Una Heurística de Clusterización para el Problema del Ruteo de Vehículos Multidepósito

Rósulo Hilarión Pérez Cupe[†], Luis Ernesto Flores Luyo[‡] y Rolando Raul Palomino Vildoso[‡]
Escuela Profesional de Matemática. Facultad de Ciencias. IMCA
Universidad Nacional de Ingeniería;
[†]*rperezc@uni.edu.pe* [‡]*lflores@imca.edu.pe,* [‡]*rpalominov@uni.edu.pe*

16 de noviembre del 2020; aceptado el 22 de diciembre del 2020

El problema VRP (**V**ehicle **R**outing **P**roblem) es uno de los problemas de optimización mas importantes y desafiantes en el campo de la Investigación de Operaciones, consiste en la construcción de un conjunto óptimo de rutas para una flota de vehículos que deberán satisfacer la demanda de un conjunto de clientes, el problema está clasificado como un problema combinatorio computacionalmente difícil (NP-Hard). En las aplicaciones prácticas, diferentes versiones del VRP han ido apareciendo tal como el problema MDVRP (**M**ulti **D**epot **V**ehicle **R**outing **P**roblem) [1], cuando un problema NP-Hard no puede ser resuelto de manera exacta se buscan soluciones aproximadas obtenidas mediante heurísticas. En el presente trabajo estudiamos, formulamos y resolvemos (por medio de heurísticas) el problema MDVRP, la solución aproximada se trata desde el punto de vista práctico a través de la formulación e implementación (en el lenguaje de programación JULIA 1.0.5) de las heurísticas de construcción y mejora (siendo ésta la contribución del trabajo de investigación). En cuanto a la heurística de construcción, se presentan dos propuestas de agrupamiento o clusterización basadas en la ubicación geográfica de los clientes y los almacenes, así como de su proximidad entre sí, en cuanto a las heurísticas de mejora, las estrategias del vecino más cercano y de separación fueron usados. Finalmente, se presentan los resultados y comparaciones con respecto a la solución BKS (Best Know Solution) disponible en la literatura.

Palabras clave: Heurística, NP-Hard, Clusterización, MIP.

The vehicle routing problem VRP is one of the most important and challenging optimization problems in the field of Operations Research, consists of building an optimal set of routes for a fleet of vehicles that should satisfy the demand of a set of customers, the problem is classified as a computationally hard combinatorial problem. In practical application, different needs have emerged that made it necessary to formulate extensions or variants of the VRP problem, such as the MDVRP problem (**M**ulti **D**epot **V**ehicle **R**outing **P**roblem) [1], when a computationally hard combinatorial problem like the one mentioned cannot be solved exactly, the approximate solutions obtained by heuristic methods are used. In the present work we study, formulate and solve (by means of heuristics) the MDVRP problem, the approximate solution is dealt with from the practical point of view through the formulation and implementation (in the JULIA 1.0.5 programming language) of the construction and improvement heuristics (this being the contribution of the research work). Regarding the construction heuristics, two proposals for grouping or clustering are presented based on the geographic location of customers and warehouses as well as their proximity to each other, in terms of improvement heuristics, the strategies of the closest neighbor and separation were used. Finally, the results and comparisons with respect to the best-known BKS solution (Best Know Solution) available in the literature are presented.

Keywords: Heuristic, NP-Hard, clustering, MIP.

1. Introducción

El problema MDVRP consiste en la distribución de algún producto o bien que se encuentra almacenado en depósitos y debe ser distribuido a los clientes cada uno de los cuales tiene una demanda conocida, la distribución se realiza mediante vehículos de capacidad limitada. Específicamente se tiene n depósitos en cada una de las cuales existe una flota de vehículos que trasladarán los productos a los m clientes, cada vehículo sale de un depósito visita una sola vez a cada cliente, satisface su demanda y regresa al mismo depósito de donde partió. Siendo el costo de transporte proporcional a la distancia, el objetivo será encontrar aquella ruta que minimiza la distancia total recorrida por todos los vehículos, formulándose así un problema de programación entera mixta, el problema así formulado es de naturaleza $NP-Hard$ [2], ésta es la razón por la cual encontrar una solución exacta para tamaños considerables de clientes y/o depósitos es difícil de obtener, por ello se requiere la formulación de heurísticas que se diseñarán en el desarrollo del presente trabajo.

Por ejemplo si suponemos que una empresa cervecera tiene tres depósitos principales A, B y C (tal como se muestra en la figura 1) desde los cuales debe distribuir su mercadería a 18 clientes (círculos numerados en su interior y en cuya parte superior aparece su demanda) y para ello dispone de una flota de 9 vehículos cada una con una capacidad máxima, finalmente cada depósito también posee una capacidad máxima en cuanto a productos y vehículos, se plantea así un problema de ruteo de vehículos cuya solución optimizará la operatividad de esta empresa.

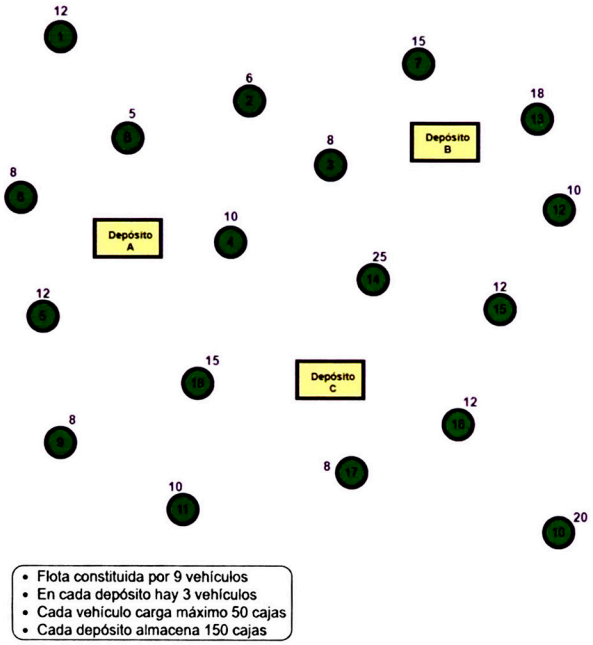

Figura 1. *Una instancia del problema del ruteo de vehículos multidepósito MDRVP con 3 depósitos y 18 clientes.*

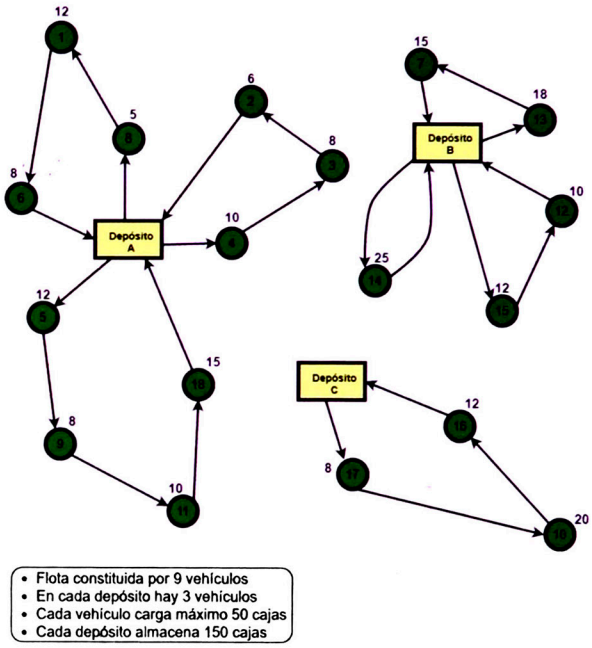

Figura 2. *Una solución factible para el problema propuesto en la figura 1. Esta solución usa 7 vehículos y tiene una longitud total de recorrido de 59.5 kilómetros y en consecuencia un costo total igual a 4760 soles.*

Como parámetros a priori se tiene información respecto a las ubicaciones de clientes y depósitos, flota de vehículos, carga máxima de cada vehículo, demanda de cada cliente, capacidad máxima de cada depósito. Consideramos la distancia entre clientes y depósitos dado en kilómetros y consideramos un costo por kilómetro igual a 80 soles (por ejemplo puede considerarse consumo de combustible, mantenimiento del vehículo y otros).

