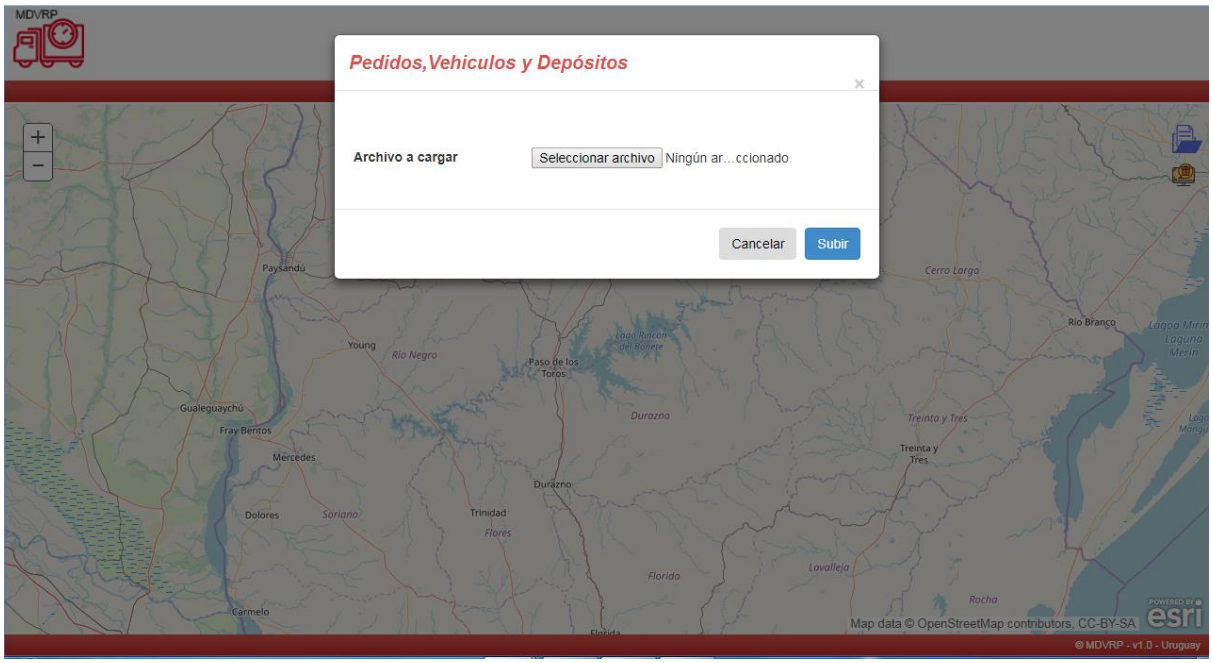



***Figura 18. Importar datos al Sistema Web***

Si no hubo ningún inconveniente en la carga, los depósitos y pedidos se podrán observar de forma inmediata en el mapa. Para cargar los datos al sistema el archivo Excel a utilizar debe contar con un formato determinado que se le proporciona al usuario del sistema.

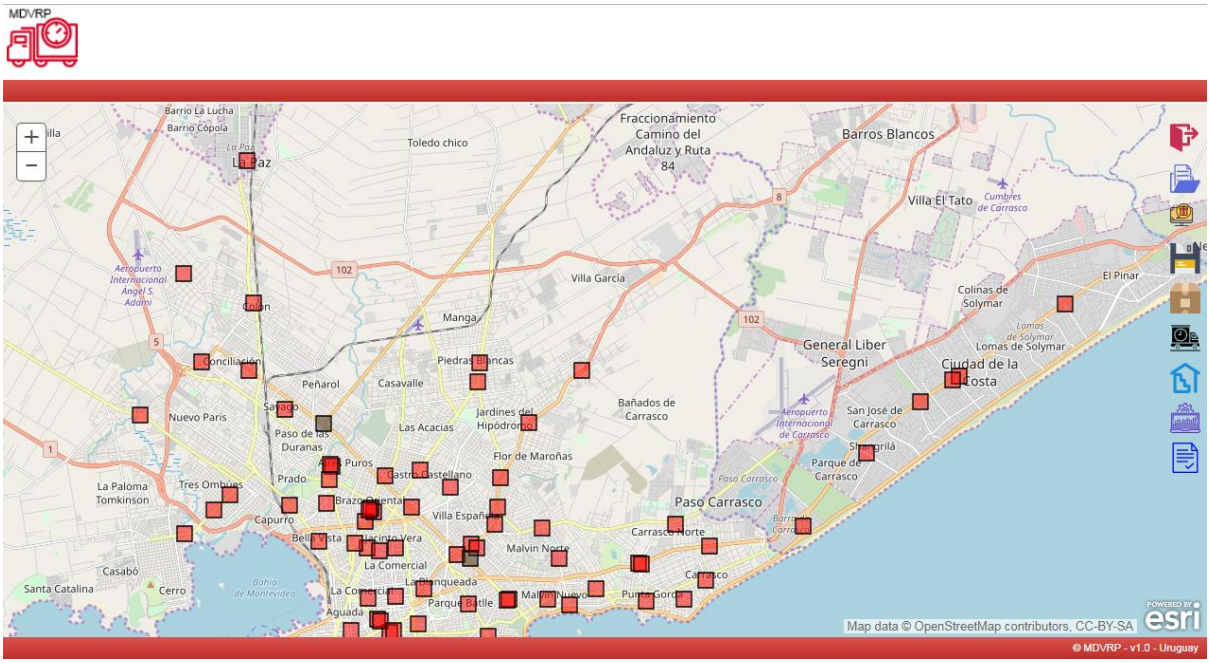



***Figura 19. Visualización de datos cargados.***

Luego de cargar los datos del problema a resolver, en el menú de funcionalidades se habilitan herramientas que nos permiten trabajar en el escenario del problema.

## Agregar pedido de forma manual

Luego de haber generado un problema a resolver, mediante la carga masiva de un escenario con pedidos, vehículos y depósitos, se permitirá agregar pedidos de forma manual mediante la herramienta *"Cargar pedido"* .

Para agregar un pedido de forma manual se debe hacer clic sobre el ícono y luego hacer clic sobre el mapa donde se quiere localizar el pedido. Cuando se localiza el pedido, el sistema despliega una ventana con datos y coordenadas del mismo como se muestra a continuación.

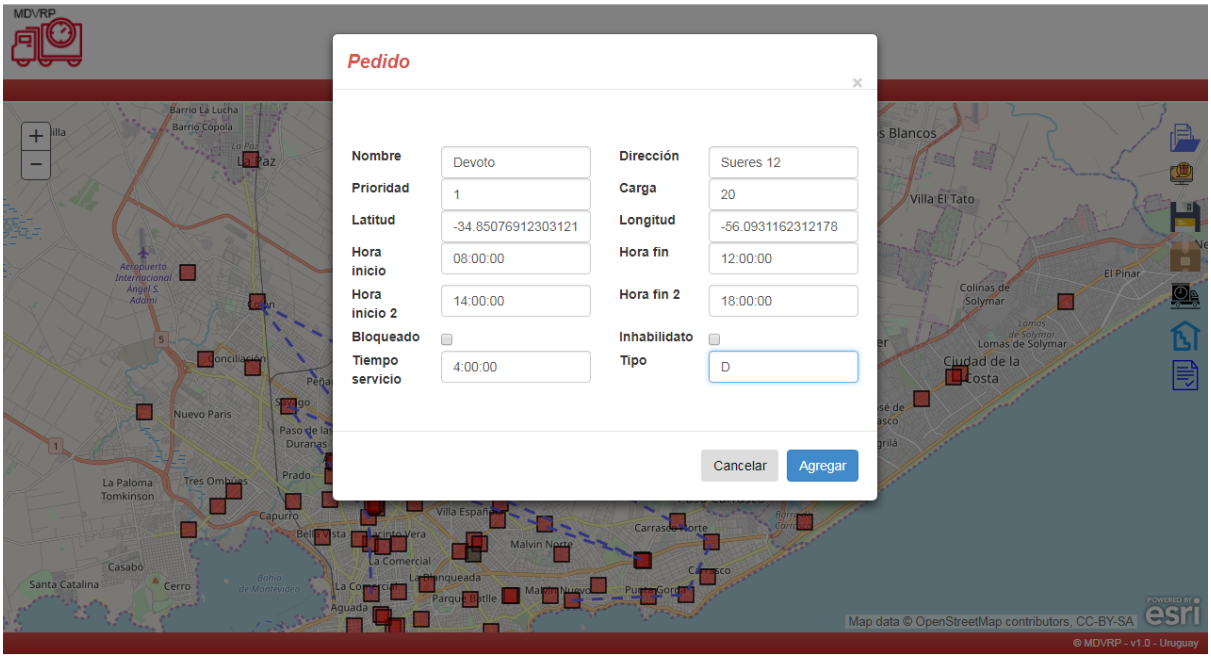



***Figura 20. Agregar Pedido***

Luego de ingresar los datos del pedido, se hace clic sobre el botón *"Agregar"* y el sistema almacena el nuevo pedido y lo asocia al problema que se está intentando resolver, agregándolo al mapa como un nuevo gráfico de pedido.

## Agregar depósito de forma manual

Luego de haber generado un problema a resolver, mediante la carga masiva de un escenario con pedidos, vehículos y depósitos, se permitirá agregar depósitos de forma manual mediante la herramienta "Cargar depósito" . Para agregar un depósito de forma manual se debe hacer primero clic sobre ícono y luego hacer clic sobre el mapa donde se quiere localizar el depósito. Cuando se localiza el depósito, el sistema despliega una ventana con datos y coordenadas del mismo como se muestra a continuación.

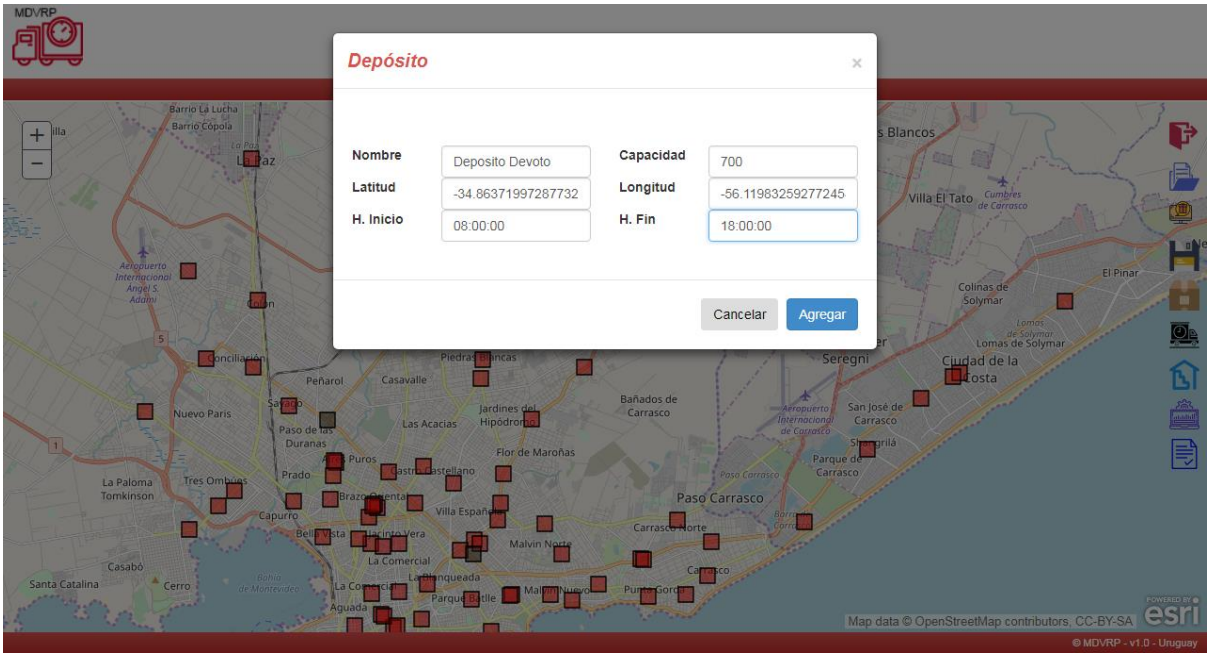



***Figura 21. Agregar depósito***

Luego de ingresar los datos del depósito, se hace clic sobre el botón "*Agregar*" y el sistema almacena el nuevo depósito y lo asocia al problema que se está intentando resolver, agregándolo al mapa como un nuevo gráfico de depósito.

## Agregar vehículo de forma manual

Si se desea agregar un vehículo para que forme parte del problema activo en el sistema, se debe hacer clic en el ícono  y se desplegará una ventana para que se ingresen los datos del vehículo.

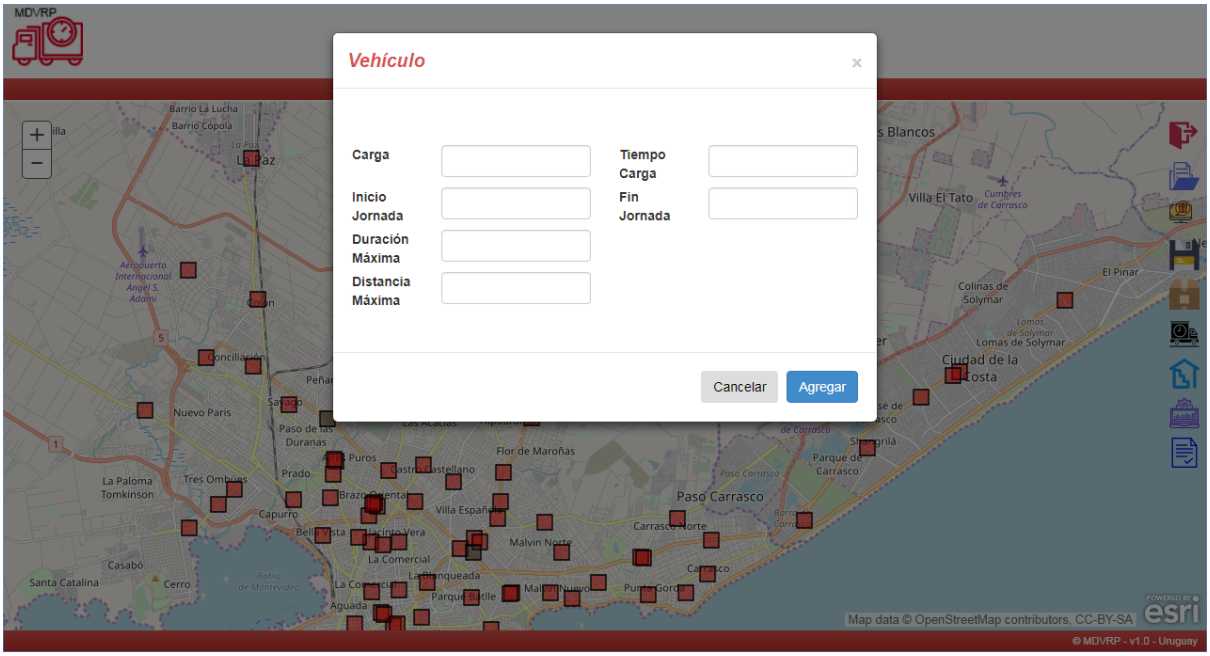



***Figura 22. Agregar vehículo***

Luego de ingresar los datos del vehículo, se hace clic sobre el botón "*Agregar*" y el sistema almacena el nuevo vehículo y lo asocia al problema que se está intentando resolver.

## Identificar pedidos

Si se desea obtener los datos de los pedidos que se encuentran ubicados en el mapa, el usuario puede hacer clic sobre el pedido y como resultado de esta interacción obtiene una ventana informativa con los datos del pedido que seleccionó.

A continuación, se observa la ventana de identificación del pedido.

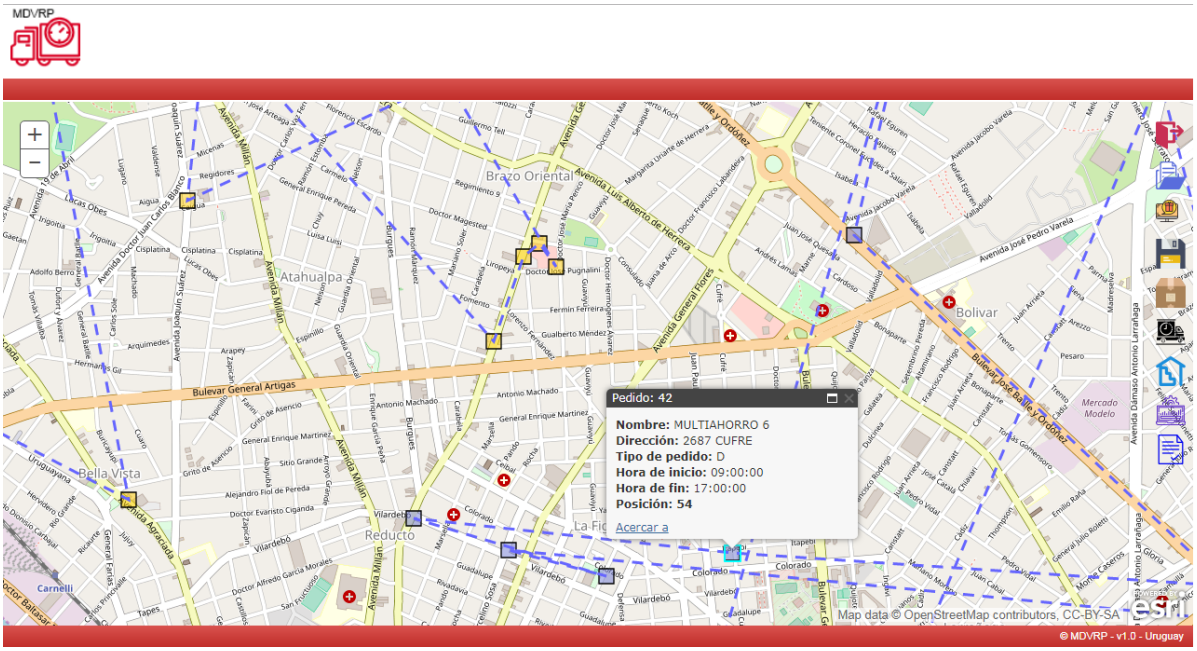



***Figura 23. Identificación de pedidos***

## Identificar depósitos

Si se desea obtener los datos de los depósitos que se encuentran ubicados en el mapa, el usuario hace clic sobre el depósito y como resultado de esta interacción obtiene una ventana informativa con los datos del depósito que seleccionó.

A continuación, se observa la ventana de identificación del depósito.

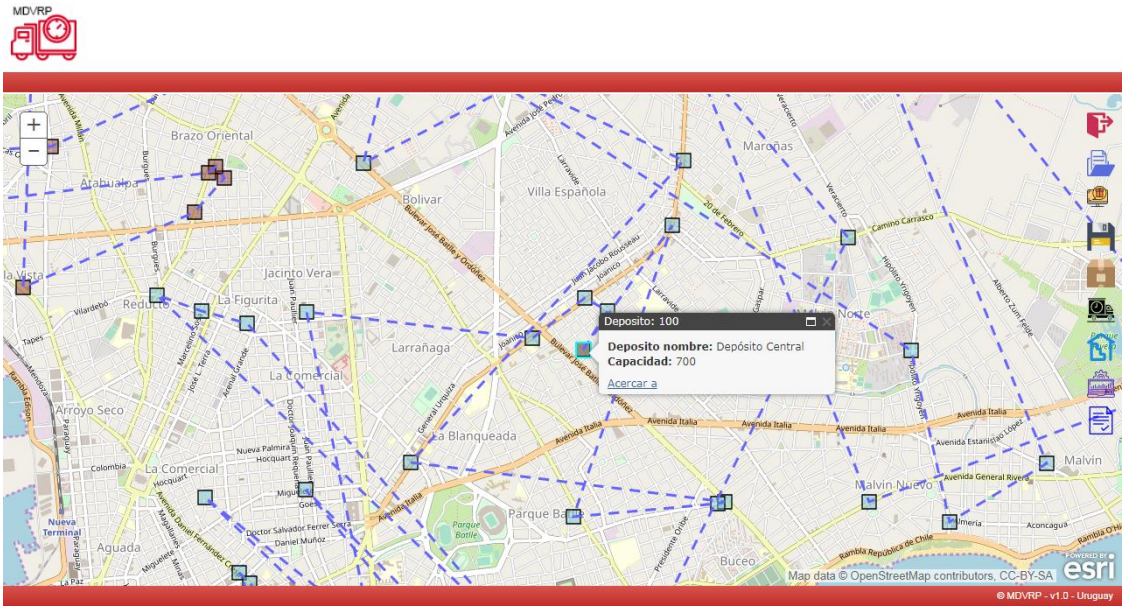



***Figura 24. Identificación de depósitos***

## Guardar escenario

El sistema permite guardar el escenario activo, para su posterior estudio o consulta. Para guardar el escenario se debe hacer clic sobre el ícono y se desplegará una ventana donde se deberá indicar el nombre con el que se desea guardar al escenario.

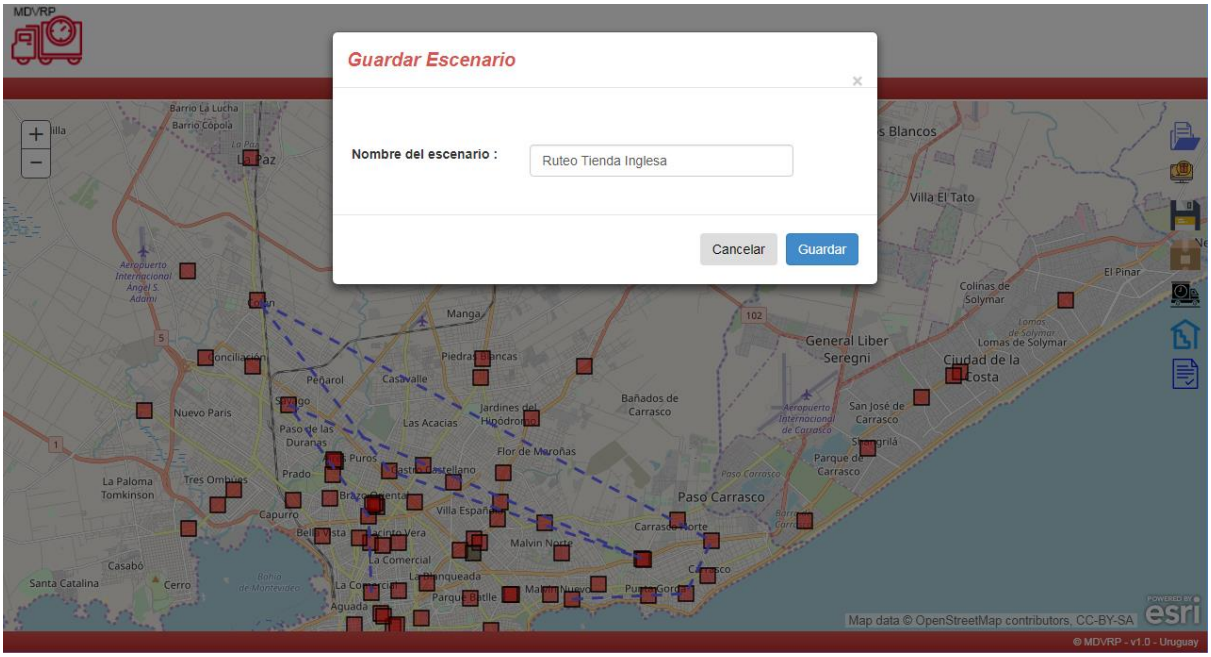



***Figura 25. Guardar escenario***

## Abrir escenario

Si el usuario desea continuar estudiando y trabajando con un problema que fue guardado en el sistema anteriormente, lo puede realizar haciendo clic sobre el ícono y se desplegará una ventana donde se listan los problemas almacenados, para seleccionar el problema a cargar en el sistema.

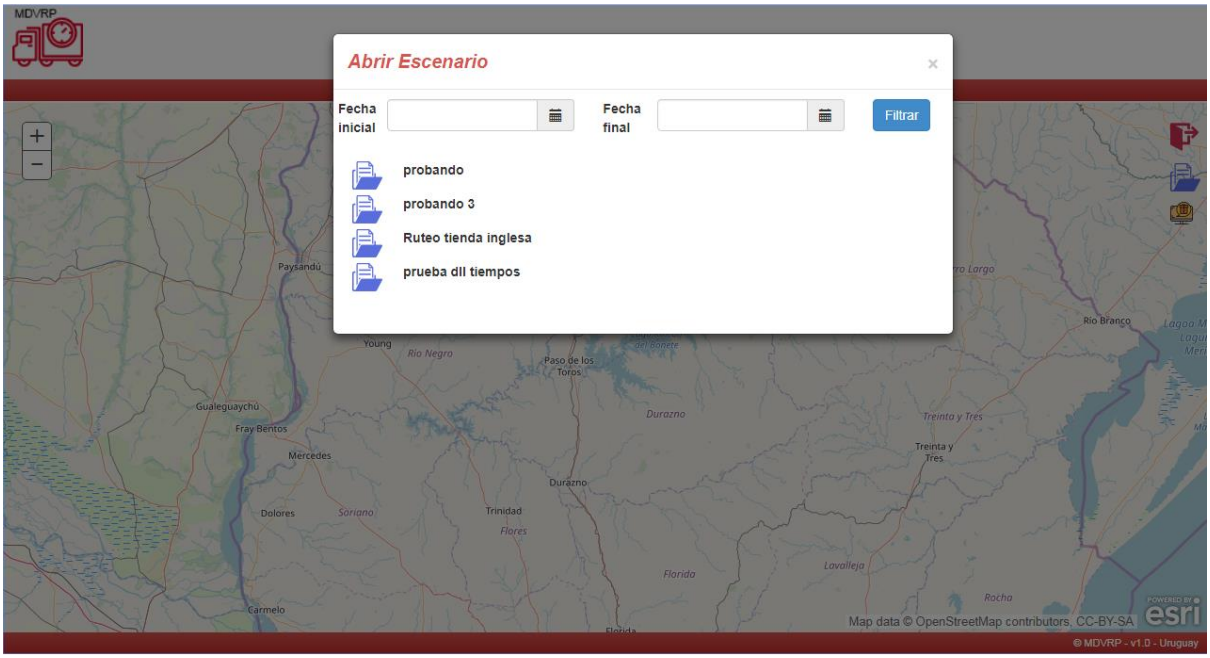



***Figura 26. Abrir escenario***

## Ejecutar proceso de optimización

La funcionalidad de ejecutar proceso de optimización permite al usuario obtener un conjunto de rutas para el problema de MDVRP de la simulación.

Para enviar a ejecutar la resolución del problema MDVRP el usuario debe tener datos cargados en el sistema y hacer clic en la herramienta de ejecutar proceso de optimización de la barra de funcionalidades.

Al hacer clic en la herramienta de procesar algoritmos  se despliega una ventana donde se pueden configurar los parámetros necesarios para la ejecución de los algoritmos de optimización como se observa a continuación.

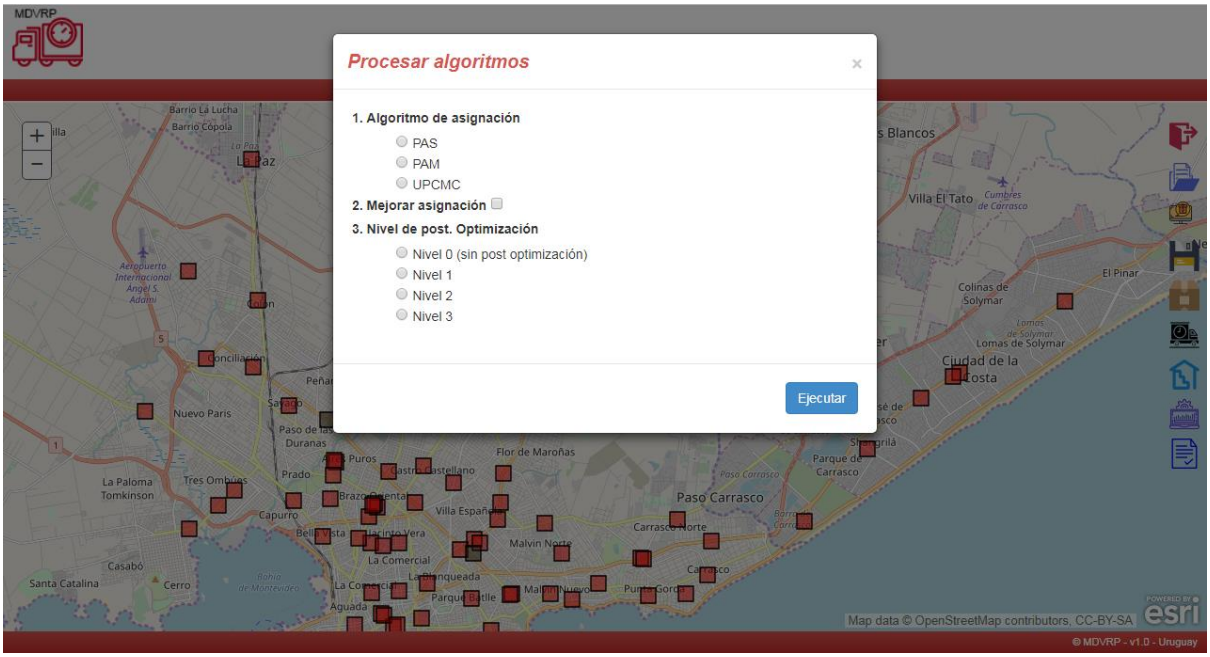



***Figura 27. Modal procesar algoritmos***

Entre los parámetros que se pueden configurar del algoritmo se encuentran la posibilidad de elegir qué algoritmo de asignación se utilizará entre los cuales figura *"Algoritmo Paralelo Simplificado" (PAS), "Partición Alrededor de Medoids" (PAM)* y *"UnweightedPairGroupsMethodCentroid" (UPGMC)*. Además, se podrá seleccionar si se realiza una mejora de asignación y el nivel de post optimización que utilizará el algoritmo.

Luego de realizar la configuración de los parámetros del algoritmo se hace clic en el botón "*Ejecutar*" y se procede a la ejecución del algoritmo.

Cuando el proceso del algoritmo termina se obtiene como resultado los clústeres que agrupan a los pedidos y las rutas que van a realizar los vehículos para poder resolver el problema de optimización. Los resultados de asignación y rutas se observan en el mapa luego de la finalización del proceso de ejecución.

