CAPÍTULO 5
METODOLOGÍA APLICADA AL PROBLEMA

En este capítulo se describen las características del algoritmo GRASP propuesto para generar soluciones para el PLIDMC.

GRASP es un heurístico iterativo. Cada iteración está compuesta por una fase de construcción y una fase de mejora. En la primera fase se selecciona un conjunto de ubicaciones para plantas y almacenes, de tal manera que se cumplan todas las restricciones. En la segunda fase se aplica un procedimiento de búsqueda local, para mejorar la solución obtenida en la primera fase. Se exploran tres entornos: 1) intercambio entre las instalaciones seleccionadas y las no seleccionadas, 2) agregar instalaciones a los conjuntos ya seleccionados y 3) eliminar instalaciones de las ya seleccionadas. A continuación se describen las dos fases del algoritmo.

FASE DE CONSTRUCCIÓN

Como una estrategia para generar soluciones factibles de una manera sencilla, se estructura el problema en dos partes. La estrategia propuesta convierte el PLIDMC en dos subproblemas de un sólo nivel.

En el primer subproblema se consideran el conjunto de ubicaciones posibles para los almacenes, el conjunto de clientes con su respectiva demanda de cada uno de los productos, los costos de distribución desde los almacenes hasta los clientes, así como los costos por operación de los almacenes. El objetivo es seleccionar un conjunto de ubicaciones para los almacenes, de tal manera que toda la demanda de los clientes esté asignada a alguno de los almacenes seleccionados. Además, toda la demanda de un producto por parte de un cliente

debe ser asignada a un solo almacén, es decir, el primer subproblema es un problema de localización de almacenes con asignación única, al cual denominaremos subproblema *almacenes-clientes*.

Una vez que se encuentre una solución factible para abastecer toda la demanda de los clientes desde un conjunto de ubicaciones para los almacenes, se requiere determinar el conjunto de ubicaciones para las plantas. Este segundo subproblema es muy similar al subproblema *almacenes-clientes*, ya que, en este caso, cada almacén puede ser considerado como un cliente cuya demanda es igual a la demanda de todos los clientes que será satisfecha desde dicho almacén. En este sentido la cantidad demandada por un almacén debe de ser satisfecha desde las plantas. La única diferencia entre este subproblema y el subproblema *almacenes-clientes* es la manera en que se asigna la demanda de los almacenes a las plantas, ya que en este caso se permite la asignación múltiple. Es decir, la cantidad demandada de cualquier producto desde cada uno de los almacenes puede ser satisfecha desde varias plantas. Por lo tanto, el segundo subproblema considera el conjunto de ubicaciones para los almacenes seleccionados en el subproblema *almacenes-clientes*, el conjunto de ubicaciones potenciales para las plantas, la demanda asignada a cada uno de los almacenes, los costos de distribución de los productos desde las plantas hacia los almacenes, así como los costos de operación de las plantas. Como la demanda que se debe de satisfacer en el segundo subproblema es la de los almacenes, se selecciona, del conjunto de ubicaciones potenciales para las plantas, un subconjunto de ubicaciones, de tal manera que se cuente con la capacidad suficiente para satisfacer las cantidades de producto demandas por los almacenes. El segundo subproblema es llamado subproblema *plantas-almacenes*. Una vez que se resuelven ambos subproblemas, también se obtiene una solución factible para el PLIDMC.

A continuación, se describe el método utilizado para obtener soluciones factibles de ambos subproblemas.

Para facilitar la lectura, se muestran en la tabla 3 los conjuntos y parámetros utilizados en esta parte del proceso.

TABLA 1 NOTACION UTILIZADA EN EL MODELO MATEMATICO

I	Conjunto de índices para los clientes.
J	Conjunto de índices para los almacenes.
K	Conjunto de índices para las plantas.
L	Conjunto de índices para los productos.
c_{ijl}	Costo variable por distribuir una unidad del producto *l* del almacén *j* al cliente *i*.
d_{jkl}	Costo variable por transportar una unidad del producto *l* desde la planta *k* al almacén *j*, además incluye producir el producto *l* en la planta *k*.
f_k	Costo fijo de apertura y operación de una planta en la ubicación *k*.
g_j	Costo fijo de apertura y operación de un almacén en la localización *j*.
a_{il}	Demanda requerida del producto *l* para el cliente *i.*
V_j	Capacidad máxima del almacén en la ubicación *j*.
D_k	Capacidad máxima de la planta en la ubicación *k*.

