CAPÍTULO 4

MÉTODOS PROPUESTOS

4.1 Algoritmos que no usan derivadas

El problema (3.1)-(3.2) puede reescribirse como:

$$\min_y \ F(x(y), y) \text{ s.a. } 0 \leq y_a \leq u_a, \text{ para todo } a \in E \tag{4.1}$$

donde $x(y)$ es la solución única del problema (3.2) correspondiente a y dado.

Algunos intentos se han hecho para resolver (4.1) directamente usando métodos de gradiente (ver, por ejemplo, [1, 19, 41]). Sin embargo, se presentan dificultades por el hecho de que la función objetivo $F(x(y), y)$, aunque continua, en general no es convexa o diferenciable como función del argumento y.

De aquí que una opción para resolver (4.1) es usar un algoritmo que no use derivadas, sino solamente evaluaciones de la función $F(x(y), y)$ para cada $x(y)$ obtenida en un y dado como solición única de (3.2).

Aplicaremos el algoritmo de Nelder-Mead (ver p.e. [9, 34, 46]) al problema (4.1) donde

$$F(x(y), y) = \sum_{a \in E} (c_a(x_a(y), y_a))x_a(y) + \theta_a G_a(y_a))$$

con $c_a(x_a(y), y_a)) = A_a + B_a \left(\frac{x_a(y)}{K_a + y_a}\right)^4$, $G_a(y_a) = y_a$, $\bar{c}_a(x_a, y_a) = A_a x_a + \frac{B_a x_a^5}{5(K_a+y_a)^4}$.

4.1.1 Descripción del algoritmo de Nelder-Mead adaptado al PPBN

El algoritmo de Nelder-Mead(NM) [45] es un método para minimizar funciones de n variables, con el uso de un simplex de $n+1$ vertices y por medio de reemplazos en el vértice con mayor valor en la función por otro punto derivado de tres operaciones (reflexión, contracción y expansión). Este método fue propuesto en los años sesentas y en la literatura reporta tener buenos resultados [45].

Por conveniencia, para la ilustración de este algoritmo se ocupa F_i como la función de objetivo de nivel superior del problema (4.1). Es decir $F_i = F(x(y), y)$. Este algoritmo comienza con la

creación del simplex con $|E|+1$ vértices (cada vértice es una solución de y para el problema (4.1)). Los vertices se crean al generar un número aleatorio (RND) entre 0 y 1, si RND es menor que un valor prefijado (en esta adaptación 0.3), el valor de esa variable será otro RND multiplicado por u_a y en caso contrario se multiplica otro RND por $u_a/2$. Este procedimiento se repetirá hasta completar el simplex de $|E|+1$ vertices. Por lo tanto tendremos $|E|+1$ soluciones y cada solución contiene $|E|$ variables. Este procedimiento puede verse en el algoritmo 1.

ALGORITMO 1 Generar Simplex

```
Inicializar y_ia ← 0 ∀ i ← 1 TO (|E| + 1), a ∈ E
for i ← 1 to(|E| + 1) do
  for a ∈ E do
    if RND < 0.3 then
      y_ia ← RND · u_a
    else
      y_ia ← RND · (u_a/2)
    end if
  end for
end for
```

Las $|E|+1$ soluciones se deben evaluar a fin de obtener h como el subíndice de la solución con el valor más alto en la función objetivo de nivel superior F_i y l como el subíndice de la solución con el valor más pequeño en la función objetivo de nivel superior F_i. Cabe aclarar que para obtener los valores de la función objetivo de nivel superior, se debe resolver primero el nivel inferior z_i con las variables y_{ia}.

Ahora que tenemos las evaluaciones de nivel superior del simplex proseguimos a definir el centroide $\bar{P}_a$ de los puntos, que se calcula con la expresión (4.2).

$$\bar{P}_a \leftarrow \sum_{i=1}^{|E|+1} y_{ia} - y_{ha} \cdot (1/|E|) \forall a \in E \tag{4.2}$$

Continuamos con la operación de *reflexión* que se denota por P_a^{ref} y su expresión esta dada por (4.3),

$$P_a^{ref} \leftarrow (1+\alpha)\bar{P}_a - \alpha P_{ha} \forall a \in E \tag{4.3}$$

en donde α es el coeficiente de reflexión y $\alpha > 0$.

El nuevo punto de P^{ref} debe de ser verificado, ya que NM es un método irrestricto y puede generar una solución que no cumple la restricción del problema (4.1). Para solucionar este aspecto,

se ocupa un adaptación del método NM al evaluar la función objetivo de nivel superior (4.1), es decir $F^{ref} = F(x(P^{ref}), P^{ref})$. En este procedimiento se emplea la penalización a la violación de las restricciones de (4.1). Si un punto está fuera del intervalo $[0, u_a]$ se le penalizará al sumarle un valor de $1 \cdot 10^6$ en la función objetivo de nivel superior de (4.1). Esto se hace, por que de esta manera, el punto no es atractivo para reemplazar al punto con peor valor en la función objetivo del simplex y esto obliga a pasar a otra etapa de NM. El esquema de procedimiento de *penalización* se puede observar en el algoritmo 2.

