ALGORITMO DE BUSQUEDA VORAZ PARA LA OBTENCIÓN DE UN ARBOL EXTENDIDO DE COSTO MINIMO MODIFICADO

Jesús Mendoza Verduzco[1] y Fidel González Gutiérrez[2]

[1]Instituto Tecnológico de Colima, Tecnologías de la Información; Avenida Tecnológico No. 1, Colonia liberación, C.P: 28976, Villa de Álvarez, Colima, 15460679@itcolima.edu.mx

[2]Universidad Politécnica de Querétaro, Carretera estatal 420 S/N, El Marqués, C.P: 762440, Querétaro, fidel.gonzalez@upq.mx

RESUMEN

El artículo a presentar en este documento fue realizado con el objetivo de generar una solución óptima al problema de MLC (Corredor de Longitud Mínima) en un grafo poligonal a través de un algoritmo codificado. Se realizó la solución por medio de un algoritmo de búsqueda voraz en estructura de camino y otro de estructura de ciclo; estos son estrategias de búsquedas que siguen un conjunto de reglas que consisten en elegir la opción óptima en cada paso que se ha llevado a cabo, con tal de llegar a una solución general óptima.

Se llevó a cabo la codificación del algoritmo por medio de un lenguaje de programación de alto nivel llamado Wolfram, el cual es un lenguaje general, sirve como el idioma de interfaz para Mathematica, en el cual ha sido realizado este proyecto. La facilidad que brinda es la esencial para realizar grafos, la principal característica de estos es que es un conjunto de nodos o vértices y arcos o aristas. Generalmente, cualquier problema puede representarse mediante un grafo.

ABSTRACT

The article to present in this document was made with the objective of generate an optimal solution to the problem of MLC (Minimum Length Corridor) in a polygonal graph through a coded algorithm. The solution was made through a greedy search algorithm of path structure and other of loop structure; these are search strategies that follow a set of rules that consists in choosing the optimal solution in each step that has taken place, to find the general optimal solution.

The coding of the algorithm was made through a high level programming language called Wolfram, is a general language, it serves as the interface language for Mathematica, in which this project has been made. The facility that provides is the essential for make graphs; the main feature of these is that it is a set of nodes or vertices and arcs or edges. Usually, any problem can be represented through a graph.

Palabras Clave: Algoritmo, grafo poligonal, búsqueda voraz, lenguaje de programación, vértices y aristas.

INTRODUCCIÓN

Los algoritmos nos ayudan en nuestra vida cotidiana, estos nos permiten realizar una actividad mediante pasos sucesivos que no generen dudas para la persona que los realizará, mediante el seguimiento de un conjunto de reglas o instrucciones claramente definidas se podrá llegar a un estado final obteniendo así una o más soluciones dependiendo de su eficacia. Los algoritmos resaltan por su eficacia, lo que significa que todas las operaciones deben ser suficientemente sencillas para poder ser realizadas de manera ordenada en modo exacto y en un tiempo finito.

Los algoritmos de este artículo han sido realizados con el objetivo de dar solución al problema MLC (Corredor de Longitud Mínima) en un grafo poligonal, para llevar a cabo una simulación, así la solución óptima es por medio de un camino tenga en su paso a todos los polígonos que se encuentran en el grafo, pasando por al menos un vértice o nodo de su representación gráfica. Los grafos nos proporcionan una forma gráfica de ver rutas, conexiones o elementos relacionados en un problema, realizando procesos que nos lleven a obtener la ruta o las conexiones que menos recursos utilicen, permitiéndonos así aprovecharlos al máximo.

El objetivo de la investigación a grandes rasgos es codificar una serie de algoritmos que nos permitan obtener el árbol de expansión mínima en un grafo poligonal cumpliendo con ciertos requisitos primordiales, como lo son la codificación del algoritmo en lenguaje Wolfram en la IDE Mathematica, obtener el árbol o camino de expansión mínima, pasando por todos los polígonos del grafo, el vértice raíz siempre debe iniciar en los vértices de la periferia, realizar una comparativa con todas las posibles soluciones de expansión mínima con los vértices de la periferia.

MARCO TEÓRICO

El estudio de la teoría de grafos se dio inicio con una teoría que tuvo lugar en 1736, en un artículo de Leonhard Euler. El trabajo surgió de un problema conocido como el problema de los puentes de Königsberg.

Durante el Siglo XVIII, la ciudad de Königsberg, en Prusia Oriental estaba dividida en cuatro zonas por el río Pregel. Había siete puentes que comunicaban estas regiones, tal y como se muestra en la figura 1. Los habitantes de la ciudad hacían paseos dominicales tratando de encontrar una forma de caminar por la ciudad, cruzando cada puente una sola vez, y regresando al lugar de partida.