Una solución factible σ para PLIDMC se puede representar de la siguiente manera,

$$\sigma = \left(\hat{J}, \hat{K}, a_1, a_2\right)$$

donde $\hat{J}$ es el conjunto de ubicaciones seleccionadas para los almacenes, $\hat{K}$ denota el conjunto de ubicaciones seleccionadas para las plantas, $a_1 : I \times L \to J$, tal que $a_1(i,l) = j$ si toda la demanda del cliente i para el producto l se satisface por el almacén j y $a_2 : J \times K \times L \to \mathbb{R}$, tal que $a_2(j,k,l) = r$, si r unidades del producto l son surtidos desde la planta k hacia el almacén j.

SUBPROBLEMA ALMACENES-CLIENTES

Como se ha descrito antes, el subproblema *almacenes-clientes* selecciona un conjunto de ubicaciones para los almacenes de manera tal que se satisfaga la demanda de los clientes utilizando un heurístico voraz aleatorizado. Es un procedimiento iterativo que parte de una solución vacía, y en cada iteración se añade una ubicación al conjunto de ubicaciones seleccionadas para el emplazamiento de los almacenes. Asimismo, a cada una de las ubicaciones seleccionadas para los almacenes se le asigna la demanda que será distribuida desde dicha ubicación, verificando que no se exceda la capacidad del almacén. Para ello, se utiliza una función voraz que evalúa la conveniencia de añadir dicha ubicación al conjunto de ubicaciones para los almacenes que operarán en el sistema de distribución.

La función voraz, $\varphi_j, \forall j \in J \backslash \hat{J}$, mide el costo por unidad de producto asignado al almacén j, en el caso de que dicho almacén fuera a ser seleccionado. Esto se hace intentando asignar la demanda de los productos con menor costo de distribución desde el almacén j hacia los clientes. Para obtener el valor de φ_j , primero se construye una lista, $Lista_j$, de la demanda que no sido asignada previamente a ningún almacén y que podría ser asignada al almacén. Cada elemento de dicha lista es un par $(i,l), i \in I, l \in L$ e indica que la cantidad demanda del producto l por el cliente i puede ser satisfecha en su totalidad por un almacén en la ubicación j. Los elementos de dicha lista se procesan en orden no decreciente con respecto a los costos de distribución (c_{ijl}) y se agregan a una segunda lista CP_j, que contiene la demanda que se asignará al almacén en caso de que sea seleccionado. Sea h_j la capacidad disponible de la ubicación j . Inicialmente $h_j \leftarrow V_j$. Para cada par (i,l) primero se evalúa si aún queda capacidad disponible en la ubicación j , es decir, si $a_{il} \leq h_j$. En ese caso, se agrega el par (i,l) a CP_j y se actualiza la capacidad disponible de la ubicación j ,

es decir, $h_j \leftarrow h_j - a_{il}$, En caso contrario se continúa con el siguiente elemento de la lista $Lista_j$, y así sucesivamente hasta haber procesado todos sus elementos.

El valor de la función voraz φ_j es igual al cociente entre el costo total (costos fijos de operación más costos de distribución) y el total de unidades de productos que se distribuirán a los clientes desde dicha ubicación. De esta manera se obtiene el costo por unidad de producto asignado al almacén j. Por tanto, la función voraz (φ_j), que evalúa el costo por unidad de producto asignado a la ubicación j es,

$$\varphi_j = \frac{g_j + \sum_{(i,l)\in CP_j} c_{ijl}\, a_{il}}{\sum_{(i,l)\in CP_j} a_{il}} \qquad \forall j \notin \hat{J}$$

La tabla 4 describe la notación utilizada en esta parte de la fase de construcción de la solución inicial factible.

TABLA 2 NOTACION UTILIZADA EN LA PARTE 1 DE LA FASE DE CONSTRUCCION

h_j	Capacidad disponible de la ubicación j.
$Lista_j$	Demanda que no ha sido asignada a algún almacén, y puede ser asignada al almacén j.
CP_j	Demanda que será asignada al almacén j en caso de que se seleccione.
$\hat{J}$	Conjunto de ubicaciones seleccionadas para los almacenes.
a_1	Conjunto de la demanda asignada a los almacenes seleccionados.
α	Parámetro de aleatoriedad.