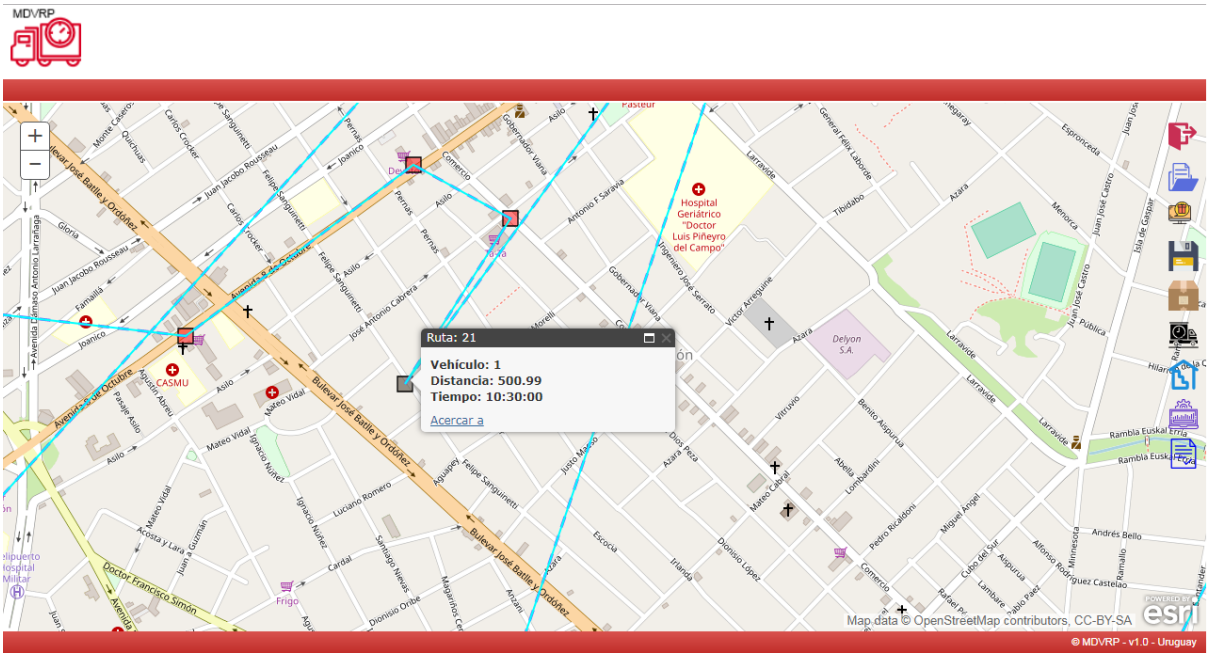



*Figura 28. Identificación de ruta*

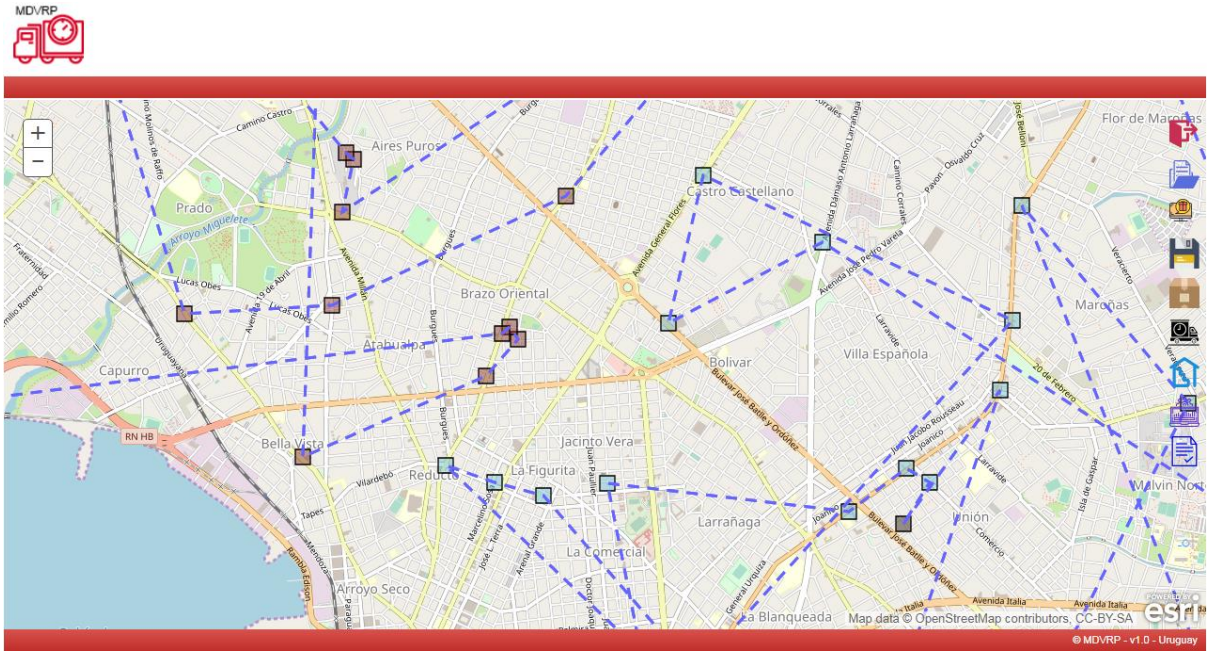



*Figura 29. Rutas resultado de procesar algoritmos*

## Identificar rutas

Si deseamos obtener datos sobre las rutas que se observan en el mapa, el usuario puede hacer clic sobre las mismas y como resultado obtiene una ventana con datos descriptivos de la ruta.

A continuación, se observa la ventana informativa que se despliega sobre la ruta

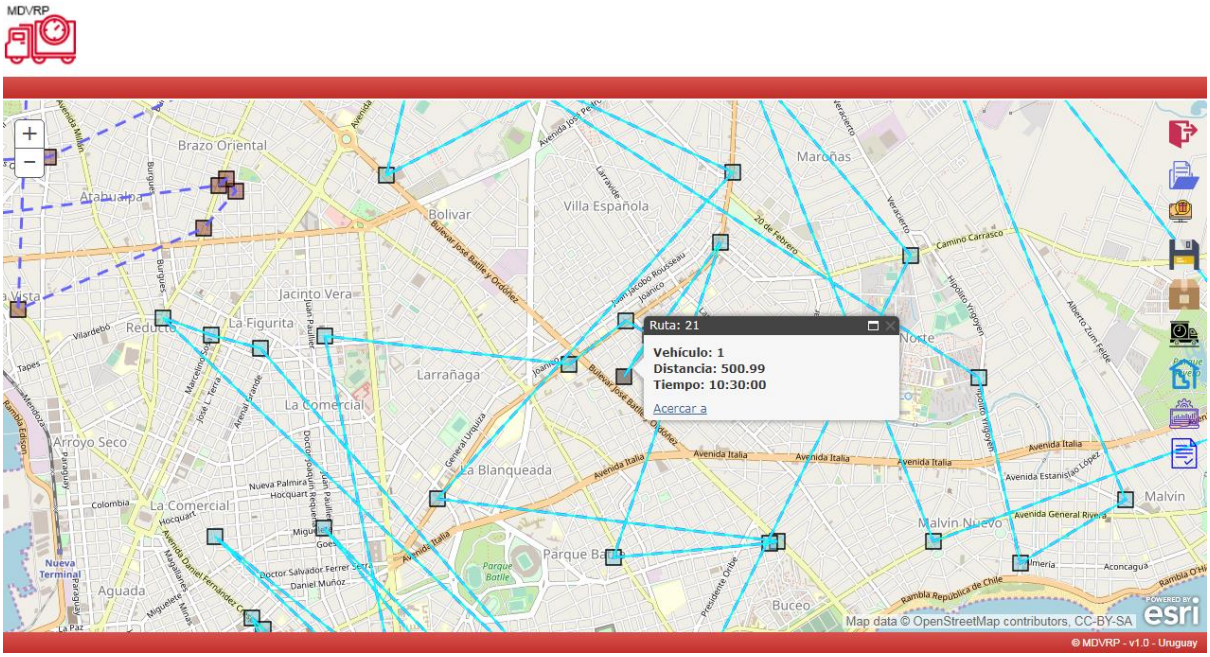



***Figura 30. Identificar rutas***

## Obtener reporte de resultados de optimización

Si el usuario desea obtener en forma tabular los resultados de la optimización puede consultar los mismos haciendo clic en la herramienta del menú de funcionalidades donde se desplegará una ventana con los datos de las rutas resultantes indicando el tiempo, distancia y vehículo que realiza la ruta como se observa en la imagen a continuación.

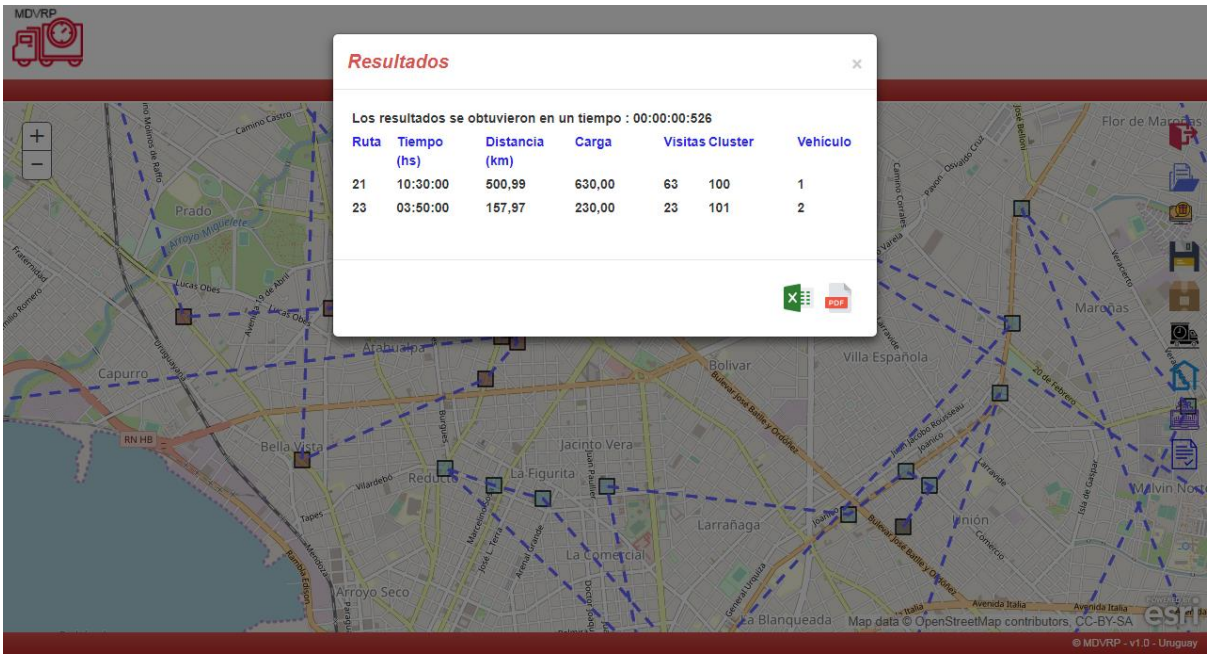



***Figura 31. Ventana de resultados obtenidos al procesar algoritmo***

## Exportar resultados en formato Excel

El sistema cuenta con la posibilidad de extraer un archivo Excel con los resultados del algoritmo ejecutado para resolver el problema. Para descargar en formato Excel los resultados de las rutas, se debe realizar clic en el botón de la ventana "*Resultados*".

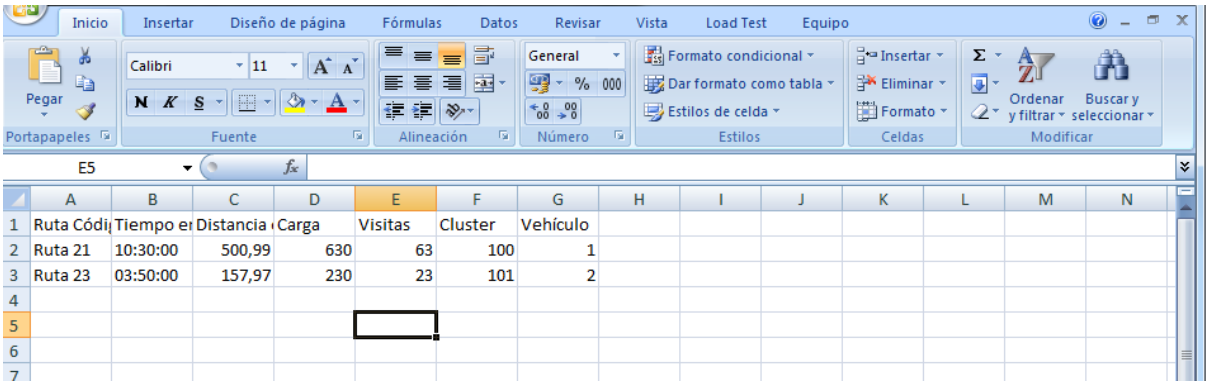



***Figura 32. Exportación a Excel***

## Exportar resultados en formato PDF

El sistema cuenta con la posibilidad de extraer un archivo PDF con los resultados del algoritmo ejecutado para resolver el problema. Para descargar en formato PDF los resultados de las rutas, se debe realizar clic en el botón de la ventana "*Resultados*".

A continuación, se muestra un ejemplo de un reporte PDF.

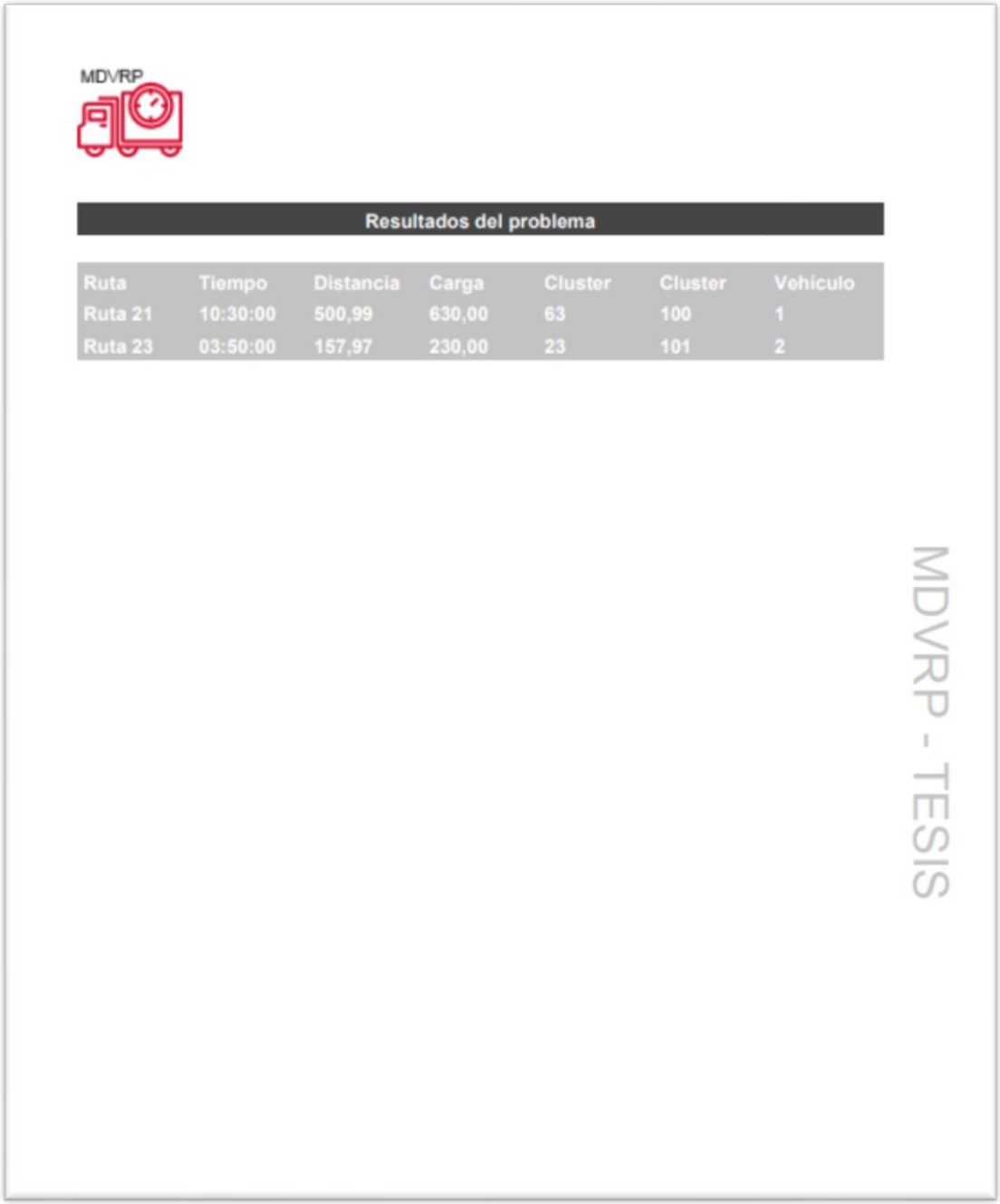



***Figura 33. Exportación a PDF***

# Estado del arte

## Índice General

# Resumen

En este documento se presenta el Estado del Arte sobre el Problema de Ruteo de Vehículos con Múltiples Depósitos en el contexto del Proyecto de Grado de la carrera de Ingeniería en Computación.

El objetivo específico del documento es realizar una revisión del estado del arte del Problema de Ruteo de Vehículos con Múltiples Depósitos (MDVRP por su nombre en inglés), abarcando desde la definición del problema y sus múltiples variantes hasta las soluciones propuestas a los mismos.

El documento se organiza en las secciones que se presentan a continuación.

En el Capítulo 1 de este documento se presenta la descripción del Problema de Ruteo de Vehículos, así como algunas de sus variantes. Se hace particular énfasis en la variante del problema que considera múltiples depósitos (MDVRP).

En el Capítulo 2 se realiza una revisión de las técnicas exactas para resolver el problema del ruteo de vehículos.

Luego, en el Capítulo 3 se realiza una revisión de las técnicas heurísticas para resolver el problema del ruteo de vehículos.

En el Capítulo 4 se realiza una revisión de las técnicas metaheurísticas para resolver el problema del ruteo de vehículos.

En el Capítulo 5 se realiza una revisión de las técnicas exactas para resolver el problema del ruteo de vehículos con múltiples depósitos.

Luego, en el Capítulo 6 se realiza una revisión de las técnicas heurísticas para resolver el problema del ruteo de vehículos con múltiples depósitos y algunas de sus variantes.

En el Capítulo 7 se presenta una revisión de las técnicas metaheurísticas para resolver específicamente el problema del ruteo de vehículos para múltiples depósitos.

Finalmente, se presenta un resumen de las ideas principales del tema y las conclusiones a las que se llegaron.

# Introducción

La distribución de productos desde los depósitos o plantas hasta los consumidores finales es un problema clave para la gestión logística, teniendo un rol importante en la misma debido a los costos que implica. [3]

El problema de ruteo de vehículos (VRP) en su forma más simple busca resolver la problemática de la distribución de productos y/o servicios, diseñando rutas que atienden la demanda de los clientes dispersos geográficamente desde un depósito utilizando una flota de vehículos. [33].

Como menciona Montoya-Torres la resolución de VRP es de suma importancia para los sistemas de distribución [3] y es por este motivo que ha sido estudiado ampliamente durante los últimos 60 años.

Distintas necesidades reales y restricciones en el campo de la logística llevaron a la definición de variantes del problema original de ruteo de vehículos generando así una variedad importante de problemas a resolver. Para todas las variantes se mantiene el objetivo principal de optimizar el uso de los recursos de transporte y la satisfacción de los clientes.

El problema de MDVRP, generaliza el problema de VRP, agregándole la posibilidad de contar con múltiples depósitos pudiendo estar asociados a los mismos tanto clientes como vehículos. La literatura clasifica esta variante como un problema NP-Hard al igual que lo es el problema original, lo cual implica que no es posible hallar una solución exacta en un tiempo de cómputo polinomial a medida que el tamaño del problema aumenta. [1][3]

Debido a la característica de NP-Hard del problema, se han desarrollado a lo largo de las últimas décadas una serie de heurísticas y metaheurísticas con el fin de obtener buenas soluciones en menor tiempo que los métodos exactos.

En este trabajo se presentan algunas de las distintas propuestas de solución tanto para VRP como para MDVRP que van desde métodos exactos hasta metaheurísticas.

# Capítulo 1

## Problema de Ruteo de Vehículos

El Problema de Ruteo de Vehículos, definido en 1958 por Dantzig y Ramset, consiste en, dado un conjunto de clientes dispersos geográficamente, un depósito y una flota de vehículos, determinar un conjunto de rutas de costo mínimo que comiencen y terminen en el depósito, para que los vehículos visiten a los clientes. [33]

Las clientes, depósitos y vehículos, puede presentar distintas características y las rutas puede requerir distintas restricciones operativas, lo cual va a dar lugar a diferentes variantes del problema.

En lo que respecta a los clientes, estos deben ser visitados exactamente una vez y podrían contar con restricciones relacionadas a un horario de atención, así como compatibilidad con los distintos vehículos del problema. [6]

El depósito es el punto de partida de las rutas y suele contener la mercadería a ser repartida por los vehículos. Lo habitual es que cada vehículo inicie y finalice la ruta en el depósito, pero puede ser de utilidad para la operativa permitirle al vehículo finalizar en otro lugar. Al igual que los clientes, el depósito podría tener restricciones de tiempo que limitaran los horarios de las rutas. [6]

Los vehículos por lo pronto pueden tener restricciones tanto para la duración de una ruta como para la cantidad de mercadería que pueden transportar. Si la flota vehicular del problema de ruteo tiene vehículos idénticos en capacidad y demás restricciones se dice que es homogénea, en caso contrario se denomina heterogénea. En la versión más simple del problema un vehículo recorre solamente una ruta, pero existen modelos que consideran que puede recorrer más de una. [6]

El objetivo más habitual del problema de ruteo de vehículos es minimizar la cantidad de vehículos que se utilizan y el costo de la ruta que recorren los mismos [9].

El problema del vendedor ambulante, TSP es un problema que ha llamado mucho la atención de matemáticos e informáticos específicamente porque es muy fácil de describir y muy difícil de resolver.

El problema puede definirse como, un vendedor desea visitar exactamente una vez cada una de las m ciudades de su lista y luego regresar a la ciudad local, ¿Cuál es la ruta menos costosa que el vendedor ambulante puede tomar?

Un desarrollo histórico completo de este y otros problemas relacionados se pueden encontrar en Hoffman y Wolfe (1985), Applegate et al. Alabama. (2006), y Cook (2011).

Dado que VRP generaliza TSP, es también un problema NP-Hard donde los métodos exactos tienden a tener un mal desempeño a medida que el tamaño de la instancia aumenta. Este motivo sumado a la importancia que tiene VRP para disminuir costos de distribución ha hecho que a lo largo de los años se definan numerosas heurísticas que buscan encontrar una buena solución. [3]

Una heurística trata de la resolución de problemas aplicando soluciones parciales, a menudo intuitivas. Se evalúan los resultados intermedios obtenidos para aproximarse poco a poco al resultado o solución final. También se aplican atajos que funcionan, aunque no se sepa exactamente por qué.

De acuerdo con ANSI/IEEE Std 100-1984, la heurística trata de métodos o algoritmos exploratorios durante la resolución de problemas, gracias a los cuales las soluciones se descubren evaluando progresos intermedios. Los métodos heurísticos se basan en la experiencia más que en la razón.

El campo de variantes del problema de ruteo de vehículos es extenso por lo que se van a mencionar únicamente las que han recibido mayor atención en los últimos años. Generalmente las variantes son producto de restricciones reales de la operativa. [9]

El problema de ruteo de vehículos con ventanas de tiempo (VRPTW por su nombre en inglés, *Vehicle Routing Problem with Time Windows*) considera que los clientes pueden tener un intervalo de tiempo en el cual deben ser atendidos por alguno de los vehículos. [4]

La variante que considera restricciones de capacidad en los vehículos (CVRP por su nombre en inglés, *Capacitated Vehicle Routing Problem*) es utilizada en la práctica ya que modela una problemática más real y no tan teórica. En esta variante los vehículos no pueden transportar una cantidad mayor de productos que un máximo preestablecido. [7]

El problema de ruteo puede incluir recolección de productos en determinados lugares para su posterior distribución (VRPPD por su nombre en inglés, *Vehicle Routing Problem with Pickup and Delivery*). En ese caso la dificultad del problema se encuentra en que la carga de los vehículos tiene fluctuaciones a lo largo de la ruta en lugar de solo incrementarse/decrementarse. [4]

En los casos donde la demanda del cliente excede la capacidad del vehículo se define una variante donde dicho cliente puede ser atendido por más de un vehículo (SDVRP, por su nombre en inglés, *Split Delivery Vehicle Routing Problem*). De acuerdo con G. Laporte y F.V. Louveaux esta relajación del problema le añade complejidad a VRP. [34]

Ciertas problemáticas de ruteo pueden presentar alguna variable aleatoria, ya sea la cantidad de clientes, su demanda, su tiempo de servicio, el tiempo de viaje, entre otros. En general, la función objetivo de esta variante (SVRP, por su nombre en inglés, *Sthocastic Vehicle Routing Problem*) está diseñada para optimizar el valor esperado de todos los posibles escenarios.

La variante con más relevancia para el presente estado del arte es la del Problema de Ruteo de Vehículos con Múltiples Depósitos, MDVRP por su nombre en inglés, *Multi-Depot Vehicle Routing Problem.* En este caso, el problema cuenta con la posibilidad de utilizar múltiples depósitos para atender la demanda de sus clientes. [5]

De acuerdo con Montoya Torres [3], la tasa de artículos por año sobre MDVRP en el período 1984-2000 era de 1.12, mientras que para el período de 2011-2014 había aumentado a 12.75. Esto se debe a que diversas aplicaciones en la vida real, como por ejemplo la distribución de gas, de productos químicos, de alimentos y de bebidas entre otras han motivado la investigación acerca de métodos para resolver el problema de MDVRP debido a la considerable importancia económica que tiene en esos casos. [22][46]

## Problema de Ruteo de Vehículos con Múltiples Depósitos

Como se menciona previamente MDVRP es una generalización de VRP donde se considera que existe más de un depósito. La solución al problema con múltiples depósitos es un conjunto de rutas que cumple con las siguientes condiciones [3]:

e) cada ruta comienza y finaliza en el mismo depósito.
f) cada cliente es atendido una sola vez por un solo vehículo.
g) los productos entregados en una ruta no exceden la capacidad del vehículo.
h) el costo de la solución es el mínimo posible.

El problema se puede modelar mediante un grafo $G = (V, A)$ completo, donde $V$ es el conjunto de vértices y $A$ es el conjunto de arcos. El conjunto $V$ se puede particionar en dos subconjuntos donde $VC = \{1 \ldots N\}$ es el conjunto de los clientes y $VD = \{N + 1 \ldots N + M\}$es el conjunto de los depósitos. Dichos conjuntos son disjuntos y la unión de estos es $V$.

Los arcos de $A$ tienen costo mayor o igual que cero (sea $(i, j) \in A, c_{ij} \geq 0$) que puede estar asociado a la distancia, tiempo de viaje o cualquier costo definido por la operativa.

Cada cliente tiene una demanda asociada $d_i$que debe ser atendida por alguno de los $K$ vehículos que tiene la flota con capacidad $P_k$ cada uno. Para

simplificar el problema se asume que la demanda de cada cliente es menor o igual que la capacidad de cada vehículo.

El tiempo de servicio del cliente $i$ es $t_i$ y la duración máxima de la ruta es $T$.

A los efectos de simplificar la formulación se asume que el costo, la distancia y el tiempo entre dos nodos tienen el mismo valor.

Montoya Torres explica en su trabajo que el modelo matemático de MDVRP requiere la definición de una variable binaria$x_{ijk}$ que indica si los clientes $i$ y $j$ se encuentran en la ruta del vehículo $k$. Del mismo modo son necesarias variables auxiliares $y_i$para eliminar la posibilidad de ciclos en las rutas. [3]

De acuerdo con lo anterior, el modelo que se define es el siguiente:

Minimizar $\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}\sum_{k=1}^{K}(c_{ij}.x_{ijk})$ (1)

s.a.

$\sum_{i=1}^{N+M}\sum_{k=1}^{K}x_{ijk} = 1\ con\ j = 1,..,N$
(2)

$\sum_{j=1}^{N+M}\sum_{k=1}^{K}x_{ijk} = 1\ con\ i = 1,..,N$
(3)

$\sum_{i=1}^{N+M}x_{ihk} - \sum_{j=1}^{N+M}x_{hjk} = 0\ con\ k = 1,..,K;\ h = 1,..,N+M$
(4)

$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}d_i.x_{ijk} \leq P_k\ con\ k = 1,..,K$
(5)

$\sum_{i=1}^{N+M}\sum_{j=1}^{N+M}(c_{ij} + t_i)x_{ijk} \leq T_k\ con\ k = 1,..,K$
(6)

$\sum_{j=N+1}^{N+M}\sum_{i=1}^{N}x_{ijk} \leq 1\ con\ k = 1,..,K$
(7)

$\sum_{i=N+1}^{N+M}\sum_{j=1}^{N}x_{ijk} \leq 1\ con\ k = 1,..,K$
(8)

$y_i - y_j + (M+N)\,x_{ijk} \leq N + M - 1\ Para\ 1 \leq i \neq j \leq N\ y\ 1 \leq k \leq K$

(9)

$x_{ijk} \in \{0,1\} \forall i,j,k$

(10)

La función objetivo es el costo total de la solución. Las restricciones (2) y (3) aseguran que cada cliente es atendido una sola vez por un único vehículo. La restricción (4) se encarga de asegurar la continuidad de la ruta mientras que (5) y (6) aseguran que se cumpla con las restricciones de demanda y tiempo.

La disponibilidad de los vehículos se asegura mediante (7) y (8) mientras que la restricción (9) impide que la solución contenga ciclos. [3]

# Capítulo 2

## Métodos exactos

El problema de ruteo de vehículos, con sus distintas variantes, ha sido estudiado extensamente en la literatura. Desde que fue presentado en 1959 por Dantzig se han desarrollado diversos métodos de solución, en su mayoría heurísticos debido a la complejidad computacional del problema. [33]

Los métodos de solución exactos son algoritmos capaces de solucionar de forma óptima los sistemas de ecuaciones lineales derivados de la formulación de los problemas de ruteo de vehículos [3].