La solución exacta es posible determinarla utilizando solvers como son CPLEX versión 12.6.1 y GUROBI versión 9.1.0 o también programando el modelo en el lenguaje de programación JULIA version 1.0.5, cabe mencionar que solo es posible obtener soluciones exactas para tamaños de problema pequeños (menos de 30 clientes) en tiempos razonablemente cortos, para tamaños mayores el tiempo podría ser de orden exponencial, el análisis de complejidad de algoritmos puede encontarse en [3].

Por ejemplo en la figura 2 se tiene una solución factible para el problema planteado en la figura 1, se observa que en este caso el costo de la solución es igual a 4760 soles (de acuerdo a los datos considerados para la cuantificación del costo). Se dice que el problema está resuelto si encontramos aquella solución que minimiza el costo, al ser el problema mencionado de tipo $NP-hard$ la cantidad de soluciones factibles es de tipo combinatorio o exponencial por lo tanto una solución por exploración de todos los casos es inviable, en el presente trabajo se construirán heurísticas que permitan encontar soluciones razonablemente cercanas a la exacta.

2. Modelo Matemático

Presentamos a continuación un modelo matemático para la solución del problema planteado, consideremos para ello los siguientes conjuntos:

I : Depósitos ($i \in I$) $\qquad$ J : Clientes ($j \in J$)
K : Vehículos ($k \in K$), también denota la ruta k

Debe observarse que los conjuntos I, J y K son subconjuntos finitos de $\mathbb{N}$.

Asimismo consideremos los siguientes parámetros: (los cuales se conocen a priori)
N : cantidad de vehículos
C_{ij} : distancia entre los puntos i y j $\qquad$ $i, j \in I \cup J$
D_i : capacidad máxima del depósito i
d_j : demanda del cliente j
Q_k : capacidad máxima del vehículo (o ruta) k

A continuación se definen las variables de decisión:

$$x_{ijk} = \begin{cases} 1 & ; \text{ si el cliente } i \text{ precede inmediatamente} \\ & \quad \text{al cliente } j \text{ en la ruta } k \\ 0 & ; \text{ en otro caso} \end{cases}$$

$$z_{ij} = \begin{cases} 1 & ; \text{ si el cliente } j \text{ es asignado al depósito } i \\ 0 & ; \text{ en otro caso} \end{cases}$$

Asimismo consideramos adicionalmente las variables auxiliares U_{jk} : ($j \in J$ y $k \in K$) usados para las restricciones de eliminación de subrutas en la ruta k.

La función objetivo del modelo matemático consistirá en minimizar la distancia total de recorrido de todos los vehículos, es decir

$$\text{mín}\left\{\sum_{i\in I\cup J}\sum_{j\in I\cup J}\sum_{k\in K} C_{ij}x_{ijk}\right\}$$

sujeto a las restricciones:

$$\sum_{k\in K}\sum_{i\in I\cup J} x_{ijk} = 1 \quad \forall\, j \in J \tag{1}$$

Esta restricción nos dice que a cada cliente se le asignará una sola ruta.

$$\sum_{j\in J}\sum_{i\in I\cup J} d_j x_{ijk} \leq Q_k \quad \forall\, k \in K \tag{2}$$

Esta restricción asegura que la suma de demandas de los clientes de una determinada ruta es menor o igual a la capacidad del vehículo asignado a dicha ruta.

$$U_{lk} - U_{jk} + N x_{ijk} \leq N-1 \quad \forall\, l,j \in J \quad ; \quad \forall\, k \in K \tag{3}$$

Restricciones de eliminación de subrutas.

$$\sum_{j\in I\cup J} x_{ijk} - \sum_{j\in I\cup J} x_{jik} = 0 \quad \forall\, k \in K \quad \forall\, i \in I\cup J \tag{4}$$

Restricción referida a la conservación de flujo, para cada cliente o depósito la cantidad de arcos que ingresa al nodo es igual a la cantidad de arcos que sale de dicho nodo.

$$\sum_{i\in I}\sum_{j\in J} x_{ijk} \leq 1 \quad \forall\, k \in K \tag{5}$$

Establece que cada ruta o vehículo es usado a lo más una vez.

$$\sum_{j\in J} d_i z_{ij} \leq D_i \quad \forall\, i \in I \tag{6}$$

Esta restricción asegura que en cada depósito la suma de las demandas de los clientes abastecidos por el mencionado depósito debe ser menor que la capacidad máxima del depósito.

$$-z_{ij} + \sum_{u\in I\cup J} (x_{iuk} + x_{ujk}) \leq 1 \quad \forall\, i \in I,\ \forall\, j \in J,\ \forall\, k \in K \tag{7}$$

Esta restricción nos indica que un cliente es asignado a un depósito solo si hay ruta desde el depósito.

$$x_{ijk} \quad ; \quad z_{ij} \in \{0;1\} \quad ; \quad U_{lk} \geq 0 \tag{8}$$

Restricción de la naturaleza de las variables.

En los siguientes artículos ([5], [1] y [6]) se puede encontrar propuestas de heurísticas y metaheurísticas para el modelo planteado.

Dentro de la literatura existen dos tipos de heurísticas:

Heurísticas de Construcción: Estas heurísticas utilizan estrategias específicas para obtener una solución factible inicial, generalmente estas soluciones no son tan buenas, en el sentido de encontrarse lejos del óptimo, sin embargo gracias a las heurísticas de mejora podemos acercarnos a dicho óptimo.

Heurísticas de Mejora: Tomando la solución factible inicial obtenida por la heurística de construcción, estas heurísticas realizan pequeñas variaciones a la solución obtenida con la intención de mejorar la función objetivo.

3. Heurísticas de Clusterización

Las heurísticas propuestas en el presente trabajo construyen clústeres (grupos de clientes), cada cluster será asignados a un único depósito (un depósito puede contener mas de un cluster en general) y la demanda de cada cluster será satisfecha por la flota de vehículos existente en el depósito. Asimismo luego de obtener una solución factible inicial se proponen heurísticas de mejora para aproximarse mucho más a la mejor solución conocida (BKS)

3.1. La Heurística de clusterización por Distancias (HD)

En esta heurística se construirán tantos clústeres como depósitos existentes, para finalmente asignar a cada clúster el depósito más cercano. Cabe mencionar que en ésta propuesta de heurística no se tiene aún en cuenta las demandas de los clientes, ni la capacidad máxima de los depósitos, asi como tampoco la capacidad de cada vehículo; estos detalles serán considerados al final del proceso de clusterización y asignación de clústeres a los depósitos.