ALGORITMO 2 Penalización

$Infactible \leftarrow FALSE$
for $a \in E$ **do**
  if $P_a^{ref} < 0$ **or** $P_a^{ref} > u_a$ **then**
    $Infactible \leftarrow TRUE$
  end if
end for
if $Infactible \leftarrow TRUE$ **then**
  $F(x(P^{ref}), P^{ref}) = \sum_{a \in E}(c_a(x_a(P^{ref}), P_a^{ref}))x_a(P^{ref}) + \theta_a G_a(P_a^{ref}) + 1 \cdot 10^6$
else
  $F(x(P^{ref}), P^{ref}) = \sum_{a \in E}(c_a(x_a(P^{ref}), P_a^{ref}))x_a(P^{ref}) + \theta_a G_a(P_a^{ref})q$
end if

Con la evaluación F^{ref}, si F^{ref} se encuentra entre el intervalo (F_l, F_h), entonces la solución y_{ha} se reemplaza por P_a^{ref} y se reinicia NM con este nuevo simplex.

Si $F^{ref} < F_l$, es decir que la reflexión produjo un nuevo mínimo, procedemos a calcular la operación *expansión*, que se da por la relación (4.4),

$$P_a^{exp} \leftarrow \gamma P_a^{ref} + (1 - \gamma)\bar{P}_a \forall a \in E \tag{4.4}$$

γ, es el coeficiente de *expansión*, en donde $\gamma > 1$. Como se explicó anteriormente es necesario verificar la factibilidad de este punto, debido a esto se ocupa el mismo procedimiento de penalización del algoritmo 2 antes mencionado, sólo se necesita adaptarlo al utilizar el punto P^{exp} para obtener la evaluación de nivel superior $F^{exp} = F(x(P^{exp}), P^{exp})$.

Si $F^{exp} < F_l$, se reemplaza la solución y_{ha} por P_a^{exp}; en caso contrario es decir $F^{exp} > F_l$, se toma la solución y_{ha} por P_a^{ref} antes de reiniciar el algoritmo.

Si cuando se lleva a cabo la operación de reflexión P^{ref}, se halla que $F^{ref} > F_i$ para todo $i \neq h$, y además $F^{ref} > F_h$, pasamos a calcular la operación de *contracción*, pero si $F^{ref} > F_h$,

substituimos la solución y_{ha} por P_a^{ref} antes de realizar *contracción*. La contracción se forma por la ecuación 4.5, que tienen como parámetro β y $0 < \beta < 1$.

$$P_a^{con} \leftarrow \beta P_h + (1-\beta)\bar{P}_a \forall a \in E \tag{4.5}$$

De la misma forma que se hizo con las operaciones de *reflexión* y *expansión* se examina la factibilidad con el procedimiento de *factibilidad* con el algoritmo 2, intercambiando P_a^{ref} por P_a^{con}, para adquirir $F^{con} = F(x(P^{con}), P^{exp})$.

Una vez que se contrajo el punto, se acepta el punto P_a^{con} por P_h, a menos que $F^{con} > min(F_h, F^{ref})$ entonces se obtiene una contracción insatisfactoria y debido a esto se cambia todos los puntos con la expresión 4.6 y se reinicia el algoritmo de NM.

$$y_{ia} \leftarrow (y_{ia} + y_{la})/2 \ \forall \ i \leftarrow 1 \ \textit{TO} \ (|E|+1), \ a \in E \tag{4.6}$$

Es conveniente tener un criterio de paro para el algoritmo de NM, ya que como se observa en la descripción del método, este reinicia al término de cada etapa, así que como se menciona en [45], se puede ocupar la desviación estándar entre las evaluaciones de nivel superior del simplex que esta dado por la ecuaciones (4.8) y (4.7). Cuando se termina una etapa se comprueba, si $parar$ es menor que un valor ϵ, entonces para el algoritmo, en caso contrario se continua iterando.

$$M \leftarrow (\textstyle\sum_{i=1}^{|E|+1} F_i)/(|E|+1) \tag{4.7}$$

$$parar \leftarrow \sqrt{(\textstyle\sum_{i=1}^{|E|+1}(F_i - M)^2)/(|E|+1)} \tag{4.8}$$

Por otra parte, también es recomendable utilizar algún otro criterio de paro si se tiene pocos recursos computacionales, que puede ser el uso de una variable que cuente el número de veces en que en cada iteración no se mejora la solución con mejor valor en la función objetivo, si el valor de esta variable es mayor que un valor prefijado se termina el algoritmo de NM.

El esquema general para NM adaptado para el CNDP se puede ver en el algoritmo 3.

4.1.2 Descripción del algoritmo de Búsqueda Dispersa adaptado al PPBN

Scatter search, también conocido como Búsqueda dispersa (BD) en español, es un metaheurístico que se ocupa para resolver problemas de optimización. Este método pertenece a la familia de los algoritmos evolutivos y presenta similitudes con los algoritmos genéticos, solo que este método