Dantzig define en 1947 el Método Simplex, a partir del cual se han desarrollado todo tipo de técnicas como Branch and Bound, Column Generation y Branch and Cut que buscan relajar el problema facilitando la búsqueda de soluciones que realizan los algoritmos. [3]

Laporte y Norbert, en su publicación de 1987 presentan una clasificación de los métodos exactos, donde definen tres grandes grupos [35]:

- Direct Tree Search Methods
- Dynamic Programming
- Integer Linear Programming

Dentro de los métodos del tipo Integer Linear Programming se encuentra la mayor cantidad de investigación presentada para soluciones exactas a los VRP, por lo cual estos problemas se subdividen según su formulación en [35]:

- Set Partitioning Formulations

- Vehicle Flow Formulations
- Commodity Flow Formulations

### Direct Tree Search Methods

Los métodos del tipo Direct Tree Search son una construcción secuencial de rutas que se realiza a través de un árbol Branch and Bound. De acuerdo con el trabajo de Herazo Padilla, existe un algoritmo Branch and Bound aplicado a una formulación orientada al tiempo desarrollado por Baker en 1983, mientras que Christofides y Eilon en 1969 presentan otra aplicación del algoritmo Branch and Bound en el que el número de vehículos es presentado como un parámetro y se ramifica sobre los arcos. [41] [32]

### Dynamic programming

Christofides et al presentan en 1981 trabajos referidos a este tipo de métodos donde estudian relajaciones para enfoques de programación dinámica aplicadas al problema del agente viajero con ventanas de tiempo. [36]

Eilon et al en 1974 presentan una formulación de programación dinámica para el VRP con un algoritmo de relajación para reducir el espacio de búsqueda a través de criterios de dominancia. [37]

### Integer linear programming

La técnica Branch and Bound es una de las más utilizadas en la resolución de problemas de programación lineal. Un artículo de Dastghaibifard [38] llamado “A Parallel Branch and Bound Algorithm for Vehicle Routing Problem” presenta un algoritmo Branch and Bound aplicado a un problema de ruteo de vehículos con capacidad fija.

Otro modelo de programación lineal que aplica una variante del Branch and Bound llamada Branch and Cut es estudiado por Ralphs en 2003, para el problema de ruteo de vehículos. [39]

En el trabajo de Herazo Padilla se hace mención de una generalización del problema de los *m* agentes viajeros utilizando Column Generation en una formulación del tipo Set Partitioning la cual es resuelta a través del método Simplex y Branch-and-Bound. [41]

Laporte y Norbert en 1987 presentan una revisión de los métodos exactos que resuelven el problema de ruteo de vehículos. [35]

Años más tarde Laporte, presenta otra revisión como continuación del trabajo anterior incluyendo a los métodos aproximados. En 2009 continúa nuevamente con los desarrollos hechos hasta la fecha en general de todos los métodos de solución para problemas de ruteo de vehículos en su trabajo denominado “Fifty Years of Vehicle Routing”. [40]

Para los casos donde se maneja grandes cantidades de clientes (o ciudades, puntos, etc.), los métodos exactos requieren demasiado procesamiento de cómputo, por lo cual el enfoque utilizado para encarar este tipo de problemas (sobre todo con grandes cantidades de clientes) es heurístico. Dicho enfoque aplica para todas las variantes de ruteo de vehículos. Tanto para TSP, VRP, MDVRP, y cualquiera de sus variantes dado que la complejidad de los problemas no parece disminuir en ningún caso.

En el trabajo de Isaza et al. [41] se resolvió el problema de MDVRP con ventanas de tiempo y flota heterogénea para un conjunto de 20 clientes utilizando una formulación matemática de programación lineal entera mixta MILP. En dicha publicación se resuelve el problema a nivel óptimo en un tiempo de 0.5 a 3000 segundos, sirviendo estos como punto de comparación para futuras investigaciones en el campo de las heurísticas y metaheurísticas.

# Capítulo 3

## Heurísticas para VRP

En este capítulo se describen técnicas, métodos, algoritmos y estrategias que permiten ser aplicados para la búsqueda de soluciones al problema VRP.

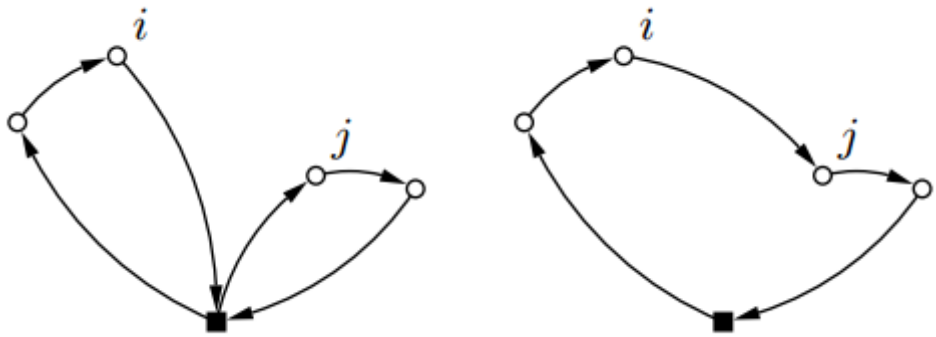
Dado que el problema MDVRP es una generalización del problema VRP las heurísticas descritas para VRP en este capítulo son de suma importancia dado que las mismas serán utilizadas y combinadas en los métodos metaheurísticos aplicados para MDVRP. [9]

A lo largo de este capítulo se describen algunas de las heurísticas clásicas que se utilizan para la resolución de VRP.

### Algoritmo de ahorros

El algoritmo de ahorros propuesto por Clarke y Wright en 1964 combina rutas diferentes de una solución, para conformar nuevas rutas que produzcan un ahorro en el total de la distancia recorrida, cumpliendo con la restricción de capacidad del vehículo que realiza la ruta [52].

A continuación, se observa una combinación entre dos rutas diferentes.




Analizando la imagen anterior, al combinar las siguientes rutas utilizando el algoritmo de ahorros y definiendo las rutas como:

- ruta 1 $(Depósito, \ldots., vértice\ i, , Depósito)$
- ruta 2 $(Depósito, vértice\ j, \ldots, Depósito)$

Se puede producir un ahorro en la distancia recorrida si la suma de distancia de los arcos $(vértice\ i, Depósito)$, $(Depósito, vértice\ j)$ del grafo, que son

eliminados en la combinación, superan a la distancia del arco $(vértice\ i, vértice\ j)$ agregado en la ruta producto de la combinación.

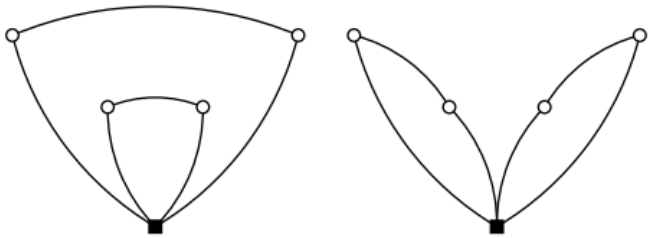
El ahorro ganado en distancia al combinar las rutas anteriores es:

$$s_{ij} = (c(i, depósito)\ + c(depósito, j)\ )\ - c(i, j)$$

Para aplicar la heurística de algoritmo de ahorros es necesario partir de una solución inicial. Mediante la aplicación de reiteradas combinaciones a la solución inicial, se busca alcanzar soluciones cada vez mejores, verificando en cada una de las mismas que se sigue cumpliendo con las restricciones de capacidad. [9]

Si el ahorro ganado, producto de la combinación de diferentes rutas es negativo, significa que la combinación entre las rutas produce un aumento en la distancia recorrida en la ruta resultante. El aumento de distancia recorrida puede observarse reflejado con una disminución de la cantidad de rutas de la solución y por lo tanto una menor cantidad de vehículos utilizados. Dado las particularidades de cada problema se debe pensar si es debido considerar combinaciones que aumenten la distancia recorrida. [9]

En muchas aplicaciones del algoritmo de ahorros se ha observado que pueden producirse rutas circulares, lo que puede ser no deseado para la resolución del problema. [9]



(a) Rutas Circulares (b) Rutas Radiales

Para evitar este problema, se puede definir la función de ahorro como se observa a continuación, con la inclusión de un parámetro $\lambda$ que penaliza la unión de rutas con clientes que se encuentran demasiado lejos.

$$s_{ij} = (c(i, depósito)\ + c(depósito, j)\ )\ - \lambda c(i, j)$$

El parámetro $\lambda$ es también utilizado para generar diferentes conjuntos de soluciones mediante la ejecución del algoritmo con diferentes valores de $\lambda$.

A continuación, se presenta el pseudocódigo para el algoritmo de ahorros tanto para la versión paralela como para la versión secuencial.

Algoritmo de Ahorros (Versión paralela)

- Paso 1 (inicialización). Para cada cliente $i$ construir la ruta (0, $i$, 0), donde 0 es el depósito.
- Paso 2 (cálculo de ahorros). Calcular $s_{ij}$ para cada par de clientes $i$ y $j$.
- Paso 3 (mejor unión). Sea $s_{i*j*} = máx\, s_{ij}$, donde el máximo se toma entre los ahorros que no han sido considerados aún. Sean $r_i^*$ y $r_j^*$las rutas que contienen a los clientes $i^*$ y $j^*$ respectivamente. Si $i^*$ es el último cliente de $r_i^*$ y $j^*$ es el primer cliente de $r_j^*$ y la combinación de $r_i^*$ y $r_j^*$ es factible, combinarlas. Eliminar $s_{i*j*}$ de futuras consideraciones. Si quedan ahorros por examinar ir a 3, si no terminar.

Algoritmo de Ahorros (Versión secuencial)

- Paso 1 (inicialización). Para cada cliente $i$ construir la ruta (0, $i$, 0), donde 0 es el depósito.
- Paso 2 (cálculo de ahorros). Calcular $s_{ij}$ para cada par de clientes $i$ y $j$.
- Paso 3 (selección). Si todas las rutas fueron consideradas, terminar. Si no, seleccionar una ruta que aún no haya sido considerada.
- Paso 4 (extensión). Sea $(0, i, \ldots, j, 0)$la ruta actual. Si no existe ningún ahorro conteniendo a $i$ (o a $j$), ir a 3. Sea $s_{k*i}$ (o $s_{jl*}$ ) el máximo ahorro conteniendo a $i$ (o a $j$). Si $k^*$(o $l^*$) es el último (o primer) cliente de su ruta y la combinación de dicha ruta con la actual es factible, realizar dicha combinación. Eliminar $s_{k*i}$ (o $s_{jl*}$) de futuras consideraciones. Ir a 4.

El algoritmo de Clarke y Wright es muy utilizado en la práctica debido a la facilidad con el que se aplica, retornando soluciones aceptables en un corto tiempo.

## Heurísticas de Inserción

Las heurísticas de inserción son métodos que permiten la creación de una solución, mediante múltiples y sucesivas inserciones de clientes en las rutas. En cada iteración se tiene una solución parcial del problema con rutas que se encuentran formadas por un subconjunto de clientes, donde se elige un nuevo cliente no visitado para ser parte de la solución del problema. [53]

La heurística de inserción puede ser un método secuencial donde solo se puede insertar clientes en la última ruta creada. Este método presenta la desventaja de que los últimos clientes a insertar que se encuentran muy dispersos pueden provocar un alto costo en las últimas rutas construidas. [68]

Para evitar este problema se puede implementar una heurística de inserción en paralelo, permitiendo insertar un cliente en cualquiera de las rutas de la solución.

A continuación, se describen las heurísticas de inserción de Mole y Jameson [53] y el método de inserción en paralelo de Christofides, Mingozzi y Toth [54].

### Heurística de inserción Mole & Jameson

La heurística de inserción de Mole & Jameson considera dos medidas para saber cuál es el próximo cliente por insertar en la ruta de la solución parcial. [53]

Una primera medida es calcular para cada cliente no visitado la mejor posición a insertar en la ruta actual, siendo la ruta actual $(v0, v1, \dots, vt, vt+1)$ donde $v0 = vt+1$.

Si $w$ es el cliente que es examinado para insertar en la ruta, su costo de inserción está dado por la siguiente definición:

- $c(vi, w) = c(vi, w) + c(w, vi+1) - \lambda c(vi, vi+1)$
  si $(v0, v1, \dots, vi, w, vi+1, \dots vt, vt+1)$ es una inserción factible con respecto a la restricción del problema.

- $c(vi, w) = \infty$ cuando la inserción no es factible.

La mejor posición a insertar el cliente $w$ en la ruta actual está dada por la siguiente definición:

$$i(w) = arg\ min\ c(i, w) \text{ donde } i \text{ varía de } 0 \dots t$$

Si el enfoque es únicamente en el costo de inserción se puede dar que clientes lejanos nunca sean tenidos en cuenta para ser insertados en la ruta actual

hasta las últimas iteraciones del algoritmo. Es necesario entonces, considerar un segundo costo denominado *urgencia* definido de la siguiente manera:

$$c_2\ (vi, w) = \ uc0w - \ c(i, w)$$

La medida de urgencia tiene el cometido de otorgar un incentivo adicional para la inserción de clientes que se encuentran lejanos en términos de distancia.

A continuación, se presenta el pseudocódigo del algoritmo de inserción de Mole & Jameson.

- Paso 1 (Creación de la ruta): Si todos los clientes se encuentran asignados a alguna ruta terminar. Si no tomar un cliente no asignado y crear la ruta (0,$w$,0).
- Paso 2 (Inserción): Sea la ruta $(v0, v1, \ldots, vt, vt + 1)$ donde $v0 = \ vt + 1,$ para cada cliente $w$ no seleccionado se calcula $i(w) = arg\ min\ c(i, w)$ donde i varía de $0 \ldots t$. Si no existen inserciones posibles, ir al paso 1. Calcular $w^* = arg\ max c_2(vi(w^*), w)$Insertar $vi(w^*)$ en $r$
- Paso 3 (Optimización): Optimizar aplicando el algoritmo 3-opt [47] sobre $r$. Ir paso 2.

Para seleccionar el cliente que iniciará la ruta pueden utilizarse diferentes alternativas, por ejemplo, el más lejano al depósito. [53]

Sucede por lo general que los últimos clientes no visitados están muy dispersos y, por ende, las últimas rutas construidas son de peor calidad que las primeras.

Para corregir esta deficiencia, se propone realizar una serie de intercambio de clientes entre las rutas una vez que el algoritmo finaliza su ejecución con el objetivo de mejorar el costo general de la solución.

### Heurística de inserción en paralelo de Christofides, Mingozzi y Toth

La heurística de Christofides, Mingozzi y Toth [54] consta de 2 fases. En la primera fase, se determina la cantidad de rutas que se van a utilizar, junto con un cliente para comenzar cada una de las rutas. En la segunda fase se crean dichas rutas y se insertan los clientes restantes en ellas.

En la primera fase del algoritmo se utiliza un algoritmo de inserción secuencial, obteniendo rutas compactas.

La fase 1 de la heurística tiene los siguientes pasos:

- Paso 1: Hacer $k = 1$ (Ruta 1)
- Paso 2: Elegir un cliente no visitado $vk$ para insertar en la ruta. Para cada cliente $w$ no visitado se calcula $\delta w, vk = c0w + \lambda cw, vk$ (costo de insertar $w$en la ruta que contiene a $vk$, siendo $vk$ el cliente inicial de la ruta)
- Paso 3: Calcular $w^* = argmin_w \delta w, vk$ sobre los clientes $w$ no visitados. Insertar $w^*$ en la ruta y aplicar 3-opt para optimizar la ruta. Ir al punto 3 si quedan clientes no visitados.
- Paso 4: Si todos los clientes pertenecen a alguna ruta terminar, si no $k = k +$ 1e ir al paso 2. (Próxima ruta)

En la segunda fase del algoritmo se crean $k$ rutas que son inicializadas con los clientes seleccionados en el paso 2 de la fase 1.

A continuación, se describen los siguientes pasos que conforman la fase 2 de la heurística de inserción.

- Paso 1 (inicialización): Crear $k$ rutas $rt = (0, vt, 0)$ para $t = 1..k$, donde $k$ es la cantidad rutas obtenidas en la fase 1. Sea $J = \{r1..rk\}$}.
- Paso 2 (asociación): Para cada cliente $w$ que no se haya seleccionado, calcular $tw = argmin_{t|rt\epsilon J} \delta w, vt$.
- Paso 3 (Medidas de urgencia): Seleccionar $rt$ perteneciente a $J$ y hacer $J$= $J\backslash\{rt\}$. Para cada cliente $w$ que $tw = t$, calcular $t' = argmin_{t|rt\epsilon J} \delta w, vt$ y $\tau w = t'w - tw$.
- Paso 4: Calcular $w^* = argmax_{t|t=tw} \tau w$. Insertar $w^*$ en la ruta $rt$ y aplicar el algoritmo 3-opt [42]. Si quedan clientes que pueden ser insertados en $rt$ ir al paso 3 de la fase 2.
- Paso 5: Si el conjunto $J$ es vacío ir al paso 2 de la fase 2. Si se han visitado todos los clientes terminar. Si no, aplicar el algoritmo nuevamente comenzando desde la fase 1.

Como se puede ver en los pasos que se describen anteriormente, para determinar el orden de inserción de los clientes en una ruta se calcula la urgencia de inserción de cada uno y se inserta el que tiene mayor urgencia. La urgencia se define en este caso como la diferencia entre el costo de insertar el cliente en la ruta que se está armando y el costo de insertarlo en la segunda mejor opción para él.

Solomon

Solomon [98] propone varias heurísticas de inserción secuencial para resolver el VRPTW.

Todas se basan en extensiones de las funciones de costo para incorporar tanto las distancias como el tiempo.

Por un lado, se define una heurística de inserción más cercana.

Si se tiene una ruta ($ds \ldots, v_i, .. dl$), se define el costo de insertar el pedido $v_j$ a continuación de $v_i$ en la ruta como:

$$\widehat{c_{ij}} = \eth_1 c_{ij} + \eth_2 T_{ij} + \eth_3 V_{ij}$$
$$T_{ij} = b_j - (b_i + s_i)$$
$$V_{ij} = I_j - (b_i + s_i + t_{ij})$$

Donde:

- los parámetros $\eth_1, \eth_2$ $y$ $\eth_3$ son no negativos y suman 1.
- $b_j$ es el tiempo de arribo al nodo $v_j$ y $b_i$ es el tiempo de arribo al nodo $v_i$.
- $s_i$ es el tiempo de servicio en el nodo $v_i$.
- $t_{ij}$ es el tiempo de viaje del nodo $v_i$ al nodo $v_j$.
- El valor de $T_{ij}$ indica la diferencia entre la hora de comienzo del servicio en $v_j$ y la del fin del servicio en $v_i$ , midiendo la cercanía de los pedidos en términos temporales.
- $V_{ij}$ mide la urgencia de realizar la inserción como la diferencia entre la hora de arribo a $v_j$ (sin incluir la espera) y la última hora a la que se podría atender dicho pedido. Valores de $V_{ij}$ cercanos a 0 indican que dentro de "pocas" iteraciones, el pedido $v_i$ no podrá ser insertado en esta ruta.

El algoritmo selecciona el pedido inicial de una ruta como el más cercano al depósito de salida (según la medida ĉ) dentro de los pedidos no insertados. En cada paso se selecciona el pedido no insertado que sea más cercano al último pedido de la ruta (considerando solamente las inserciones factibles) y se lo agrega al final. Cuando no hay más pedidos para insertar en la ruta actual, se crea una nueva o se termina si todos los pedidos han sido atendidos.

Por otro lado, se definen tres heurísticas más elaboradas que responden a los mismos criterios genéricos. Se tiene la ruta ($v_0, \ldots, v_{t+1}$) con $v_0 = ds$ y $v_{t+1} = dl$. Al igual que en la heurística de Mole & Jameson, se utilizan dos medidas para decidir el próximo pedido a insertar en una ruta: $C_1(V_i, W)$ considera la ventaja de agregar el pedido no atendido *w* entre los pedidos consecutivos $v_i$ , $v_{i+1}$ y $C_2(V_i, W)$ indica la urgencia de realizar dicha inserción. El algoritmo es esencialmente el mismo (salvo porque no se aplica ninguna post optimización a las rutas): se crea

una nueva ruta, se selecciona un pedido a insertar según $c_2$ y se inserta según $c_1$ , repitiendo el proceso hasta que no haya inserciones factibles.

Se proponen tres alternativas para seleccionar el primer pedido de la ruta: el más lejano al depósito de salida, el pedido cuya ventana de tiempo termine antes y el que minimice la suma del tiempo y distancia al depósito de salida. Para las medidas $c_1$ y $c_2$ también se realizan tres propuestas.

1. En esta propuesta se considera la ventaja de atender el pedido dentro de la ruta parcial y no en una ruta específica para él. En este caso se tiene:

$$C_1(v_i, w) = \alpha_1 C_{11}(v_i, w) + \alpha_2 C_{12}(v_i, w)$$
$$C_{11}(v_i, w) = d_{i,w} + d_{w,i+1} - \mu d_{i,i+1}$$
$$C_{12}(v_i, w) = b^w{}_{i+1} + b_{i+1}$$
$$C_2(v_i, w) = \lambda d_{ow} - C_1(v_i, w)$$

   Donde:
   - $b^w{}_{i+1}$ indica el tiempo de arribo al nodo *i*+1 cuando *w* es insertado en la ruta.
   - $b_{i+1}$ indica el tiempo de arribo al nodo *i*+1 antes de la inserción del nodo *w*.
   - $d_{i,j}$ es la distancia ente los nodos *i* y *j*.
   - los parámetros verifican: $\alpha_1$ , $\alpha_2$, $\mu$, $\lambda \geq 0$ y $\alpha_1 + \alpha 2 = 1$.

   Para calcular la mejor inserción se tienen en cuenta dos factores. Por un lado, se mide el ahorro en la distancia si se insertara *w* ente $v_i$ y $v_{i+1}$. Además, se tiene en cuenta el retardo que provoca insertar el pedido en la ruta. Finalmente, para seleccionar el próximo pedido a insertar se considera tanto el valor de $C_1$ como la distancia del pedido al depósito de salida, procurando privilegiar a los clientes para los que sería demasiado crear una ruta individual.

2. En este caso se intenta seleccionar el pedido que, si fuera insertado en la ruta, minimiza una medida de la distancia y el tiempo total de la misma. Aquí, c 1 es igual que en la parte anterior y:

$$C_2(v_i, w) = \beta_1 R_d(w) + \beta_2 R_t(w)$$

   Donde $R_d(w)$ y $R_t(w)$ son el tiempo y la distancia total de la ruta si *w* es insertado entre $v_i$ y $v_{i+1}$. En este caso, el pedido seleccionado es el que minimiza $C_2$, contrariamente a los demás casos (en que se busca maximizar la medida $C_2$). Los parámetros verifican $\beta_1 \geq 0$, $\beta_2 \geq 0$ y $\beta_1 + \beta_2 = 1$.
3. En la última propuesta se define:

$$C_1(v_i, w) = \alpha_1 C_{11}(v_i, w) + \alpha_2 C_{12}(v_i, w) + \alpha_3 C_{13}(v_i, w)$$
$$C_{13}(v_i, w) = I_w - b_w$$
$$C_2(v_i, w) = C_1(v_i, w)$$

Siendo $C_{11}$ y $C_{12}$ los de la propuesta 1. Se agrega un nuevo término a $C_1$ , que mide que tan cerca del final de su ventana de tiempo se arribaría al nodo *w* si éste fuera insertado en la ruta. Finalmente, se utiliza $C_1$ para seleccionar el próximo pedido a insertar.

### Heurística Potvin y Rousseau

Esta heurística propuesta por Potvin y Rousseau [68], es una versión paralela de la heurística de Inserción Secuencial de Solomon.

La cantidad de rutas de la solución se decide al comienzo y no se altera durante la ejecución del algoritmo. Para decidir cuantas rutas crear, se ejecuta el algoritmo de

Inserción Secuencial y se crea una nueva solución con la cantidad de rutas de la solución obtenida. Además, de cada una de las rutas dadas por el algoritmo de Inserción Secuencial, se selecciona el pedido más lejano al depósito (o el más cercano) como pedido inicial de una ruta en la nueva solución.

De esta manera, el algoritmo de Inserción Secuencial se utiliza para decidir cuantas rutas iniciales crear y también para inicializar dichas rutas.

En cada iteración se debe decidir que pedido agregar a la solución y dónde ubicarlo.

Para esto se utilizan dos medidas para decidir que pedido insertar y dónde insertarlo.

Se define la medida $C_{1_r}(V_i, W)$ como:

$$C_{1_r}(V_i, W) = \alpha_1 C_{11_r}(V_i, W) + \alpha_2 C_{12_r}(V_i, W)$$
$$C_{11_r}(V_i, W) = d_{i,w} + d_{w,i+1} - \mu d_{i,i+1}$$
$$C_{12_r}(V_i, W) = b^w{}_{i+1} - b_{i+1}$$

Donde:

- $b^w{}_{i+1}$ indica el tiempo de arribo al nodo *i*+1 cuando w es insertado en la ruta
- $b_{i+1}$ indica el tiempo de arribo al nodo i+1 antes de la inserción del nodo w
- $d_{i,j}$ es la distancia ente los nodos *i* y *j*
- los parámetros verifican: $\alpha_1, \alpha_2 \geq 0$ y $\alpha_1 + \alpha_2 = 1$.

Es decir, la suma ponderada del incremento en distancia y el retardo provocados por la inserción del pedido w entre los nodos consecutivos (pedidos y/o depósitos) $V_i$ y $V_{i+1}$ en la ruta *r*.

Notar que hay un valor de la medida para cada ruta. Luego, para cada pedido *w* se calcula la mejor alternativa $C^*{}_{1_{r^*}}(V^*{}_i, W) = minC_{1_r}(V_i, W)$, donde el mínimo se toma de todas las posibles rutas r y todos los pares de nodos consecutivos en dicha ruta.

Finalmente, para decidir cuál de todas las inserciones realizar, se calcula, para cada pedido $C_2(W) = \sum_{r \neq r^*}(C_{1_r}(V_i, W) - C^*{}_{1_{r^*}}(V^*{}_i, W))$

Esta expresión mide la urgencia de insertar *w* en la solución como la diferencia del costo de la mejor alternativa con todas las demás.

Valores elevados de $C_2(W)$ indican una gran diferencia entre el costo la mejor opción y el costo de insertarlo en otras rutas. Estos pedidos deben considerarse primero, pues la cantidad de alternativas interesantes para ellos es reducida. Por lo tanto, se calcula $W^* = \arg max_w C_2(W)$ y ese pedido es insertado en la ruta *r**, en la posición dada por $C_{1_{r^*}}$

Para dos pedidos con igual cantidad de alternativas, la medida discrimina por los costos de inserción. Los únicos parámetros del algoritmo son $\alpha 1$ y $\alpha 2$ en la definición de $C_{1_r}$ .

## Búsqueda Local

La heurística de búsqueda local no se preocupa tanto por la búsqueda de nuevas rutas, si no que se enfoca en el estado actual de la solución y los intercambios de clientes que llevan a la solución a otro estado mejor. [9]

Los algoritmos de búsqueda local presentan las siguientes ventajas:

- Son ahorrativos: usan poca memoria en donde no se guarda la secuencia de estados.
- Razonables: Ofrecen buenas soluciones, cuando el espacio de estados posibles de solución es infinito.

Los algoritmos de búsqueda local son propensos a encontrar óptimos locales que no son la mejor solución, muchas veces encontrar un óptimo global en tiempo limitado es generalmente imposible por el tamaño del espacio de soluciones. [42]

Muchas metaheurísticas utilizan la heurística de búsqueda local en la búsqueda de una mejor solución en el espacio de soluciones. [9]

Mediante el procedimiento de búsqueda local se puede obtener una mejor solución al problema partiendo de una solución inicial $s$. El algoritmo de búsqueda local define un conjunto de soluciones vecinas $N(s)$ a $s$, del cual se selecciona una solución $s^*$ de menor costo y se reemplaza $s$ por $s^*$ y se repite el procedimiento hasta que la solución $s$ no se pueda mejorar más.

En problemas de optimización de rutas, como por ejemplo en VRP el conjunto de soluciones vecinas a la solución $s$ es obtenido mediante la aplicación de reglas y procedimientos sencillos llamados movidas.

En el problema VRP los movimientos de los clientes pueden ser en una misma ruta o multi-ruta. En las movidas en ruta no se cambia el conjunto de clientes de la ruta luego de aplicar la regla, lo que varía es el orden en el que los clientes son visitados en la ruta. En las movidas multi-ruta los clientes además de poder variar en orden en que son visitados en la ruta pueden ser cambiados de ruta, luego de aplicar las reglas para obtener el conjunto de soluciones vecinas a la solución actual. [9]

Operador $\lambda$ intercambio de Lin

El operador $\lambda$ intercambio de Lin [42], utilizado en el procedimiento de búsqueda local aplicado para una ruta consiste en eliminar $\lambda$ arcos de esta y reconectar los $\lambda$ segmentos restantes de la ruta solución.

Se define $\lambda - opt$ al algoritmo de búsqueda local en el cual se utiliza el operador de $\lambda$intercambio para obtener la solución óptima. La solución de óptima obtenida por el operador de $\lambda$intercambio se llama solución $\lambda - \acute{o}ptima$.

La cantidad de $\lambda$intercambios aplicados a una ruta que atiende a $n$ clientes está dado por la ecuación:

$$2^{\lambda-1}(\lambda-1)!\binom{n+1}{\lambda}$$

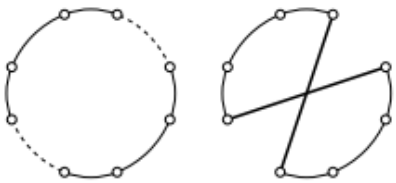
Donde$\binom{n+1}{\lambda}$ son las posibles formas de eliminar $\lambda$ arcos de la ruta y $2^{\lambda-1}(\lambda-1)!$ es la cantidad de formas que hay de reconectar la ruta.