En cada paso del algoritmo se tiene una determinada cantidad de clústeres (inicialmente cada cliente constituye un cluster, por lo tanto la cantidad de clústeres al inicio es igual a la cantidad de clientes), entonces el algoritmo identifica los dos clústeres más cercanos (considerando la distancia entre dos clústeres como la distancia entre sus centroides) y une a ambos clústeres, constituyendo así un nuevo cluster de mayor tamaño (en cuanto a cantidad de nodos y demanda). El proceso se detiene cuando se ha logrado construir tantos clústeres como depósitos. Luego de considerar las siguientes notaciones: $m = |I|$ (cantidad de depósitos), $n = |J|$ (cantidad de clientes), $\mathcal{C}$:(conjunto finito de clústeres, cada elemento de $\mathcal{C}$ es un conjunto de clientes) y $d(C_i, C_j)$: distancia entre los clústeres C_i y C_j, se muestra el algoritmo

Algoritmo 1 (HD) Algoritmo de clusterización por distancias

Entrada: Clientes:(ubicación, demandas y cantidad). Depósitos:(ubicación, capacidad, cantidad). Vehículos:(capacidad y cantidad)

Salida: Conjunto de clústeres

1: $\mathcal{C} = \{\{c_1\}, \{c_2\}, \cdots, \{c_n\}\}$ (cada cliente es un cluster)
2: **mientras** $|\mathcal{C}| > m$ **hacer:**
3: Hallar dos índices i_{min} y j_{min} tales que $d(C_{i_{min}}, C_{j_{min}}) = min\{d(C_i, C_j) \,/\, C_i, C_j \in \mathcal{C}\}$
4: $C_{min} = C_{i_{min}} \cup C_{j_{min}}$
5: $\mathcal{C} = (\mathcal{C} \setminus \{C_{i_{min}}, C_{j_{min}}\}) \cup C_{min}$
6: **fin de mientras**
7: **retorna** $\mathcal{C}$

Se aplicará esta heurística a los datos mostrados (instancia) en la figura 3, donde tenemos 15 clientes enumerados desde 1 hasta 15, con sus respectivas demandas (los cuales están señalados en la parte superior de los respectivos nodos) y 5 depósitos (A , B , C , D y E), cuyas capacidades máximas también están señalados en la parte superior de cada depósito. También se observa que en la instancia de prueba se dispone de una flota de 6 vehículos $V_1, V_2, \cdots, V_6$ con sus respectivas capacidades máximas de carga (flota heterogénea en este caso).

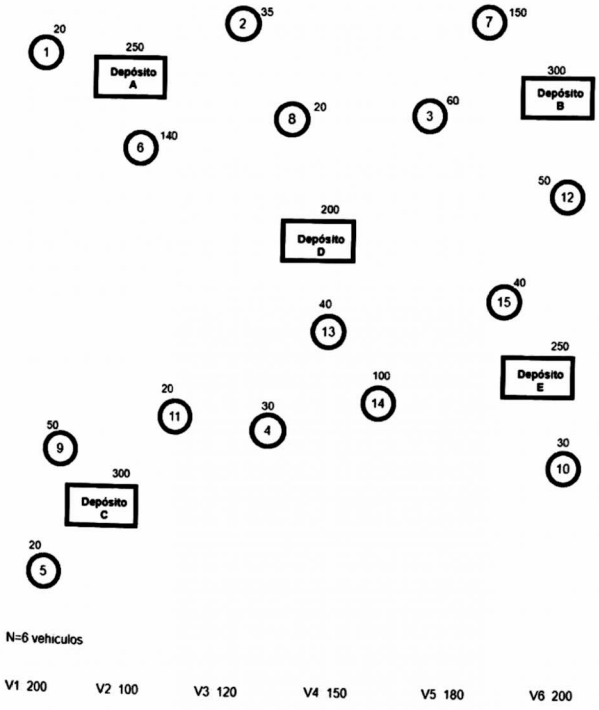

Figura 3. *Instancia de prueba a la cual se le aplicará la heurística* **HD**.

Como resultado de la aplicación del algoritmo a la instancia mencionada se obtuvieron 5 clústeres (C_1 , C_2 , C_3 , C_4 y C_5) uno por cada depósito, tal como se muestra en la figura 4.

Esta heurística presenta un inconveniente, si bien la clusterización de los clientes y depósitos es correcta, al no tener control sobre la capacidad total de cada clúster obtenido y las capacidades del vehículo o vehículos a asignar podría tornarse complicado o incluso imposible de asignar rutas. Por ejemplo según las demandas totales de los clústeres C_1 y C_2 y teniendo en cuenta las capacidades máximas de los vehículos disponibles sería necesario asignar dos vehículos a cada uno de dichos clústeres, usando así 4 vehículos de los 6 en total quedando 3 clústeres por asignar y solo dos vehículos disponibles los cuales de ninguna manera lograrán satisfacer la demanda de los otros clústeres.

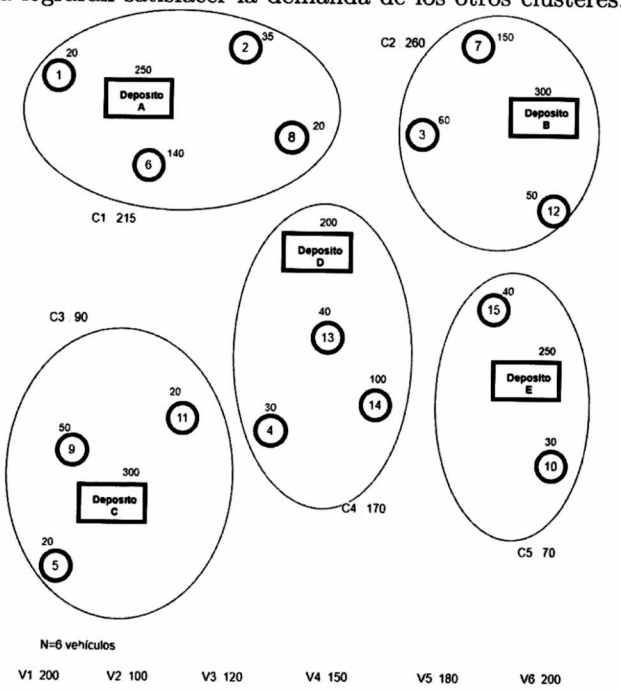

Figura 4. *Se obtuvieron 5 clústeres (igual al número de depósitos) C_1, C_2, C_3, C_4 y C_5 se muestran las demandas totales de cada clúster formado.*

Debe notarse que en cada paso del algoritmo la cardinalidad de $\mathcal{C}$ (que representa a la cantidad de clústeres) disminuye en una unidad garantizando de este modo la finalización del algoritmo. Sin embargo el algoritmo tiene algunas deficiencias, los cuales pasamos a detallar:

- El hecho de construir tantos clústeres como depósitos constituye una pérdida de generalidad, en casos donde hay pocos depósitos y muchos clientes podría definir algún cluster con muchos clientes y en consecuencia una demanda grande en comparación con la capacidad del vehículo.