ALGORITMO 3 Nelder - Mead

$iter \leftarrow 1, parar \leftarrow \infty$, $\epsilon \leftarrow 0.02$, $Solmin \leftarrow \infty, ContSolmin \leftarrow 0$ $SolucionesIguales \leftarrow FALSE$
while $parar > \epsilon$ **and** $SolucionesIguales = FALSE$ **do**
  if $iter = 1$ **then**
    Generar Simplex
    $z_i \leftarrow$Resolver Nivel Inferior $\forall i \leftarrow 1$ **to** $(|E| + 1)$
    $F_i \leftarrow$ Evaluar Nivel Superior $\forall i \leftarrow 1$ **to** $(|E| + 1)$
    $h \leftarrow argmax\{F_i\}$ $\forall i \leftarrow 1$ **to** $(|E| + 1)$, $l \leftarrow argmin\{F_i\}$ $\forall i \leftarrow 1$ **to** $(|E| + 1)$
  end if
  $\bar{P}_a \leftarrow$ Calcular Centroide $\forall a \in E$, $\bar{F} \leftarrow$ Evaluar Nivel Superior
  $P_a^{ref} \leftarrow$ Calcular Reflexión $\forall a \in E$, $F^{ref} \leftarrow F$ Evaluar Nivel Superior
  if $F^{ref} < F_l$ **then**
    $P_a^{exp} \leftarrow$Calcular Expansión$\forall a \in E$, $F^{exp} \leftarrow$ Evaluar Nivel Superior
    if $F^{exp} < F_l$ **then**
      Reemplazar $y_{ha} \leftarrow P_a^{exp}$ $\forall a \in E$, $F_h \leftarrow F^{exp}$
    else
      Reemplazar $y_{ha} \leftarrow P_a^{ref}$ $\forall a \in E$, $F_h \leftarrow F^{ref}$
    end if
  else
    if $F^{ref} > F_i$ **and** $i \neq h$ **then**
      if $F^{ref} > F_h$ **then**
        $P_a^{con} \leftarrow$ Calcular Contracción $\forall a \in E$, $F^{con} \leftarrow$ Evaluar Nivel Superior
      else
        Reemplazar $y_{ha} \leftarrow P_a^{ref}$ $\forall a \in E$
        $P_a^{con} \leftarrow$ Calcular Contracción $\forall a \in E$, $F^{con} \leftarrow$ Evaluar Nivel Superior
      end if
      if $F^{con} > F_h$ **then**
        Reemplazar $y_{ia} \leftarrow (y_{ia} + y_{l,a})/2$ $\forall i \leftarrow 1$ **to** $(|E| + 1)$, $a \in E$
        $F_i \leftarrow$ Evaluar Nivel Superior $\forall i \leftarrow 1$ **to** $(|E| + 1)$
      else
        Reemplazar $y_{ha} \leftarrow P_a^{con}$ $\forall a \in E$, $F_h \leftarrow F^{con}$
      end if
    else
      Reemplazar $y_{ha} \leftarrow P_a^{ref}$ $\forall a \in E$, $F_h \leftarrow F^{ref}$
    end if
  end if
  $h \leftarrow argmax\{F_i\}$ $\forall i \leftarrow 1$ **to** $(|E| + 1)$, $l \leftarrow argmin\{F_i\}$ $\forall i \leftarrow 1$ **to** $(|E| + 1)$
  $parar \leftarrow$ Calcular desviación estándar entre las evaluaciones del simplex
  if $F_l < Solmin$ **then**
    $Solmin \leftarrow F_l$, $ContSolmin \leftarrow 0$
  end if
  if $F_l = Solmin$ **then**
    $ContSolmin \leftarrow ContSolmin + 1$
  end if
  if $ContSolmin = |E| + 1$ **then**
    Se encontraron $|E| + 1$ soluciones iguales, $SolucionesIguales \leftarrow TRUE$
  end if
end while

usa estrategias sistemáticas en lugar de aleatorias. El algoritmo de BD se presenta por primera vez en [27], en donde se establece los principios fundamentales y en [35] establecen estrategias avanzadas para BD.

El diseño básico de BD se basa en cinco elementos [28] o métodos que se describen a continuación (mantenemos la acepción en inglés):

1. *Diversification Generation Method*. Método que genera un conjunto P de soluciones diversas (alrededor de 100).

2. *Improvement Method*. Método que transforma las soluciones a fin de mejorar soluciones tanto del conjunto de referencia *RefSet* como las combinadas antes de estudiar la introducción al conjunto de referencia. Es clave mencionar que este método debe ser capaz de que a partir de soluciones no factibles, obtener una factible y además mejorar su valor. En el caso en que el método no logre mejorar a la solución inicial, se considera como resultado la solución inicial.

3. *Reference Set Update Method*. Este método se encarga de construir y mantener el conjunto de referencia que consiste en las *b* ($b = |RefSet|$) soluciones que reúnen las características de calidad y diversidad (en donde el valor de b es generalmente pequeño, $b \leq 20$).

   a) *Creación.* El *RefSet* se construye inicialmente con las $b/2$ soluciones con mejor valor en la función objetivo de P. Las $b/2$ restantes se extraen de P con el criterio de maximizar la minima distancia con las que se encuentran en el *RefSet*. Se necesita establecer una función de distancia para este problema.

   b) *Actualización.* Al realizar *Solution Combination Method* estas combinaciones pueden entrar y reemplazar a alguna de las que se tiene en el *RefSet*, en caso de que las mejoren. De esta manera el conjunto de referencia mantendrá un tamaño constante *b* a lo largo de la ejecución del algoritmo. Otro enfoque a la actualización es realizar los intercambios por diversidad en lugar a calidad.

4. *Subset Generation Method*. Se encarga de operar al *RefSet*, para producir un subconjunto de sus soluciones como bases para crear la combinación de soluciones. Este método especifica la forma en que se selecciona los subconjuntos para aplicarles el método de combinación. Existen diferentes maneras para hacer estos subconjuntos, por ejemplo se elige entre buscar parejas, trios, etc., y una vez que se tiene la forma en que se va a combinar, se realiza este

método a los elemento de *RefSet* y a todos los subconjuntos se les aplica el método de combinación.

