CAPÍTULO 4
METODOLOGÍA PROPUESTA

En este capítulo, se describen los métodos utilizados en el procedimiento propuesto para obtener soluciones del PLIDMC.

En la actualidad, muchos de los problemas de optimización combinatoria no pueden ser resueltos computacionalmente de manera óptima por la naturaleza del problema. Por ello, en las últimas décadas se han desarrollado métodos que generen soluciones de buena calidad, sin la necesidad de examinar cada una de las posibles soluciones. Las técnicas de la optimización combinatoria han incrementado la capacidad para resolver problemas reales de gran escala. Algunas de las metodologías clásicas aplicadas son: buenas formulaciones matemáticas, teoría de planos de corte, algoritmos de descomposición y los métodos heurísticos.

Por otro lado, el avance de la tecnología ha facilitado la implementación de métodos de solución eficientes, y la resolución de problemas de gran tamaño, por lo que, en los últimos años, se ha generado un gran avance en la capacidad de los algoritmos para resolver los problemas de optimización. Los métodos heurísticos, también llamados metaheurísticas, se han aplicado exitosamente en el área de la optimización. Por ello, se proponen estos métodos para obtener soluciones factibles, y en muchos casos se utilizan para problemas NP-hard. Lo que intentan estas estrategias es generar buenos resultados de la manera más sencilla y eficaz, con un uso razonable de recursos, basándose en un conjunto de reglas, sin ser totalmente rigurosas.

Sin embargo, estos métodos no necesariamente aseguran el óptimo, aunque pueden generar soluciones de muy buena calidad. Una forma de clasificar estos algoritmos es en base al método de resolución: construcción, mejora o

combinados. Algunos de los métodos más utilizados son: GRASP, algoritmos genéticos y evolutivos, templado simulado, búsqueda tabú, sistemas de colonia de hormigas, búsqueda dispersa, re-encadenamiento de trayectoria, entre otros.

La efectividad de estos métodos, se basa en la capacidad de aprovechar la estructura básica del problema, así como evitar quedar atrapado en óptimos locales. Para esto se han propuesto técnicas que proveen beneficios, tales como aleatorización controlada, estructuras de datos eficientes, pre-procesamientos, entre otros. Además, existen muchas propuestas que incluso combinan distintas metodologías y generan nuevas heurísticas.

Debido a la complejidad del problema estudiado en este trabajo y el número de posibles soluciones que se incrementa exponencialmente conforme a la cantidad de instalaciones a ubicar, no existe actualmente un algoritmo exacto que pueda resolver cualquier PLIDMC, con un número polinomial de operaciones. Por lo tanto, este trabajo propone una heurística GRASP, detallando su procedimiento en este capítulo.

ALGORITMO VORAZ

Un algoritmo voraz es un procedimiento que construye una solución factible para el problema estudiado. Es un procedimiento iterativo, que inicia con una solución vacía y en cada iteración va añadiendo elementos a su solución, en base a una función que mide el aporte de cada elemento a la función objetivo, repitiendo esto hasta completar la solución. Esta función es llamada función voraz, y mide el beneficio o contribución de cada elemento candidato a la función objetivo. En cada una de las iteraciones, se ordenan los elementos por agregar en base a esta función generando una lista de candidatos. Se selecciona el elemento con mayor aporte a la función objetivo y se agrega a la solución. Después se reevalúa la función voraz para los candidatos restantes y se repite el proceso. Es

importante notar que esta función es miope en el sentido que no analiza el efecto que tendrá el elemento seleccionado en las selecciones sucesivas.

Sea S, una solución vacía. En la ilustración 2 se muestra el pseudocódigo de un algoritmo voraz.

```
S ← ∅
Mientras la solución no sea factible
        Evaluar la función voraz para todos los candidatos
        Elegir el candidato x con mejor evaluación
        Eliminar x de los candidatos
        S ← S ∪ {x}
Fin-Mientras
```

ILUSTRACIÓN 1. PSEUDOCÓDIGO DEL ALGORITMO GREEDY.

Estos algoritmos seleccionan los elementos más prometedores del conjunto de candidatos hasta encontrar una solución que en la mayoría de los casos no es la óptima. Además se dice que el método es adaptativo, ya que en cada iteración se va actualizando la función voraz, calculando las aportaciones parciales a la solución para cada uno de los elementos candidatos. Es decir, la función voraz para un elemento no tiene que tener el mismo valor de una iteración a otra.