- El haber construido clústeres sin considerar sus demandas podría derivar en la no existencia de un vehículo que satisfaga la demanda de algún cluster, tal como ocurre en la clusterización obtenida para la instancia mostrada en la figura 3

El último ejemplo mostrado en las figuras 3 y 4 demuestran que en general el algoritmo propuesto no funcionará, sin embargo tomando como base la idea de la clusterización, en la siguiente subsección, construimos los clústeres ya pensando también en las futuras rutas, es decir ya asignando (de manera eficiente) los vehículos disponibles, de tal manera que al final del proceso ya se tienen las rutas con sus respectivos vehículos asignados.

3.2. La heurística de clusterización por distancias y demandas (HDD)

Para evitar lo ocurrido con la heurística **HD**, construiremos los clústeres considerando no solo las distancias (cercanías) sino también las demandas y la asignación de vehículos a cada clúster, esta heurística a la que llamaremos **HDD** está compuesto de tres partes: Clusterización, Asignación óptima y TSP, cada una de ellas se describen en las siguientes subsecciones.

3.2.1. Clusterización

Para esta parte de la heurística definimos algunos términos y notaciones que serán utilizados en la heurística.

1. Se definen los conjuntos $\mathcal{C}_{temp}$: que contiene a los clústeres temporales aún en proceso de depuración y construcción y $\mathcal{C}_{def_i}$: que contiene a los clústeres definitivos.

2. Se dice que la construcción de clústeres temporales **colapsa** cuando se elige dos clústeres más cercanos (cada uno con vehículo asignado) y NO existe vehículo que pueda atender la demanda de la unión de dichos clústeres.

3. Si $V_1, V_2, \cdots, V_t$ son vehículos disponibles para ser asignados al clúster C (cuya demanda total es d_C) y $Q_{V_1}, Q_{V_2}, \cdots, Q_{V_t}$ son las cargas máximas de los respectivos vehículos. El vehículo V_k $(k = 1, 2, \cdots, t)$ se llama **mejor vehículo** si y solo si

$$Q_{V_k} - d_C = \min_{1 \leq i \leq t} \{Q_{V_i} - d_C \ / \ Q_{V_i} > d_C\}$$

   Por ejemplo si un clúster C tiene demanda total $d_C = 190$ y se tienen $t = 4$ vehículos disponibles de capacidades 200, 160, 250 y 300, el **mejor vehículo** es el de capacidad 200.

4. Dados dos clústeres con vehículos asignados (C_1, V_1) y (C_2, V_2) y demandas totales conocidas, diremos que (C_1, V_1) es **más saturado** que (C_2, V_2) si $Q_{V_1} - d_{C_1} \leq Q_{V_2} - d_{C_2}$.

5. El control de los vehículos asignados se lleva a cabo a través de la función $v : \mathcal{C} \to \{0, 1\}$, donde $\mathcal{C}$ es el conjunto actual de clústeres y para cada $C \in \mathcal{C}$

$$v(C) = \begin{cases} 1 & ; \quad C \text{ tiene vehículo asignado} \\ 0 & ; \quad \text{en caso contrario} \end{cases}$$

Con respecto a la heurística anterior han sido mejorados varios aspectos, los cuales mencionamos a continuación:

- Los clústeres serán construidos teniendo en cuenta las distancias pero también la posibilidad de asignárseles el **mejor vehículo**.

- La idea principal en la presente heurística es seleccionar los dos clústeres más cercanos, intentar unirlos en un solo clúster y asignarle el **mejor vehículo**. Este proceso repetitivo inevitablemente **colapsa** en la imposibilidad de unir dos clústeres más cercanos con vehículos ya asignados. Al ocurrir este caso se envía el clúster **más saturado** al conjunto $\mathcal{C}_{def_i}$ dejando al otro clúster en $\mathcal{C}_{temp}$, este proceso continúa hasta que solo queda un clúster en el conjunto $\mathcal{C}_{temp}$.

- Dependiendo del caso, un vehículo que previamente fue asignado a un clúster podría ser liberado posteriormente. Por ejemplo si el algoritmo selecciona dos clústeres C_i y C_j(más cercanos) cuyas demandas totales hasta el momento son 30 y 80 respectivamente y tienen asignados dos vehículos cuyas capacidades son 100 y 150 respectivamente, entonces los clústeres mencionados se unirán para constituir un solo clúster de capacidad $30 + 80 = 110$ al cual se le asigna el vehículo de capacidad 150 quedando libre el vehículo de capacidad 100 para que sea utilizado en otro cluster.

El algoritmo selecciona en cada paso los dos clústeres más cercanos C_i y C_j, y dependiendo de los vehículos asignados a dichos clústeres se analizan los siguientes tres casos:

1. **Ambos no tienen asignado un vehículo:** (esto ocurre en los primeros pasos), se escoge el **mejor vehículo** para cada uno o si es posible se une a ambos en un solo cluster y se le asigna el **mejor vehículo** disponible siempre que exista.

2. **Ambos tienen asignado un vehículo:** en este caso se intenta unir ambos clústeres en uno solo con el mejor vehículo escogido entre los dos vehículos que estan siendo usados o algún otro disponible. Cuando no es posible realizar la unión ocurre el **colapso**.

3. **Solo uno de los clústeres tiene un vehículo asignado:** en este caso se intenta aglutinar en el clúster con vehículo todo el contenido del otro clúster, en caso no fuera posible se busca el mejor vehículo disponible para el clúster sin vehículo.

El algoritmo termina cuando hay un solo clúster en el conjunto $\mathcal{C}_{temp}$ el cual pasa a formar parte del conjunto $\mathcal{C}_{def_i}$ quedando el conjunto de clústeres temporales vacío $(\mathcal{C}_{temp} = \phi)$, el cumplimiento de ésta condición está garantizado dado que en cada paso se eligen los dos clústeres más cercanos y se unen para formar otro cluster de mayor demanda y debido a la capacidad limitada de los vehículos asignados inevitablemente se llegará al **colapso** lo cual obliga a eliminar un clúster de $\mathcal{C}_{temp}$ y enviarlo al conjunto $\mathcal{C}_{def_i}$ permitiendo así que la cantidad de clústeres en $\mathcal{C}_{temp}$ disminuya sistemáticamente. Finalmente los clústeres obtenidos luego del proceso anterior deberán ser asignados de manera óptima a los depósitos para construir las rutas. Este procedimiento se detalla en la siguiente subsección.