5. *Solution Combination Method*. Este método es el encargado de combinar todas las soluciones del *RefSet*. Para ello se considera los subconjunto de *Subset Generation Method* y se les aplica el método de combinación. Estas soluciones que se obtienen de las combinaciones se pueden introducir inmediatamente al *RefSet* (actualización dinámica) o se pueden almacenar temporalmente en una lista hasta que se termina de combinar y después ver que soluciones entran al *RefSet* (actualización estática).

En el caso particular del CNDP, BD inicia con *Diversification Generation Method*. El método que se utiliza para generar las soluciones y_i de P, con $|P| = 100$ (las soluciones se refieren al conjunto de incrementos de capacidad para las conexiones $a \in E$) se puede ver en el algoritmo 4, de esta manera al multiplicar un número aleatorio RND por u_a obligamos a que todas las soluciones sean factibles (*Improvement Method*) desde su creación, además de incluir diversidad en el conjunto P. Ahora se necesita resolver el problema (4.1) para toda solución que se encuentra en P.

ALGORITMO 4 Generación P

```
for i ← 1 to(|P|) do
  for a ∈ E do
    if RND < 0.3 then
      y_ia ← RND · u_a
    else
      y_ia ← RND · (u_a/2)
    end if
  end for
end for
```

El conjunto de referencia $|RefSet| = b = 10$ se construye con el *Reference Set Update Method* en el apartado de *creación* con las $b/2$ soluciones con mejor valor en la función objetivo de nivel superior de P para generar $RefSet = \{y_1, \ldots, y_{b/2}\}$ y las $b/2$ soluciones mas dispersas a las ya incluidas en el $RefSet$ para generar $RefSet = \{y^{(b/2)+1}, \ldots, y^b\}$. La función para medir la dispersión entre la soluciones es la distancia Euclidiana (4.9) en donde de manera general $d(r, s)$ es la distancia Euclidiana entre una solución cualquiera r y otra s. Se calcula para cada solución en $P - RefSet$ el mínimo de de las distancias euclidianas (4.10). Entonces se selecciona las

soluciones con el máximo de estas distancias mínimas y se agregan al $RefSet$. Ahora se pasa a ordenar el $RefSet$ de acuerdo a su evaluación de la función objetivo de nivel superior, tal que, y_1 es la mejor solución y y_b es la peor.

$$d(r,s) \leftarrow \sqrt{\sum_{i=1}^{n} (s_i - r_i)^2} \tag{4.9}$$

$$d_{min}(r) \leftarrow Min\{d(r,s) | s \in RefSet\} \tag{4.10}$$

De acuerdo con *Subset Generation Method*, en esta adaptación el método de generación de subconjuntos será en parejas. Ya que tenemos los subconjuntos pasamos a *Solution Combination Method* en el que se realizan tres tipos de combinaciones lineales [28] en las que asumimos que las soluciones del $RefSet$ son $y_{(j)}$ y $y_{(k)}$ y RND es un número aleatorio en el rango $(0,1)$:

$$\begin{aligned} &C1 : y = y_{(j)} - v \\ &C2 : y = y_{(j)} + v \\ &C3 : y = y_{(k)} + v \\ &v = RND \cdot \tfrac{y_{(j)} - y_{(k)}}{2} \end{aligned} \tag{4.11}$$

Las siguientes reglas se usan para generar soluciones con estos tres tipos de combinaciones lineales:

- Si $j \leq b/2$ y $k \leq b/2$ entonces se generan 4 soluciones con C1y C3 una vez y C2 dos veces.
- Si $j \leq b/2$ y $k > b/2$ entonces se generan 3 soluciones con C1, C2 y C3 una vez.
- Si $j > b/2$ y $k > b/2$ entonces se generan 2 soluciones con C2 y al elegir aleatoriamente entre C1 o C3.

De esta manera, al tener un $|RefSet = 10|$ y al utilizar parejas se crean 135 nuevas soluciones, que se agregan al conjunto $NuevasSoluciones$. La generación del conjunto $NuevasSoluciones$, no garantiza que se cumplan las restricciones del problema (4.1), por eso se necesita aplicar *Improvement Method*, que en este caso lo que hace es verificar si alguna solución es menor que 0, si se cumple esto se tomara el valor absoluto de esa solución. En el caso en que la solución sea mayor que u_a se le restará a cada solución un número aleatorio RND en el rango $(0,1)$ dividido entre 10 y este multiplicado por u_a, este procedimiento se aplicará hasta que la solución sea factible. El procedimiento se puede verificar en el algoritmo 5.

ALGORITMO 5 Factibilidad para $NuevasSoluciones$

```
for i ← 1 to|NuevasSoluciones| do
  for a ∈ E do
    if y_ia < u_a then
      y_ia ← abs(y_ia)
    end if
    while y_ia > u_a do
      y_ia ← y_ia − (u_a · (RND/10))
    end while
  end for
end for
```

El conjunto de $NuevasSoluciones$ se evalúa con (4.1) y para las b mejores soluciones (calidad), si esa solución no pertenece a $RefSet$ y su evaluación es menor que la peor solución de $RefSet$, entonces eliminamos del $RefSet$ la peor solución y agregamos la solución que se comparó. Se ordena el $RefSet$ de acuerdo a su evaluación en (4.1) tal que y_1 es la mejor solución y y_b la peor y se repite el proceso anterior, este procedimiento corresponde a *Reference Set Update Method* con el apartado de *actualización*. Este procedimiento de *actualización* se le conoce como actualización estática [35].