Verificar si una solución es $\lambda - \acute{o}ptima$ puede realizarse en tiempo O ($n^{\lambda}$) en el peor caso. [42]

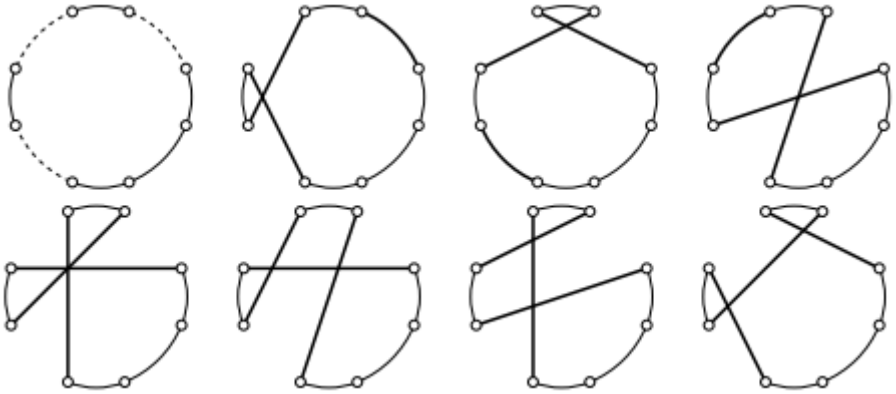
Generalmente los operadores de $\lambda$ intercambios más utilizados son “2-intercambios” y “3-intercambios”.

Asumiendo que el aplicar el operador $\lambda$ no afecta la factibilidad y los costos de los arcos son simétricos, pueden buscarse soluciones que mejoren el costo, sin tener que explorar todos los intercambios posibles.

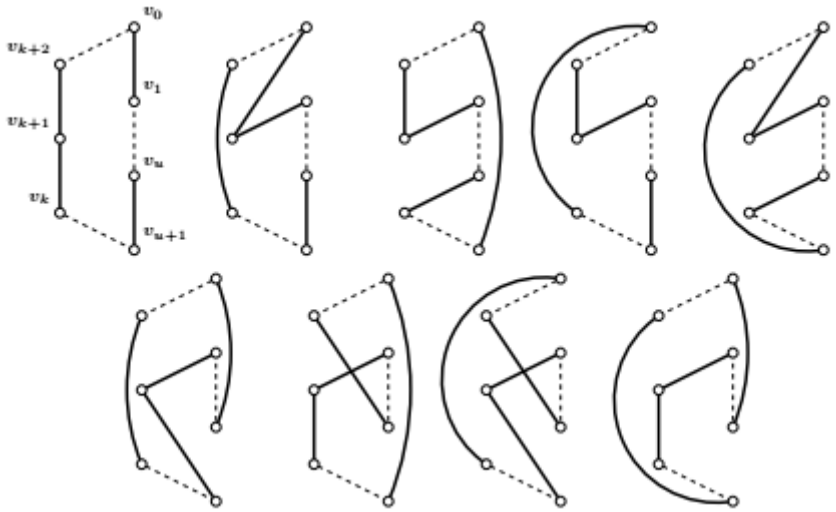
A continuación, se observa el operador 2-intercambio, 3-intercambio y 4-intercambio aplicado a un grafo que representa una ruta que atiende a $n$ clientes.



*Operador 2-intercambio*



*Operador 3-intercambio*



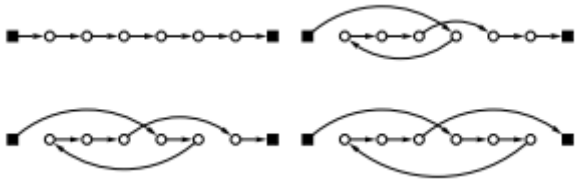

*Operador 4-intercambio*

### Operador Or-Opt

El algoritmo Or-opt [69] consiste en eliminar una secuencia de $k$ clientes consecutivos de la ruta y colocarlos en otra posición de la ruta, de modo que permanezcan consecutivos y en el mismo orden.

En la versión 3-opt se realizan las movidas con $k = 3$, luego con $k = 2$ y finalmente con $k = 1$.

En la siguiente figura se muestra una ruta y todas las posibles maneras de reubicar los 3 primeros clientes a la manera de Or-opt. Si una ruta visita $n$ clientes existen O ($n^2$) de estas movidas.

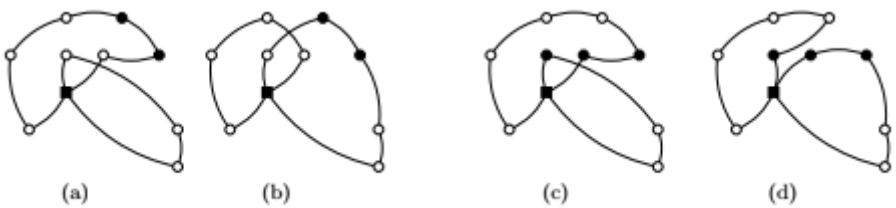


### Operadores Van Breedam

Van Breedam propone en 1995 dos operadores para intercambiar clientes entre un par de rutas. [70]

En el operador String Relocation, una secuencia de $m$ nodos es transferida de una ruta a la otra manteniendo el orden en la ruta original.

En el operador String Exchange una ruta envía una secuencia de $m$ clientes a la otra y esta última envía otra secuencia de $n$ clientes a la primera.




*String Relocation y String Exchange*

## Asignar primero y luego rutear

Un enfoque alternativo para resolver el problema del ruteo de vehículos es asignar primero lugar y luego rutear (cluster first, route second). Como es de esperar este tipo de método tiene dos fases. [9]

La primera fase consiste en generar clústeres de clientes, donde cada grupo de clientes (clúster) pertenece a una ruta de la solución final. En esta primera etapa de clusterización se debe contemplar las restricciones capacidad donde la demanda total de la ruta no puede exceder la capacidad del vehículo que atiende a la ruta de clientes perteneciente al clúster. [9]

La segunda fase consiste en crear una ruta para cada clúster, en el cual, dependiendo de la cantidad de clientes de este, se puede aplicar métodos exactos o aproximados para la creación de la ruta solución. [9]

### Heurística Sweep (barrido)

La heurística Sweep [55] es un algoritmo para generar clústeres de clientes, que consiste en trazar una semirrecta con origen en el depósito y luego ir barriendo con la misma para ir incorporando los clientes al clúster teniendo en cuenta las restricciones de capacidad. Este algoritmo es aplicable sólo a problemas de dos dimensiones, donde la ubicación del cliente está dada por las coordenadas polares $(\rho, \theta i)$ y donde se toma la ubicación del depósito como origen.

A continuación, se describe las etapas del algoritmo Sweep (barrido):

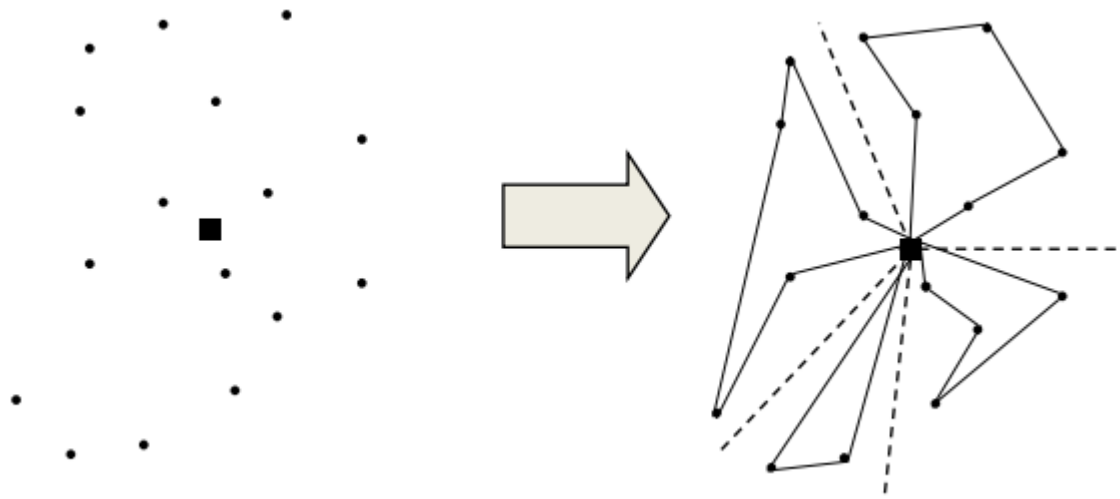
- Paso 1: Ordenar los clientes según $\theta$creciente, si dos clientes tienen el mismo valor de $\theta$ se selecciona primero el cliente que tiene menor $\rho$. Para comenzar se elige un cliente $w$ y hacer $k = 1$ y el conjunto $C_k = \{w\}$.

- Paso 2: Si todos los clientes pertenecen a algún clúster ir al paso 3. En otro caso seleccionar el siguiente cliente $w_i$. Si el cliente $w_i$ se puede agregar al conjunto $C_k$, porque cumple con la restricción de capacidad, entonces hacer que $C_k = C_k + w_i$. En caso de no satisfacer la restricción de capacidad hacer $k = k + 1$y crear el nuevo clúster $C_k = \{w_i\}$ y volver a ejecutar el paso 2.

- Paso 3: Para cada clúster $C_k con\ k = 1 \ldots k$, resolver el problema TSP (Traveling Salesman Problem) aplicando métodos exactos o métodos aproximados dependiendo de la cantidad de clientes pertenecientes a cada clúster.

Los pasos descritos del algoritmo “Sweep”, se repiten $n$ veces comenzando por un cliente diferente en cada ejecución.

En las figuras que se muestra a continuación se observa la aplicación del algoritmo “Sweep” para un conjunto de clientes dispersos en el mapa.



*Algoritmo de barrido*

### Heurística de Fisher y Jaikuman

La heurística de Fisher y Jaikuman propone en principio fijar $k$ clientes semillas $s_k$ con $k = 1,..k$ a partir de los cuales se van a formar los clústeres de clientes. El segundo paso del algoritmo es resolver el problema de asignación generalizada (GAP), donde los clientes restantes son asignados a cada clúster [57].

El GAP se define de la siguiente manera:

$$min \sum_{k=1}^{K} \sum_{i \epsilon V \backslash \{0\}} d_{ik} x_{ik} \quad (1)$$

$$s.a$$

$$\sum_{k=1}^{K} x_{ik} = 1 \qquad \forall i \, \epsilon \, V \backslash \{0\} \quad (2)$$

$$\sum_{i \epsilon \, V\{0\}} q_{ik} x_{ik} \; < Q \qquad \forall \; k = 1..k \quad (3)$$

$$x_{ik} \, \epsilon \{0,1\} \qquad \forall i \, \epsilon V \backslash \{0\}, \; \forall \; k \; = 1..k$$

La variable $x_{ik}$ determina si el cliente $i$ se encuentra en el clúster $k$. La ecuación (1) es la función objetivo del GAP donde se desea minimizar el costo. Las ecuaciones (2) y (3) son las restricciones a la que está sujeta la función objetivo. La

ecuación (2) refiere que el cliente $i$ se encuentra asignado a un solo cluster. La ecuación (3) indica que la demanda total del cluster no puede exceder la capacidad del vehículo encargado de atender la demanda del cluster.

La función $d_{ik}$ define el mejor costo de inserción del cliente $i$ en el cluster $k$

$$d_{ik} = min\{c_{oi} + c_{i,sk} + c_{sk,o}, c_{o,sk} + c_{sk,i} + c_{io}\} - \{c_{o,sk} + c_{sk,o}\}$$

Si los costos son simétricos, $d_{ik} = c_{oi} + c_{i,sk} - c_{o,sk}$

La heurística sigue los pasos que se detallan a continuación:

- Paso 1: Formar $k$ clústeres e iniciar cada clúster con un cliente $s_k \; con \; k = 1..k$.
- Paso 2: Resolver el GAP para decidir a qué clúster asignar cada cliente.
- Paso 3: Para cada clúster: resolver el TSP de los clientes del clúster.

### Heurística de Bramel y Simchi-Levi

La heurística de localización de Bramel y Simchi-Levi determina el conjunto de clientes semillas resolviendo un algoritmo un Problema de Localización de Concentradores con Capacidades (CCLP) [58].

En el CCLP si se tienen $m$ posibles ubicaciones de concentradores de capacidad $Qj(j = 1..m)$ y $n$ terminales (clientes) con demanda $d_i \; con \; i \in (1..n)$, se define que el costo de ubicar un concentrador en la ubicación $j$ es $f_j$ y el costo de conectar un terminal $i$ a un concentrador es $c_{ij}$.

Lo que se busca al tratar de resolver el CCLP, es determinar cuántos concentradores colocar y en qué terminal (cliente) centrar el concentrador, de modo que cada terminal (cliente) sea asignado a solo un concentrador.

Es importante que cada concentrador satisfaga las restricciones de capacidad del vehículo asignado al mismo y que se minimicen los costos.

La formulación del CCLP es la siguiente:

$$min \sum_{j=1}^{m} f_j y_j + \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij} x_{ij}$$

$$s.a$$

$$\sum_{i=1}^{n} d_i x_{ij} \leq Q_j y_j \;\; \forall j = 1..m \qquad (1)$$

$$\sum_{j=1}^{m} x_{ij} = 1 \;\; \forall i = 1..n \qquad (2)$$

$x_{ij}\ \epsilon\{0,1\}\ i = 1..n\ , j = 1..m$
$y_j = \epsilon\{0,1\}\ j = 1..m$

Considerando que:

- $x_{ij}$ indica si el terminal $i$ se conecta al concentrador $j$.
- $y_j$ indica si se realiza un concentrador en la ubicación $j$.
- La restricción (1) obliga a que se respete la restricción de las capacidades.
- La restricción (2) asegura que cada terminal se conecta exactamente a un concentrador.

Heurística de Localización

La heurística sigue los pasos que se detallan a continuación:

- Paso 1 (Inicialización): Seleccionar $m$ semillas $T_1 \ldots T_m$. Para cada semilla $j$ calcular $f_j$. Para cada cliente $i$ calcular $c_{ij}$.
- Paso 2: Resolver CCLP con las semillas como potenciales sitios concentradores y los clientes como terminales.
- Paso 3 (Ruteo): La solución de CCLP define los clústeres. Para cada clúster resolver un TSP.

## Algoritmo de Pétalos

Si se considera un conjunto de rutas $R$ de manera que cada ruta $r$ es una ruta factible, el problema de encontrar un subconjunto de $R$ de costo mínimo en el cual cada cliente es visitado exactamente una vez se puede expresar como un problema de "*Set Partitioning Problem*" (SPP) definido de la siguiente forma [56].

Formulación:

Minimizar: $\sum_{k\epsilon R} c_k X_k$

Sujeto a:

$\sum_{k\epsilon S} a_{ik} X_k = 1$ para todo $i$ pertenece a los vértices $V$ del grafo sin ser el vértice que representa al depósito.

$X_k \in \{0,1\} \forall k \ \in S$

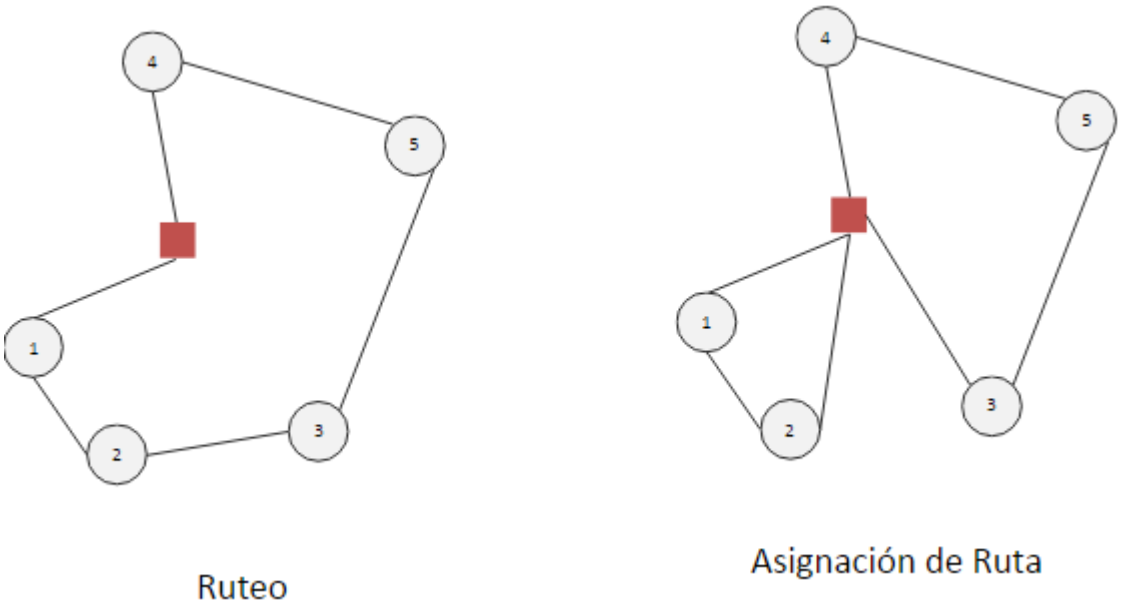
El valor de $a_{ik}$ es 1 si el cliente $i$ es visitado por la ruta $r_k$ y 0 en caso contrario. El costo de la ruta $k$ se representa por medio de $c_k$, además la variable $X_k$ indica si la ruta $k$ es considerada en la solución final o no.

En el problema "SPP" cada columna representa una ruta de $R$, en las cuales pueden aparecer una cantidad de ceros consecutivos donde se verifica la propiedad de "Columnas Circulares" y el "SPP" se puede resolver en tiempo polinomial. Esta propiedad que puede ocurrir hace que determinado ordenamiento de clientes en las rutas del problema tome la forma de pétalo en algunos casos.

## Rutear primero y asignar después

Los algoritmos de rutear primero y asignar después, presentan dos fases: en la primera fase se crea una única ruta que contiene a todos los clientes sin respetar las restricciones del problema, la segunda fase se particiona la ruta en varias sub-rutas donde cada una de las mismas, satisface las restricciones de capacidad del problema [59].

A continuación, se observan en la figura lo pasos de la heurística rutear primero y luego asignar.




*Etapas de Rutear Primero y luego asignar*

Dada $r = (0, v1, \ldots, vn, 0)$ la solución del TSP obtenida en la primera fase, se determina la mejor partición de r que respete la capacidad del vehículo.

Este problema se puede formular como el de hallar un camino mínimo en un grafo dirigido y acíclico. Para ello, se construye un grafo $G = (X, V, W)$ donde $X = \{0, v1, \ldots, vn\}$. Los arcos del $G$ conectan todo par de clientes $vi$ y $v$j con $i < j$ y tales que la demanda total de los clientes $vi + 1, \ldots, vj$ no supera la capacidad del vehículo $V = \{(vi, vj) \mid i < j, Pj\ k = i + 1\ dvk \leq Q\}$. Cada arco $(vi, vj)$se pondera con el costo de la $ruta\ (0, vi + 1, \ldots, vj, 0)$, es decir

$$w(v_i, v_j) = c_{0,vi+1} + c_{vj,0} + \sum_{k=i+1}^{j-1} c_{vk,vk+1}$$

Un arco $(vi, vj)$ representa la ruta $(0, vi + 1, \ldots, vj, 0)$. Cada camino de $0$ a $vn$ en $G$ representa una posible partición de la ruta $r$ en rutas que respetan las

restricciones de demanda. Por lo tanto, el camino de costo mínimo entre $0$ y $vn$ representa la partición de costo mínimo de la ruta original en rutas que respetan la restricción de capacidad. Como el grafo es acíclico se puede utilizar el Algoritmo de Dijkstra para hallar dicho camino.

Aunque la ruta inicial sea la solución óptima del TSP y la partición se realice de manera óptima, las rutas obtenidas no necesariamente son una solución óptima para el problema. [59]

# Capítulo 4

## Metaheurísticas para VRP

Las metaheurísticas [2] son procedimientos genéricos de exploración del espacio de soluciones para los problemas de optimización. Para encontrar la solución a problemas de optimización, las metaheurísticas recurren a técnicas para realizar una mejor exploración del espacio de soluciones consiguiendo así mejores resultados que los obtenidos por heurísticas. [44]

Las metaheurísticas que se estudian en este capítulo son algoritmos de colonias de hormigas, búsqueda tabú y algoritmos genéticos, cada uno de ellos aplicados para VRP.

Los métodos que se presentan en este capítulo son representantes de tres paradigmas diferentes. Los Algoritmos de Hormigas son procedimientos basados en agentes que mediante métodos constructivos aleatorios cooperan entre sí compartiendo información. Los algoritmos de Búsqueda Tabú son métodos de búsqueda local que aceptan empeorar las soluciones para escapar de 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.

### Algoritmo de colonias de hormigas

El Algoritmo de Hormigas se basa en el procedimiento de búsqueda de alimentos realizado por parte de colonias de hormigas. Cada hormiga cuando encuentra un camino hacia un alimento deja en el mismo una sustancia llamada feromona. La cantidad de sustancia feromona depositada, nos da una idea del largo del camino y de la calidad del alimento buscado. Cuando una hormiga no detecta ninguna sustancia feromona comienza a moverse de forma aleatoria y si en algún momento la encuentra, tomará la decisión de moverse con probabilidad alta, hacia el camino con mayor presencia de feromona, lo que producirá un aumento de sustancia en la zona. Este comportamiento indica que cuando más hormigas siguen un camino, más atractivos se vuelve para ellas.

En los algoritmos de hormigas, cada hormiga construye una solución determinada por un criterio ávido que le indica a la misma lo bueno que es tomar determinada decisión y la información histórica (feromona) que le indica que tan bueno fue haber considerado tomar dicha decisión.

[2] Glover las define en 1986 como “Métodos de solución que orquestan una interacción entre procedimientos de mejoramiento local y estrategias de más alto nivel para crear un proceso capaz de escapar de los óptimos locales y realizar una búsqueda robusta del espacio de la solución”

El primer problema al que se le aplicó el algoritmo de colonias de hormigas fue a TSP, donde la decisión que toma la hormiga en esta situación es decidir cuál es la próxima ciudad que va a visitar [9].

El denominado criterio ávido de la hormiga indica que tan bueno sería moverse al nodo $j$ estando en el nodo $i$ en un grafo que modele el problema. Al criterio que utiliza la hormiga se le denomina *visibilidad* y su media está dada por la siguiente función:

$$\eta ij = 1/cij.$$

Además, cada arco "$ij$" del grafo del problema, tiene asociada una cantidad de feromona para cada iteración "$t$" del algoritmo, dada por la siguiente función: $\tau ij(t)$.

En el modelo "Ant- Cycle" [60] se distribuyen $M$ hormigas en los $n$ nodos del grafo y en cada iteración $t$, cada hormiga selecciona un conjunto de nodos a través de una regla probabilística que combina $\eta ij\ y\ cij$para construir la solución. Por ejemplo, si la hormiga se encuentra en el nodo $i$ se puede mover al nodo $j$ con la siguiente función de probabilidad:

$$pij(t) = \frac{[\tau ij(t)]^{\alpha}[\eta ij]^{\beta}}{\sum_{h\epsilon\Omega}[\iota ih(t)]^{\alpha}[\eta ih]^{\beta}}\ \ si\ j\ \in a\ \Omega\ y\ pij(t) = 0\ si\ j\ no \in a\ \Omega.$$

Siendo $\Omega$ el conjunto de nodos aún no visitados. Los parámetros $\alpha$ y $\beta$ indican la información recolectada por la colonia y el criterio ávido de selección.

Luego de que todas las hormigas hayan creado una solución se actualiza la feromona de cada arco $(ij)$ según la siguiente función:

$$\tau ij(t+1) = \rho\tau ij(t)\ + \Delta\tau ij$$

$$donde\ \Delta\tau ij = \sum_{k=1}^{M}\Delta\tau ij^{k}\ \ y\ \Delta\tau ij^{k}\ indica\ la\ feromona\ depositada\ en\ la\ k-\acute{e}sima\ hormiga\ en\ dicho\ arco.$$

El parámetro $\rho$ se encuentra en el intervalo [0,1] y refiere a el porcentaje de feromona que permanece de una iteración a otra y surge de la necesidad de evitar la convergencia rápida del algoritmo. Al coeficiente $\rho$ se le denomina *coeficiente de evaporación*.

### Ant - System

A continuación, se describen los pasos del algoritmo “Ant - System” [9] para el problema TSP.

- Paso 1 (Inicialización): Colocar cada hormiga en cada nodo. Iniciar $t$ en cero y $\tau ij(t) = \tau 0 \; \forall \; (i,j) \; \varepsilon \; a \; E \; (conjunto \; de \; aristas)$.
- Paso 2 (Construcción): Para cada hormiga construir una solución dada por la regla probabilística $pij(t)$.
- Paso 3 (Actualización): Actualizar la feromona de cada arco según la función $\tau ij(t+1)$.
- Paso 4 (Terminación): Hacer $t = \; t+1$. Si $t \; < \; t^{max}$ colocar una hormiga en cada nodo e ir al paso 2. Si no, terminar.

El tiempo de ejecución para una iteración del algoritmo para $n$ hormigas es $\theta(n^3)$ y el orden del tiempo en que cada hormiga construye una solución es de $\theta(n^2)$ dependiendo si se consideran conjunto de posibles candidatos, se puede reducir un poco los costos en tiempos del algoritmo.

El algoritmo “Ant -System” descrito representa un esquema básico sobre el cual se han realizado algunas variantes. Algunas de las variantes desarrolladas para resolver TSP se describen a continuación.

### Ant-System con Selección Elitista

Una de las variantes realizadas al algoritmo “Ant-System” fue la incorporación de una estrategia elitista para la actualización de la feromona [61]. La idea de esta estrategia es considerar con mayor énfasis a la mejor solución obtenida en cada iteración. Si el largo de la mejor solución encontrada en la iteración $t$ es $L^*$la función de actualización de la feromona para $(i,j)$ es la siguiente:

$$\tau ij(t+1) = \rho \tau ij(t) \; + \Delta \tau ij \; + \Delta \tau^* ij$$

donde $\Delta \tau^* ij = \sigma \frac{Q}{L^*}$ si el arco $(i,j)$ es parte de la mejor solución y en otro caso $\Delta \tau^* ij$=0.

Las hormigas elitistas representadas por $\sigma$ proporcionan la idea de actualizar la feromona como si las mismas hubieran encontrado la mejor solución.

### Ant-System con Selección Elitista y Ranking

La idea del algoritmo además de considerar con mayor énfasis la mejor solución encontrada en cada iteración es generalizar la misma dando un lugar a una selección por ranking. Lo que propone el algoritmo es que antes de actualizar la feromona se ordenen las hormigas por el largo de la solución encontrada ($L1 \leq L2, \ldots \leq LM$). Donde las $\sigma - 1$ primeras hormigas elitista colocan su feromona en los arcos. La feromona aportada es ponderada por $\sigma - \mu$ donde el parámetro $\mu$ que indica la posición de la hormiga.

De esta manera la función de actualización de la feromona es la dada por $\tau ij$ pero con las variantes que $\Delta\tau ij = \sum_{\mu=1}^{\sigma-1} \Delta\tau^{\mu} ij$ y $\Delta\tau^{\mu} ij = (\sigma - \mu)\frac{Q}{L\mu}$ si la $\mu$-ésima hormiga utiliza el arco $(i, j)$ en otro caso su valor es cero [62].

### Ant-Q

En el algoritmo “Ant-Q” [63] la próxima ciudad $j$ que va a visitar la hormiga que se encuentra en el nodo $i$ está definida por la siguiente función:

$$j = argmin\ j\epsilon\Omega\ [\tau ij]^{\alpha}[\eta ij]^{\beta}$$

con probabilidad $\rho$.

Esto implica que con cierta probabilidad $\rho$ se elije la mejor ciudad a visitar basándose en la combinación de feromona y el criterio de visibilidad y si no se considera la regla probabilística del algoritmo “Ant System”.

En el algoritmo “Ant-Q” la feromona se actualiza solo para los arcos recorridos por alguna hormiga. La actualización se realiza cuando todas las hormigas construyeron su solución y esta actualización presenta dos fases una fase de actualización local y otra global. La actualización local está dada por la siguiente función:

$$\tau ij(t+1) = (1-\rho)\tau ij(t) + \rho max\ h\epsilon\Omega\ \tau jh(t),$$

donde cada hormiga actualiza la feromona de los arcos $(i, j)$recorridos.

La función de actualización global es la siguiente:

$$\tau ij(t) = (1-\rho)\tau ij(t)\ +\ \rho W/Lbest$$

donde la hormiga que presenta la mejor solución $Lbest$ actualiza la feromona en los arcos $(i, j)$.

### Ant System Híbrido para VRP

La variante “Ant System Híbrido para VRP” del algoritmo “Ant-System” propuesto por Bullnheimer, Hartl y Strauss para VRP [43], utiliza para la construcción del algoritmo la probabilidad $pij$ del algoritmo “Ant-System”, pero con la consideración de la inclusión de un nodo si no viola las restricciones del problema.

Entonces ahora el conjunto $\Omega$ contiene a los nodos no visitados que pueden ser insertados en la solución sin violar las restricciones del problema. Cuando el conjunto $\Omega$ es vacío la hormiga retorna al depósito y comienza la construcción de una nueva ruta sobre los clientes aún no visitados. El algoritmo se encuentra basado en el algoritmo “Ant-System” con selección elitista y cuando una hormiga termina la construcción de una solución se aplica el procedimiento 2-opt.

La nueva probabilidad del próximo nodo a visitar se encuentra dado por la siguiente función:

$$pij(t) = \frac{[\tau ij(t)]^{\alpha}[\eta ij]^{\beta}[\mu ij]^{\gamma}[kij]^{\lambda}}{\sum_{h\epsilon\Omega}[\tau ih(t)]^{\alpha}[\eta ih]^{\beta}[\mu ih]^{\gamma}[kih]^{\lambda}}$$

si $j$ pertenece a $\Omega$, en cambio si no pertenece $pij(t) = 0$ donde $\mu ij = ci0 + coj - cij$ es la medida de “Ahorro de Clark and Wright” y $kij = (Qi + qj)/Q$ representa la medida de utilización del vehículo si $j$ se agregara a la ruta y $Qi$ demanda acumulada en la ruta parcial. Al ser $k$ y $\mu$ medidas de la ruta parcial las probabilidades deben calcularse en cada iteración con alto costo computacional.