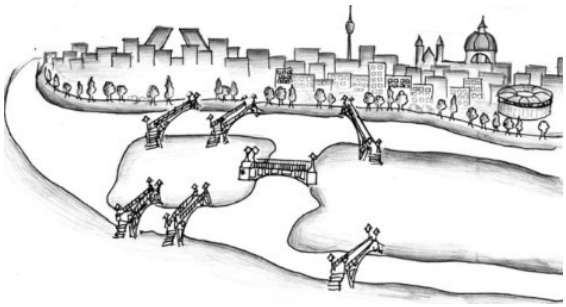


Figura 1. Dibujo de los puentes de Königsberg.

Para resolver este problema, Euler representó las cuatro zonas como cuatro puntos, y los puentes como aristas que unen los puntos, tal y como se muestra en la figura 2.

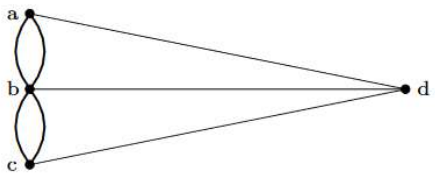


Figura 2. Representación de los puentes por medio de un grafo con vértices y aristas.

La solución que dio Euler al problema establece que:

- Un grafo contiene un circuito euleriano si y solo si todos sus vértices son pares.
- Un grafo contiene un camino euleriano si y solo si tiene dos vértices impares y los otros vértices pares.

La solución a este problema dada por Euler es considerada el primer teorema de Teoría de Grafos.

Se define a un grafo como una estructura de datos que permite representar diferentes tipos de relaciones entre objetos. En un grafo se distinguen básicamente dos elementos: los vértices y las aristas, que conectan un vértice con otro. Por ejemplo, la figura 3 representa algunas de las ciudades de México y la conexión entre ellas. Los vértices en el grafo representan a las ciudades, mientras que las aristas representan a los caminos entre ciudad y ciudad. Las aristas tienen un valor que representa el costo de viajar de una ciudad a otra.

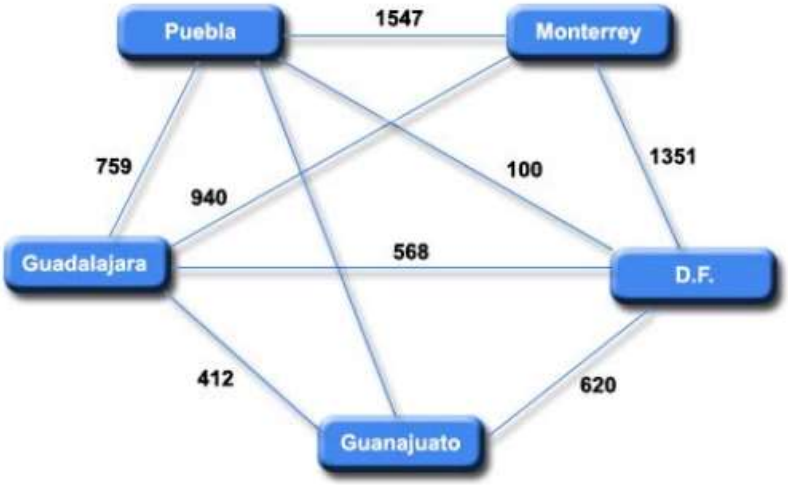


Figura 3. Ejemplo de un grafo.

El grafo de la figura 4 es un grafo poligonal, el cual ha sido asignado para resolverse con 4 distintos algoritmos para al final ser comparados y así tener una solución óptima del problema MLC (Corredor de Longitud Mínima).