Para generar la *Lista Restringida de Candidatos* (LRC) se calcula el valor umbral u de la siguiente manera. Sea $\varphi_{jmín} = mín_{j\in J\setminus\hat{J}}\{\varphi_j\}$ el valor más pequeño de la función voraz para las ubicaciones candidato, $\varphi_{jmáx} = máx_{j\in J\setminus\hat{J}}\{\varphi_j\}$ el valor más grande para las ubicaciones candidato y $\alpha \in [0,1]$ un parámetro que mide el grado de voracidad o aleatoriedad del algoritmo voraz. Entonces, se calcula el valor

umbral $u = \varphi_{jmín} + \alpha\left(\varphi_{j\max} - \varphi_{j\min}\right)$ y se agregan a la lista restringida de candidatos los índices de todas aquellas ubicaciones cuyo valor de la función voraz es menor o igual al valor umbral, es decir, $LCR = \left\{j : \varphi_j \le u\right\}$. Posteriormente se elige al azar una ubicación j_s de LCR y se añade al conjunto de ubicaciones seleccionadas para los almacenes, $\hat{J} := \hat{J} \cup \left\{j_s\right\}$. Asimismo, se actualiza la asignación de productos al almacén j_s, es decir, $a_1(i,l) = j_s, \forall (i,l) \in CP_{j_s}$. El pseudocódigo para el subproblema almacenes-clientes se muestra en la ilustración 7.

$\hat{J} \leftarrow \emptyset$
Mientras ($|a_1| < |I| \times |L|$)
Obtener $CP_j, \forall j \in J \backslash \hat{J}$

$$\varphi_j \leftarrow \frac{\left(g_j + \sum_{(i \ , \ l) \in CP_j} c_{ijl} a_{il}\right)}{\sum_{(i \ , \ l) \in CP_j} a_{il}}, \forall j \in J \backslash \hat{J}$$

Fin para
$\varphi_{mín} \leftarrow mín_{j \in J \backslash \hat{J}} \left\{\varphi_j\right\}$
$\varphi_{máx} \leftarrow máx_{j \in J \backslash \hat{J}} \left\{\varphi_j\right\}$
$u \leftarrow \varphi_{mín} + \alpha\left(\varphi_{máx} - \varphi_{mín}\right)$
$LCR = \left\{j : \varphi_j \le u\right\}$
Seleccionar j_s aleatoriamente de LRC
$\hat{J} \leftarrow \hat{J} \cup \left\{j_s\right\}$
$a_1(i,l) = j_s, \forall (i,l) \in CP_j$
Fin mientras

ILUSTRACIÓN 1 PSEUDOCÓDIGO DEL ALGORITMO VORAZ PARA LA SELECCIÓN DE UBICACIONES PARA LOS ALMACENES

Al final de procedimiento voraz, tenemos un conjunto de ubicaciones para los almacenes y una asignación factible de la demanda de todos los clientes a los almacenes seleccionados.

SUBPROBLEMA PLANTAS-ALMACENES

Como se ha descrito antes, el subproblema *plantas-almacenes* selecciona un conjunto de ubicaciones para las plantas, de manera tal que se satisfaga la demanda del subconjunto de almacenes seleccionados en el subproblema *almacenes-clientes,* utilizando un heurístico voraz aleatorizado. Es un procedimiento iterativo que parte de una solución vacía, y en cada iteración se añade una ubicación al conjunto de ubicaciones seleccionadas para el emplazamiento de las plantas. Asimismo, a cada una de las ubicaciones seleccionadas para las plantas se le asigna la demanda que será distribuida desde dicha ubicación, verificando que no se exceda la capacidad de la planta. Para ello, se utiliza una función voraz que evalúa la conveniencia de añadir dicha ubicación al conjunto de ubicaciones para las plantas que operarán en el sistema de distribución.