Los autores luego refinaron el algoritmo utilizando la función de probabilidad del algoritmo de “Ant-System” con una función que incorpora a la visibilidad, el ahorro obtenido dado por la siguiente definición:

$$\eta ij = cio + coj - gcij + f|cio - coj|.$$

Con la nueva definición de $\eta ij$, no es necesario volver a calcular en cada paso de la construcción las probabilidades, lo que nos proporciona un ahorro computacional y facilidad al aplicar el algoritmo. Además, en la refinación del algoritmo, los autores definieron el tamaño del conjunto de candidatos como $[n/4]$ vecinos más cercanos.

Otra característica del algoritmo es el uso de selección elitista con ranking y la aplicación del algoritmo “2-opt” al finalizar la construcción de una solución de la misma manera que en el algoritmo original.

## Búsqueda Tabú

El algoritmo “Búsqueda Tabú” propuesto por Glover [44], se basa en una búsqueda local donde se pueden aceptar soluciones que aumenten el costo.

El algoritmo en la iteración $t$ obtiene la solución $st$ que luego se mueve mediante una operación llamada movida a la solución $st + 1$ siendo esta última la mejor solución del subconjunto de soluciones vecinas $N(st)$. Como el costo de la solución $st + 1$puede ser mayor que el costo de $st$, la metaheurística considera mecanismos para que en las próximas iteraciones no se retorne a la solución $st$.

La solución más fácil a este problema es, por ejemplo, guardar en memoria todas las soluciones obtenidas hasta el momento por la metaheurística, con el inconveniente de un excesivo costo de almacenamiento en memoria. Otra opción es utilizar una memoria de corto plazo, donde se almacenan los atributos de las soluciones durante una cantidad $\theta$ de iteraciones. A las soluciones que no conveniente regresar, se las denomina ***soluciones tabú*** y a las operaciones que permiten volver a las soluciones tabú se las denomina ***movidas tabú***.

La metaheurística presenta un criterio, llamado criterio de aspiración donde se permite aceptar soluciones, aunque sean consideradas soluciones tabú. Cuando el el criterio de aspiración acepta una solución tabú lo hace porque la solución tabú mejora los costos obtenidos hasta el momento. El concepto de soluciones admisibles en el algoritmo hace referencia a las soluciones que no son tabú y a las que son aprobadas por el criterio de aspiración. La metaheurística Búsqueda Tabú puede tener presente un mecanismo de diversificación que permite aumentar el espacio de exploración de soluciones. Otro mecanismo que se puede aplicar es el método de intensificación, que brinda una búsqueda más exhaustiva en determinadas regiones del espacio de soluciones.

### Algoritmo Taburoute

El algoritmo fue propuesto por Gendreau, Hertz y Laporte [45] y en el mismo se aceptan soluciones no factibles durante el procedimiento de búsqueda. El algoritmo considera rutas que atienden exactamente una vez a cada cliente, pero, pueden violar las restricciones de capacidad y largo máximo de la ruta.

La función objetivo del “algoritmo Taburoute” es la siguiente:

$$c'(s) = c(s) + \alpha Q(s) + \beta L(s)$$

Donde $Q(s)$ contabiliza el exceso total de la capacidad de las rutas y $L(s)$ mide el exceso de largo total de las rutas de la solución $s$.

Los parámetros $\alpha$ y $\beta$ permiten penalizar a las soluciones que no cumplen con las restricciones de capacidad y largo, intentando de esta manera que las soluciones obtenidas pertenezcan a una región factible.

Los parámetros $\alpha$ y $\beta$ son ajustados cada diez iteraciones en el algoritmo, de manera que si en las últimas diez iteraciones no se violó la restricción de capacidad se divide el valor de $\alpha$ entre 2 y en caso contrario el valor de $\alpha$ se multiplica por 2. Para ajustar el parámetro $\beta$ se procede de forma similar verificando la restricción de largo máximo de la ruta.

Para analizar una solución, se debe examinar cada cliente $v$, eliminarlo de la ruta a la que pertenece y agregarlo a otra ruta donde se encuentren alguno de los $p$ vecinos más cercanos a $v$. Si luego del análisis se determina eliminar definitivamente a $v$ de la ruta, no se podrá volver a insertar en la misma por $\theta$iteraciones determinadas. El valor de $\theta$ se sortea entre los números del intervalo [5,10] y a las movidas que regresen a $v$ a su ruta, en las iteraciones $\theta$ se las denomina movida tabú.

En el “algoritmo Taburoute” se utilizan dos procedimientos de intensificación. Uno de los procedimientos de intensificación es aumentar el valor del parámetro de cantidad de vecinos $p$ más cercanos, cuando no se ha logrado obtener una mejor solución a la obtenida durante un determinado número de iteraciones, además se puede aplicar el operador US [47] para implementar una búsqueda local sobre la solución obtenida. El algoritmo penaliza el agrupamiento de movidas sobre los mismos clientes como método de diversificación.

El algoritmo se aplica varias veces, partiendo de un conjunto de soluciones y durante pocas iteraciones. Luego para la mejor solución obtenida, se aplica el algoritmo para un número mayor de iteraciones.

### Algoritmo Taillard

El “algoritmo Taillard” [64] se basa en particiones del conjunto de clientes y para cada partición resuelve un VRP mediante “Búsqueda Tabú” de forma independiente. Las formas de particionar a los clientes pueden ser a través de consideraciones geométricas en las cuales se tiene las coordenadas geométricas de los clientes y depósito. Otra forma en que el algoritmo puede particionar los clientes es utilizando métodos de árboles de cubrimiento formado por los caminos mínimos del depósito a cada cliente. Cada cierta iteración del algoritmo se pueden variar las particiones del conjunto de clientes.

Las movidas que son posibles de aplicar en el algoritmo son la de insertar un cliente en una ruta entre dos clientes consecutivos para minimizar el aumento de costo derivado de la inserción y al eliminar un cliente de una ruta se une el cliente anterior al cliente posterior de él. La acción inversa a cada movida se considera movida tabú por determinadas $\theta$ iteraciones. Utilizando el criterio de aspiración se puede aceptar soluciones que hayan sido penalizadas por utilizar movidas de un mismo cliente en un número repetido de veces.

En determinadas ocasiones cada ruta es optimizada mediante la resolución de TSP sobre los clientes mediante el algoritmo de Volgenant y Jonker [53].

## Algoritmos Genéticos

Los algoritmos genéticos [65] aplican ideas de la evolución natural de los seres vivos para resolver problemas de optimización y búsqueda.

La representación de las soluciones en los algoritmos genéticos se realiza mediante estructuras como vectores, matrices y árboles. Al conjunto de soluciones codificadas en alguna estructura se las denomina población y a cada solución se le llama individuo. A cada individuo se le aplica una función llamada "fitness" que cuando mayor es el valor del fitness $f(i)$aplicado al individuo $i$ mejor es la solución que el individuo representa. Cada iteración del algoritmo presenta la aplicación de operadores evolutivos que combinan y modifican a los individuos de la población para crear una nueva población. [65]

Los operadores evolutivos son selección, cruzamiento y mutación. El operador de selección se encarga de seleccionar los individuos de la población que tendrán la posibilidad de reproducirse para generar una población intermedia. En la selección los individuos seleccionados son los que presentan mayor fitness en la población. En el operador de selección proporcional, la probabilidad de seleccionar un individuo $i$ está dada por la siguiente definición: $\frac{f(i)}{\bar{f}}$ donde $\bar{f} = \frac{1}{|Población|} \sum_{i \, \epsilon \, Población} f(i)$.

Otro tipo de operador de selección es el operador por torneo en el que se eligen al azar un conjunto de individuos de la población de los cuales se selecciona el de mayor fitness para la creación de la población intermedia.

Luego de generarse la población intermedia los algoritmos evolutivos utilizan el operador de cruzamiento para combinar los individuos y generar una nueva población. Con frecuencia el operador de cruzamiento toma dos individuos de la población a los que le denomina padres y el resultado de aplicar el operador da como resultado dos individuos llamados hijos. [65]

Por ejemplo, si se codifican los individuos como vectores y los vectores padres son los siguientes $p1 = 1001001101$ y $p2 = 1100101110$ un cruzamiento posible es cortar el vector en $n$ posiciones aleatorias e intercambiar los segmentos intermedios. Para el caso de ejemplo si se cortan los vectores en los puntos 3 y 6, los vectores hijos resultados son $h1 = 1000101101$ y $h2 = 1101001110$.

El operador de cruzamiento en un punto se denomina "Single Point Crossover", el de dos puntos "Double Point Crossover". Como tercera y última fase de los algoritmos evolutivos se aplica el operador de mutación que consiste en modificar alguna propiedad del individuo, por ejemplo, si el individuo se codifica mediante un vector binario puede invertirse algún bit del vector o intercambiar

valores de dos posiciones. A cada población generada sucesivamente se la denomina generación.

A continuación, se describen algunos ejemplos de algoritmos genéticos.

### Algoritmos Genéticos para VRP

El algoritmo evolutivo “Genetic Vehicle Representation” (GVR) trabaja directamente sobre las soluciones. En el algoritmo GVR el cruzamiento entre dos individuos $p1$ y $p2$ se realiza seleccionando una subruta $r = (v1 \ldots\ldots vk)$ de $p1$ y luego se calcula cual es el cliente $wj$ más cercano a $v1$que no está en $r$. Si la ruta que contiene a $wj$ en la solución $p2$ es $r' = (0, w1 \ldots\ldots wj, wj+1, \ldots\ldots, 0)$ está última será reemplazada por la ruta $(0, w1 \ldots\ldots wj, v1 \ldots\ldots vk, wj+1, \ldots\ldots, 0)$. Está última ruta representa a uno de los hijos del cruzamiento y el otro hijo es una copia de $p1$. Por último, el algoritmo presenta cuatro operadores de mutación que son intercambiar la posición de dos clientes en la ruta, invertir el orden de una ruta, reinsertar un cliente en una ruta diferente a la que pertenece y seleccionar una sub-ruta e insertarla en otro lugar de la solución [66].

### Algoritmo de Baker y Ayechew

El algoritmo de Baker y Ayechew [54] se basa en la idea de asignar primero y rutear después. La representación de cada individuo es mediante un vector, que en la posición $i$ se encuentra el código del vehículo que es asignado al cliente asociado a la posición $i$. Los clientes cercanos en distancias se encuentran asociados a índices cercanos en el vector, para lograrlo el algoritmo utiliza una variante del algoritmo de barrido descrito en la sección de heurísticas. La función de fitness del algoritmo propuesto por Baker y Ayechew se calcula generando una ruta para cada vehículo que atienda a todos los clientes asignados a él utilizando los métodos 2-opt y 3-opt. Los operadores de cruzamiento que pueden ser utilizados en el algoritmo son el algoritmo de estado estacionario, selección por torneo y DPX. Como medida de mutación el algoritmo realiza el intercambio probabilístico entre dos posiciones del vector.

# Capítulo 5

## Métodos Exactos para MDVRP

La literatura sobre los enfoques exactos para el MDVRP es escasa. De hecho, la mayoría de los autores se han centrado en el desarrollo de métodos heurísticos y metaheurísticos para encontrar soluciones de buena calidad rápidamente [5].

Baldacci y Mingozzi [7] proponen un método basado en el procedimiento de generación sistemática de límites basado en la propuesta de Christofides et al. [36]. El algoritmo exacto utiliza tres tipos de procedimientos para generar límites inferiores, basados en relajación lineal y en relajación lagrangeana, realizadas a la formulación matemática. Los procedimientos propuestos reducen el número de variables del modelo matemático, lo que permite resolver el problema resultante de forma exacta utilizando un software comercial de programación lineal entera.

En el trabajo de Baldacci et al. [71] se describe una formulación de programación entera del problema de ruteo vehículos con visita periódicas, Periodic Vehicle Routing Problem (PVRP), que se utiliza para obtener diferentes límites inferiores. El MDVRP se puede formular como un PVRP debido a que los depósitos pueden modelarse como los múltiples períodos en el contexto de un PVRP. Por lo tanto, cualquier algoritmo que resuelve el PVRP también puede resolver el MDVRP.

Contardo y Martinelli [5] en su trabajo de 2014 presentan un algoritmo exacto considerando restricciones de capacidad y largo de ruta, el cual utiliza dos formulaciones para el MDVRP, vehicle-flow y set-partitioning, usadas en diferentes etapas del algoritmo. El límite inferior calculado con la formulación vehicle-flow se utiliza para eliminar los bordes no prometedores, reduciendo así la complejidad del subproblema de fijación de precios utilizado para resolver la formulación de set-partitioning. Los autores agregan varias clases de desigualdades válidas para fortalecer ambas formulaciones, incluida una nueva familia de desigualdades válidas utilizadas para prohibir ciclos de una duración arbitraria. Los resultados computacionales muestran que el algoritmo propuesto es competitivo frente a los métodos más avanzados para el problema de ruteo de vehículos con múltiples depósitos, y es capaz de resolver de manera óptima algunas instancias no resueltas anteriormente. Además, para las instancias que no pueden ser resueltas por el algoritmo propuesto por los autores, los límites inferiores finales resultan más fuertes que los obtenidos por métodos anteriores.

Existen otros trabajos donde se propone un modelo de programación lineal entera mixta (MILP por su nombre en inglés, Mixed Integer Lineal Programming) para modelar MDVRP teniendo en cuenta las ventanas de tiempo.

Dondo, Méndez y Cerdá en 2003 [75] proponen un marco MILP para el problema de ruteo de vehículos con múltiples depósitos, flota heterogénea y ventanas de tiempo (MDVRPHFTW). Su procedimiento proporciona un plan óptimo y también el tamaño óptimo de la flota para cada ruta. En 2007 Dondo, Méndez y Cerdá proponen un MILP para resolver una variante del problema donde se consideran entregas y recogidas simultáneas además de ventanas de tiempo. [74][5]

Isaza, Franco, y Herazo-Padilla [41] proponen un modelo basado en una formulación matemática de programación lineal entera mixta MILP para la resolución de un problema de ruteo de vehículos con múltiples depósitos, flota heterogénea y ventanas de tiempo para la minimización del costo total de la operación basada en la formulación desarrollada por Dondo & Cerdá [73] en 2007 con una modificación en las variables de carga por cada vehículo que logra una versión un poco más compacta y que entrega directamente información de balanceo de carga de la solución. La formulación propuesta no necesita restricciones de rompimiento de sub-tours, lo cual disminuye considerablemente el número de restricciones requeridas para el problema. Se observa que el modelo resuelve sin ningún problema instancias de hasta 10 clientes y hasta 20 clientes es posible obtener soluciones desviadas de la mejor solución posible de hasta 3% en tiempos de ejecución de hasta 3000 segundos.

Li, Li y Pardalos [83] resuelven el problema de ruteo de vehículos con múltiples depósitos, ventanas de tiempo y una elección flexible del depósito de llegada, es decir, MDVRPTWSDR. Ellos formulan el problema como un modelo de programación lineal entera con el costo como el tiempo total de viaje, considerando las limitaciones de la capacidad, las ventanas de tiempo, la duración de la ruta, el número de vehículos de cada depósito y el número de espacios de estacionamiento de cada depósito. En su trabajo también desarrollan un algoritmo genético híbrido para resolver el problema.

# Capítulo 6

## Heurísticas para MDVRP

Los algoritmos heurísticos han sido adoptados para resolver los problemas de MDVRP debido a que son metodologías que encuentran soluciones de forma rápida, además de ser relativamente fácil de implementar. [5]

### Heurística Multinivel

Cuando se utilizan métodos heurísticos, el objetivo principal es construir un modelo que se pueda entender fácilmente y que proporcione buenas soluciones en un tiempo razonable. [88]

En estos métodos, hay dos enfoques principales,

1. Heurística compuesta
2. Técnicas de búsqueda local

Ambos (1) y (2) comienzan con una solución inicial, preferiblemente factible, aunque no necesariamente. [88]

En la búsqueda local, la solución puede deteriorarse o convertirse en no factible. Esta flexibilidad ayuda a evitar, en muchas circunstancias, el problema de quedar atrapado en óptimos locales. Aquí los principales inconvenientes son principalmente la configuración los parámetros y la carga computacional que puede se requiere para obtener la solución. [88]

En una heurística compuesta, la solución se mejora utilizando un procedimiento de refinamiento dado, y el método se detiene cuando no hay más mejoras. Tal solución se considera entonces como la mejor solución que es un óptimo local con respecto al procedimiento utilizado. Una forma de dirigir la búsqueda fuera del óptimo local es mediante el uso de otros procedimientos de refinamiento para el cual el óptimo local actual ya no se mantiene. [88]

Un posible esquema para representar una heurística compuesta se presenta a continuación. [88]

*Level 1* : Generate an initial solution.

*Level 2* : Apply a composite heuristic of type I.

*Level 3* : Apply a composite heuristic of type II and go to level 2.

⋮

*Level p* : Apply a composite heuristic of type $p-1$ and go to level 2.

*Level* $p+1$ : Record the final solution.

En el nivel 1, la solución se puede generar al azar, utilizando la experiencia o simplemente optando por una heurística Greedy adecuada. [88]

En el nivel 2, la composición heurística de tipo I consiste en varios procedimientos de refinamiento que se implementan en secuencia. Estas los módulos deben ser rápidos en su implementación. Esto se puede lograr ya sea porque los procedimientos en sí mismos son rápidos, o porque se aplican a un conjunto pequeño y restringido. [88]

En el nivel 3, la composición heurística del tipo II consiste en procedimientos con menos refinamientos, que suelen ser más poderosos, pero requieren altos tiempos computacionales. Algunos de estos procedimientos pueden ser los mismos del nivel, pero aplicado a un vecindario más grande. [88]

En los siguientes niveles se pueden utilizar procedimientos de refinamientos cada vez más poderosos y que requieran mayor tiempo de cómputo.

### Multi-RPERT

En el trabajo de Salhi et al. [88] se propone una heurística multinivel para solucionar el problema de ruteo de vehículos con múltiples depósitos y flota heterogénea. El método en la primera parte utiliza una composición heurística denominada *borderline customers*, la cual busca crear soluciones iniciales factibles. En la segunda parte, se realiza una estrategia de búsqueda local que permite deteriorar o hacer no factibles las soluciones, con el fin de evitar óptimos locales.

El algoritmo propuesto por Salhi et al. [88] cuenta con los siguientes pasos:

Paso 0: Configurar el parámetro de borde $\varepsilon = \varepsilon_0$ y el tamaño del paso $\Delta$ .

Paso 1: Usando el valor de $\varepsilon$, determinar los clientes que están en el borde y asignar los otros clientes a sus depósitos más cercanos.

Paso 2: Para cada depósito y sus clientes asociados resolver el problema de VRP utilizando una heurística apropiada para VRP.

Paso 3: Insertar los clientes del borde en las rutas parciales con menor costo, incluidas las rutas vacías, utilizando una regla adecuada de selección / inserción.

Paso 4: Mejorar la solución para cada depósito utilizando una heurística compuesta adecuada de Tipo I.

Paso 5: Aplicar una heurística compuesta de tipo II que involucre rutas desde diferentes depósitos.

Paso 6: Repetir pasos 3 y 4 hasta que se cumpla uno de los criterios de parada se alcanza.

Considerando el esquema de niveles presentado anteriormente se puede decir que en este caso el nivel I consiste en los primeros cuatro pasos, nivel 2 y nivel 3 corresponden al paso 4 y paso 5 respectivamente.

La heurística compuesta de tipo I a la que se refieren en el paso 4 cuenta con los siguientes módulos de refinamiento:

• ***Correspondencia***: para cada ruta, se asigna el vehículo más pequeño que pueda acomodar la demanda total de los clientes atendidos en esta ruta.

• ***Asignar e Intercambiar***: reasignar clientes de una ruta a otra o cambiarlos entre las rutas.

• ***Combinar y compartir:*** combinar rutas para usar vehículos menos grandes o dividir una ruta para ser servida por vehículos más pequeños

• ***reducir***: eliminar toda la ruta asignando sus clientes a otras rutas

• ***dejar caer / relajarse:*** intentar hacer una ruta servida por el siguiente vehículo más pequeño en tamaño asignando a algunos de sus clientes en otras rutas

• ***relajar / peinar:*** combinar dos rutas si uno o dos clientes de la ruta combinada se asignan en otro lugar

• ***relajar / compartir:*** similar a relajar / peinar, excepto que es una división de una ruta en lugar de una combinación.

• ***asignar en cadena***: se ubica entre los módulos asignar y reducir. Esto se basa en eliminar una cadena de clientes, por ejemplo, de tamaño dos, e insertar a los clientes de la cadena en otras rutas.

• ***perturb-vfm***: Este procedimiento considera tres rutas al mismo tiempo. Por ejemplo, un cliente se quita de la ruta uno y se inserta en el mejor lugar de una de las otras rutas. Las restricciones de tiempo y capacidad no se imponen en esta

etapa. La ruta que atrajo a un cliente, aunque ahora se vuelve inviable, se considera para pruebas adicionales al insertar a algunos de sus clientes en una tercera ruta. Solo en esta etapa se llevan a cabo verificaciones de viabilidad para la segunda y la tercera ruta. La nueva solución es aceptada si la solución es factible y la mejora general es positiva. Este procedimiento se repite para todos los clientes y todas las rutas.

Se realizaron pruebas sobre instancias de hasta 360 clientes, con una cantidad de depósitos que variaban entre 2 y 9 con vehículos de diferentes capacidades. Los resultados obtenidos igualan a las mejores soluciones reportadas en la literatura con un tiempo de cálculo mucho menor. [88]

Caso MDVRPMPD

Basados en el trabajo de Salhi et al de 1997 [88], Nagy et al. [86] resuelven una variante del MDVRP donde se consideran entregas y recogidas simultáneas, Multi-depot VRP with mixed Pickup and Delivery (MDVRPMPD). La metodología es desarrollada en varias etapas:

i) Divide el conjunto de clientes en limítrofes y no limítrofes.
ii) Asigna los clientes no limítrofes al depósito más cercano
iii) Para cada depósito se encuentra una solución factible que se obtiene de la solución del problema de ruteo con entregas y recogidas considerando un depósito, Vehicle Routing Problem with Pickups and Deliveries (VRPPD).

Se insertan los clientes limítrofes en las rutas encontradas en el paso anterior. Luego se realizan procedimientos de mejoramiento a través de heurísticas como 2 y 3-OPT, reinserciones e intercambios

### Heurística de 5 niveles

Hadjiconstantinou y Baldacci en 1998 consideraron un problema de la vida real tomado de una empresa proveedora de mantenimiento servicios. Su problema consistía en determinar los límites de las áreas geográficas atendidas por cada depósito, la lista de clientes a visitar cada día y las rutas seguidas por vehículos. El objetivo era proporcionar un mejor servicio al cliente y minimizar el costo sujeto a restricciones en la frecuencia de visitas, tiempo de servicio requisitos, preferencias del cliente para visitar días determinados y otras restricciones de ruteo. Esta situación se resolvió utilizando la variante periódica: MDPVRP para la cual se propuso una heurística de cinco niveles: primer y segundo nivel resuelven el problema de determinar el área de servicio y días de servicio (el VRP periódico);

tercer nivel resuelve el VRP para cada día; el cuarto nivel resuelve un TSP para cada ruta, y quinto nivel busca la optimización de rutas.[80]

## Búsqueda Local

Mediante el procedimiento de búsqueda local se puede obtener una mejor solución al problema partiendo de una solución inicial $s$. El algoritmo de búsqueda local define un conjunto de soluciones vecinas $N(s)$ a $s$, del cual se selecciona una solución $s^*$ de menor costo y se reemplaza $s$ por $s^*$ y se repite el procedimiento hasta que la solución $s$ no se pueda mejorar más.

Cordeau y Maischberger [85] presentan una heurística de búsqueda local para resolver diferentes variantes del VRP, entre ellas el MDVRP. En su trabajo utilizan búsqueda tabú como método de búsqueda local. La heurística combina la Búsqueda Tabú con un simple mecanismo de perturbación para garantizar la exploración del espacio de búsqueda y además se describe la implementación paralela de la heurística.

Los experimentos computacionales muestran que la incorporación de búsqueda tabú en el marco de búsqueda local iterativa trae mejoras muy significativas sobre el uso de solamente la búsqueda tabú.

El uso de la computación paralela trae mejoras y permitió la identificación de las nuevas mejores soluciones para algunos de los problemas utilizados en las pruebas. Una gran fortaleza del enfoque es su simplicidad. El algoritmo se basa en una simple heurística de búsqueda tabú y un solo tipo de perturbación para escapar de óptimos locales. El método también es muy flexible en el sentido que puede abordar muchas variantes de problemas. Finalmente, el algoritmo es razonablemente rápido y efectivo, siendo competitivo con heurísticas conocidas como las de Christofides y Solomon para cada variante del problema en particular.

En Toro-Ocampo et al. se propone una metodología híbrida que combina las técnicas de clusterización para generar soluciones iniciales con un algoritmo de búsqueda local iterada (ILS por su nombre en inglés, iterated location search) para resolver el problema. Aunque en trabajos previos se proponen los métodos de clusterización como estrategias para generar soluciones de inicio, en este trabajo se potencia la búsqueda sobre el sistema de información obtenido después de aplicar el método de clusterización. Además, se realiza un extenso análisis sobre el desempeño de las técnicas de clusterización y su impacto en el valor de la función objetivo. El desempeño de la metodología propuesta es factible y efectivo para resolver el problema en cuanto a la calidad de las respuestas y los tiempos computacionales obtenidos, sobre las instancias de la literatura evaluadas. [90]

## Programación Dinámica

Lee et al. [82] plantean el MDVRP como un problema de programación dinámica determinística, con estados finitos, luego usan el algoritmo heurístico de la ruta más corta para resolverlo.

El trabajo de Tsirimpas et al. estudió el caso de las entregas de productos múltiples cuando la demanda de cada cliente para el producto i 2 {1, ..., K} no es una variable aleatoria pero un número constante. También asumieron que el vehículo visita a cada cliente solo una vez y diseñaron un algoritmo de programación dinámica adecuada para la determinación de la política óptima. [91]

## Agrupar Primero y Rutear Después

Giosa et al. [27] implementó la estrategia de “agrupar primero y rutear después”, aplicada a una variante del MDVRP que considera ventanas de tiempo, “MDVRP with Time Windows” (MDVRPTW). En la etapa de clusterización se asignan los clientes a los depósitos, luego cada clúster se resuelve usando el algoritmo de ahorros. El artículo realiza un estudio computacional con seis diferentes técnicas de clusterización.

El problema de MDVRPTW puede observarse como un problema de asignación y de ruteo. Es un problema de asignación de generación de clústeres, porque se debe vincular a los clientes con los depósitos de los cuales los vehículos tomarán las mercaderías para atender a los mismos. Además, es un problema de ruteo porque luego de la generación de clústeres, se deben generar las rutas de vehículos que realizarán la atención de clientes de cada uno de los clústeres.

La cantidad de clústeres generados es igual a la cantidad de depósitos del problema.

Una mala decisión al momento de la generación de clúster puede afectar el posterior ruteo generando rutas de alto costo. Por eso la generación de clústeres y la posterior generación de rutas no son independientes.

### Algoritmos de asignación para MDVRPTW

En esta sección se analizarán algoritmos de generación de clústeres de clientes y la generación de las rutas para cada clúster generado como forma de resolver el problema de Multi Depot Vehicle Routing Problem Time Windows [27].

Algoritmo de Asignación:

Un pseudocódigo de general para todos los algoritmos de asignación es el siguiente:

Hasta que todos los clientes sean asignados a un depósito
Determinar el próximo mejor cliente a ser asignado y a cuál deposito,
tomando en cuenta la demanda de los clientes, la capacidad de los
depósitos y las ventanas de tiempo de los clientes y depósitos.

Los próximos algoritmos de asignación que se describen a continuación presentan diferentes tipos de medidas para asignar un vehículo a un depósito. Estas medidas son medidas de urgencia, de asignación cíclica y asignación vía clúster. [27]

*Asignación a través de urgencia:*

La urgencia es la manera de expresar la relación de precedencia entre clientes para ser asignados a los depósitos, siendo una forma de representar la prioridad de cada cliente al momento de ser asignado a un depósito. Esta medida de urgencia nos indica que el cliente con mayor urgencia entre los demás, va a ser el primero por seleccionarse para ser asignado a un depósito. [27]

Entre los algoritmos de urgencia se encuentran, “Parallel Algorithm” y “Simplified Algorithm” que varían entre sí, sólo en la forma en que calculan la medida de urgencia y algoritmo de asignación de intercambio. [27]

*Parallel Algorithm:*

La idea de este algoritmo es que la medida de urgencia para cada cliente es calculada considerando todos los depósitos al mismo tiempo. [27]

La fórmula de urgencia para este algoritmo es la siguiente:

$$uc = (\sum_{dep\ \epsilon\ Depositos} d(c, dep)) - d\ (c, dep')$$

Donde $d(c, dep)$ representa a la distancia entre el cliente y un depósito del conjunto de Depósitos.

La función $d(c, dep')$ representa la distancia entre el cliente y su depósito más cercano.

Está heurística compara los costos de asignar un determinado cliente a su depósito más cercano, con el costo de asignar el cliente a los otros depósitos. El cliente donde la función $uc$sea máxima, indica que es el mismo presenta la mayor prioridad para ser asignado a un depósito. [27]

El costo del cómputo de este algoritmo en el peor caso es de orden $\theta(3CD + CD^2 + C^2D)$, siendo $C\ y\ D$ la cantidad total de clientes y depósitos del problema respectivamente. [27]

*Simplified Algorithm:*

Este algoritmo de asignación es similar al "Parallel Algorithm", variando solamente en la forma de calcular la urgencia, la cual se encuentra definida por la siguiente ecuación [27]:

$$uc = d(c, dep'') - d(c, dep')$$

Siendo $dep''$ el segundo depósito más cercano en distancia al cliente $c$ y $dep'$ el depósito más cercano en distancia al mismo cliente $c$ .

La función $d$ hace referencia a la distancia entre el cliente y un depósito. [27]

De forma similar el cliente con mayor $uc$ va a tener la mayor prioridad para ser asignado a su depósito más cercano. [27]

El orden del cómputo de este algoritmo en el peor caso es $\theta(3CD + CD^2 + C^2D)$ [27].

*Algoritmo de asignación de intercambio:*

Está heurística calcula la urgencia dependiendo si cliente se siente atraído al depósito con mayor demanda insatisfecha. [27]

La función de urgencia del algoritmo de asignación de intercambio es la siguiente [27]:

$$uc = d(c, dep^*) - d(c, dep')$$

Siendo $dep^*$ y $dep'$el depósito con la mayor demanda insatisfecha y el depósito más cercano en distancia al cliente "$c$"respectivamente [27].