BÚSQUEDA LOCAL

La búsqueda local es una técnica que se utiliza en los problemas de optimización para mejorar una solución inicial. Todo problema de optimización tiene un conjunto de soluciones posibles, ya sea finito o infinito. Estos algoritmos son iterativos, parten de una solución factible y exploran dentro del conjunto de soluciones posibles, intentando mejorar la solución, aplicando distintas estrategias de búsqueda.

Definición 1: Cada solución s tiene un conjunto de soluciones asociadas $N(s)$, que se denomina **entorno** de s.

Definición 2: Dada una solución s, cada solución s' de su entorno $N(s)$, puede obtenerse directamente a partir de s mediante una operación llamada **movimiento**.

Definición 3: Dada una solución s, se dice que es un **óptimo local** si no existe una mejor solución en $N(s)$ [17].

La idea básica de estos algoritmos es, a partir de una solución inicial s_0, explorar su entorno $N(s_0)$ y escoger una nueva solución $s_1 \in N(s_0)$.

Para determinar un procedimiento de búsqueda local se debe definir una estructura de vecindad y el criterio de selección de una solución. Por ende, la definición de *entorno-movimiento* depende en gran medida de la estructura del problema a resolver, así como de la función objetivo. Es decir, dada una solución factible, un algoritmo de búsqueda local explora su entorno y busca una solución que mejore la función objetivo. Si cumple con el criterio de selección se realiza el movimiento asociado, y se repite el proceso hasta que la solución no pueda ser mejorada. A esta solución encontrada se le denomina óptimo local. En la ilustración 3 se muestra el pseudocódigo del algoritmo de búsqueda local en general. Sea $f(x)$ la función a minimizar, x una solución inicial factible y $N(x)$ la estructura de vecindad.

```
Criterio de parada ← falso
Mientras no se cumpla el criterio de parada hacer
        x' = arg min {f(y)|y ∈ N(x)}
        Si (f(x') < f(x)) entonces
                x← x'
        De lo contrario
                Criterio de parada ← verdadero
        Fin-si
Fin- mientras
```

ILUSTRACIÓN 2 ALGORITMO DE BÚSQUEDA LOCAL

GRASP (GREEDY RANDOMIZED ADAPTATIVE SEARCH PROCEDURE)

El método fue desarrollado en 1989 por Feo y Resende [18], con el objetivo principal de solucionar problemas de recubrimiento de conjuntos, aunque el término GRASP fue introducido hasta 1995 por Feo y Resende [19]. Dicho término viene de sus siglas en ingles “Greedy Randomized Adaptative Search Procedure”.

GRASP es una técnica iterativa de muestreo aleatorio, de tal manera que repite su procedimiento un número definido de veces con la intención de generar buenas soluciones. Cada iteración se compone de dos fases. En la primera, se aplica una heurística constructiva para generar una solución factible. A partir de la solución generada, la segunda fase utiliza una heurística de mejora para la búsqueda de un óptimo local. El algoritmo selecciona la mejor solución obtenida en todas las iteraciones como una solución del problema.

FASE DE CONSTRUCCIÓN

En la fase de construcción, por medio de un algoritmo voraz, iterativamente se construye una solución factible. En cada iteración de este procedimiento, se agrega un nuevo elemento a la solución, en base al valor de la función voraz. Asimismo, la heurística es adaptativa, debido al hecho de que el beneficio asociado a cada uno los elementos candidatos se actualiza en cada iteración de la fase de construcción, reflejando los cambios generados por el elemento previamente seleccionado. Aunque los algoritmos voraces pueden producir soluciones iniciales razonables para búsquedas locales, su principal desventaja es la falta de diversidad, por lo que como una estrategia de diversificación, se aleatoriza el algoritmo voraz con respecto a la selección de elementos para la solución. En lugar de seguir la regla común de seleccionar el mejor elemento en base al valor de la función voraz, se construye una lista de candidatos que incluye los mejores candidatos y se selecciona uno aleatoriamente. Para implementar esto es necesario establecer un criterio de selección para los elementos que van a

formar parte de la lista de candidatos, llamado comúnmente umbral, que es un valor límite permitido para los elementos. En base a este valor, se reduce la lista de candidatos a una lista restringida de candidatos (LRC) y de ellos se toma un elemento al azar. Este elemento seleccionado se añade a la solución. El procedimiento se repite hasta completar la solución factible. De esta manera, la técnica de selección permite construir distintas soluciones iniciales en cada iteración del GRASP.