Con todas las consideraciones anteriores mostramos el algoritmo:

Algoritmo 2 (HDD) Algoritmo de clusterización por distancias y demandas

Entrada: Clientes:(ubicación, demandas y cantidad). Depósitos:(ubicación, capacidad, cantidad). Vehículos:(capacidad y cantidad)

Salida: Conjunto de clústeres $\mathcal{C}$

$\mathcal{C}_{temp} = \{\{c_1\}, \{c_2\}, \cdots, \{c_n\}\}$
$\mathcal{C}_{defi} = \{\phi\}$
$v(C) = 0 \ \forall C \in \mathcal{C}_{temp}$
mientras $|\mathcal{C}_{temp}| > 1$ **hacer:**
  Hallar los índices i_{min} y j_{min} tales que $d(C_{i_{min}}, C_{j_{min}}) = min\{d(C_i, C_j) \ / \ C_i \ , C_j \in \mathcal{C}_{temp}\}$
  si $(v(C_{i_{min}}) + v(C_{j_{min}}) = 0)$ **entonces**
    Juntarlos en un solo clúster y asignarle el mejor vehículo en caso se pueda o asignarle a cada clúster un mejor vehículo
  sino
    si $(v(C_{i_{min}}) + v(C_{j_{min}}) = 2)$ **entonces**
      si ocurre **colapso entonces**
        $C =$ más saturado entre $C_{i_{min}}$ o $C_{j_{min}}$
        $\mathcal{C}_{temp} = \mathcal{C}_{temp} \setminus \{C\}$
        $\mathcal{C}_{defi} = \mathcal{C}_{defi} \cup \{C\}$
      sino
        $C = C_{i_{min}} \cup C_{j_{min}}$
        Asignar el mejor vehículo disponible a C y liberar el otro vehículo
        $\mathcal{C}_{temp} = (\mathcal{C}_{temp} \setminus \{C_{i_{min}}, C_{j_{min}}\}) \cup \{C\}$
      fin de si
    sino
      si $(v(C_{i_{min}}) + v(C_{j_{min}}) = 1)$ **entonces**
        $C_1 =$ clúster con vehículo asignado
        $C_0 =$ clúster sin vehículo asignado
        si C_0 puede aglutinarse en C_1 **entonces**
          $C_1 = C_1 \cup C_0$
          $\mathcal{C}_{temp} = \mathcal{C}_{temp} \setminus \{C_0\}$
        sino
          Asignar el mejor vehículo disponible a C_0
        fin de si
      fin de si
    fin de si
  fin de si
fin de mientras
$\mathcal{C} = \mathcal{C}_{temp} \cup \mathcal{C}_{defi}$
retorna $\mathcal{C}$

3.2.2. Asignación óptima de clústeres a depósitos

Luego de obtener los clústeres con vehículos ya asignados y teniendo toda la información respecto a los depósitos, a continuación debemos asignar los clústeres obtenidos a los depósitos. Eventualmente algunos depósitos podrían quedar sin ningún clúster asignado o un depósito podría albergar a más de un clúster siempre que su capacidad máxima no haya sido alcanzada. Para resolver éste subproblema, planteamos un problema de optimización binaria (dado que la variable de decisión solo puede tomar el valor de 0 o 1), para ello consideremos los siguientes parámetros:

p : cantidad de clústeres obtenidos
m : cantidad de depósitos
$i = 1; 2; \cdots; m$ (índice de depósitos)
$j = 1; 2; \cdots; p$ (índice de clústeres)
M_{ij} : representa la distancia del depósito i al cluster j
d_{C_j} : es la demanda total del clúster j
D_i : es la capacidad máxima del depósito i
NVD : es la cantidad de vehículos en cada depósito
Consideramos la variable de decisión:

$$x_{ij} = \begin{cases} 1 & ; \quad \text{si el cluster } j \text{ es asignado al depósito } i \\ 0 & ; \quad \text{en otro caso} \end{cases}$$

La función objetivo para este subproblema consistirá en minimizar la distancia entre clústeres asignados y los depósitos, es decir,

$$\text{mín} \left\{ \sum_{j=1}^{p} \sum_{i=1}^{m} M_{ij} x_{ij} \right\}$$

sujeta a las restricciones:

$$\sum_{j=1}^{p} d_{C_j} x_{ij} \leq D_i \quad \forall \, i = 1; 2; \cdots; m \tag{9}$$

Esta restricción asegura que la suma de demandas de los clústeres asignados al depósito i no superan su capacidad.

$$\sum_{i=1}^{m} x_{ij} = 1 \quad \forall \, j = 1; 2; \cdots; p \tag{10}$$

Esta restricción garantiza que cada clúster es asignado a un solo depósito.

$$\sum_{j=1}^{p} x_{ij} \leq NVD \quad \forall \, i = 1; 2; \cdots; m \tag{11}$$

Esta restricción garantiza que la cantidad de clústeres asignados a cada depósito es menor o igual a la cantidad de vehículos en dicho depósito.

$$x_{ij} \in \{0; 1\} \tag{12}$$

Restricción que expresa el tipo de variable.

Esta parte de la heurística se implementa en el lenguaje de programación JULIA 1.0.5, dado que en general el tamaño del problema (expresado en términos de cantidad de clústeres y depósitos) es relativamente pequeña, respecto al tamaño inicial del problema.

Para la misma instancia de prueba mostrada en la figura 3, se puede observar en este caso la obtención de 6 clusteres (color rojo) con el mejor vehículo ya asignado vea la figura (5). A continuación la solución del problema de asignación óptima de clústeres a depósitos permite asignar a cada clúster un único depósito, obteniendo clústeres de mayor tamaño (a los que llamaremos superclústeres) que incluyen a los clústeres asignados a un determinado depósito y también al depósito. Estos superclusteres (que agrupan a varios clústeres en torno

a un mismo depósito asignado) están representados en la figura (5) de color azul, existen tantos superclústeres como depósitos.

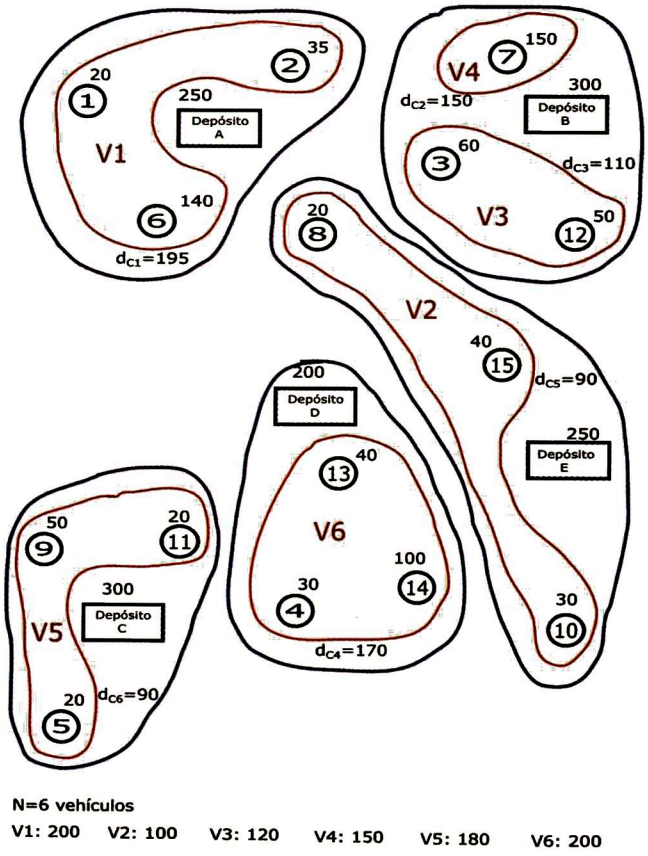

Figura 5. *Los clústeres de color rojo son los 6 originales de acuerdo a las distancias y demandas además cada clúster ya tiene asignado un vehículo, mientras que los superclústeres de color azul en general aglutinan a varios clústeres y sus depósitos asignados, eventualmente algún depósito podría quedar sin clústeres asignados.*