Un alto valor de la medida de urgencia significa que es más conveniente asignar el cliente $c$ al su depósito más cercano que al depósito de mayor demanda insatisfecha [27].

El costo de cómputo de este algoritmo en el peor caso es de orden:
$\theta(3CD + C^2D + D(D^2 + DC + C))$ siendo $C\ y\ D$ la cantidad de clientes y depósitos como se ha comentado en los algoritmos anteriores [27].

*Asignación cíclica:*

La heurística de asignación cíclica consiste en primera instancia en asignar a cada uno de los depósitos del problema su cliente más cercano. Luego de forma cíclica y procediendo de a un cliente a la vez, se selecciona el cliente más cercano en distancia al cliente anterior ya asignado a un depósito y se le asigna el mismo depósito. [27]

El costo de cómputo de este algoritmo en su peor caso es de orden $\theta(CD^2 + C)$ [27].

*Asignación vía clúster:*

Cada clúster se encuentra formado por un depósito y un conjunto de clientes asignado a él. Los algoritmos de esta clase construyen un clúster compacto de clientes para cada depósito. [27]

*Coeficiente de Atracción*

En este procedimiento la forma en que los clientes son asignados a un clúster es definida por coeficientes de atracción sobre los depósitos y sobre los clientes ya asignados a un clúster. [27]

En la heurística de asignación por clúster, en cada paso se selecciona el cliente que minimiza la escala de distancia a un depósito o a un cliente ya asignado, para ser vinculado a un clúster. [27]

La fórmula de la escala de distancia es la siguiente:

$$dscaled(c, c') = d(c, c') * coefc'$$

Si un cliente o depósito presenta un coeficiente de atracción menor que a otro, esa corta distancia con respecto a los demás clientes hace que se sienta atraído a ser vinculado al cluster que forman ellos. En cambio, si el coeficiente de atracción es mayor el cliente se siente rechazado. [27]

Como los clientes que no fueron asignados no presentan valor de atracción, cuando el mismo es asignado obtiene su coeficiente de atracción a través de una función que puede variar. Para calcular el coeficiente de atracción de un cliente $c$ con su cliente más cercano $c'$ se utiliza un coeficiente de degradación. [27]

La fórmula para el cálculo del coeficiente de atracción para un cliente es la siguiente [27]:

$$coefc = min(1.coefc' + (coefc' * degc'))$$

Este algoritmo presenta en costo de cómputo en el peor caso $\theta(C^3+C^2D)$ [27]

*Triple Criterio de Agrupamiento*

Este algoritmo es una adaptación del algoritmo que asigna clientes a días de la semana para la recolección de basura. Los criterios que se utilizan para incluir un cliente en un clúster son [27]:

4- Distancia promedio a los clientes del clúster.
5- Varianza de la distancia promedio a los clientes del clúster.
6- Distancia a los clientes más cercanos de cada clúster.

Si hay un cliente con una distancia promedio a su grupo más cercano suficientemente más pequeña que la distancia promedio a su segundo grupo más cercano (mejora del 10% o más) entonces puede ser asignado; el que maximiza la diferencia de las distancias promedio se asigna a su clúster más cercano. De lo contrario, se tiene en cuenta la varianza de la distancia promedio y si hay un cliente con una varianza suficientemente pequeña de la distancia promedio a sus clústeres más cercanos (40%) se puede asignar; de nuevo se asigna el que maximiza la diferencia. Finalmente, si las dos primeras medidas fallan, la decisión se toma en función de la distancia al cliente más cercano en su clúster más cercano, y ahora se asigna el cliente que minimiza esta distancia. [27]

La complejidad de todo el algoritmo es $O = 3C^2D + 3C^2D^2 + CD^2)$ siendo C la cantidad de clientes y D la cantidad de depósitos. [27]

### MDVRPTW

En el trabajo de Tansini & Viera se presentan dos grandes clases de algoritmos de clustering para utilizar en la variante de MDVRP con ventanas de tiempo: Particionamiento y Jerarquía. Los algoritmos de particionamiento que se incluyen en el trabajo son KNN, K-Means y PAM. Los algoritmos de jerarquía presentados son Aglomeration y Rock. [8]

#### *KNN*

KNN, K vecinos más cercanos por su nombre en inglés, utiliza los K vecinos más cercanos a un elemento para decidir a que clúster va a pertenecer. El elemento formará parte del clúster que cuente con mayor cantidad de elementos dentro de los K vecinos. [8]

La complejidad de todo el algoritmo es $O = CD)$ siendo C la cantidad de clientes y D la cantidad de depósitos. [8]

#### *K-Means*

Este algoritmo clasifica a cada elemento teniendo en cuenta su distancia al centroide. Esta distancia considera tanto las coordenadas geográficas como las ventanas de tiempo. [8]

El orden del algoritmo es $O = CD^2)$ siendo C la cantidad de clientes y D la cantidad de depósitos. [8]

#### *Partición Alrededor de Medoids*

Partición Alrededor de Medoids (PAM) utiliza una búsqueda ávida que puede no encontrar la solución óptima, pero es más rápida que la búsqueda exhaustiva. [8]

Los pasos a seguir son:

4. Inicialización: seleccionar *k* de los *n* puntos como medoids.
5. Asociar cada punto al medoid más cercano.
6. Mientras el costo de la configuración disminuya:
    1. Para cada medoid *m*, para cada no medoid *o*:
        1. Intercambiar *m* y *o*, recalcular el costo (suma de la distancia de los puntos a sus medoids).

2. Si el costo total de la configuración aumentó en el paso anterior, deshacer el intercambio.

En la figura que se muestra a continuación se puede ver el cambio de medoid del clúster claro lo que produce que un cliente del clúster oscuro pase a pertenecer al clúster claro. [8]


Los depósitos pueden ser elegidos como los primeros medoids, y posteriormente se puede elegir tanto depósitos como clientes, mientras que se cumplan todas las restricciones. [8]

La distancia utilizada en los algoritmos de particionamiento incorpora las ventanas de tiempo. [8]

Para dos elementos ei y ej la distancia se calcula como:

$$WSUM\ (i,j)\ =\ wxy\ ((x_i\ -\ x_j)^2 + (y_i\ -\ y_j)^2)^{1/2}\ +\ wt\ |t_i\ -\ t_j|$$

donde $x$ e $y$ son coordenadas, $t$ se calcula como $t\ =\ (e+\ l)/2$ donde e y l son el principio y fin de la ventana. $wxy$ y $wt$ son coeficientes tales que $wxy\ >0$ y $wt\ >\ 0$y $wxy\ +wt\ =1$. [8]

La complejidad del algoritmo es O($DC^3$) donde C y D son el número de clientes y depósitos respectivamente. [8]

*Agglomeration*

Este es un acercamiento ascendente: cada punto comienza en su propio clúster, y en cada paso los clústeres más cercanos se fusionan en un nuevo clúster mientras que el número de clústeres no exceda el número de depósitos y se cumplan todas las restricciones. [8]

La distancia entre clústeres se puede calcular de distintas formas:

9- Single Linkeage: la distancia entre los clústeres se calcula como la mínima distancia entre un par de elementos pertenecientes a distintos clústeres. Es sensible a elementos que se alejan mucho de la mediana. [8]
10- Complete Linkeage: la distancia entre los clústeres se calcula como la máxima distancia entre un par de elementos pertenecientes a distintos clústeres. Es sensible a elementos que se alejan mucho de la mediana. [8]
11- Average Linkeage (Unweighted Pair Groups Method Centroid): es un método que calcula la distancia utilizando los centroides de los clústeres. Tiene dificultades para encontrar clústeres compactos cuando difieren mucho en los tamaños. [8]
12- Weighted Averrage Linkeage (WPGMC, Weighted Pair-groups Method Centroid): Difiere del anterior dando peso al miembro más reciente de un clúster para eliminar de esta forma la influencia del tamaño del clúster. [8]
13- Centroid (Unweighted Pair Groups Method Average): los dos clústeres con la menor distancia promedio se unen para formar un nuevo clúster. [8]
14- Median (Weighted Pair Groups Method Average): difiere del anterior dándole peso a los miembros más recientes del clúster. [8]
15- Within Grups(WG): es similar a UPGMA, pero los clústeres se fusionan con el objetivo de minimizar la varianza interna. Esto produce clústeres más compactos. [8]
16- Ward: es similar a WG y la pertenencia a un clúster se determina calculando la suma de las desviaciones de la media al cuadrado. El criterio de fusión determina que debe producirse el menor incremento posible en el error de la suma de los cuadrados. [8]

La distancia entre dos elementos cualesquiera se calcula de la siguiente manera:

$$Angle(i,j) = Cos^{-1}\left(\frac{x_i . x_j y_i . y_j . t_i . t_j}{(x_i^2 + y_i^2 + t_i^2)^{1/2}(x_j^2 + y_j^2 + t_j^2)^{1/2}}\right)$$

donde $x$ e $y$ son coordenadas y $t$ se calcula como $t = (e + l)/2$ donde $e$ y $l$ son el comienzo y fin de la ventana de tiempo. [8]

La complejidad de estos algoritmos es $O((D+C)^2 + (D+C)^2 \log(D+C))$ donde D es el número de depósitos y C es el número de clientes. [8]

Para evaluar esta nueva heurística de asignación para la búsqueda de soluciones al problema MDVRP, debemos generar las rutas para cada clúster de clientes. [8]

Para la generación de rutas se utiliza el algoritmo de ahorros de Clarke and Wright, considerando que la flota de vehículos es ilimitada y que una ruta se considera completa cuando no se puede agregar un cliente a la misma, porque se violan las restricciones de capacidad del vehículo que la realiza. [8]

*Rock*

Este algoritmo de agrupamiento se basa en el concepto de “link” entre elementos para determinar si dos clústeres deben o no ser combinados. El link entre elementos se define como la cantidad de vecinos que tienen en común. Los clústeres con mayor cantidad de vecinos en común serán agrupados. [8]

Para determinar si dos elementos son vecinos se utiliza una función de similitud (Sim) y un valor límite (T). Dos elementos, i y j son considerados vecinos si:

$$Sim(i,j) \leq T$$

En este caso la función Sim es la siguiente:

$$Sim(i,j) = |distancia(i) - distancia(j)| \;\; para\ los\ elementos\ i,j.$$

La complejidad de este algoritmo es $O = (C+D)^2 + (D+C) + ((D+C) - 1)^{an} + (D+C)^2 \log(D+C)$ donde $an$ es el numero promedio de vecinos, D es la cantidad de depósitos y C es la cantidad de clientes. [8]

# Capítulo 7

## Metaheurísticas y algoritmos para MDVRP

Como se menciona en capítulos anteriores, MDVRP es una generalización de VRP y por este motivo es importante estudiar las heurísticas y metaheurísticas más conocidas para VRP. Sin embargo, existe una serie de trabajos a lo largo de los últimos 60 años donde se buscan soluciones específicas para MDVRP [3]. A continuación, se describen algunas de las metaheurísticas y algoritmos aplicados a MDVRP presentes en dichos trabajos. [5]

### Algoritmos Genéticos

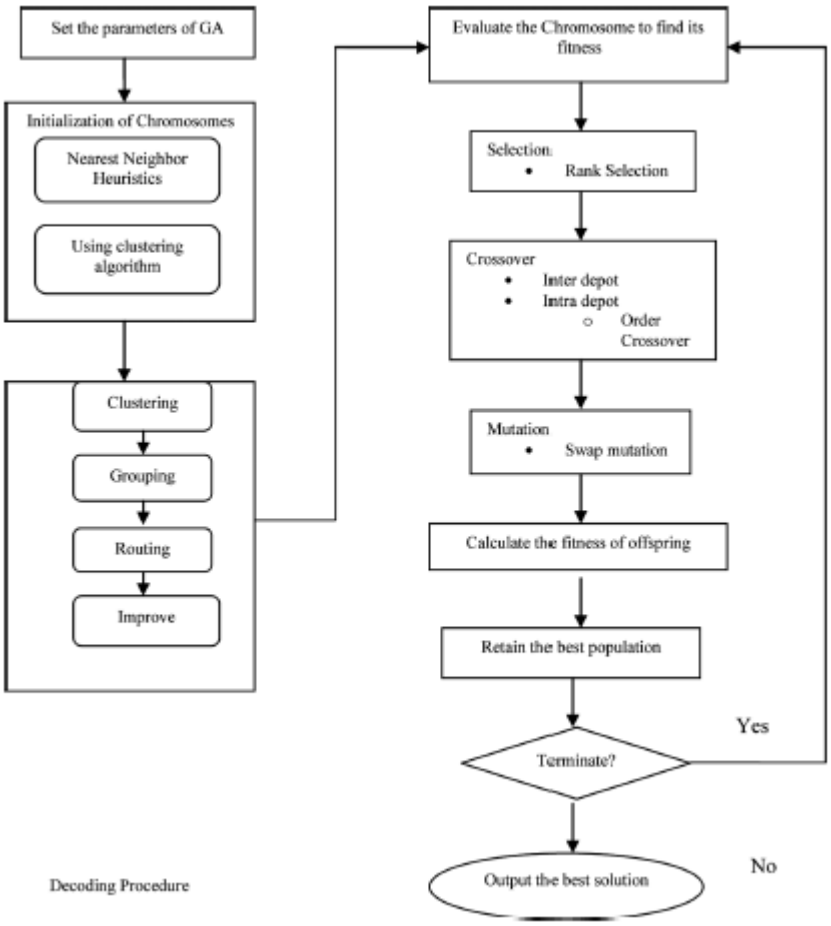
Los algoritmos genéticos pertenecen a una clase de metaheurísticas que imitan la teoría de la evolución de Darwin y la supervivencia del más apto. [67]

Mientras que los métodos clásicos y más convencionales optan por operar en una única solución, los algoritmos genéticos operan en una población entera de soluciones. Cada solución dentro de la población se denomina *cromosoma*. Estos cromosomas evolucionan a través de la iteración llamada generación. Durante cada generación, se evalúa la aptitud de los cromosomas. Los cromosomas que mejor se ajustan tienen una alta probabilidad de ser seleccionados para operaciones genéticas tales como el cruce y la mutación. Después de la generación predeterminada, GA converge hacia la solución casi óptima del problema. [67]

En general un algoritmo genético cuenta con los siguientes pasos [67]:

1- Inicializar la población de *n* soluciones de forma aleatoria.
2- Evaluar la aptitud (fitness) de cada cromosoma de la población.
3- Repetir hasta que se cumpla la condición de parada:
    a. Seleccionar un par de padres de acuerdo con su valor de aptitud.
    b. Cruzar los padres.
    c. Mutar los hijos resultantes
    d. Evaluar la aptitud de los nuevos cromosomas.
    e. Reemplazar población vieja con población nueva en función de su valor de aptitud.

En lo que respecta a la resolución de un problema de MDVRP el flujo de un algoritmo genético que lo resuelve se puede apreciar en la siguiente figura.




Tang, Yin y Man trabajaron una optimización basada en genética para MDVRP. Plantean un novedoso diseño de la estructura de los cromosomas y proponen un algoritmo genético de objetivos múltiples para hacer frente al problema, de modo que la distancia total de viaje y el tiempo total de viaje son minimizados. [89]

Gonzalez y Gonzalez en 2007 presentan una metaheurística denominada Genetic Clustering and Tabú Routing que soluciona el problema de MDVRP en dos fases. Utiliza algoritmos genéticos para la primera fase, donde genera los clústeres y en la segunda fase rutea utilizando heurísticas de barrido y búsqueda local. [78]

Bolaños propone en 2018 un algoritmo para MDVRP con flota heterogénea. Utiliza un algoritmo genético de Chu-Beasley modificado que precisa de una población inicial diversa obtenida mediante un procedimiento hibrido el cual involucra un proceso heurístico y la solución del modelo del problema de la ruta más corta. En la etapa de mejoramiento son utilizadas ocho estrategias de búsqueda local inter-rutas. [79]

Surekha y Sumanthi [50] resuelven el problema a través de un algoritmo genético. Los clientes son agrupados con base en la distancia a los depósitos más cercanos y luego son ruteados con el algoritmo de ahorros de Clarke and Wright [52]. Luego se realiza el procedimiento de optimización usando el algoritmo genético.

### Algoritmo genético de Surekha y Sumanthi

A continuación, se describe el algoritmo propuesto por Surekha y Sumanthi para MDVRP, utilizando algoritmos genéticos [50].

En el algoritmo genético de la solución propuesta para MDVRP se realiza una selección natural y de supervivencia relacionada con una función de evolución, que en los conceptos y definiciones de algoritmos genéticos se denomina “Fitness”. Además, en el algoritmo, el término de cromosoma representa una codificación de los caminos de clientes que son visitados en orden y asociados a un depósito en particular. [50]

Cada ruta formulada en un cromosoma debe respetar la restricción de que la demanda de mercaderías de los clientes de la ruta no puede exceder la capacidad de carga del vehículo que la realiza. También se denota que cada ruta en el cromosoma comienza en el depósito y termina en el mismo. [50]

Para llegar a obtener los cromosomas debemos primeramente realizar una clusterización (agrupamiento) de clientes a los depósitos que satisface la demanda de estos.

En la etapa de clusterización todos los clientes serán asignados a su depósito más cercano en términos de distancia Euclidiana. [50]

Luego de realizarse el agrupamiento, cada cliente formará parte de una asociación al depósito asignado (“link”). [50]

En la etapa de ruteo, cada cliente perteneciente a un mismo “link” es asignado a rutas utilizando el algoritmo de ahorro “Clark y Wright”. El algoritmo de “Clark y Wright” se encuentra descrito de forma detallada en el capítulo de heurísticas.

En la etapa de planificación y optimización de rutas las mismas se construyen a partir del primer cliente y luego el próximo cliente en la secuencia se elige de manera que el mismo sea el más cercano al anterior. Este proceso se repite hasta que todos los clientes de cada ruta sean seleccionados y procesados para construir una posible solución inicial al problema MDVRP. [50]

Para cada cromosoma se debe evaluar la operación de “Fitness” y calcular la aptitud promedio de “fitness”. Para MDVRP la función objetivo es minimizar el tiempo máximo de entrega entre todos los depósitos. La operación de entrega

comienza al mismo tiempo en cada depósito, pero lleva distintos tiempos atender a todos los clientes asignados. [50]

Luego de evaluar la función de evaluación, se construye una nueva población de soluciones siguiendo por ejemplo los pasos básicos de los algoritmos genéticos, selección, cruzamiento y mutación. [50]

Durante la etapa de selección se eligen dos individuos padres de la solución parcial para la reproducción utilizando el método de “Tournament selection”. [50]

El procedimiento de selección se describe de la siguiente forma:

- Paso 1: Se selecciona un conjunto $g$ de individuos de manera random para formar un conjunto torneo.
- Paso 2: Se elige un número $rs$ en el rango $[0,1]$.
- Paso 3:
    - Si $rs < a$ se selecciona el individuo más apto dentro del conjunto torneo para reproducirse
    - Si $rs >= a$ se toman cualesquiera dos cromosomas de manera aleatoria del conjunto torneo para reproducirse.
- Paso 4: Se aplica elitismo para garantizar que el mejor individuo fue seleccionado.

La técnica de cruzamiento utilizada en el trabajo de Suretha es conocida como “Best Cost Route Crossover” (BCRC). [50]

La técnica “Best Cost Route Crossover” se compone de los siguientes pasos.

- Paso 1: Elegir individuos del conjunto torneo.
- Paso 2: Seleccionar una ruta de cada individuo de manera aleatoria.
- Paso 3: Remover los clientes de la $ruta\ 1$pertenecientes a uno de los individuos
- Paso 4: Cada cliente perteneciente a la $ruta\ 1$
    - Computar el costo de inserción de la $ruta\ 1$en cada ubicación del otro individuo y computar el costo en una lista ordenada.
    - Por cada ubicación de inserción validar si es posible la inserción
    - Generar un número aleatorio “$rn$”entre $[0,1]$
    - Elegir la primera ubicación de inserción si $rn$ es menor que un límite dado
    - Si “$rn$”es mayor que el límite dado elegir el primero de la lista ordenada a pesar de la validez.
- Paso 5: Repetir el paso 4 para los clientes pertenecientes a la ruta del otro individuo del cruzamiento.

La técnica de mutación utilizada toma una cadena del cromosoma de manera aleatoria y aplica la mutación inversa a la misma, lo cual incrementa la diversidad de población, así como su calidad. [50]

Los resultados del trabajo de Suretha et al sugieren que la eficiencia de los algoritmos genéticos en términos computacionales es muy buena. Otro aspecto importante de los mismos es la velocidad con que exploran un amplio espacio de soluciones de forma paralela [50], lo cual permite encontrar buenas soluciones en un espacio corto de tiempo.

### Algoritmos Genéticos Híbridos

Los algoritmos genéticos híbridos consideran para la generación de la población inicial, técnicas de generación aleatorias y técnicas heurísticas para obtener mejores valores. Del mismo modo tienen en cuenta procesos de minimización para una solución no viable, procesos de intercambio génico, procesos de intercambio de ruta y utilizan una tasa de mutación flexible. [81]

Vidal et al. [92] proponen un algoritmo genético híbrido donde se permiten tanto soluciones factibles como no factibles, las cuales son separadas en dos grupos. Se aplica sucesivamente un número de operadores para seleccionar dos individuos padres y combinarlos. El nuevo individuo hijo, generado inicialmente, es mejorado usando procedimientos de búsqueda local y luego es incluido en la subpoblación apropiada con relación a su factibilidad. Este es un método muy eficiente y alcanza todos los óptimos de la literatura.

Jeon et al. [81] proponen un algoritmo genético híbrido para el MDVRP, el cual considera el mejoramiento de una solución inicial mediante tres heurísticas y una tasa de mutación flotante para escapar de óptimos locales.

Ho et al. [2] desarrollaron dos algoritmos genéticos híbridos (HGA1 y HGA2). La mayor diferencia entre los genéticos es la forma en que se generan las soluciones iniciales; en el HGA1 se generan de forma aleatoria. El algoritmo de ahorros de Clarke y Wright [52] y el algoritmo del vecino más cercano fueron incorporados en el proceso de inicialización del HGA2. El desempeño de ambos algoritmos es evaluado usando dos instancias generadas de forma aleatoria, donde se consideran 2 depósitos, 50 y 100 clientes. En el estudio se demuestra que utilizando algoritmo HGA2 se obtienen mejores resultados.

Wang, Sun y Ren estudiaron un algoritmo hibrido para MDVRP que tiene en cuenta pickups y deliveries. En la primera etapa utilizan un algoritmo genético hibrido con el fin de simplificar el problema. Las mejoras de los operadores de cruce parcialmente coincidentes pueden evitar la destrucción de partes buenas de genes durante el curso del cruce. El estudio adopta la estrategia de protección de genes en

su conjunto; introduce dos operadores de mutación de cambio. En la segunda etapa el shock de elite fusiona el algoritmo genético con búsqueda tabú para mejorar la velocidad de convergencia. [93]

Liu, Jiang y Geng proponen en 2012 un algoritmo genético hibrido para la variante del problema denominada MDOVRP (Problem de ruteo de vehículos con múltiples depósitos abierto) en la que los vehículos parten del depósito, pero no están obligados a regresar. En el enfoque propuesto tres algoritmos clásicos se adoptan para proporcionar buenas soluciones iniciales. Un método de división es diseñado para calcular la amplitud exacta de cada cromosoma. Varios métodos simples y potentes de búsqueda local se utilizan para mejorar la descendencia generada en el cruce. [84]

## Particle Swarm Optimization

Al igual que los algoritmos genéticos, la optimización por enjambre de partículas (PSO, por su nombre en inglés, Particle Swarm Optimization) toma conceptos del área de la biología y naturaleza para plantear una metaheurística que resuelva el problema. [22]

PSO permite optimizar un problema a partir de una población de soluciones candidatas, denotadas como "partículas", moviendo éstas por todo el espacio de búsqueda según reglas matemáticas que tienen en cuenta la posición y la velocidad de las partículas. El movimiento de cada partícula se ve influido por su mejor posición local hallada hasta el momento, así como por las mejores posiciones globales encontradas por otras partículas a medida que recorren el espacio de búsqueda. El fundamento teórico de esto es hacer que la nube de partículas converja rápidamente hacia las mejores soluciones. [22]

PSO es una metaheurística, ya que asume pocas o ninguna hipótesis sobre el problema a optimizar y puede aplicarse en grandes espacios de soluciones candidatas. Sin embargo, como toda metaheurística, PSO no garantiza la obtención de una solución óptima en todos los casos. [22]

En 2012, Geetha et al desarrollan un enfoque metaheurístico para resolver MDVRP. Utilizan una búsqueda que puede ser vista como la búsqueda del mejor elemento de un conjunto de elementos discretos. La metodología de agrupar primero y rutear después es adaptada y metaheurísticas de algoritmos genéticos (GA) y enjambre de partículas (PSO) son utilizadas para resolver el MDVRP. Del mismo modo proponen una metaheurística de PSO hibrida donde las partículas iniciales son generadas con base en el agrupamiento de K-Means y KNN. Las partículas son decodificadas en grupos y múltiples rutas son generadas dentro de

los clusters. La heurística de búsqueda local 2-opt es utilizada para optimizar las rutas obtenidas. [22]

Ezugwu et al. [77] presentan en 2018 una metaheurística basada en enjambres, inspirada en las características de las gotas de agua en un río y los cambios ambientales que resultan del curso del río. En su trabajo presentan un algoritmo de caída de agua inteligente mejorado para resolver problemas de ruteo de vehículos con múltiples depósitos. Introducen un algoritmo de Simulated Annealing propuesto como metaheurística de búsqueda local para evitar que el algoritmo inteligente de gota de agua quede atrapado en los mínimos locales y también mejore la calidad de su solución.

El rendimiento del algoritmo híbrido propuesto se evaluó mediante el uso de 33 problemas de prueba estándar, con los resultados obtenidos en comparación con las soluciones ofrecidas por cuatro técnicas bien conocidas de la literatura en cuestión. Los resultados experimentales y las pruebas estadísticas muestran que el nuevo método posee un rendimiento sobresaliente en términos de calidad de la solución y tiempo de ejecución consumido. Además, el algoritmo propuesto es adecuado para resolver problemas a gran escala. [77]

## Búsqueda Tabú

La Búsqueda Tabú es considerada una metaheurística de optimización de aplicación global, que intenta explorar el espacio de soluciones moviéndose de una solución a otra solución vecina mejor [46].

Crevier et al. [73] proponen una metaheurística que combina el método de búsqueda tabú con la programación lineal entera, en esta aproximación los vehículos pueden ser reabastecidos en depósitos intermedios a lo largo de la ruta. El problema fue pensado para aplicarlo en la vida real en Montreal, Canadá.

En el trabajo de Escobar et al. [76] se propone un algoritmo búsqueda tabú granular híbrido, el cual considera diferentes vecindarios y estrategias de diversificación para mejorar la solución inicial obtenida, mediante un procedimiento híbrido. Los resultados obtenidos son competitivos en cuanto a calidad de las respuestas y tiempos computacionales.

Cordeau, Gendreau y Laporte propusieron una metaheurística de búsqueda tabú capaz de resolver tres problemas de ruteo, PVRP, PTSP Y MDVRP. Experimentos computacionales llevados a cabo en casos tomados de la literatura indicaron que el método propuesto superaba heurísticas existentes para los tres problemas. [72]

### FIND

Renaud, Laporte y Boctor plantean una metaheurística basada en Búsqueda Tabú en 1995 denominada FIND que se presenta a continuación. [30]

En términos del problema MDVRP que se aborda, la metaheurística Búsqueda Tabú FIND se conforma de dos partes:

- Construcción de una solución inicial.
- Búsqueda Tabú.

Construcción de una solución inicial:

Para obtener la solución inicial, a cada cliente se le asigna el depósito más cercano utilizando por ejemplo la heurística “Improved Petal” de Jacques Renaud et al [31].

Algoritmo FIND

A continuación, se describe el algoritmo FIND (**F**ast Improvement **IN**tensification **D**iversification). El núcleo de algoritmo FIND se encuentra constituido por Tabú search. [30]

El algoritmo FIND para la resolución del problema MDVRP consiste en tres fases:

- Fast Improvement
- INtensification
- Diversification

En cada una de estas fases mencionadas del algoritmo FIND, se utilizan algunos de los siguientes procedimientos básicos:

- 1-route: Este procedimiento es utilizado como post optimizador de rutas. Consiste en aplicar el mecanismo de mejora $4-opt$ desarrollado por los autores de el problema “Travel Salesman Problem”. [30]
- 2-route: Determina que las mejoras a veces se producen cuando se mueven vértices entre diferentes rutas asignadas a uno o dos depósitos. Por ejemplo, si tomamos la secuencia de clientes representados por los vértices $(vih, vjh, vkh, vlh)$representando con “$h$” a cada una de la ruta siendo “$h$” igual a 1 y a 2. Con lo siguientes movimientos que se enumeran a continuación se intentará mantener la factibilidad de la solución. [30]
  - insertar $vj1$ entre los vértices $v12\ y\ vj2$
  - insertar $vj2$ entre los vértices $v11\ y\ vj1$
  - intercambiar $vj1\ y\ vj2$
  - insertar $(vj1, vk1)$ entre $vi2\ y\ vj2$
  - insertar $(vj2, vk2)$ entre $vi1\ y\ vj1$
  - intercambiar $(vj1, vk1)$y $(vj2, vk2)$

  Los movimientos enumerados corresponden a una familia de movimientos considerado como procedimiento $\lambda$ intercambio descrito en el capítulo de heurísticas del documento. [30]

- 3-route: Es utilizado cuando la capacidad de la ruta es muy cercana a la capacidad del vehículo que la realiza, donde no es posible mejorar la solución utilizando el mecanismo 2-route. 3-route propone un esquema de intercambio entre tres rutas cercanas. Considerando el vértice“$vih$”donde cada “$h$”representa a qué ruta pertenece el vértice, tomando para “$h$”los valores 1, 2 y 3. La siguiente combinación de movimientos de mejoras intentan mejorar y mantener la factibilidad de la solución. [30]

  Se considera $(vrh, vsh)$arco del grafo que modela el problema.