La función voraz, $\varphi_k, \forall k \in K \backslash \widehat{K}$, mide el costo por unidad de producto asignado la planta k, en el caso de que dicha planta fuera a ser seleccionada. Esto se hace intentando asignar la demanda de los productos con menor costo de distribución desde la planta k hacia los almacenes. Para obtener el valor de φ_k, primero se construye una lista, $Lista_k$, de la demanda que no sido asignada previamente a ninguna planta y que podría ser asignada a la planta. Cada elemento de dicha lista es un par $(j,l), j \in \hat{J}, l \in L$ e indica que la cantidad demandada del producto l por el almacén j puede ser satisfecha, ya sea en parte o en su totalidad, por una planta en la ubicada en k. Los elementos de dicha lista se procesan en orden no decreciente con respecto a los costos de distribución (d_{jkl}) y se agregan a una segunda lista AP_k, que contiene la demanda que se asignaría a la planta en caso de que dicha planta fuera seleccionada. Cada elemento de la lista AP_k estará formado por una terna de la forma (j, l, m_{jl}) donde m_{jl} denota el número de unidades del producto l demandadas por el almacén j que serán distribuidas desde la planta k. Sea t_k la capacidad disponible de la ubicación k y s_{jl} la cantidad del producto l demandada por el almacén j que aún

no ha sido asignada a ninguna planta. Inicialmente $t_k \leftarrow D_k$, $s_{jl} \leftarrow \sum_{(i,l)\in CP_j} a_{il}$ y $p \leftarrow \sum_{j\in J} \sum_{(i,l)\in CP_j} a_{il}$. Para cada par (j,l) de la lista, $Lista_k$, primero se evalúa la cantidad disponible del producto l demandada por el almacén j que puede ser distribuida desde la planta k, es decir, si $t_k > 0$, entonces $m_{jl} \leftarrow mín\{s_{jl}, t_k\}$. En ese caso, se agrega la terna (j,l,m_{jl}) a AP_k. Asimismo, se actualiza la capacidad disponible de la ubicación k, es decir, $t_k \leftarrow t_k - m_{jl}$ y las unidades del producto l demandadas por el almacén j que aún no han sido asignada a ninguna planta, es decir, $s_{jl} \leftarrow s_{jl} - m_{jl}$. Si $s_{jl} = 0$, entonces se continúa con el siguiente elemento de la lista $Lista_k$. En caso contrario se habrá agotado la capacidad de la planta k. Toda la demanda de los almacenes seleccionados en el subproblema *almacenes-clientes* debe de estar cubierta sin exceder las capacidades de las plantas. Además en esta parte no existe asignación única, es decir, el mismo producto puede ser entregado por más de una planta a un almacén.

El valor de la función voraz φ_k es igual al cociente entre el costo total (costos fijos de operación más costos de distribución) y el total de unidades de productos que se distribuirán a los almacenes desde dicha ubicación. De esta manera se obtiene el costo por unidad de producto asignado a la planta k. Por tanto, la función voraz (φ_k), que evalúa el costo por unidad de producto asignado a la ubicación k es,

$$\varphi_k = \frac{f_k + \sum_{(j,l)\in AP_k} d_{jkl}\, m_{jl}}{\sum_{(j,l)\in AP_k} m_{jl}} \qquad \forall k \notin \widehat{K}$$

La tabla 5 describe la notación utilizada en esta parte de la fase de construcción de la solución inicial factible.

TABLA 3 NOTACIÓN UTILIZADA EN LA PARTE 2 DE LA FASE CONSTRUCTIVA

$List_k$	Demanda que no ha sido asignada previamente a alguna planta y podría ser asignada a la planta *k*.
AP_k	Demanda que será asignada a la planta *k,* en caso de que sea seleccionada.
m_{jl}	Número de unidades del producto *l* demandadas por almacén *j* que serán distribuidas desde la planta *k*.
s_{jl}	Cantidad de producto *l* demanda por el almacén *j* que aún no ha sido asignada a alguna planta.
p	Cantidad de demanda que aún no ha sido asignada a alguna planta.
t_k	Capacidad disponible de la planta *k*.
α	Parámetro de aleatoriedad.
$\widehat{K}$	Conjunto de ubicaciones seleccionadas para las plantas.
a_2	Conjunto de la demanda asignada a las ubicaciones seleccionadas.