Una vez que se actualizó el $RefSet$, se calcula el primer criterio de paro, que tiene como finalidad detener el algoritmo de BD cuando la desviación estándar entre las evaluaciones de nivel superior del $RefSet$ es menor que 0.05 y además poner un limite en las regeneraciones del algoritmo, que se explicará en párrafos posteriores. Para el cálculo de la desviación estándar se puede ocupar las ecuaciones de NM (4.8) y (4.7), solo se sustituiría $(|E| + 1)$ por b.

El algoritmo itera los métodos de *Reference Set Update Method*, *Subset Generation Method*, *Solution Combination Method* y desviación estándar entre soluciones del $RefSet$, hasta que ya no se pueda ingresar una de $NuevasSoluciones$ al $RefSet$ o hasta un número de iteraciones prefijado en este caso 5.

En el algoritmo de BD se puede utilizar estrategias avanzadas, como es la regeneración del conjunto $|P|$ [35], que se utiliza para mejorar la dispersión de las soluciones. Para esta adaptación la regeneración sólo se lleva a cabo cuando se tiene estos tres casos simultáneamente:

1. En *Reference Set Update Method* no se encontró al menos una nueva solución.

2. La desviación estándar es menor que 0.05 (esto indica que no hay dispersión en $RefSet$).

3. Un contador de regeneraciones se menor o igual a 2.

Al Utilizar los tres casos anteriores se mantendrá una dispersión considerable que es la clave para el buen funcionamiento del algoritmo de BD. Se regenera el conjunto P y se mantiene las $b/2$ mejores soluciones (calidad), y de ahí se vuelve a calcular la distancia euclidiana para obtener las $b/2$ restantes de $RefSet$. De este punto el algoritmo continua normalmente.

El esquema general de BD se encuentra en el algoritmo 6.

4.1.3 Algoritmo de Búsqueda Dispersa con Nelder-Mead (BD-NM)

Como se observa en la sección anterior, el algoritmo de BD ocupa *Improvement Method*. Este método se ocupa en dos etapas del algoritmo. La primera se utiliza al término de la generación de P. Para complementar este método, se ocupa el algoritmo de NM. Es decir, al terminar de generar P, la solución con el mejor valor en la función objetivo de nivel superior de este conjunto se ingresa como un punto del simplex de NM. Para generar las soluciones restantes del simplex, se multiplica números aleatorio RND en el rango $(0, 1)$ por la solución que existe y de esta manera se tiene las soluciones necesarias para iniciar NM. Una vez que se ejecuta NM, las soluciones del simplex reemplazan a las $|E| + 1$ mejores soluciones (calidad) de P para continuar el algoritmo normalmente.

La segunda etapa en la que se utiliza *Improvement Method* es al finalizar *Solution Combination Method*. Cabe mencionar que se continua el uso del algoritmo 5 como parte del método mejora, pero al terminar de evaluar las soluciones del conjunto $NuevasSoluciones$, de la misma manera que se explicó en el párrafo anterior, se elige la mejor solución (calidad) de $NuevasSoluciones$ y se agrega a NM como un vértice del simplex y se multiplica varios RND por esta solución para completar el simplex y comenzar NM. Al terminar se regresan las soluciones de NM reemplazando las $|E|+1$ mejores soluciones (calidad) de P para continuar el algoritmo de la forma antes descrita.

Los procedimientos descritos anteriormente se realizan debido a que se debe aplicar la rutina de mejora de forma selectiva, ya que aplicar este método a todas los soluciones de los conjuntos respectivos, no garantiza el obtener mejores soluciones finales [28]. Además, hacer la mejora de soluciones de todos los conjuntos representa un gran gasto de recursos computacionales por los tamaños de los conjuntos y se pierde la dispersión entre las soluciones.