  Donde $vr2\ <>\ vi2\ y\ vs2\ <>\ vi2$

  - insertar $vi1$entre $vr2\ y\ vs2$
  - insertar $vi2$ entre $vr3\ y\ vs3$

  Cuando un vértice es movido de la ruta a que pertenece, y en los sucesivos

movimientos el vértice retorna a la misma, el movimiento es declarado “tabú” por $\theta$ iteraciones, donde $\theta$ es elegido de manera randomica en el intervalo entre [4..10].

A continuación, se describen las tres fases del algoritmo FIND:

Fast Improvement:

En esta fase se intenta aplicar repetidamente los siguientes pasos:

- inter-depot: Se aplica en el intercambio entre rutas de diferentes depósitos
- intra-depot: Se aplica en el intercambio entre rutas de un mismo depósito
- 3-route: Aplicado en el intercambio de vértices entre tres rutas.

Los pasos anteriores son repetidos hasta llegar al límite definido por la variable $\theta 1$, que se determina experimentalmente. En cada uno de los pasos de “Fast Improvement” cualquier movimiento que genere una mejora es implementado inmediatamente. En los movimientos implementados se aplica el procedimiento 1-route a cada ruta involucrada en el movimiento. [30]

Para poder definir la distancia entre una ruta a un depósito o entre un par de rutas, cada ruta es representada por su centro de gravedad en la fase “Fast Improvement”. [30]

En el paso inter-depot se consideran los intercambios de rutas entre cada depósito $vi$y sus $p$ depósitos más cercanos, siendo $p$ un parámetro dado. Para el paso “intra depot” se consideran todas las rutas de cada depósito. Finalmente, el paso “3-route” se aplicará a todas las rutas $h1, h2, h3$, donde se cumple que la ruta $h2$ es la más cercana a la ruta $h1$ $y$ $h3$ es la ruta más cercana a la ruta $h2$ cumpliendo que las rutas $h2$ $y$ $h1$ no son la misma. [30]

INtensification:

El cometido de esta fase es intensificar la búsqueda de las mejores rutas partiendo de la mejor solución conocida. Para lograr el objetivo se aplica el paso “intra-depot” a cada depósito del problema MDVRP, hasta cumplir $\theta 2$ iteraciones sin lograr una mejora en la solución. Siendo $\theta 2$ un parámetro dado del algoritmo FIND. [30]

Diversification:

El cometido de esta fase es ampliar la exploración el espacio de soluciones. Esta fase está compuesta de dos pasos que son repetidos un número determinado de veces. En el primer paso se realiza la búsqueda que produzca la mejor reinserción de un vértice de una ruta a otra ruta que pertenezca a otro depósito al que no se encuentra asignado el vértice. El mismo vértice no puede ser reutilizado en futuras reinserciones, por una cantidad definida de iteraciones. En el segundo paso se aplican “inter-depot” e “intra-depot” por un número $\theta 3$ dado de iteraciones

consecutivas. Siendo $\theta 3$ un parámetro dado del algoritmo FIND. [30]

A continuación, se observa una imagen de los costos computacionales del algoritmo FIND, obtenidos para 11 problemas clásicos descritos por Christofides y Eilon y 12 nuevos problemas de Chao. Además, se puede observar el costo computacional incurridos en cada fase del algoritmo. [30][32]

| | Gillett and Johnson | Chao, Golden and Wasil (CGW) | | FIND algorithm | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Fast improvement | | Intensification | | Diversification | |
| | Cost | Cost | Time* | Cost | Time† | Cost | Time† | Cost | Time† |
| 1 | 593.2 | 582.4 | 1.1 | **576.86** | 0.5 | **576.86** | 0.6 | **576.86** | 3.2 |
| 2 | 486.2 | **476.6** | 1.2 | 484.30 | 1.4 | 484.30 | 1.5 | 476.66 | 4.8 |
| 3 | 652.4 | **641.2** | 1.8 | 650.42 | 0.6 | 650.42 | 0.8 | 645.14 | 5.8 |
| 4 | 1066.7 | 1026.9 | 2.2 | **1025.09** | 1.3 | **1022.98** | 2.6 | **1016.13** | 11.4 |
| 5 | 778.9 | 756.6 | 2.4 | 784.76 | 6.2 | 775.70 | 7.1 | **754.20** | 12.8 |
| 6 | 912.2 | 883.6 | 2.1 | 886.29 | 0.6 | **878.31** | 1.7 | **876.50** | 8.4 |
| 7 | 939.5 | 898.5 | 4.8 | 905.77 | 1.2 | 905.75 | 1.6 | **897.86** | 6.8 |
| 8 | 4832.0 | 4511.6 | 24.1 | 4576.61 | 6.4 | **4500.48** | 27.5 | **4500.48** | 69.4 |
| 9 | 4219.7 | **3950.9** | 20.9 | 4122.70 | 5.4 | 4071.01 | 13.1 | 3669.31 | 41.2 |
| 10 | 3822.0 | 3815.6 | 7.2 | **3804.92** | 8.6 | **3754.52** | 13.5 | **3720.88** | 43.0 |
| 11 | 3754.1 | 3733.0 | 16.7 | **3727.70** | 7.1 | **3696.37** | 10.3 | 3670.25 | 36.4 |
| 12 | | 1327.3 | 2.8 | **1318.95** | 1.0 | **1318.95** | 1.4 | **1318.95** | 5.4 |
| 13 | | 1345.9 | 0.7 | **1318.95** | 1.1 | **1318.95** | 1.4 | **1318.95** | 4.8 |
| 14 | | 1372.5 | 1.3 | **1365.68** | 1.1 | **1365.68** | 1.1 | **1365.68** | 2.6 |
| 15 | | 2610.3 | 2.3 | **2551.45** | 4.1 | **2551.45** | 5.0 | **2551.45** | 15.5 |
| 16 | | 2605.3 | 6.1 | **2572.23** | 2.9 | **2572.23** | 3.6 | **2572.23** | 11.1 |
| 17 | | 2816.6 | 6.5 | **2731.37** | 2.3 | **2731.37** | 2.9 | **2731.37** | 5.8 |
| 18 | | 3877.4 | 8.6 | **3814.62** | 3.9 | **3814.62** | 5.2 | **3789.96** | 23.2 |
| 19 | | 3863.9 | 22.3 | **3828.61** | 4.5 | **3828.61** | 5.6 | **3827.06** | 22.0 |
| 20 | | 4272.0 | 14.6 | **4097.05** | 3.6 | **4097.05** | 4.5 | **4097.05** | 10.0 |
| 21 | | 5791.5 | 78.5 | **5678.50** | 17.8 | **5678.50** | 19.9 | **5678.50** | 48.7 |
| 22 | | 5857.4 | 132.4 | **5718.00** | 6.0 | **5718.00** | 7.7 | **5718.00** | 33.5 |
| 23 | | 6494.6 | 24.4 | **6145.58** | 5.7 | **6145.58** | 7.2 | **6145.58** | 17.3 |

*Minutes on a Sun 4/370.
†Minutes on a Sun Sparcstation 10 (cumulative time).

Tabla de costos computacionales

## Ant Colony Optimization

La metaheurística “Ant Colony Optimization” fue creada por Marco Dorigo y demuestra su eficacia por haberse aplicado a varios problemas de optimización basándose en el estudio del comportamiento social de las hormigas en sus colonias [10].

Este estudio permitió la creación de múltiples algoritmos de optimización que parten de la observación de cómo las hormigas en sus colonias se comunican entre sí. [10]

Dicha comunicación entre hormigas se realiza de forma indirecta a través de cambios que cada hormiga realiza en su entorno. Estos cambios se refieren a que cada hormiga marca el suelo con una sustancia química llamada feromona. [10]

Yu et al. presentaron un algoritmo paralelo basado en una búsqueda mediante colonia de hormigas (ACO) para la resolución del MDVRP como evolución a un proyecto previo que introducía técnicas de mutación propias de los algoritmos genéticos para facilitar la diversificación y una nueva estrategia para actualizar el nivel de feromona del recorrido. [94]

Ma y Yuan en su artículo presentan el algoritmo de colonia de hormigas para FTMDVRP (MDVRP con el tiempo de finalización más corto). Al organizar vehículos y designar depósitos para una secuencia dada de clientes, FTMDVRP se convierte en el problema de encontrar la secuencia óptima de clientes para cada ruta, entonces el algoritmo de la colonia de hormigas es adoptado para determinar dicha secuencia óptima. Utilizan el algoritmo de división mejorado para organizar vehículos y el algoritmo de flujo máximo para designar depósitos. [12]

Narasimha, Kivelevitch y Kumar trabajaron en una técnica ACO para resolver Min-Max MDVRP que se presenta a continuación. [10]

### Ant Colony para Min Max MDVRP

A continuación, se describe una metaheurística diseñada para buscar soluciones al problema MDVRP, basada en la técnica “Ant Colony Optimization”. [10]

Primeramente, se formula el problema MDVRP y luego se intenta resolver el problema de optimización enfocados para cada depósito por separado como SDVRP (“Single Depot Vehicle Routing Problem”) y finalmente utilizando técnicas de

partición en regiones se extenderá la solución para resolver el problema MDVRP. [10]

Enfoque “Single Depot Vehicle Routing Problem”:

La idea es utilizar el método Ant Colony para resolver el problema “SDVRP”, buscando un espacio de soluciones a través de hormigas artificiales que simulan a las hormigas reales en su ambiente. [10]

El objetivo se basa en considerar la distancia entre la comida y el nido de la hormiga donde la memoria corresponde al camino formado por la feromona. Los algoritmos de colonias de hormigas frecuentemente utilizan algunos parámetros que incluyen heurísticas de atractivos, reglas de actualización de feromonas y probabilidades de transición estocásticas. [10]

La heurística de atractivo $nij$ de la visita del cliente $j$ luego de haber atendido al cliente $i$ es representada de forma inversamente proporcional al costo del arco entre $i\ y\ j$, $nij\ =\ 1/dij$. [10]

Otra opción es que la función de la heurística de atractivo sea la siguiente:

$$nij = di0 + d0j - gdij + f|dio - doj|$$

Donde $f$ y $g$ son parámetros de optimización y $di0$ representa la distancia entre el cliente $i$ y el depósito. [10]

La regla de la transición probabilística “$pij$” que representa la probabilidad de elegir movernos a $j$ desde el cliente $i$.

$$pij = \frac{\tau ij^{\alpha} nij^{\beta}}{\sum_{h\ \epsilon \Omega} \tau ih^{\alpha} nih^{\beta}}$$ si $vj\ \epsilon\ \Omega$. Si no $pij$ es igual a 0.

$\tau ij$ representa a la concentración de feromonas en el camino entre el cliente $i$ e $j$.

$\alpha$ y $\beta$ asocian los perjuicios del camino de la feromona y la visibilidad del camino respectivamente. [10]

El algoritmo presenta restricción de distancia $L$, donde las hormigas artificiales construyen rutas de vehículos, eligiendo y determinando el próximo cliente a visitar en la ruta por la regla de transición probabilística descrita anteriormente.

Cuando se determina el próximo cliente a visitar en la ruta se calcula el largo de la ruta para satisfacer la restricción de distancia. [10]

Cuando se elige un cliente para pertenecer a la ruta y la ruta conteniendo el mismo no satisface la restricción de distancia $L$, se crea una nueva ruta de un nuevo vehículo, eligiendo un nuevo cliente para comenzar. [10]

Otra restricción que es considerada en el algoritmo es la capacidad de carga del vehículo que debe satisfacer la demanda de los clientes de la ruta que realiza el mismo. [10]

Enfoque “Multiple Depot Vehicle Route Problem”:

Este enfoque intenta resolver el problema MDVRP particionando el problema en múltiples enfoques “SDVRP”, donde en cada uno de ellos se utiliza el método descrito anteriormente para buscar la solución.

El particionado consiste en dividir en un número determinado de regiones convexas y equitativas que contienen a los clientes. El número de regiones es determinado por el número de depósitos presentes en el problema. [10]

La partición en regiones debe ser óptima y equitativa con respecto al dominio de depósitos del problema. [10]

A continuación, se observa una imagen donde se representan las particiones de regiones para 10 depósitos.

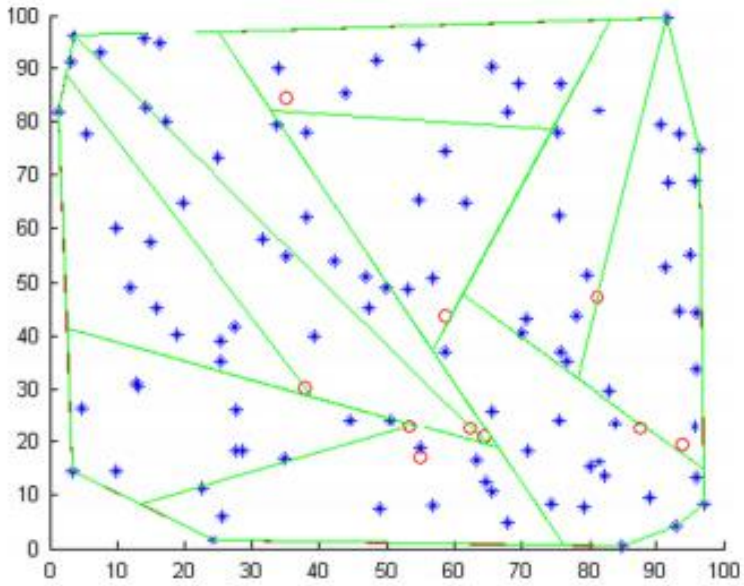



**Fig. 1.** A convex equitable partition for a case of 10 depots.

Algunas de las técnicas de partición en regiones utilizadas y mencionadas en distintos artículos en son:

- Triangulation method
- Hertel Mehlhorn
- Chazelle´s complex cubic algorithm
- Centroidal Voronoi Tessellation.

El objetivo de las particiones es obtener subregiones de igual área.

El algoritmo de partición utilizado en este enfoque descrito se basa en el método de “Carlsson`s region partitioning”. [10]

La técnica utilizada por Carlsson es considerar un conjunto de puntos que representan a los clientes en un plano 2-d, distribuidos en el mismo de forma uniforme. [10]

Este método utiliza un algoritmo de aproximación para determinar las particiones provistas por las búsquedas binarias sobre los conjuntos de puntos que representan a los depósitos y a los clientes en el plano 2-d. [10]

Cada partición resultante, presenta las siguientes propiedades:
- Todas las particiones son polígonos convexos.
- Todas las particiones contienen exactamente un depósito.
- Cada partición tiene la misma área.

A continuación, se observa el algoritmo para obtener la solución al problema “MDVRP”, en el mismo se puede observar la subrutina del método aplicado para resolver el problema “SDVRP” que fue descrito anteriormente.

Ant colony optimization algorithm to solve the min–max MDVRP.

**Ant colony optimization algorithm to solve the min–max MDVRP**

/∗**Inputs**∗/
1. *cust_points*=*get_custpoints*()/∗***set of x,y coordinates of the cities***∗/
2. *depots*=*get_depotpoints*()/∗***set of x,y coordinates of the depots***∗/

/∗**Generate convex hull**∗/
3. *poly_points*=[*cust_points; depots*]/∗***set of combination of depot and cust_points***∗/
4. *poly_vertices*=*convhull*(*poly_points*)/∗***get the vertices of the convex hull using convhull function***∗/

/∗**Region Partitioning using Carlsson algorithm**∗/
5. *subregions*=*region_partition*(*poly_vertices,depots*)/∗***get the vertices of the partitioned polygon and also get the depot points corresponding to these s vertices***∗/

/∗**Main Loop**∗/
*no_of_depots*=*size*(*depots*)
6. **For** *i*=1 to *no_of_depots* do
   *IN*=*inpolygon*()/∗***assign the cust_points that are in partition i to depot i, basically check the points if it inside or outside the polygon which i***
   *V*=*cust_points*(*IN*)/∗***V is the set of customer points that are inside region 'i'***∗/
   $v_0$=*depots*(*i*)/∗***depot(i) corresponds to the x,y coordinate of the ith depot***∗/

/∗**Generate the min–max SDVRP tour and calculate optimum tour length and optimum vehicle tour for each vehicle**∗/
[*opt_vehic_tour opt_tour_length*]=SDVRP($V$, $v_0$)
   **End For**

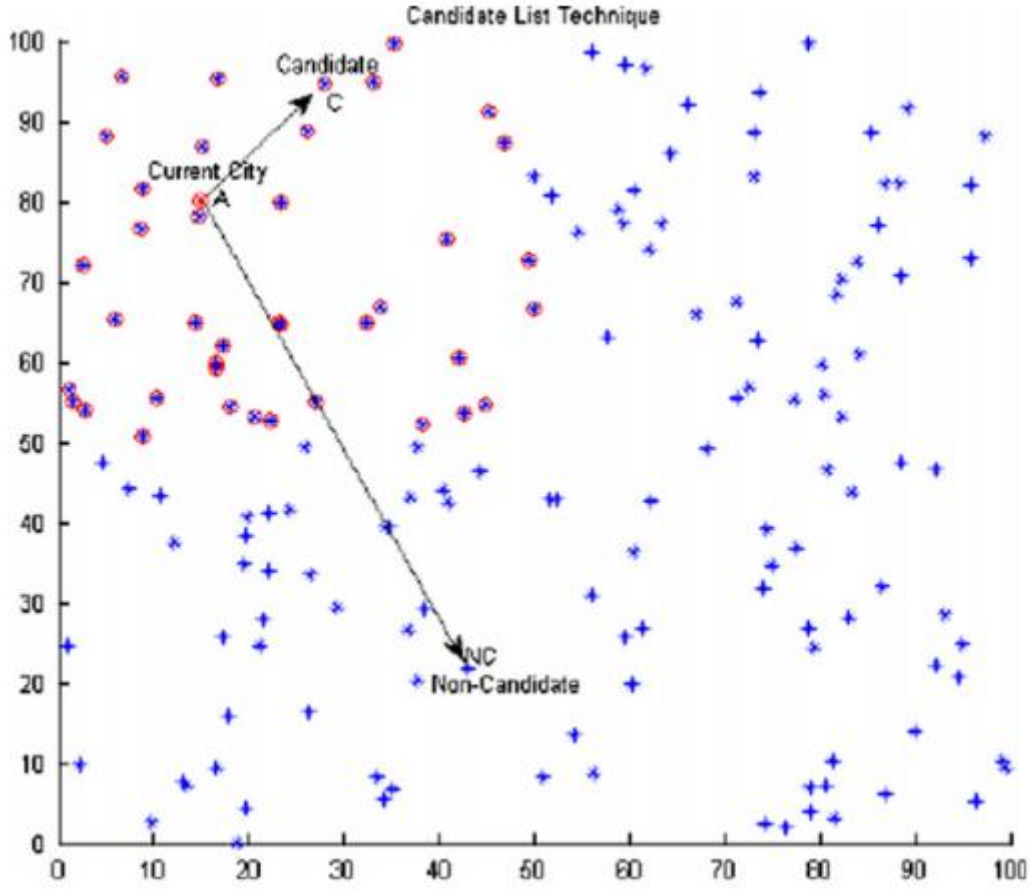
/∗**Plot the Vehicle Tours**∗/
7. **For** *i*=1 *to no_of_depots* do
   **For** *k*=1 *to no_of_vehic* do
   plot()/∗**Plot the vehicle tour**∗/
   **End For**
**End For**

Existen técnicas para mejorar el recorrido de las rutas de los vehículos, esto se refiere a manejar una lista de clientes candidatos posibles, de la cual se va a tomar el próximo cliente a visitar en la simulación del camino realizado por la hormiga. [10]

Por ejemplo, si el vehículo (hormiga artificial) se encuentra atendiendo un cliente, los demás clientes se agrupan en una lista y se la ordena de forma creciente en términos de distancia con respecto al cliente en el que se encuentra el vehículo. [10]

Solamente los clientes más próximos a la ciudad en la que se encuentra el vehículo, se seleccionan como clientes candidatos del cual se va a tomar el próximo cliente a visitar en el recorrido de la ruta del vehículo. [10]

Un ejemplo de lo descrito se observa en la siguiente imagen, en la cual se observa que el cliente $C$ es el mas optimo a elegir a ser el próximo cliente, en vez del cliente $NC$ en el recorrido, si el vehículo se encuentra ubicado en el cliente $A$ actualmente. [10]



Frecuentemente el tamaño de la lista de candidatos de próximos clientes a visitar en el recorrido es de $n/4$ siendo $n$ el total de clientes del problema MDVRP.

Considerar una la lista de candidatos de clientes presenta dos ventajas:

- El cálculo de tiempo del recorrido es menor dado que se considera un número reducido clientes para el cálculo.
- La calidad de la solución es mejorada porque la probabilidad de selección del próximo cliente no se toma sobre soluciones altamente improbables.

Otra estrategia para optimizar el recorrido puede ser la propuesta por Lin, en la cual luego de haber obtenido una solución óptima, utiliza la heurística r-opt para aplicar una serie de intercambios de a pares de cliente a la solución, con el cometido de mejorar la misma. [10]

A continuación, se observa un comparativo de los resultados obtenidos para diferentes escenarios propuestos, entre el algoritmo de colonia de hormigas descrito y el de Carlsson aplicados al problema. [10]

**Table 6**
Summary of solution obtained by the Carlsson's LP based method.

| Scenario number | No. of depot × no. of cities × no. of vehicles | Best solution | Worst solution | Mean solution | Standard deviation of solution | Avera time |
|---|---|---|---|---|---|---|
| 6 | 3 × 80 × 6 | 193.7000 | 264.5546 | 224.8572 | 23.3321 | 1.783 |
| 7 | 3 × 140 × 9 | 172.6441 | 202.5625 | 190.5611 | 7.3586 | 2.064 |
| 8 | 4 × 80 × 8 | 207.6464 | 260.8396 | 213.9139 | 11.0426 | 1.620 |
| 9 | 4 × 140 × 12 | 149.4203 | 223.2752 | 181.6720 | 20.6708 | 2.427 |
| 10 | 5 × 140 × 12 | 129.0139 | 176.6352 | 150.7001 | 12.6358 | 2.530 |
| 11 | 5 × 140 × 15 | 129.7286 | 200.5404 | 167.2335 | 20.7763 | 5.581 |

**Table 7**
Summary of solution obtained by the ant colony based method.

| Scenario number | No. of depot × no. of cities × no. of vehicles | Best solution | Worst solution | Mean solution | Standard deviation of solution | Avera time |
|---|---|---|---|---|---|---|
| 6 | 3 × 80 × 6 | 186.5168 | 199.7390 | 190.9416 | 4.2773 | 241.4 |
| 7 | 3 × 140 × 9 | 184.5539 | 190.4588 | 188.4617 | 1.4237 | 602.4 |
| 8 | 4 × 80 × 8 | 205.1906 | 205.1906 | 205.1906 | 0.0000 | 225.7 |
| 9 | 4 × 140 × 12 | 146.8195 | 150.9697 | 149.8148 | 1.1232 | 484.5 |
| 10 | 5 × 140 × 12 | 125.0439 | 125.8830 | 125.4495 | 0.2485 | 432.5 |
| 11 | 5 × 140 × 15 | 110.2564 | 200.5404 | 112.0314 | 0.5435 | 443.8 |

## Simulated Annealing

Esta técnica está basada en el proceso físico de tratamiento térmico de los metales denominado recocido, en donde un metal es llevado a altas temperaturas alcanzando

altos niveles energéticos llegando al punto de fusión y luego es enfriado gradualmente, en un proceso por fases en donde el sólido puede alcanzar el equilibrio térmico para cada fase, hasta volver de nuevo al estado sólido obteniendo un estado de energía mínimo que es definido previamente. De esta manera se puede construir el modelo de optimización comparando las posibles soluciones con los estados del sistema físico y el costo de la solución con la energía del estado, teniendo en cuenta que el proceso de recocido debe tener una configuración en el recocido simulado se debe tener una solución factible; la solución óptima en el problema de optimización está relacionada con la configuración que se debe tener para alcanzar el mínimo de energía nombrado anteriormente, y finalmente el manejo de la temperatura en el proceso de recocido se compara con un parámetro dado en el modelo de optimización.

El proceso de recocido simulado (SA) al igual que el de otras metaheurísticas es iterativo y cada iteración corresponde a una fase de enfriamiento, en cada iteración se realiza una perturbación de la solución, el reemplazo que produce esta perturbación se haría si es mejor que la solución actual o si cumple cierta regla de aceptación que es probabilística y está en función de la temperatura.

Chen et al desarrolló un algoritmo híbrido que incorporó la regla de aceptación de Simulated Annealing con un algoritmo genético. En el algoritmo híbrido, el algoritmo genético (GA) combina la búsqueda global y la búsqueda local para buscar los resultados óptimos y el Simulated Annealing (SA) utiliza cierta probabilidad para evitar quedar atrapado en un óptimo local. El estudio computacional mostró que el algoritmo propuesto es un enfoque factible y efectivo para el problema de ruteo de vehículos con varios depósitos. [95]

Mirabi et al. presentaron tres heurísticos híbridos para resolver el MDVRP combinando heurísticos de búsqueda y mecanismos de mejora de tipo determinístico, estocástico y de Simulated Annealing, respectivamente. [19]

# Búsqueda Local e Iterativa

## GRASP

El algoritmo de GRASP crea una solución inicial para luego mejorar la calidad de la solución. Es un algoritmo iterativo, donde cada iteración tiene una fase donde se construye una solución y una fase donde se mejora. GRASP se introdujo por primera vez en Feo y Resende (1989). [96][5]

### Fase de Construcción

La primera fase consiste en la definición de una función Greedy que orienta la construcción de la solución eligiendo el mejor movimiento disponible sin tener en cuenta que puede pasar en el futuro.

Se le agrega a las funciones Greedy un parámetro de relajación para evitar el determinismo a la hora de elegir la lista restringida de candidatos.

La fase constructiva no asegura una solución óptima. Para resolver esta situación GRASP incorpora una fase de mejora en la iteración con el fin de realizar una optimización local mediante una función de búsqueda local. [96]

### Fase de Mejora

Un algoritmo de búsqueda local explora reiteradamente el vecindario de la solución de la fase constructiva en busca de la una mejor solución. El método realiza movimientos siempre y cuando mejore el valor de la función objetivo y finaliza cuando no encuentra movimientos que la mejoren. [96]

La estructura básica de un algoritmo GRASP es la siguiente:

```
Mientras no se cumpla la condición de parada
Hacer
Fase Constructiva
      Construir solución aleatoria Greedy
      Considerar una lista restringida de los mejores candidatos
      Seleccionar un elemento aleatoriamente de la lista restringida
Fase de mejora
            Realizar un procedimiento de búsqueda local a partir de
      la solución construida hasta que no se pueda mejorar mas.
Actualización
            Si en la solución obtenida mejora a la mejor almacenada,
      actualizarla.
Fin Mientras
```

Priore-Moreno et al. [97] en su trabajo de 2012 describen el desarrollo e implementación de un Sistema de Soporte a la Decisión que ayuda en el proceso de cálculo de rutas y llenado de camiones, que tienen que transportar un número considerable de vehículos desde 8 orígenes y distribuirlos entre más de 3.000 posibles destinos repartidos por España y Portugal. Se analiza, en primer lugar, el comportamiento de distintas metodologías a la hora de resolver un problema de rutas del tipo MDVRP y VRPTW. Tras ello, optan por la utilización de la heurística GRASP como núcleo del optimizador. El resultado es una mejora en la utilización del cubicaje de los vehículos, y una racionalización en las rutas que se traducen en un descenso de los costos de transporte.

Método Híbrido IVNDS

El método híbrido IVNDS [17] para la resolución MDVRP se basa en una conjunción de heurísticas conocidas y descritas en el documento con algunas modificaciones específicas. Aunque principalmente la columna vertebral del algoritmo IVNDS se basa en una búsqueda local iterada. El pseudocódigo de la búsqueda local iterada se puede observar a continuación.

Algoritmo de búsqueda local iterada

$S_0$ solución inicial y $S^*$ solución actual

$S_0$=Generar solución inicial
$S^* =$ Búsqueda local ($S_0$)
Mientras Condición de parada hacer
    $S'$ =Perturbación de $S^*$
    $S^{*\prime}$ =Búsqueda local ($S'$)
    $S^*$=Criterio de aplicación ($S^*$, $S^{*\prime}$,history)
Fin mientras

Como forma de no confundir óptimos locales con óptimos globales, la metaheurística de IVNDS propone que en cada iteración se aplique un método de perturbación seguido de una búsqueda local. Además, el algoritmo IVNDS permite utilizar criterios de aceptación de la solución obtenida en la búsqueda local para determinar si es válida para la evolución del algoritmo. [17]

El algoritmo IVNDS presenta aspectos fundamentales de búsqueda local, criterio de aceptación y perturbación. Donde cada uno de estos aspectos presentan algunas especificidades concretas descritas a lo largo del desarrollo escrito del algoritmo. [17]

A continuación, se observa el pseudocódigo del algoritmo IVNDS.

// $S$ solución actual, $S^*$ mejor solución
$S\leftarrow$Asignar clientes a depósitos con el algoritmo del depósito más cercanos
$S \leftarrow$construir solución inicial
Set $S^* \leftarrow S$, iterations $\leftarrow$0
Mientras iterations < n hacer
Búsqueda Local ($S$)
// criterio de aceptación
    si f($S$) < f($S^*$) entonces
        $S^* \leftarrow S$
fin si

si iterations >= Ω entonces
$S \leftarrow S^*$
fin si
Perturbación ($S$)
fin mientras
devolver $S^*$

Como se observa en el pseudocódigo del algoritmo, en la fase de inicialización de este se asigna cada cliente a cada depósito y se genera un conjunto de rutas factibles para esta asignación inicial. Este proceso produce una solución inicial válida para el comienzo de la mejora. Para comenzar con una buena solución inicial y no presentar ningún inconveniente luego, es necesario que la misma no sea generada de manera aleatoria. [17]

Existen diferentes métodos para asignar cada cliente a un depósito en la fase de inicialización del algoritmo como, por ejemplo, los que presenta el estudio de asignación por Giosa et al [27].