3.2.3. TSP para definir las rutas

Luego de culminar el proceso de clusterización y la asignación de cada uno de los clústeres a los depósitos, se construyen las rutas resolviendo para cada cluster y su depósito asignado un problema del agente viajero TSP, definiendo la ruta de menor distancia, obteniendo así un conjunto de rutas que constituye una solución inicial factible. La heurística está diseñada para ser aplicada a un problema MDVRP con flota mixta y con restricciones de capacidad máxima para los depósitos. Sin embargo con el objetivo de realizar las pruebas para las instancias definidas en el marco del presente trabajo `https://github.com/fboliveira/MDVRP-Instances/` se considera la misma cantidad de vehículos en cada depósito y todos los vehículos con la misma capacidad es decir flota homogénea. Resumimos la heurística **HDD** simbólicamente en la expresión:

$$\text{HDD} = \text{Clusterización} + \text{Asignación Óptima} + \text{TSP}$$

Esta heurística ha sido implementada en el lenguaje de programación JULIA 1.0.5 y se usará una instancia creada manualmente de tamaño pequeño (15 clientes y 5 depósitos) con el fin de obtener también la solución exacta para efectos de comparación, se muestran además las ubicaciones iniciales de los clientes y depósitos, el resultado de la clusterización por distancias y demandas y la solución exacta para la instancia creada llamada **DataPrueba**

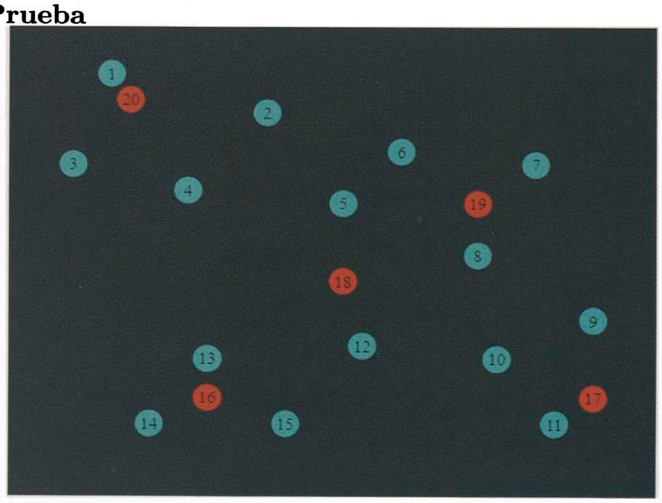

Figura 6. *Nube de puntos correspondiente a la instancia manual* **DataPrueba** *con 5 depósitos (de color rojo y enumerados de 16 a 20), 15 clientes (de color turquesa y enumerados de 1 a 15) y dos vehículos cuya carga máxima es 60 en cada depósito.*

Al aplicar la heurística por distancias y demandas (HDD) desarrollada en esta sección a la instancia **DataPrueba** se obtiene como resultado la solución que se muestra en la siguiente figura.

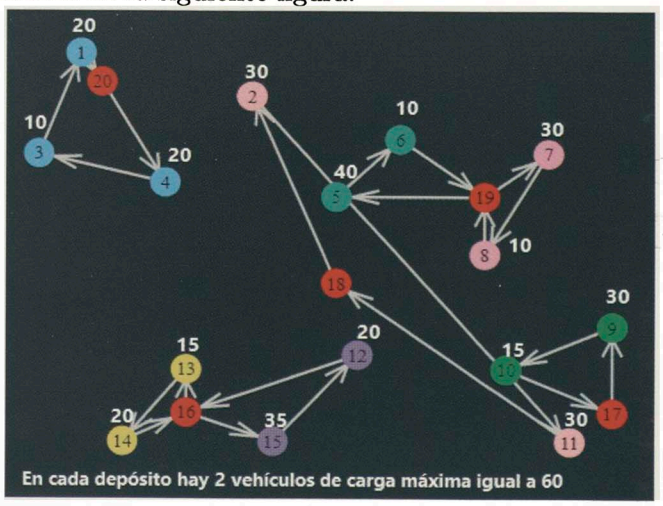

Figura 7. *Resultados obtenidos por la aplicación de la heurística HDD a la instancia* **DataPrueba**, *los conjuntos de nodos con el mismo color representan nodos del mismo clúster cuyas demandas serán satisfechas por un vehículo de la flota, los depósitos estan representados con el color rojo, el número mostrado en la parte superior de cada nodo cliente representa su demanda. La función objetivo que resulta de la aplicación de la heurística proporciona el valor* **165.15** *utilizando 7 vehículos (de la flota de 10 vehículos).*

Se muestra a continuación para la misma instancia la solución exacta (por implementación del modelo matemático en JULIA 1.0.5) se observa similitud en alguno de los clústeres obtenidos entre la solución por heurística

y la solución exacta.

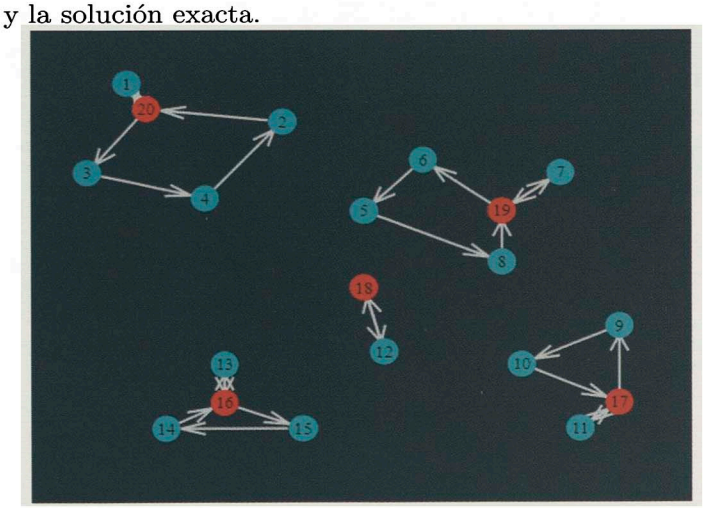

Figura 8. *Solución exacta para la misma instancia, los depósitos estan representados de color rojo y sus respectivos clusteres de color turquesa alrededor del depósito, el valor objetivo óptimo es* **116.70** *en este caso se usan 9 vehículos (del total de 10 vehículos).*

Finalmente elaboramos una tabla comparativa con el desempeño de la heurística **HDD** para la instancia creada **DataPrueba** la cual tiene $m = 5$ depósitos, $n = 15$ clientes, así mismo otras características de la instancia son la existencia de 2 vehículos por depósito, es decir $TV = 10$ (tamaño de flota) y para ésta instancia en particular (tamaño pequeño) es posible hallar la solución exacta, la cual es 116,70 hallada en 6.33 segundos. Todos estos datos y resultados obtenidos por la ejecución de las heurísticas y el modelo exacto son resumidas en la tabla 1

Método	costo	Nro de Vehículos	tiempo(seg)
solución exacta	116.70	9	6.33
Heurística HDD	165.15	7	1.09

Cuadro 1. Resultados obtenidos para la solución exacta.