Para generar la *Lista Restringida de Candidatos* (LRC) se calcula el valor umbral u de la siguiente manera. Sea, $\varphi_{km\acute{\imath}n} \leftarrow m\acute{\imath}n_{k\in K\setminus\widehat{K}}\{\varphi_k\}$ el valor más pequeño de la función voraz para las ubicaciones candidato, $\varphi_{km\acute{a}x} \leftarrow m\acute{a}x_{k\in K\setminus\widehat{K}}\{\varphi_k\}$ el valor más grande para las ubicaciones candidato y $\alpha \in [0,1]$ un parámetro que mide el grado de voracidad o aleatoriedad del algoritmo voraz. Entonces, se calcula el valor umbral $u \leftarrow \varphi_{km\acute{\imath}n} + \alpha(\varphi_{km\acute{a}x} - \varphi_{km\acute{\imath}n})$ y se agregan a la LRC los índices de todas aquellas ubicaciones cuyo valor de la función voraz es menor o igual al valor umbral, es decir, $LRC = \{k: \varphi_k \leq u\}$. Posteriormente se elige al azar una ubicación k_s de LRC y se añade al conjunto de ubicaciones seleccionadas para las plantas, $\widehat{K} \leftarrow \widehat{K} \cup \{k_s\}$. Asimismo, se actualiza la asignación de productos a la planta k_s, es decir, $a_2(j, k_s, l) = m_{jl}$, $\forall(j, l, m_{jl}) \in AP_k$ y $p = p - m_{jl}$, $\forall(j, l, m_{jl}) \in AP_k$. El pseudocódigo para el subproblema *plantas-almacenes* se muestra en la ilustración 8.

$\widehat{K} \leftarrow$ ø

$p \leftarrow \sum_{j \in J} \sum_{(i,l) \in CP_j} a_{il}$

Mientras $(p > 0)$

Obtener $AP_k, \forall k \in K \backslash \widehat{K}$

$$\varphi_k = \frac{f_k + \sum_{(j,l) \in AP_k} d_{jkl}\, m_{jl}}{\sum_{(j,l) \in AP_k} m_{jl}} \qquad \forall k \notin \widehat{K}$$

Fin para

$\varphi_{kmín} \leftarrow mín_{k \in K \backslash \widehat{K}} \{\varphi_k\}$

$\varphi_{kmáx} \leftarrow máx_{k \in K \backslash \widehat{K}} \{\varphi_k\}$

$u \leftarrow \varphi_{kmín} + \alpha(\varphi_{kmáx} - \varphi_{kmín})$

$LRC = \{k : \varphi_k \leq u\}$

Seleccionar k_s aleatoriamente de *LRC*

$\widehat{K} \leftarrow \widehat{K} \cup \{k_s\}$

$a_2(j, k_s, l) = m_{jl}\,, \forall (j, l, m_{jl}) \in AP_k$

$p = p - m_{jl}\,, \quad \forall (j, l, m_{jl}) \in AP_k$

Fin mientras

ILUSTRACIÓN 2 PSEUDOCÓDIGO DEL GREEDY ALEATORIZADO PARA LA APERTURA DE PLANTAS

Como se muestra en el pseudocódigo, este procedimiento se repite hasta que toda la demanda de los almacenes seleccionados se ha asignado a alguna planta, también seleccionada.

Además con el objetivo de asegurar la mejor asignación de la demanda al conjunto de instalaciones seleccionadas, por medio del software de optimización XPRESS™ se resuelve el *subproblema de flujo óptimo*. De tal manera que se obtiene la asignación óptima de la demanda para el conjunto de instalaciones seleccionadas en la fase de construcción. El modelo que se resuelve es el siguiente:

$$MinZ = \sum_i \sum_j \sum_l a_{il} c_{ijl} X_{ijl} + \sum_j \sum_k \sum_l d_{jkl} Y_{jkl} \qquad (9)$$

$Sujeto\ a$:

$$\sum_{j} X_{ijl} = 1 \qquad \forall i \in I, \forall l \in L \qquad (10)$$

$$\sum_{i}\sum_{l} a_{il} X_{ijl} \leq V_j \qquad \forall j \in \hat{J} \qquad (11)$$

$$\sum_{i} a_{il} X_{ijl} = \sum_{k} Y_{jkl} \qquad \forall j \in \hat{J}, \forall l \in L \qquad (12)$$

$$\sum_{j}\sum_{l} Y_{jkl} \leq D_k P_k \qquad \forall k \in \widehat{K} \qquad (13)$$

$$X_{ijl} = \{0,1\} \qquad \forall i \in I, \forall j \in \hat{J}, \forall l \in L, \forall k \in \widehat{K} \qquad (14)$$

$$Y_{jkl} \geq 0 \qquad \forall j \in \hat{J}, \forall k \in \widehat{K}, \forall l \in L \qquad (15)$$

Por medio de la fase de construcción se obtiene una solución inicial factible, generando una cota superior.

