Universidad Nacional Autónoma de México
Facultad de Ciencias
Heuristicas de Optimización Combinatoria

Resolución de `k-MST` por medio de una variante de `WAO`

Autor: Medina Peralta Joaquín

Introducción

En este documento, veremos la manera de resolver un problema `NP-duro`, el cual es el encontrar el árbol de tamaño k de peso mínimo, donde usando algoritmos tradicionales o variantes de algunos conocidos, como *Prism* o *Kruskal*, no será posible calcularlo en tiempo humanamente posible.

De este modo, utilizaremos una heuristica de optimización combinatoria que en este caso es una variante del algoritmo de optimización por ballena (*WOA* por sus siglas en ingles), donde se usará para realizar búsqueda local y usando una discretización inspirada en su versión binaria propuesta en artículos posteriores al original, la cual utiliza una función sigmoid para transformar valores $\mathbf{R} \longrightarrow (0, 1)$ y usando un limite podemos definir si un nodo está o no en el árbol.

Por último, revisaremos la experimentación con tres instancias de gráficas $|V(G)| \approx 1000$ y con $|E(G)| \approx \{150000, 40000, 2500\}$ respectivamente para calcular valores de $k = 40$ y $k = 150$, donde veremos que se debe de ajustar los parámetros para obtener soluciones factibles y en poco tiempo.

Problema

En este trabajo, estaremos revisando el problema de gráficas del árbol generador de peso mínimo de tamaño k, el cual consiste en dado una gráfica $G = (V, E)$ con vértices y donde para cada $u, v \in V$, $0 \leq E(u, v)$ y dado un entero positivo $k \leq |V|$ es buscar un árbol T con k vértices de G y donde el tamaño $l(T) = \sum_{i,j \in V(T)} d_{i,j}$ sea el mínimo.

Dado que, debemos de escoger combinaciones de los vértices de tamaño k donde el orden no importa y no existen elementos repetidos, por lo revisar cada uno para encontrar el de mayor tamaño, donde nos costaría un tiempo aproximado de:

$$\frac{|V|!}{(|V| - k)!k!}$$

De esta manera, no se puede seleccionar una mejor solución en tiempo polinomial y es dificultad de resolver este tipo de problemas, aunque se puede intentar por medio de algoritmos greedy o de programación dinámica.

Heurística

En este ocasión, se decidió utilizar la heurística de `WAO`, el cual busca inspiración como se menciona en Mirjalili y Lewis, 2016, en el método de caza de la ballena jorobada llamado red de burbujas donde se pudieron asociar dos maniobras:

- Durante la etapa denominada como *espirales ascendentes*, las ballenas se zambullen a unos 12 metros de profundidad y luego comienzan a crear burbujas en forma de espiral alrededor de las presa mientras nadan hacia la superficie.
- En la segunda maniobra llamada *doble bucle*, constituye en tres etapas diferentes: bucle coralino, golpe de cola y bucle de captura.

De esta manera, usando este comportamiento se modela la optimización donde como se menciona en Mirjalili y Lewis, 2016, consiste en crear un conjunto de soluciones aleatorias donde cada agente (ballena) se caracteriza por un vector $X \in \mathbf{R}^n$. Posteriormente se calcula la mejor solución del conjunto de agentes y es el objetivo para todos los demás.

En este punto, el algoritmo `WOA` consta en una fase de explotación y de explooración, con las siguientes características:

1. La fase de exploración, las ballenas actuales buscan de manera aleatoria presas usando la posición de otras ballenas alejadas, donde se modela con la ecuación 3 y 4 del anexo.
2. En la fase de exploatción, las ballenas se acercan a la posición de la presa que representa la mejor solucón encontrada hasta el momento, donde usando las ecuaciones 1 y 2 son para acercarse a la mejor ballena. Mientras que la ecuación 5 y 6 es para simular el ataca de las ballenas hacia la presa usando una red de burbujas en forma de espiral.

   Para este caso, se utiliza una probabilidad $p \in [0, 1]$ para decidir si se utiliza el acercamiento directo o el ataque en espiral.

De esta manera, se puede observar que el algoritmo `WOA` está diseñado para problemas de optimización continua, por lo que se debe de adaptar para problemas discretos, como el `k-MST`.

Para este caso, se decidió utilizar una versión binaria del `WOA`, inspirada en K. et al., 2019, donde se propone tres funciones para transformar los valores reales a binarios, siendo las siguientes funciones:

- Una función por medio de una tangente hiperbólica (7) donde si el valor mayor a un umbral aleatorio se asigna un 1, en otro caso un 0 (8).
- Una función utilizando una transformación de arcotangente (9) donde si el valor mayor a un umbral aleatorio se asigna un 1, en otro caso un 0 (10).
- Una función sigmoide (11) donde si el valor mayor a un umbral aleatorio se asigna un 1, en otro caso un 0 (12).