3.3. Heurísticas de Mejora (HM)

Luego de obtener una solución inicial factible mediante la heurística de construcción descrita anteriormente, se aplicará a la solución obtenida otra heurística llamada de mejora, dentro de éstas se han considerado las siguientes estrategias:

- **Intercambio de nodos** Consiste en trabajar en clústeres cercanos e intercambiar nodos con el objetivo de mejorar la función objetivo
- **Partición de un clúster** Luego de detectar un clúster con mucha variabilidad o irregularidad (por ejemplo es geométricamente muy grande o cruza a otros clústeres) se divide el cluster en dos o mas partes dentro de las condiciones de factibilidad.

La aplicación sucesiva de estas dos heurísticas permiten mejorar la función objetivo hasta determinado límite, se ha registrado en las tablas la cantidad de mejoras realizadas y el tiempo adicional utilizado. Para una descripción algorítmica del proceso de mejora consideramos las siguientes observaciones:

- $\mathcal{S}$ es el conjunto de soluciones factibles para el problema de optimización planteado, es decir,

  $\mathcal{S} = \{s \ / \ s \text{ es factible para el problema planteado}\}$

- La función $f : S \longrightarrow \mathbb{R}^+$ definida mediante $f(s) =$ valor objetivo asociada a la solución s
- Consideramos ya implementado el procedimiento **Mejora** que recibe una solución factible $s \in S$ y retorna una solución mejorada s_m (también factible), en el sentido siguiente: luego de aplicar

  $$s_m = \textbf{Mejora}(s)$$

  se cumple

  $$f(s_m) \leq f(s)$$

Donde se verifica $f(s_m) = f(s)$ cuando ya no se puede mejorar la solución s. En caso que $f(s_m) < f(s)$ se ha logrado mejorar la función objetivo, describimos a continuación el algoritmo de mejoras sucesivas

Algoritmo 3 (**HM**) Algoritmo de Mejoras sucesivas

Entrada: s_0: Solución inicial dada por la heurística de construcción **HDD**
Salida: s_1: Mejor solución

1: $s_1 = \textbf{Mejora}(s_0)$
2: **mientras** $f(s_1) \neq f(s_0)$ **hacer:**
3: $\quad s_0 = s_1$
4: $\quad s_1 = \textbf{Mejora}(s_0)$
5: **fin de mientras**
6: **retorna** s_1

En la siguiente figura se muestra la secuencia de mejoras sucesivas aplicadas a la instancia p01, cuya solución factible inicial se muestra en la siguiente figura, en la que se señala el cluster de color amarillo que será modificado localmente.

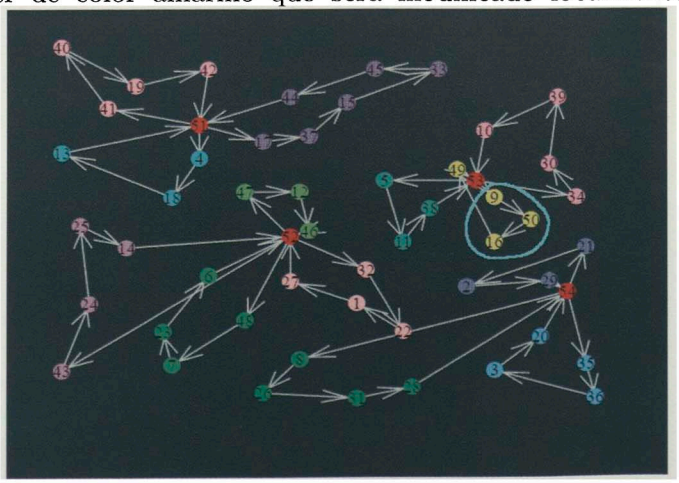

Figura 9. *Instancia p01 con 50 clientes, 4 depósitos, 4 vehículos por depósito c/u con capacidad 80. Tiempo=0.86 seg y Valor objetivo=643.699.*

A continuación se muestra una aplicación de la heurística de mejora en el cluster señalado, se observa que

los clientes 16 y 50 que originalmente pertenecían al cluster de color amarillo fueron reasignados a otros clústeres cercanos, mejorando de esta manera la función objetivo.

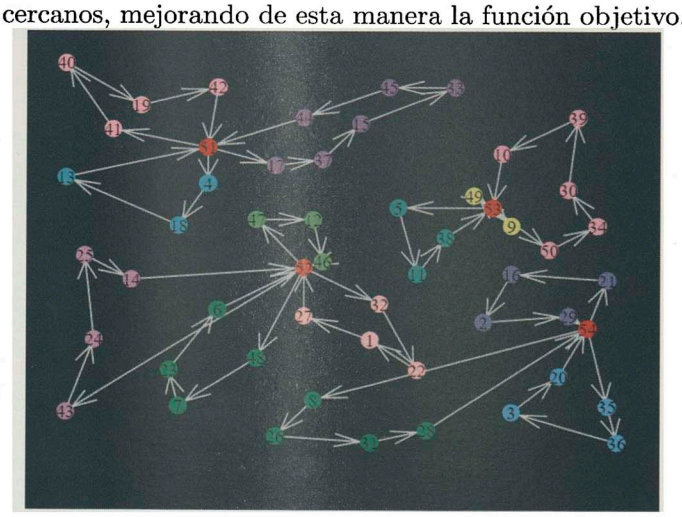

Figura 10. *Valor objetivo mejorado: 633.237 .*

Al aplicar sucesivamente las heurísticas de mejora, hasta que la función objetivo ya no se pueda mejorar, se muestra la secuencia de mejoras.

Mejora	Valor objetivo
0	643.69
1	633.23
2	627.78
3	624.55
4	614.18
5	610.92
6	603.78
7	588.49
8	588.49

Cuadro 2. Secuencia de mejoras aplicada a la instancia p01, desde la solución inicial s_0 obtenida por la heurística de construcción HDD (cuyo valor fue 643.69) hasta la solución mejorada s_1 (por aplicación sucesiva de las heurísticas de mejora) cuyo mejor valor es 588.49

Obteniendo finalmente la siguiente configuración, llamado también mejor solución para la instancia p01

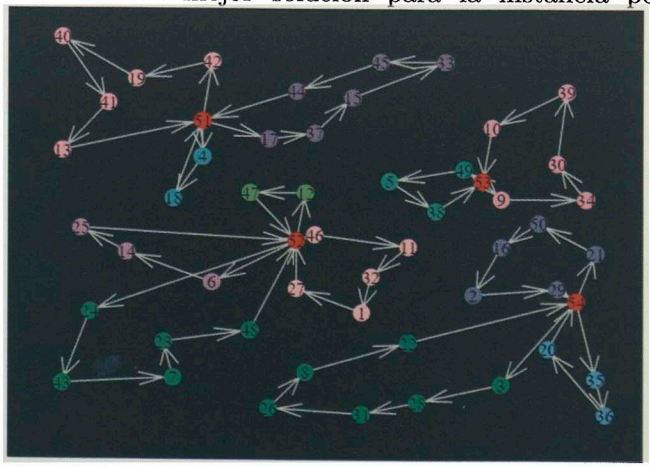

Figura 11. *Valor objetivo mejorado al máximo: 588.495.*

Para la instancia p01 en particular se ha hallado la solución exacta (al 27.2% de gap y limitada a casi 5 horas), obteniendo la siguiente configuración.