FASE DE MEJORA

La segunda fase del GRASP consiste en mejorar la solución inicial generada en la fase de construcción. La fase de búsqueda local explora los entornos de la solución actual, con la finalidad de encontrar soluciones factibles que mejoren la función objetivo del problema.

El procedimiento para la fase de búsqueda local explora tres tipos de estructura de vecindad: intercambio, apertura y cierre de instalaciones. Cada uno de los entornos se aplica en dos partes, una para las plantas y otra para los almacenes.

Donde σ es una solución factible, conformada por el conjunto de ubicaciones para los almacenes y plantas seleccionadas, así como la demanda asignada a cada instalación. Los vecindarios se describen a continuación:

1. Intercambio de dos plantas (exploración N_1).

   El entorno $N_1(\sigma)$ contiene todas las soluciones que se pueden obtener a partir de la solución σ, cerrando una planta abierta, $k_1 \in \widehat{K}$, abriendo una planta cerrada, $k_2 \in K \backslash \widehat{K}$, y asignando toda la demanda asignada a la planta k_1 a la planta k_2. Se evalúan todos los posibles pares (k_1, k_2), tales que $D_{k_1} \leq D_{k_2}$ y se selecciona el mejor intercambio con respecto al valor de la función objetivo.

2. Intercambio de dos almacenes (exploración N_2).

   El entorno $N_2(\sigma)$ contiene todas las soluciones que se pueden obtener a partir de la solución σ, cerrando un almacén abierto, $j_1 \in \hat{J}$, abriendo un almacén cerrado, $j_2 \in J \backslash \hat{J}$, y asignando toda la demanda asignada al almacén j_1 al almacén j_2. Se evalúan todos los posibles pares (j_1, j_2), tales que $V_{j_1} \leq V_{j_2}$ y se escoge el mejor intercambio con respecto al valor de la función objetivo. En este entorno es más complicado evaluar en comparación con N_1, debido a que influyen los costos de distribución de los dos niveles.

Después de realizar cualquier movimiento en N_1 y N_2, se optimiza la asignación de la demanda a las instalaciones resolviendo el subproblema de flujo óptimo por medio del software Xpress™. También, para los siguientes entornos (N_3, N_4, N_5, N_6) se utiliza el software de optimización Xpress™ para calcular la función objetivo en la evaluación de los movimientos.

3. Abrir una planta (exploración N_3).

   El entorno $N_3(\sigma)$ contiene todas las soluciones que se pueden obtener a partir de la solución σ, abriendo una planta cerrada, $k_2 \in K \backslash \widehat{K}$, y

reasignando toda la demanda resolviendo el subproblema de flujo óptimo. Se evalúan todos los elementos de $K\backslash\widehat{K}$ y se realiza el movimiento cuando encuentra mejora en la función objetivo. El procedimiento termina cuando la solución es localmente óptima para el entorno.

4. Abrir un almacén (exploración N_4).

   El entorno $N_4(\sigma)$ contiene todas las soluciones que se pueden obtener a partir de la solución σ, abriendo un almacén cerrado, $j_2 \in J\backslash\hat{J}$, y reasignando toda la demanda utilizando el subproblema de flujo óptimo. Se evalúan todos los elementos de $J\backslash\hat{J}$. Se realiza el movimiento si existe mejora en la función objetivo. El procedimiento termina cuando la solución es localmente óptima para el entorno N_4.

5. Cerrar una planta (exploración N_5).

   El entorno $N_5(\sigma)$ contiene todas las soluciones que se pueden obtener a partir de la solución σ, cerrando una planta abierta, $k_1 \in \widehat{K}$, y reasignando toda la demanda por medio del subproblema de flujo óptimo. Se evalúan todos los elementos de $\widehat{K}$, siempre y cuando las plantas restantes, $\widehat{K}\backslash k_1$, tienen la capacidad suficiente para satisfacer la demanda de todos los almacenes. Se realiza el movimiento si mejora la función objetivo y el procedimiento termina cuando la solución es localmente óptima para el entorno N_5.