En este caso, se decidió uutilizar la función de la transformación del arcotangente, para evitar saturación en los valores altos, lo cual podría suceder para la función sigmoide y tangente

hiperbólica.

Finalmente, en el anexo se puede observar el pseudocódigo del algoritmo `WOA`, donde para cada valor de la posición de la ballena se utilizara la función para discretizar los valores y así utilizar la heuristica para una optimización combiinatoria discreta.

Diseño de la solución

En este proyecto, se decidió utilizar el lenguaje de programación `Rust` para optimizar el rendimiento y manejo de memoria, ya que se trabajará con instancias grandes de gráficas. Además, se utilizará paquetes de `cargo.io` como:

Tecnologías a utilizar

Implementación del problema

Experimentación

Resultados

Conclusión

Referencias

K., S. R., Panwar, L., Panigrahi, B. K., & Kumar, R. (2019). Binary Whale Optimization Algorithm: A New Meta-heuristic Approach for Profit-Based Unit Commitment Problems in Competitive Electricity Markets. *Engineering Optimization*, *51*(3), 369-389. https://doi.org/10.1080/0305215X.2018.1463527

Mirjalili, S., & Lewis, A. (2016). The Whale Optimization Algorithm. *Advances in Engineering Software*, *95*, 51-67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Anexo

Algoritmo 1 Algoritmo de Optimización de Ballenas (Whale Optimization Algorithm - WOA)

Inicializar la población de ballenas X_i $(i = 1, 2, \ldots, n)$
Calcular el fitness de cada agente de búsqueda
X^* = el mejor agente de búsqueda
while $t <$ número máximo de iteraciones **do**
    for cada agente de búsqueda **do**
        Actualizar a, A, C, l, y p
        if $p < 0.5$ **then**
            if $|A| < 1$ **then**
                Actualizar la posición del agente de búsqueda actual por la Ec. (1)
            else
                Seleccionar un agente de búsqueda aleatorio (X_{rand})
                Actualizar la posición del agente de búsqueda actual por la Ec. (3)
            end if
        else
            Actualizar la posición del agente de búsqueda actual por la Ec. (5)
        end if
    end for
    Comprobar si algún agente de búsqueda sale del espacio de búsqueda y modificarlo
    Calcular el fitness de cada agente de búsqueda
    Actualizar X^* si hay una mejor solución
    $t \leftarrow t + 1$
end while
return X^*

Donde las ecuaciones son:

$$X_t = X^* - A \cdot D \tag{1}$$
$$D = |C \cdot X^* - X_t| \tag{2}$$

$$X = X_{rand} - A \cdot D \tag{3}$$
$$D = |C \cdot X_{rand} - X_t| \tag{4}$$

$$X_t = D \cdot e^{bl} \cdot \cos(2\pi l) + X^* \tag{5}$$
$$D = |X^* - X_t| \tag{6}$$

Con el valor de las constantes como:

- $A = 2a \cdot r - a$
- $C = 2 \cdot r$
- $a \in [2, 0]$ es un valor que va en decremento durante la cantidad de iteraciones.
- $r \in [0, 1]$ es un valor aleatorio.
- b es la constante definida de la forma logarítmica de la espiral.
- $l \in [-1, 1]$ es un valor aleatorio.

$$\vec{\vartheta}(i+1) = \begin{cases} 0, & T(\vec{X}(i+1)) > R \\ 1, & \text{otherwise} \end{cases} \tag{7}$$

$$T(\vec{X}(i+1)) = \tanh(\vec{X}(i+1)) = \frac{\exp^{-\tau(\vec{X}(i+1))} - 1}{\exp^{-\tau(\vec{X}(i+1))} + 1} \tag{8}$$

$$A(\vec{X}(i+1)) = \arctan(\vec{X}(i+1)) = \left| \frac{2}{\pi} \arctan\left(\frac{\pi}{2}\vec{X}(i+1)\right) \right| \tag{9}$$

$$\vec{\vartheta}(i+1) = \begin{cases} 0, & A(\vec{X}(i+1)) > R \\ 1, & \text{otherwise} \end{cases} \tag{10}$$

$$S_2\{\vec{X}(i+1)\} = \text{sigmoid}_2(\vec{X}(i+1)) = \frac{1}{1 + e^{-10\{(\vec{X}(i+1)) - 0.5\}}} \tag{11}$$

$$\vec{\vartheta}(i+1) = \begin{cases} 0, & S_2(\vec{X}(i+1)) > R \\ 1, & \text{otherwise} \end{cases} \tag{12}$$