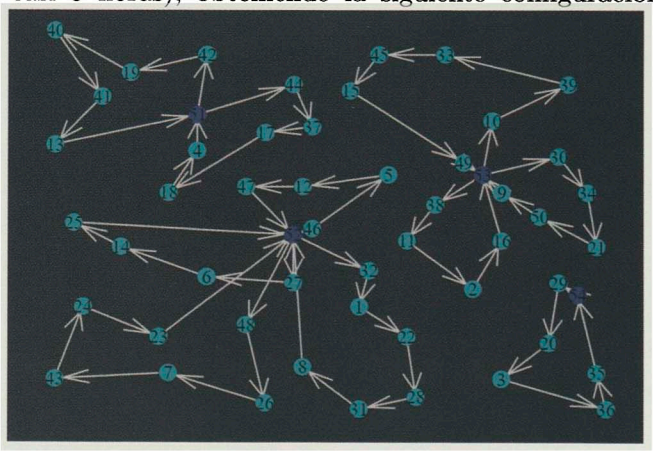

Figura 12. *Instancia p01. Tiempo=17881 seg=4.96 horas, Mejor Valor objetivo=584.43 (27.2 % de gap), Mejor conocida=576.87.*

En la siguiente sección se analiza cada una de las instancias existentes en la literatura (desde p01 hasta p17)

4. Resultados y conclusiones

4.1. Resultados para la heurística HDD

Se muestra a continuación los resultados obtenidos al aplicar la heurística a diferentes instancias existentes en la literatura.

Instancias		MEJOR SOLUC.	Soluc. Inicial por **HDD**		Solución Mejorada **HDD+HM**	
Nro	N	BKS	dist.	t_{ini}	dist.	t_{acum}
p01	50	576.87	643.69	0.85	588.49	10.54
p02	50	473.53	630.10	0.86	585.87	14.90
p03	75	641.19	720.27	0.93	689.38	10.74
p04	100	1001.59	1232.25	0.70	1145.16	45.16
p05	100	750.03	996.45	0.74	831.44	139.86
p06	100	876.50	1119.44	0.73	1017.71	27.29
p07	100	881.97	1137.23	0.71	997.12	54.32
p08	249	4372.78	5727.57	0.90	5181.44	564.37
p09	249	3858.66	4647.73	0.90	4346.48	391.33
p10	249	3631.11	4602.74	0.92	4202.72	241.26
p11	249	3546.06	4393.01	0.91	4154.11	790.99
p12	80	1318.95	2707.42	0.79	1770.32	131.23
p13	80	1318.95	2707.42	0.77	1770.32	78.14
p14	80	1360.12	2707.42	0.78	1770.32	92.28
p15	160	2505.42	5008.94	0.91	4430.36	108.80
p16	160	2572.23	5008.94	0.89	4430.36	89.24
p17	160	2709.09	5008.94	0.88	4430.36	44.93

Cuadro 3. Solución factible inicial obtenida por la heurística de construcción **HDD** basada en clusterización y la aplicación sucesiva de las heurísticas de mejora **HM**.

En la tabla anterior se muestran las características de cada una de las 17 instancias(existen 23 en la literatura), como son cantidad de clientes(N), cantidad de depósitos(M), cantidad de vehículos en cada depósito(K) y la carga máxima de cada vehículo(Q). Asimismo para cada

instancia se consigna la mejor solución conocida hasta el momento o BKS (Best Know Solution). Para cada instancia y respecto a la solución inicial tenemos registrada la solución obtenida por aplicación de la heurística en cuanto a distancia total del ruteo y tiempo de cómputo. Respecto a la solución mejorada tenemos registrado la mejor distancia obtenida, el tiempo de cómputo acumulado que incluye al tiempo empleado en hallar la solución inicial. En la tabla anterior se muestra la mejor solución conocida (BKS) hasta el momento, la distancia total obtenida por la solución, el tiempo que le toma a la heurística de construcción y el tiempo total que incluye el tiempo utilizado para hallar la solución inicial. El desempeño promedio se muestra en la siguiente gráfica donde se observa la evolución de la heurística desde la solución inicial hasta la mejora sucesiva.

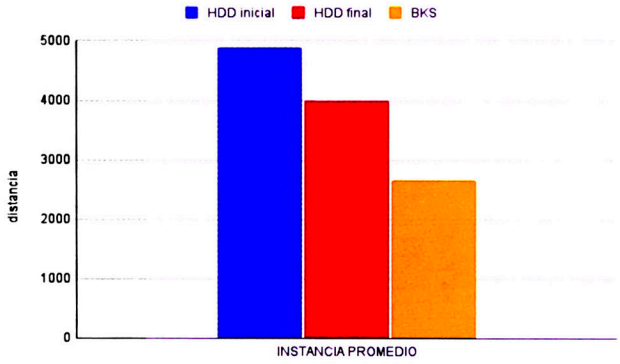

Figura 13. *Desempeño promedio de la heurística de clusterización por distancias y demandas* **HDD**.

La gráfica muestra la solución inicial promedio (azul) la mejora promedio (rojo) y la mejor solución conocida (amarillo)

4.2. Conclusiones y recomendaciones

La heurística **HD** no funciona en general sin embargo constituye la base para la elaboración de la heurística **HDD**, en promedio la heurística **HDD** (incluida su mejora) aún se puede perfeccionar dado que la brecha respecto al BKS es posible disminuirla, escencialmente el trabajo de mejora se debe enfocar en el proceso de clusterización dado que las otras dos componentes de la heurística (asignación óptima y TSP ya son óptimos). La solución exacta solo se puede hallar para tamaños muy pequeños ($n \leq 30$ clientes).

Las heurísticas de mejora utilizadas en el presente trabajo **vecino más cercano** y **spliting** permiten mejorar la función objetivo a partir de una solución inicial pero consumen un tiempo adicional.

El lenguaje de programación JULIA (de libre distribución) es una herramienta fundamental en la implementación de las heurísticas del problema de investigación planteado en el presente trabajo. Aún es posible mejorar el proceso de clusterización con otras estrategias mas específicas como $k-$means, quedando aún pendiente la propuesta de Metaheurísticas como son los algoritmos genéticos o la búsqueda tabú.

Finalmente también está pendiente la propuesta de otras estrategias de clusterización que mejoran los resultados obtenidos, estas serán presentadas en un posterior trabajo.

1. Surekha, Paneerselvam and Sumathi, Sai,World Applied Programming,1,3,118–131,2011.

2. J.K. Lenstra and A.H.G. Kan, Networks, 11, 2, 221–227, 1981, Wiley Online Library.

3. Cormen, Thomas H and Leiserson, Charles E and Rivest, Ronald L and Stein, Clifford, *Introduction to algorithms*, 2009, MIT press.

4. Pichpibul, Tantikorn and Kawtummachai, Ruengsak, ScienceAsia, 38, 3,307–318, 2012.

5. Shi, Yanjun and Lv, Lingling and Hu, Fanyi and Han, Qiaomei, Applied Sciences, 10, 7, 2403, 2020, Multidisciplinary Digital Publishing Institute.

6. Stodola, Petr, Algorithms, 11, 5, 74, 2018, Multidisciplinary Digital Publishing Institute.