6. Cerrar un almacén (exploración N_6).

   El entorno $N_6(\sigma)$ contiene todas las soluciones que se pueden obtener a partir de la solución σ, cerrando un almacén abierto, $j_1 \in \hat{J}$, y reasignando toda la demanda por medio del subproblema de flujo óptimo. Se evalúan todos los elementos de $\hat{J}$, siempre y cuando los almacenes restantes, $\hat{J}\backslash j_1$, tienen la capacidad suficiente para satisfacer la demanda de todos los clientes con abastecimiento de fuente única. Se realiza el movimiento si

mejora la función objetivo. El procedimiento termina cuando la solución es localmente óptima para el entorno N_6.

En la ilustración 9 se muestra el orden en que son explorados cada uno de los entornos descritos anteriormente, donde cada entorno se detiene cuando encuentra su óptimo local. El siguiente pseudocódigo muestra el procedimiento:

```
Procedimiento búsqueda local (σ)
    Sea σ la solución incumbente y f(σ) el costo generado por la solución σ.
    terminar = falso
    Mientras (terminar = falso)
        Explorar N1
        σ1 es la mejor solución de N1
        Si (f(σ^1) < f(σ))
            σ ← σ^1
        Si no
            terminar = verdadero
        Fin-si
    Fin-mientras
    terminar = falso
    Mientras (terminar = falso)
        Explorar N2
        σ2 es la mejor solución de N2
        Si (f(σ^2) < f(σ))
            σ ← σ^2
            Resolver el subproblema de flujo óptimo
        Si no
            terminar = verdadero
        Fin-si
    Fin-mientras
    Para toda k ∉ solución hacer
        Explorar N3 (agrega ubicaciones de plantas si existe mejora)
        σ3 es la solución de N3
        Si (f(σ^3) < f(σ))
            σ ← σ^3
        Fin-si
    Fin-hacer
    Para todo j ∉ solución hacer
        Explorar N4 (agrega ubicaciones de almacenes si existe mejora)
        σ4 es la solución de N4
        Si (f(σ^4) < f(σ))
            σ ← σ^4
        Fin-si
    Fin-hacer
    Para toda k ∈ solución hacer
        Explorar N5 (elimina ubicaciones del conjunto seleccionado)
        σ5 es la solución de N5
        Si (f(σ^5) < f(σ))
            σ ← σ^5
        Fin-si
    Fin-hacer
    Para todo j ∈ solución hacer
        Explorar N6 (elimina ubicaciones del conjunto seleccionado)
        σ6 es la solución de N6
        Si (f(σ^6) < f(σ))
            σ ← σ^6
        Fin-si
    Fin-hacer
    Regresa σ
Fin
```

ILUSTRACIÓN 3 PSEUDOCÓDIGO PARA LA BÚSQUEDA LOCAL.

El objetivo que tiene esta fase es aproximar lo más posible la solución obtenida al óptimo del PLIDMC.

Se ha descrito la implementación de las dos fases que componen el heurístico GRASP para el PLIDMC. Como estrategia de diversificación, dicho algoritmo se repite 15 veces, además se varía el valor del parámetro α cada dos iteraciones. Dicho procedimiento genera diferentes soluciones iniciales; entonces, la fase de mejora parte de distintos puntos y se amplía el área de búsqueda entre todas las soluciones posibles, generando soluciones de mayor calidad.

En la ilustración 10 se muestra el pseudocódigo para el algoritmo propuesto para la generación de soluciones del PLIDMC. Donde *msol* contiene la mejor función objetivo encontrada y *MS* es el conjunto de instalaciones seleccionadas y demanda asignada que corresponden a *msol*.

```
Algoritmo GRASP

        f(σ) es el costo generado por la solución σ
        msol ← ∞
        MS ← ∅
        iter = 0
        Mientras (iter < 15 )
                iter = iter + 1
                Construir σ, algoritmo voraz aleatorio
                Aplicar búsqueda local (σ)
                Si (iter mod 2 = 0) entones
                        α = α + 0.06
                Fin- si
                Si (msol > f(σ)) entonces
                        msol ← f(σ)
                        MS ← σ
                Fin-si
        Fin- mientras
        Regresa (msol, MS)
Fin
```

ILUSTRACIÓN 4 ALGORITMO GRASP

Para evaluar el comportamiento del algoritmo desarrollado en este estudio, en el siguiente capítulo se presentan los resultados obtenidos en la implementación y la descripción de la generación de las instancias utilizadas.