ALGORITMO 6 Búsqueda Dispersa

$iterreg \leftarrow 0$, $SolucionNueva \leftarrow FALSE$, $b \leftarrow 10$, $flag \leftarrow FALSE$, $SolucionesDispersas \leftarrow FALSE$
while $SolucionNueva = FALSE$ **and** $iterreg \leq 2$ **and** $SolucionesDispersas \leftarrow FALSE$ **do**
  Iniciar con $P = \emptyset$.
  for $i \leftarrow 1$ **to**$(|P|)$ **do**
    Método de Generación para construir una nueva solución ; sea y_i la solución obtenida. Si $y_i \notin P$ entonces añadir y_i a P, sino rechazar y_i.
  end for
  Aplicar el Método de Mejora para todo P.
  Resolver Nivel Inferior y Evaluar Nivel Superior para todas las soluciones de P.
  $b \leftarrow |RefSet|$
  if $iterreg = 0$ **then**
    Construir el conjunto de referencia con las mejores $b/2$ soluciones en P para generar $RefSet = \{y_1, \ldots, y_{b/2}\}$ y las $b/2$ soluciones de P más diversas a las ya incluidas para generar $RefSet = \{y_{(b/2)+1}, \ldots, y_b\}$
  else
    Mantener las mejores $b/2$ soluciones del $RefSet$ de la ultima iteración y completar el conjunto de referencia con las $b/2$ soluciones de P más diversas a las ya incluidas para generar $RefSet = \{y_{(b/2)+1}, \ldots, y_b\}$
  end if
  Ordenar $RefSet$ de acuerdo a su evaluación de la función objetivo, tal que, y_1 es la mejor solución y y_b es la peor
  Hacer $SolucionNueva \leftarrow TRUE$, $iter \leftarrow 1$ $SolucionesParcidas \leftarrow FALSE$
  while $SolucionNueva = TRUE$ **and** $iter <= 5$ **and** $SolucionesParcidas \leftarrow FALSE$ **do**
    Generar los $NuevasParejas$ de $RefSet$ en los que incluya al menos una nueva solución. $NuevasSoluciones \leftarrow \emptyset$
    for $y_{(j)}, y_{(k)} \in NuevasParejas$ **do**
      Seleccionar la siguiente pareja $(y_{(j)}, y_{(k)})$ en $NuevasParejas$
      Obtener una nueva solución p como una combinación lineal de $(y_{(j)}, y_{(k)})$ y añadirla a $NuevasSoluciones$
    end for
    Aplicar el Método de Mejora para todo $NuevaSoluciones$.
    $SolucionNueva = FALSE$
    for Las b mejores soluciones de $NuevaSoluciones$ p **do**
      if $y \notin RefSet$ **and** $F(x(y), y) < F(x(y_b), y_b)$ **then**
        Hacer $y_b \leftarrow y$
        $SolucionNueva = TRUE$
      end if
      Ordenar $RefSet$ de acuerdo a su evaluación de la función objetivo, tal que, y_1 es la mejor solución y y_b es la peor
    end for
    $STD \leftarrow$ Calcular desviación estándar de las evaluaciones de $RefSet$
    if $STD < 0.05$ **then**
      $flag \leftarrow TRUE$
      $SolucionesParcidas \leftarrow TRUE$
    end if
  end while
  if $flag = TRUE$ **then**
    $iterreg \leftarrow iterreg + 1$
  else
    $SolucionesDispersas \leftarrow TRUE$
  end if
end while

4.2 Ejemplos numéricos

Los algoritmos fueron codificados en Xpress 7.1, ejecutados en una computadora HP COMPAQ 8000 ELITE SFF BUSINESS PC con procesador INTEL CORE QUAD Q9650 a 3.00 GHz con 4.00 GB en RAM y probados en dos instacias que presenta [19].Todos los algoritmos se ejecutaron cinco veces para realizar las pruebas.

La primera instancia es una red de 9 nodos y 14 conexiones que se muestra en la figura 4.1. Este ejemplo numérico incluye 9 pares OD y 5 conexiones para el incremento de capacidad. Los datos de entradas para resolver las funciones objetivo de los problemas (3.1) y (3.2) se encuentran en las tablas 4.1 y 4.2.

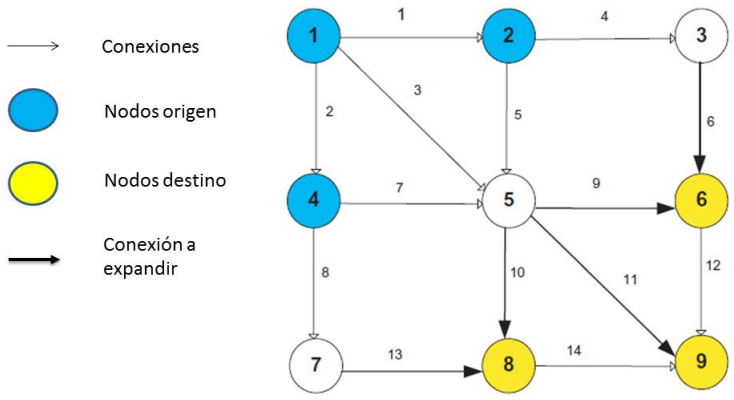

Figura 4.1: Red de 9 nodos

TABLA 4.1: OD para red de 9 nodos (en vehículos/por hora)

	6	8	9
1	120	150	100
2	130	200	90
4	80	180	110

Para NM se utilizan los siguientes parámetros: $\alpha = 0.2$, $\beta = 0.5$, $\gamma = 2$ y $\epsilon = 0.02$. El resumen de las corridas de NM en la instancia de 9 nodos se encuentra en la tabla 4.3. Como se puede observar los resultados varían considerablemente de corrida a corrida, arrojando un mínimo de 180 con un tiempo de $80.886seg$. NM proporciona resultados considerables en un tiempo corto.

TABLA 4.2: Datos para red de 9 nodos

Conexión a	A_a	B_a	K_a	θ_a	u_a
1	2.0	0.300	280	-	-
2	1.5	0.225	290	-	-
3	3.0	0.450	280	-	-
4	1.0	0.150	280	-	-
5	1.0	0.150	600	-	-
6	2.0	0.300	300	1.5	500
7	2.0	0.300	500	-	-
8	1.0	0.150	400	-	-
9	1.5	0.225	500	1.0	500
10	1.0	0.150	700	1.0	500
11	2.0	0.300	250	1.5	500
12	1.0	0.150	300	-	-
13	1.0	0.150	350	1.0	500
14	1.0	0.150	220	-	-