Para la creación de las rutas de la solución inicial, se utiliza el algoritmo del vecino más cercano, donde se selecciona un cliente de manera aleatoria, el cual va a ser el primero en ser atendido y luego por criterios de cercanía se eligen el orden en que los demás clientes son agregados a la ruta. Cada vez que se agrega un cliente a la ruta se verifica que se cumpla con la restricción de capacidad del vehículo que recorre la ruta. En caso de no cumplirse la restricción de capacidad se genera otra ruta para otro vehículo y se procede de manera similar. [17]

El método de búsqueda local desarrollado por el algoritmo IVNDS se basa en una variante de la metaheurística de búsqueda de entornos variables de Hansen. [17]

Dada una solución inicial, el algoritmo explora sus diversas estructuras de vecindarios, refiriéndose a vecindario como el rango de soluciones que se pueden obtener aplicando una transformación específica en la solución inicial. Por ejemplo, cuando se obtiene un mínimo local en un vecindario, se procede a buscar otro mínimo local en otro vecindario y así sucesivamente. Si el mínimo local obtenido en un vecindario no supone una mejora, el proceso se repetirá con el siguiente vecindario. Si por el contrario se produce una mejora, el proceso se reiniciará desde el primer vecindario. [17]

La variante de la metaheurística se denomina búsqueda por entornos descendentes. La variante con la búsqueda de entornos variables es que en cada vecindario se realiza la transformación que genera un costo menor, en vez de ser aleatoria para aplicar la mejora. Obteniendo un mínimo local con respecto los demás vecindarios al terminar el proceso. [17]

Las transformaciones que se aplican en el algoritmo IVNDS se basan en las transformaciones $\lambda$ y $\lambda$-opt por Lin [42].

Los criterios de aceptación en la metaheurística determinan si la solución generada por la búsqueda local se utilizará en las futuras iteraciones. Además, arbitra las fases de diversificación e intensificación del proceso de búsqueda. En el algoritmo IVNDS se mantiene en todo momento constancia de la mejor solución obtenida y en cada caso se selecciona la solución que se mantendrá en las siguientes iteraciones por el parámetro $\Omega$ observado en el algoritmo. [17]

La primera fase del algoritmo favorece a la diversificación donde se mantiene la solución obtenida por la búsqueda local sin tener en cuenta la calidad de esta. Luego en la segunda fase de la inicialización de los algoritmos se corresponde a la intensificación donde se obtiene la mejor solución obtenida hasta el momento centrándose los esfuerzos de búsqueda en la región más prometedora del espacio de soluciones. [17]

Como medida de que los óptimos locales no sean adquiridos a una solución y favorecer a la diversificación de la búsqueda, se aplica el método de perturbación. Para facilitar la evasión de mínimos el algoritmo presenta cuatro métodos de perturbación, de los cuales se seleccionan de manera aleatoria en cada iteración del algoritmo. De los cuatros métodos de perturbación tres son del tipo “intra-depot” y el otro es de tipo “inter-depot”. Según los experimentos se observó que el algoritmo deshacía en las iteraciones algunas transformaciones realizadas por las perturbaciones, entonces se prohibió revertir las modificaciones por un determinado número de iteraciones. [17]

Un aspecto para tomar en cuenta es el impacto que produce la perturbación representado por el parámetro $\Phi$.Donde un valor reducido de $\Phi$ puede que reflejar que la solución obtenida mediante la perturbación no se aleje lo suficiente de un mínimo local. En cambio, si el valor de del parámetro $\Phi$ es muy elevado puede reflejar en deterioro de la solución de manera que las buenas características de la solución se perderán. [17]

A continuación, se describen cuatro métodos de perturbación aplicados en el algoritmo IVNDS.

Método Swap:

Este procedimiento selecciona dos clientes asignados a un mismo depósito e intercambia sus posiciones. El primer cliente seleccionado se selecciona de manera aleatoria y el segundo se elige entre la lista de vecinos del primer cliente. Además, el segundo cliente seleccionado debe pertenecer a una ruta diferente de la que contiene al primer cliente. El método Swap se aplica una cantidad de $\Phi$veces. [17]

Método Insert:

El procedimiento de modificación “Insert” consiste en seleccionar un cliente aleatoriamente y a $\Phi$-1 clientes vecinos a él. Estos clientes son extraídos de la solución y reinsertados donde produzcan un menor costo en la solución. El método “Insert” es para los clientes pertenecientes a un mismo depósito, al momento de la inserción de menor costo. [17]

Método Exchange:

El método Exchange intercambia una cadena de clientes pertenecientes a una ruta con otros pertenecientes a una segunda ruta. El tamaño de la cadena varía entre 2 y $\Phi$. El método “Exchange” considera las rutas de un mismo depósito al momento de intercambiar las cadenas de clientes. [17]

Método intra-depot:

En este método de modificación se elige un depósito de manera aleatoria y un depósito destino próximo al primero. Luego se construye una lista para cada cliente del depósito de origen la lista de clientes vecinos pertenecientes al depósito destino. Con la construcción de la lista de clientes vecinos se permite observar la probabilidad de inserción de cada cliente del depósito origen al destino, dado por la cantidad de vecinos que el mismo posee en su lista, teniendo más probabilidad el cliente con mayor número de vecinos. La reinserción de cada cliente en este método de modificación se debe realizar de manera de producir de minimizar el costo de la solución. Este método intenta ampliar la búsqueda de soluciones hacia regiones más prometedoras. [17]

El algoritmo IVNDS se comparó con los algoritmos búsqueda Tabú Cordeau, el ALNS de Pisinger y el algoritmo evolutivo de Vidal obteniendo destacables resultados teniendo en cuenta la sencillez del mismo. [17]

# Conclusiones

Como se ha podido ver a lo largo de los capítulos de este documento, MDVRP, es un problema bajo investigación, importante para el sector logístico, donde todos los meses se pueden encontrar nuevas publicaciones académicas al respecto.

El aspecto más importante de este trabajo es la realización de un relevamiento de métodos para resolver VRP y MDVRP. El relevamiento cumplió con el objetivo de facilitar el acercamiento al problema y a sus variantes, a pesar de la dificultad que plantea la cantidad de propuestas de solución. Particularmente, este estado del arte permitió estudiar las heurísticas y metaheurísticas más difundidas y utilizadas, así como nuevas propuestas más puntuales.

El objetivo de este documento en sí es presentar tanto los algoritmos más relevantes a la hora de resolver VRP y su generalización MDVRP, así como ejemplos de algoritmos que resuelven instancias de MDVRP con determinadas características dentro de distintos enfoques.

# Bibliografía Estado del arte


[1] W. Tu, Z. Fang, Q. Li, S. L. Shaw & B. Chen (2014). A bi-level Voronoi diagram-based metaheuristic for a large-scale multi-depot vehicle routing problem. *Transportation Research Part E: Logistics and Transportation Review 61*, 84-97.

[2] W. Ho, G. T. Ho, P. Ji & H. C. Lau (2008). A hybrid genetic algorithm for the multi-depot vehicle routing problem. *Engineering Applications of Artificial Intelligence 21*, 548-557.

[3] J. R. Montoya-Torres, J. López Franco, S. Nieto Isaza & H. Felizzola Jiménez (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering 79*, 115-129.

[4] M. Adelzadeh, V. Mahdavi Asl & M. Koosha (2014). A mathematical model and a solving procedure for multi-depot vehicle routing problem with fuzzy time window and heterogeneous vehicle. *The International Journal of Advanced Manufacturing Technology 75(5)*, 793–802.

[5] C. Contardo & R. Martinelli (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization 12,* 129-146.

[6] S. Mancini (2016). A real life Multi Depot Multi Period Vehicle Routing Problem with a Heterogeneous Fleet Formulation and Adaptive Large Neighborhood Search based Metaheuristic. *Transportation Research Part C 70*, 100-112.

[7] R. Baldacci & A. Mingozzi (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming 120(2)*, 347-380.

[8] L. Tansini & O. Viera (2004). Adapted Clustering Algorithms for the Assignment Problem in the MDVRPTW. *UR. FI – INCO, Montevideo*.

[9] A. Olivera (2004). Heurísticas para Problemas de Ruteo de Vehículos. *Instituto de Computación, Facultad de Ingeniería, UdelaR, Montevideo*.

[10] K. V. Narasimha, E. Kivelevitch, B. Sharma & M. Kumar (2013). An ant colony optimization technique for solving min–max Multi-Depot Vehicle Routing Problem. *Swarm and Evolutionary Computation 13*, 63-73.

[11] E. Lalla-Ruiz, C. Expósito-Izquierdo, S. Taheripour & S. Voß (2016). An improved formulation for the multi-depot open vehicle routing problem. *OR Spectrum 38(1),* 175-187.

[12] J. Ma & J. Yuan (2010). Ant Colony Algorithm for Multiple-Depot Vehicle Routing Problem with Shortest Finish Time. E-business Technology and Strategy. *Communications in Computer and Information Science 113*, 114-123.

[13] F. B. Pereira & J. Tavares (2009). Bio-inspired Algorithms for the VRP. *Poland: Springer.*

[14] J. Rodríguez Perez (2012). Caracterización, Modelado y Determinación de las Rutas de la Flota en una Empresa de Rendering. E-Reading. *Trabajos y Proyectos de fin de estudios de la E.T.S.I Máster en Organización Industrial y Gestión de Empresas.*

[15] L. Tansini, M. Urquhart & O. Viera (2001). Comparing assignment algorithms for the Multi-Depot VRP. *UdelaR, Montevideo*.

[16] T. Tlili & S. Krichen (2015). Decision Support System for the Multi-Depot Vehicle Routing Problem. *Modelling, Computation and Optimization in Information Systems and Management Sciences 359*, 47-55.

[17] I. Gallego Mateos (2013) Desarrollo de un método híbrido para la resolución del MDVRP. *Revista de la Escuela Jacobea de Posgrado 5*, 45-64.

[18] B. Dorronsoro Díaz (2006). Diseño e Implementación de Algoritmos para Problemas Complejos (Tesis Doctoral). *Universidad de Málaga, Málaga.*

[19] M. Mirabi, S. Fatemi Ghomi & F. Jolai (2010). Efficient stochastic hybrid heuristics for the multi-depot vehicle routing problem. *Robotics and Computer-Integrated Manufacturing 26(6)*, 564-569.

[20] K. Sundary, S. Venkatachalam & S. Rathinamz (2016). Formulations and algorithms for the multiple depot, fuel-constrained, multiple vehicle routing problem*. American Control Conference (ACC), Boston*.

[21] S. Thangiah & S. Salhi (2001). Genetic clustering: An adaptive heuristic for the multidepot vehicle routing problem. *Applied Artificial Intelligence 15(4*), 361-383.

[22] S. Geetha, P. Vanathi & G. Poonthalir (2012). Metaheuristic Approach for the Multi-Depot Vehicle Routing Problem. *Applied Artificial Intelligence 26(9),* 878-901.

[23] G. González Vargas & F. González Aristizábal (2006). Metaheurísticas aplicadas al ruteo de vehículos. Un caso de estudio. Parte 1: formulación del problema. *Revista Ingeniería e Investigación 26(3),* 149-156.

[24] N. Herazo Padilla (2012). Modelación matemática del problema de ruteo de vehículos con restricciones de múltiples depósitos, flota heterogénea de vehículos y ventanas de tiempos (Tesis de Pregrado). *Corporación Universitaria de la Costa, CUC, Barranquilla, 2012.*

[25] A. Lim & F. Wang (2005). Multi Depot Vehicle Routing Problem A One Stage Approach. *IEEE Transactions on Automation Science and Engineering 2(4),* 397-402.

[26] H. Baea & I. Moon (2016). Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles. *Applied Mathematical Modelling 40(13-14)*, 6536-6549.

[27] D. Giosa, L. Tansini & O. Viera (2002). New Assignment Algorithms for the Multi-Depot Vehicle Routing Problem. *The Journal of the Operational Research Society 53(9),* 977-984.

[28] T. Tlili, S. Krichen, G. Drira & S. Faiz (2015). On Solving the Multidepot Vehicle Routing Problem. *Smart Innovation, Systems and Technologies 44*, 103-108.

[29] H. Kaur (2013). Review Paper on Multi Depot Vehicle Routing Problem. International Journal of Scientific & Engineering Research 4(8), 410-413.

[30] J. Renaud, F. Boctor & G. Laporte (1996). A fast composite heuristic for the symmetric traveling salesman problem. *INFORMS Journal on Computing 8(2),* 134-143.

[31] J. Renaud, G. Laporte & F. Boctor (1996). An improved petal heuristic for the vehicle routing problem. *Journal of the Operational Research Society 47*, 329-336.

[32] N. Christofides & S. Eilon (1969). An algorithm for one vehicle-dispatching problem. *Operational Research Quarterly 20(3),* 309-318.

[33] G. Dantzig & J. Ramser (1959). The Truck Dispatching Problem. Management Science 6(1), 80-91.

[34] G. Laporte & F. Foveaux (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations research letters 13(3)*, 133-142.

[35] G. Laporte & Y. Nobert (1987). Exact Algorithm for the Vehicle Routing Problem. *Annals of Discrete Mathematics 31,* 147-184.

[36] N. Christofides, A. Mingozzi & P. Toth (1981). Exact algorithms for the vehicle routing problem, based on spanning trees and shortest path relaxations. *Mathematical Programming 20*, 255-181.

[37] S. Eilon, W.G. CDT & N. Christofides (1974). Distribution management: Mathematical modelling and practical analysis. *Transactions on Systems Man and Cybernetics 21(6)*, 589-589.

[38] G. Dastghaibifard, E. Ansari, S. Sheykhalishahi, A. Bavandpouri & E. Ashoor (2008). A Parallel Branch and Bound Algorithm for Vehicle Routing Problem. *Proceedings of the International Multiconference of Engineers and Computer Scientists 2*, 19-21.

[39] T. Ralphs (2003). Parallel branch and cut for capacitated vehicle routing. *Parallel Computing 29(5)*, 607-629.

[40] G. Laporte (2009). Fifty Years of Vehicle Routing. *Transportation Science 43(4),* 408-416.

[41] S. Nieto Isaza, J. López Franco & N. Herazo Padilla (2012). Desarrollo y Codificación de un Modelo Matemático para la Optimización de un Problema de Ruteo de Vehículos con Múltiples Depósitos. *LACCEI Latin American and Caribbean Conference (LACCEI'2012), Megaprojects: Building Infrastructure by fostering engineering collaboration, efficient and effective integration and innovative planning*. Ciudad de Panamá.

[42] S. Lin (1965). Computer solutions of travelling salesman problem. *The Bell System Technical Journal 44(10),* 2245 - 2269.

[43] B. Bullnheimer, R. F. Hartl & C. Strauss (1999). Applying the ant system to the vehicle routing problem. *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization*, 285-296.

[44] Glover (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research 13(5)*, 533-549.

[45] M. Gendreau, A. Hertz & G. Laporte (1994). A tabu search heuristic for the vehicle routing problem. *Management Science 40(10),* 1276-1290.

[46] J. Renaud, G. Laporte & F. Boctor (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research 23(3)*, 229-235.

[47] Gendreau (1992). New insertion and post optimization procedures for the traveling salesman problem. *Operations research 40*, 1082-1094.

[48] T. Volgenant & R. Jonker (1983). The symmetric traveling salesman problem and edge exchanges in minimal 1-tree. *European Journal of Operational Research 12(4),* 394-403.

[49] B. Baker & M. Ayechew (2003). A genetic algorithm for the vehicle routing problem. *Computers & Operations Research 30(5)*, 787-800.

[50] P. Surekha & S. Sumathi (2011). Solution to Multi-Depot Vehicle Routing Problem Using Genetic Algorithms. World Applied Programming 1(3), 118-131.

[51] J. Carlsson, G. Dongdong, A. Subramaniam, A. Wu & Y. Ye. (2009). Solving Min-Max Multi-Depot Vehicle Routing Problem. *Lectures on Global Optimization 55*, 31-46.

[52] G. Clarke & W. Wright. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research 12*, 568–581.

[53] R.H. Mole & S.R. Jameson (1976). A sequential route-building algorithm employing a generalized savings criterion. *Operational Research Quarterly 27,* 503–511

[54] Christofides, Mingozzi & Toth (1979). The Vehicle Routing Problem. *Combinatorial Optimization. Wiley, Chichester,* 315–338

[55] A. Wren (1971). Computers in transport planning and operation. Ian Allan

[56] M. Balinski & R. Quandt (1964). On an integer program for a delivery problem. *Operations Research 12*, 300–304

[57] M. Fisher & R. Jaikumar (1981). A generalized assignment heuristic for the vehicle routing problem. *Networks 11*, 109–124

[58] J. Bramel & D. Simchi-Levi (1995). A location-based heuristic for general routing problems. *Operations Research 43,* 649–660

[59] J. Beasley (1983). Route first – cluster second methods for vehicle routing. *Omega 11,* 403–408

[60] A. Colorni, M. Dorigo & V. Maniezzo (1991). Distributed optimization by ant colonies. *Proceedings of the European Conference on Artificial Life,* 134–142

[61] A. Colorni, M. Dorigo & V. Maniezzo (1996). The Ant System: Optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 26*, 29–41

[62] B. Bullnheimer, R. Hartl & C. Strauss (1997). A new rank-based version of the ant system – a computational study. *Technical report, University of Vienna, Institute of Management Science.*

[63] L.M. Gambardella & M. Dorigo (1995). Ant-Q: A reinforcement learning approach to the traveling salesman problem. *International Conference on Machine Learning.* 252–260.

[64] E.D. Taillard (1993). Parallel iterative search methods for vehicle routing problems. *Networks 23*, 661–673.

[65] J. Holland (1975). Adaptation in Natural and artificial systems. *The University of Michigan Press.*

[66] Francisco Pereira & Jorge Tavares (2002). A new genetic representation for the vehicle routing problem. *Proceedings of the 13th.Conference on Artificial Intelligence and Cognitive Science*, 95–102.

[67] B. Ombuki-Berman & F. Hanshar (2009). Using genetic algorithms for multi-depot vehicle routing. *Bio-Inspired Algorithms for the Vehicle Routing Problem 161*, 77-99.

[68] J. Potvin & J. Rousseau (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Research 66,* 331-340.

[69] I. Or (1976). Travelling salesman-type combinatorial optimization problems and their relation to the logistics of regional blood banking.

[70] A. Breedam (1995). Improvement Heuristics for the vehicle routing problem based on simulated annealing. *European Journal of Operations Research*, 480-490.

[71] R. Baldacci, A. Mingozzi, E. Bertolini & A. Valletta (2001). An Exact Algorithm for the Period Routing Problem. *Journal Operations Research 59(1),* 228-241.

[72] J.F. Cordeau, M. Gendreau & G. Laporte (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks 30(2)*, 105-119.

[73] B. Crevier, J.F. Cordeau & G. Laporte (2007). The Multi-Depot Vehicle Routing Problem with Inter-Depot Routes. *European Journal of Operational Research 176(2),* 756-773.

[74] R. Dondo, C. Mendez & J. Cerdá (2007). A cluster-based optimization approach for the multidepot heterogeneous fleet vehicle routing problem with time windows. *European Journal of Operational Research 176(3),* 1478–1507.

[75] R. Dondo, C. Mendez & J. Cerdá (2003). An optimal approach to the multiple depot heterogeneous vehicle routing problem with time window and capacity constraints. *Latin American Applied Research 33(2)*, 129–134.

[76] J. Escobar, R. Linfati, P. Toth & M. Baldoquin (2014). A Hybrid Granular Tabu Search algorithm for the Multi-Depot Vehicle Routing Problem. *Journal of Heuristics 20,* 1-27.

[77] A. Ezugwu, F. Akutsah, M. Olusanya, & A. Adewumi (2018). Enhanced intelligent water drops algorithm for multi-depot vehicle routing problem. *PLoS ONE 13(3).*

[78] G. González & F. González (2007). Metaheurísticas Aplicadas Al Ruteo de Vehículos. Un Caso de Estudio. Parte 2: Algoritmo Genético, Comparación Con Una Solución Heurística. *Revista Ingeniería e Investigación 27 (1),* 149–157.

[79] M. Granada-Echeverri, R. Bolaños & J. Escobar (2018). A metaheuristic algorithm for the multi-depot vehicle routing problem with heterogeneous fleet. *International Journal of Industrial Engineering Computations 4(9),* 461- 478.

[80] E. Hadjiconstantinou & R. Baldacci (1998). A multi-depot period vehicle routing problem arising in the utilities sector. *Journal of the Operational Research Society 49(12),* 1239-1248.

[81] G. Jeon, H.R. Leep & J.Y. Shim (2007). A vehicle routing problem solved by using a hybrid genetic algorithm. *Computers & Industrial Engineering 53(4)*, 680-692.

[82] C.G. Lee, M.A. Epelman, C.C White & Y.A. Bozer (2006). A shortest path approach to the multiple-vehicle routing problem with split pick-ups. *Transportation Research Part B: Methodological 40(4),* 265-284.

[83] J. Li, Y. Li & P. Pardalos (2016). Multi-depot vehicle routing problem with time windows under shared depot resources. *Journal of Combinatorial Optimization 31(2),* 515-532.

[84] R. Liu, Z. Jiang & N. Geng (2012). A hybrid genetic algorithm for the multi-depot open vehicle routing problem. *OR Spectrum 36(2),* 401-421.

[85] M. Maischberger & J.F Cordeau (2011). Solving Variants of the Vehicle Routing Problem with a Simple Parallel Iterated Tabu Search. *International Conference on Network Optimization,* 395-400.

[86] G. Nagy & S. Salhi (2007). Location-Routing: Issues, models and methods. *European Journal of Operational Research,* 649-672.

[87] B. Ombuki - Berman & F. Hanshar (2009). Using Genetic Algorithms for Multi-Depot Vehicle Routing. *Bio-inspired Algorithms for the Vehicle Routing Problem 161*, 77-99.

[88] S. Salhi & M. Sari (1997). A multi-level composite heuristic for the multi-depot vehicle fleet mix problem. *European Journal of Operational Research 103(1)*, 95-112.

[89] K. Tang, J. Ying & K. Man (2010). A genetic-based optimization for multi-depot vehicle routing problem. *IEEE. Int. Symp. Ind. Elec.,* 1545-1549.

[90] E. Toro Ocampo (2016). Solución del problema de localización y ruteo usando un modelo matemático flexible y considerando efectos ambientales. *Pereira: Universidad Tecnológica de Pereira.*

[91] P. Tsirimpas, A. Tatarakis, I. Minis & E. Kyriakidis (2008). Single vehicle routing with a predefined customer sequence and multiple depot returns. *European Journal of Operational Research 187(2),* 483-495.

[92] T. Vidal, T.G. Crainic, M. Gendreau & C. Prins (2014). Implicit depot assignments and rotations in vehicle routing heuristics. *European Journal of Operational Research 237(1)*, 15-28.

[93] X. Wang, J. Sun & C. Ren (2009). Study on Hybrid Heuristic Algorithm for Multi-Depot Vehicle Routing Problem with Hybrid Picking Delivery Strategy. *Proceedings of the Eighth International Conference on Machine Learning and Cybernetics*, 1-6.

[94] Z. Yu, Yang & J. Xie (2011). A parallel improved ant colony optimization for multi-depot vehicle routing problem. *Journal of the Operational Research Society 62(1),* 183-188.

[95] P. Chen & X. Xu (2008). A hybrid algorithm for multi-depot vehicle routing problem. *2008 IEEE International Conference on Service Operations and Logistics, and Informatics*, 2031-2034.

[96] T.A. Feo & M.G. Resende (1989). A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters 8(2)*, 67-71.

[97] P. Priore Moreno, C. Martínez, V. Villanueva, J. Lozano & I. Fernández (2012). Aplicación de la metodología GRASP al problema de Rutificación de Vehículos (VRP). *Dirección y Organización 48* , 46-55.

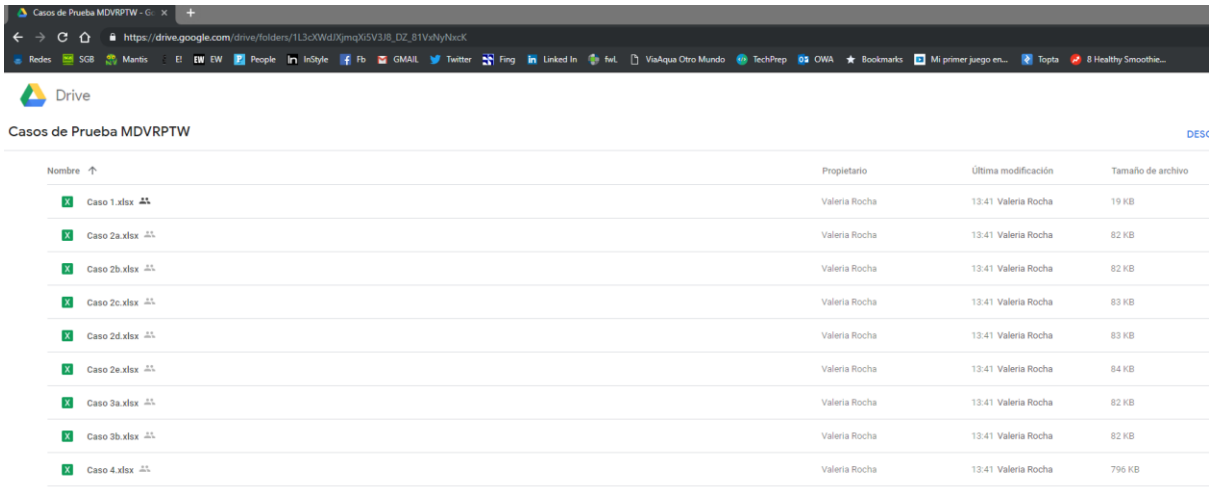
[98] Solomon, M.: Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research 35 (1987) 254-264

# Casos de Prueba

Los casos de prueba se encuentran en formato de Excel (xlsx) y no se podían apreciar correctamente como tablas en este informe.

Se puede acceder a los mismos utilizando el enlace: https://drive.google.com/drive/folders/1L3cXWdJXjmqXi5V3J8_DZ_81VxNyNxcK?usp=sharing

Los resultados de la ejecución de cada caso se encuentran en el capítulo de Testeos, específicamente en la sección de Ejecución y Resultados.



# Bibliografía informe final


[1] Baldacci, R., & Mingozzi, A. (2009). A unified exact method for solving different classes of vehicle routing problems. *Mathematical Programming, 120*(2), 347-380.

[2] Baldacci, R., Mingozzi, A., Bertolini, E., & Valletta, A. (2001). An Exact Algorithm for the Period Routing Problem. *Journal Operations Research, 59*(1), 228-241.

[3] Chirstofides, N., Mingozzi, A., & Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning trees and shortest path relaxations. *Mathematical Programming, 20*, 255-181.

[4] Christofides, Mingozzi, & Toth. (1979). The Vehicle Routing Problem. *Combinatorial Optimization. Wiley, Chichester.*, 315-338.

[5] Clarke, G., & Wright, W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research (12)*, 568-581.

[6] Contardo, C., & Martinelli, R. (2014). A new exact algorithm for the multi-depot vehicle routing problem under capacity and route length constraints. *Discrete Optimization, 12*, 129-146.

[7] Crevier, B., Cordeau, J.-F., & Laporte, G. (2007). The Multi-Depot Vehicle Routing Problem with Inter-Depot Routes. *European Journal of Operational Research, 176*(2), 756-773.

[8] Dantzig, G. B., & Ramser, J. H. (1959). The Truck Dispatching Problem. *Management Science, 6*(1), 80-91.

[9] Dondó, R., Mendez, C., & Cerdá, J. (2003). An optimal approach to the multipledepot heterogeneous vehicle routing problem with time window and capacity constraints. *Latin American Applied Research, 33(2)*, 129–134.

[10] Geetha, S., Vanathi, P., & Poonthalir, G. (2012). Metaheuristic Approach for the Multi-Depot Vehicle Routing Problem. *Applied Artificial Intelligence, 26*(9), 878-901.

[11] Giosa, D., Tansini, L., & Viera, O. (2002). New assignment algorithms for the multi-depot vehicle routing problem. *Journal of the Operational Research Society, Volume 53 Issue 9*, 977-984.

[12] Glover. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13*(5), 533-549.

[13] Laporte, G., & Nobert, Y. (1987). Exact Algorithm for the Vehicle Routing Problem. *Annals of Discrete Mathematics, 31*, 147-184.

[14] Li, J., Li, Y., & Pardalos, P. (2016). Multi-depot vehicle routing problem with time windows under shared depot resources. . *Journal of Combinatorial Optimization 31(2)*, 515-532.

[15] Lin, S. (1965). Computer solutions of travelling salesman problem. *The Bell System Technical Journal , 44*(10), 2245 - 2269.

[16] Mole, R., & Jameson, S. (1976). A sequential route-building algorithm employing a generalized savings criterion. *Operations Research Quarterly (27)*, 503-511.

[17] Montoya-Torres, J. R., López Franco, J., Nieto Isaza, S., & Felizzola Jiménez, H. (2015). A literature review on the vehicle routing problem with multiple depots. *Computers & Industrial Engineering, 79*, 115-129.

[18] Nieto Isaza, S., Lopez Franco, J., & Herazo Padilla, N. (2012). Desarrollo y Codificación de un Modelo Matemático para la Optimización de un Problema de Ruteo de Vehículos con Múltiples Depósitos. *Tenth LACCEI Latin American and Caribbean Conference (LACCEI'2012), Megaprojects: Building Infrastructure by fostering engineering collaboration, efficient and effective integration and innovative planning.* Ciudad de Panamá.

[19] Olivera, A. (2004). *Heurísticas para Problemas de Ruteo de Vehículos.* Montevideo: Instituto de Computación, Facultad de Ingeniería, UdelaR.

[20] Or, I. (1976). *Travelling salesman-type combinatorial optimization problems ans their relation to the logistics of regional blood banking.*

[21] Potvin, J., & Russeau, J. (1993). A parallel route building algorithm for the vehicle routing and scheduling problem with time windows. *European Journal of Operational Research, 66*(3), 331-340.

[22] Renaud, J., Laporte, G., & Boctor, F. (1996). A tabu search heuristic for the multi-depot vehicle routing problem. *Computers & Operations Research, 23*(3), 229-235.

[23] Tansini, L., & Viera, O. (2004). *Adapted Clustering Algorithms for the Assignment Problem in the MDVRPTW.* Montevideo: UR. FI – INCO.

[24] Tansini, L., Urquhart, M., & Viera, O. (s.f.). *Comparing assignment algorithms for the Multi-Depot VRP.* Montevideo: http://www.fing.edu.uy/inco/pedeciba/bibliote/reptec/TR0108.pdf.