Para controlar la diversidad de las soluciones encontradas se utiliza el parámetro $\alpha \in [0,1]$. Cuando $\alpha=0$ el algoritmo es un algoritmo voraz puro, mientras que cuando $\alpha=1$ es totalmente aleatorio. Es importante encontrar un equilibrio en la aleatoriedad de la generación de la solución, lo que se controla con el valor asignado, puesto que si es demasiado aleatorio generará soluciones iniciales pobres, en contraste con poca aleatoriedad siempre se obtendrá la misma solución.

Se puede resumir la fase de construcción descrita anteriormente en la ilustración 4, que muestra el pseudocódigo basado en la propuesta de Feo y Resende [19]:

```
Solución = {}
Mientras la solución no se complete
        Calcular función voraz para todos los candidatos
        Calcular el umbral
        Construir Lista Restringida de Candidatos (LRC)
        s=Seleccionar Elemento aleatoriamente de (LRC)
        Solución = Solución ∪ {s}
Fin-mientras
```

ILUSTRACIÓN 3 PSEUDOCÓDIGO DE LA FASE DE CONSTRUCCIÓN DE SOLUCION POR MEDIO DE UN ALGORITMO VORAZ ALEATORIZADO PARA EL CASO GENERAL.

Dado que la fase constructiva no garantiza la optimalidad local con respecto a la estructura de entorno en la que se trabaja, la segunda fase aplica una búsqueda local para intentar mejorar la solución construida.

FASE DE MEJORA

La búsqueda local desempeña un papel importante en el GRASP, ya que sirve para buscar soluciones localmente óptimas en regiones prometedoras del espacio de soluciones. El algoritmo trabaja de modo iterativo, reemplazando sucesivamente la solución actual por una mejor solución del entorno de la solución actual. Esto se repite hasta encontrar el óptimo local. Es posible explorar distintos entornos de búsqueda. Asimismo, se pueden tener distintos criterios para seleccionar las nuevas soluciones de los entornos, siendo lo más simple evaluar las soluciones con respecto a la función objetivo.

La ilustración 5, muestra los pseudocódigos para la fase de mejora, también propuesta por Feo y Resende [19]:

```
Mientras  s no óptimo local
        Encontrar una mejor solución t ∈ N(s)
        s = t
Fin-mientras
Regresar (s como un óptimo local del problema)
```

ILUSTRACIÓN 4 PSEUDOCÓDIGO DE LA FASE DE BÚSQUEDA LOCAL PARA EL CASO GENERAL.

El éxito de un algoritmo de búsqueda local radica en una buena selección de la estructura de vecindad, en técnicas de búsqueda efectivas y en la calidad de la solución inicial.

A manera de resumen se presenta en la ilustración 6 el pseudocódigo propuesto por Feo y Resende [19], para el GRASP en general:

```
Procedimiento GRASP
        Leer datos
        Mientras no se satisfaga el criterio de parada de GRASP
                Construir solución por medio del algoritmo voraz aleatorizado
                Búsqueda Local
                Actualizar solución (solución, mejor solución encontrada)
        Fin de mientras
        Regresar (mejor solución encontrada)
Fin
```

ILUSTRACIÓN 5 ALGORITMO DEL HEURÍSTICO GRASP

La finalidad de realizar muchas iteraciones del procedimiento es hacer un muestreo del espacio de soluciones. De esta manera se evita quedar atrapado en un óptimo local, que pueda estar muy alejado del óptimo global.

Es difícil analizar formalmente la calidad de las soluciones obtenidas utilizando la metodología GRASP. Sin embargo, existe una justificación intuitiva que plantea el GRASP como una técnica de muestreo repetitivo, ya que cada iteración del método produce una muestra de una distribución no conocida, generada de todas las soluciones posibles, cuya media y varianza depende de la naturaleza restrictiva de la lista de candidatos. Por ello, en función de la lista de candidatos se puede obtener una media y una varianza de la distribución. Si la función voraz es efectiva el valor de la solución promedio es bueno, pero probablemente no el óptimo, aunque algunas veces después de aplicar la fase de mejora la mejor solución obtenida es la óptima.

A pesar de que las características de GRASP pueden ser encontradas en otras metodologías de búsqueda heurística, la implementación en la práctica es distinta. Se propone esta metodología, debido a la capacidad que tiene para la obtención de soluciones para problemas de optimización combinatoria de gran tamaño. Además, es un método sencillo y fácil de implementar. En el siguiente capítulo se describe detalladamente la implementación del heurístico GRASP al PLIDMC.