TABLA 4.3: Resultados de red de 9 nodos

NM 9 NODOS								
Prueba No.	1	2	3	4	5	Media	Máximo	Mínimo
$F(x(y), y)$	781.085	679.471	180	478.529	492.353	522.2876	781.085	180
z	3578.01	3569.71	3579.27	3560.79	3583.55	3574.266		
Tiempo (seg)	82.583	81.151	80.886	88.644	88.094	84.2716		

TABLA 4.4: Resultados de red de 9 nodos con BD

BD 9 NODOS								
Prueba No.	1	2	3	4	5	Media	Máximo	Mínimo
$F(x(y), y)$	255.41	323.69	128.775	233.865	175.75	223.498	323.69	128.775
z	3588.66	3586.34	3590.78	3591.94	3593.28	3590.2		
Tiempo (seg)	524.852	519.387	539.682	539.576	534.125	531.5244		

TABLA 4.5: Resultados de red de 9 nodos con BD-NM

BD-NM 9 NODOS								
Prueba No.	1	2	3	4	5	Media	Máximo	Mínimo
$F(x(y), y)$	21.7782	21.7845	21.7856	21.7821	21.7783	21.78174	21.7856	21.7782
z	3594.72	3594.72	3594.72	3594.72	3594.72	3594.72		
Tiempo (seg)	1176.81	1158.6	1186.69	1225.8	1225.87	1194.754		

La tabla 4.4 presenta los resultados de las corridas del algoritmo de BD en donde todos los resultados son mejores que los de NM, pero el esfuerzo computacional es mucho mayor aportando resultados en tiempos mayores.

Los resultados de la inclusión en NM a BD se muestran en la tabla 4.5, en la que se puede observar que esta combinación de algoritmos proporciona los mejores resultados en comparación de NM y BD.

La segunda instancia es una red de mayor tamaño que cuenta con 25 nodos, 44 conexiones y 5 conexiones para incrementar su capacidad. La figura 4.2 presenta la instancia antes mencionada y en las tablas 4.6 y 4.7 se encuentras los datos de entrada para los problemas (3.1) y (3.2).

En las tablas 4.8, 4.9 y 4.10 se exponen los resultados de la red de 25 nodos de NM, BD y BD-NM respectivamente. De la misma manera que los resultados anteriores NM es el que demuestra tener las peores soluciones pero con un esfuerzo computacional menor a los otros dos algoritmos, sin embargo BD mejora las soluciones de NM pero con un incremento en el tiempo computacional y finalmente BD-NM ilustra las mejores soluciones aunque el tiempo en que tarda en encontrar estas soluciones es mucho mayor.

TABLA 4.6: OD para red de 25 nodos (en vehiculos/por hora)

	20	24	25
1	120	150	100
2	130	200	90
6	80	180	110

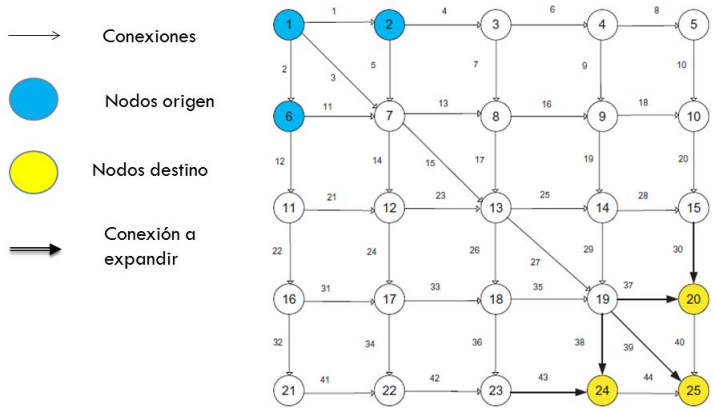

Figura 4.2: Red de 25 nodos

TABLA 4.7: Datos para red de 25 nodos

Conexión a	A_a	B_a	K_a	θ_a	u_a	Conexión a	A_a	B_a	K_a	θ_a	u_a
1	2.0	0.300	280	-	-	23	2	0.300	500	-	-
2	1.5	0.225	290	-	-	24	1.0	0.150	700	-	-
3	3.0	0.450	280	-	-	25	1.5	0.225	500	-	-
4	2.0	0.300	280	-	-	26	1.0	0.150	700	-	-
5	1.0	0.150	600	-	-	27	2.0	0.300	250	-	-
6	1.0	0.150	280	-	-	28	1.5	0.225	500	-	-
7	1.0	0.150	600	-	-	29	1.0	0.150	700	-	-
8	1.0	0.150	280	-	-	30	1.0	0.150	300	1500	1500
9	1.0	0.150	600	-	-	31	2.0	0.300	500	-	-
10	2.0	0.300	300	-	-	32	1.0	0.150	400	-	-
11	2.0	0.300	500	-	-	33	2.0	0.300	500	-	-
12	1.5	0.225	290	-	-	34	1.0	0.150	700	-	-
13	2.0	0.300	500	-	-	35	1.5	0.225	500	-	-
14	1.0	0.150	600	-	-	36	1.0	0.150	700	-	-
15	3.0	0.450	280	-	-	37	1.5	0.225	500	1.0	1500
16	1.5	0.225	500	-	-	38	1.0	0.150	700	1.0	1500
17	1.0	0.150	600	-	-	39	2.0	0.300	250	1.5	1500
18	1.5	0.225	500	-	-	40	1.0	0.150	300	-	-
19	1.0	0.150	600	-	-	41	1.0	0.150	350	-	-
20	2.0	0.300	300	-	-	42	1.0	0.150	350	-	-
21	2.0	0.300	500	-	-	43	1.0	0.150	220	1.0	1500
22	1.0	0.150	400	-	-	44	1.0	0.150	220	-	-

TABLA 4.8: Resultados de red de 25 nodos

NM 25 NODOS								
Prueba No.	1	2	3	4	5	Media	Máximo	Mínimo
$F(x(y), y)$	1012.4	1157.82	1121.22	1000.31	989.967	1056.3434	1000.31	989.967
z	9346.53	9217.56	9196.38	9344.56	9254.56	9271.918		
Tiempo (seg)	711.77	721.844	728.362	739.659	720.288	724.3846		

TABLA 4.9: Resultados de red de 25 nodos con BD

BD 25 NODOS								
Prueba No.	1	2	3	4	5	Media	Máximo	Mínimo
$F(x(y), y)$	336.066	491.122	393.784	635.021	565.411	484.2808	635.021	336.066
z	9381.24	9421.98	9347.06	9181.37	9504.75	9367.28		
Tiempo (seg)	3666.2	3683.7	3627.74	3713.08	3824.33	3703.01		

Para realizar las comparaciones, se utilizan los incrementos de capacidad para las conexiones a que muestra [19] y se evaluaron resolviendo (4.1) en el optimizador que se menciona al principio de esta sección, para obtener las funciones objetivo tanto de nivel superior e inferior. Las abreviaciones de los algoritmos con los que se compara los resultados de este trabajo los contiene la tabla 4.11.

La tablas de resultados 4.12 y 4.13 presentan los resultados de 10 algoritmos en instacias de 9 nodos y 25 nodos respectivamente. Para las instancias de 9 nodos y 25 nodos el algoritmo de BD-NM nos proporciona los mejores resultados. En la instancia de 9 nodos se alcanza el resultado de SO, mientras que en la de 25 nodos se mejora la mejor solución reportada en este caso SO.

TABLA 4.10: Resultados de red de 25 nodos con BD-NM

BD-NM 25 NODOS								
Prueba No.	1	2	3	4	5	Media	Máximo	Mínimo
$F(x(y), y)$	66.8245	66.8338	67.0368	66.9074	67.5018	67.02086	67.5018	66.8245
z	9482.33	9485.11	9544.97	9494.76	9481.24	9497.682		
Tiempo (seg)	7275.56	7249.97	6711.93	7855.76	6782.02	7175.048		

TABLA 4.11: Abreviaciones de heurísticos para CNDP

Abreviación	Nombre de la heurística	Fuentes
EDO	Equilibrium Decomposed Optimization	Suwansirikul et al. (1987)
SAB	Sensitivity Analysis Based algorithm	Yang and Yagar (1995)
GP	Gradient Projection method	Chiou (2005)
CG	Conjugate Gradient projection method	Chiou (2005)
QNEW	Qusai-Newton projection method	Chiou (2005)
PT	PARATAN version of gradient projection method	Chiou (2005)
SO	CNDP confined to the system optimal network flows	Chiou (2005)
NM	Nelder Mead	Esta tesis
BD	Búsqueda Dispersa	Esta tesis
BD-NM	Búsqueda Dispersa con Nelder Mead	Esta tesis

TABLA 4.12: Comparación de Resultados de red de 9 nodos

RESULTADOS 9 NODOS										
algoritmo / a	SAB	GP	CG	QNEW	PT	EDO	SO	NM	BD	BD-NM
6	0	0	0	0	0	0.0002	0.0002	1.53386	9.16159	1.39003E-06
9	0	0	0	0	0	0.0002	0.0002	67.9108	7.93232	0.000120387
10	0	0	0	0	0	0.0002	0.0002	22.8407	4.0173	0.000274095
11	0	0	0	0	0	0.0002	0.0002	10.3062	47.9428	0.000249401
13	128.827	137.445	130.564	130.564	127.79	137.445	0.0002	49.7407	9.4101	0.000653404
$F(x(y), y)$	150.515	159.123	152.25	152.25	149.479	159.124	21.778	180	128.775	21.7782
z	3567.4	3566.45	3567.2	3567.2	3567.52	3566.45	3594.72	3579.27	3590.78	3594.72
Tiempo (seg)	-	-	-	-	-	-	-	80.886	539.682	1176.81

TABLA 4.13: Comparación de Resultados de red de 25 nodos

RESULTADOS 25 NODOS										
algoritmo / a	SAB	GP	CG	QNEW	PT	EDO	SO	NM	BD	BD-NM
6	0	0	0	0	0	0.0007	0.0007	120.011	117.082	0.00177662
9	0	0	0	0	0	0.0007	0.0007	163.251	2.14221	0.00568669
10	0	0	0	0	0	0.0007	0.0007	234.994	84.725	0.000482702
11	0	0	0	0	0	0.0007	0.0007	129.859	0.607907	0.00455116
13	477.836	479.778	493.953	493.953	479.83	499.324	242.266	210.009	64.1234	0.0106503
$F(x(y), y)$	544.032	546.189	560.326	560.326	546.214	565.648	309.193	989.967	336.066	66.8245
z	9176.08	9176.04	9173.94	9173.94	9175.49	9181.36	9239.75	9254.56	9381.24	9482.33
Tiempo (seg)	-	-	-	-	-	-	-	720.288	3666.2	7275.56