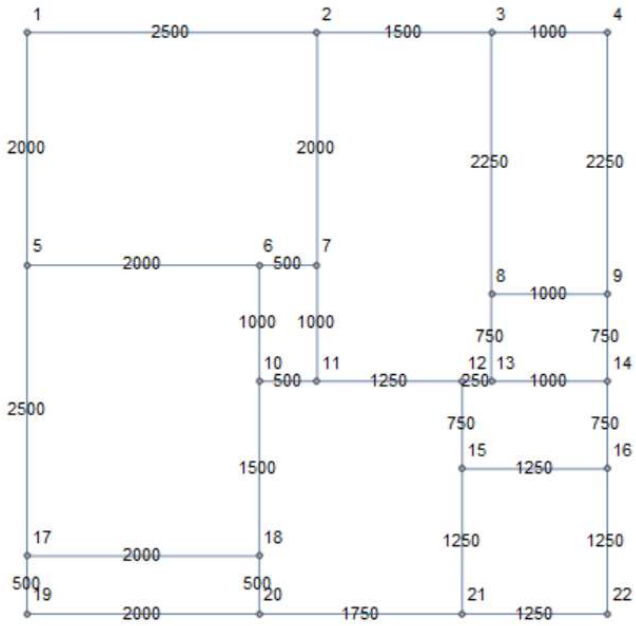
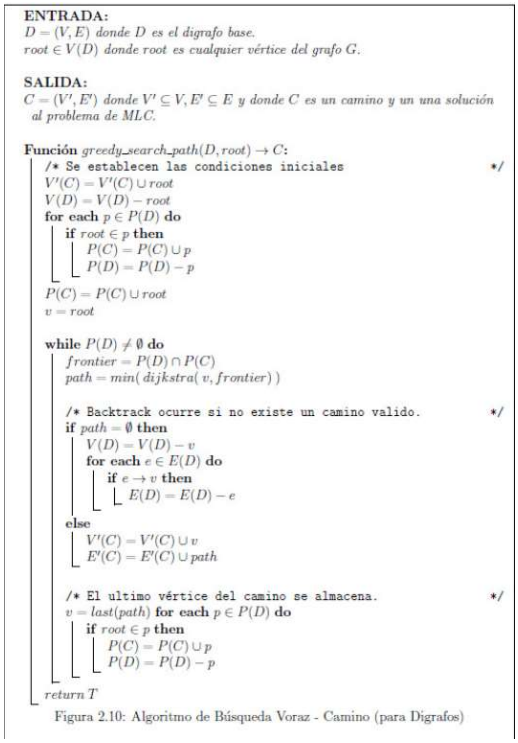


Figura 4. Grafo poligonal propuesto.

Los grafos tienen la necesidad de ser resueltos, pues se requiere de una ruta o árbol de expansión mínima para poder tener una solución óptima de los recursos o cualquier objetivo al que se quiera llegar, para la resolución de esto, se requieren algoritmos, que, según Osvaldo Cairó: *"Formalmente definimos un algoritmo como un conjunto de pasos, procedimientos o acciones que nos permiten alcanzar un resultado o resolver un problema".*

Los algoritmos en el área de informática son representados a manera de pseudocódigo, una forma escrita de un algoritmo de forma que cualquier persona del mundo de la programación pueda comprenderlo y lo pueda escribir en forma de código en cualquier lenguaje de programación. Es una forma de diagramar un algoritmo para resolver un determinado problema, sin atenerse a ningún lenguaje de programación en específico.

Se ha presentado el pseudocódigo de la figura 5, el cuál refiere a una serie de instrucciones para resolver el grafo poligonal de la figura 4 por medio de la codificación de un algoritmo voraz con estructura de camino y de ciclo.



ENTRADA:
$D = (V, E)$ *donde* D *es el digrafo base.*
$root \in V(D)$ *donde root es cualquier vértice del grafo* G.

SALIDA:
$C = (V', E')$ *donde* $V' \subseteq V, E' \subseteq E$ *y donde* C *es un camino y un una solución al problema de MLC.*

Función $greedy\_search\_path(D, root) \rightarrow C$:

```
/* Se establecen las condiciones iniciales                    */
V'(C) = V'(C) ∪ root
V(D) = V(D) − root
for each p ∈ P(D) do
    if root ∈ p then
        P(C) = P(C) ∪ p
        P(D) = P(D) − p
P(C) = P(C) ∪ root
v = root

while P(D) ≠ ∅ do
    frontier = P(D) ∩ P(C)
    path = min( dijkstra( v, frontier) )

    /* Backtrack ocurre si no existe un camino valido.        */
    if path = ∅ then
        V(D) = V(D) − v
        for each e ∈ E(D) do
            if e → v then
                E(D) = E(D) − e
    else
        V'(C) = V'(C) ∪ v
        E'(C) = E'(C) ∪ path

    /* El ultimo vértice del camino se almacena.              */
    v = last(path) for each p ∈ P(D) do
        if root ∈ p then
            P(C) = P(C) ∪ p
            P(D) = P(D) − p
return T
```

Figura 2.10: Algoritmo de Búsqueda Voraz - Camino (para Digrafos)

Figura 5. Algoritmo de búsqueda voraz - Estructura de camino.

MÉTODOS Y MATERIALES

Se hizo uso de una metodología con características prototipales, haciendo uso de un repositorio en la plataforma GitLab, en el que se llevaba un control de versiones y se compartían recursos con el resto de los integrantes. A su vez se generaban reportes semanales de manera individual, describiendo lo realizado durante la semana, las complicaciones tenidas y la forma en que fueron resueltas. Se comentó todo proceso realizado dentro del código y se dio una explicación de lo que el código realiza.

Los materiales utilizados son los siguientes:

- PDF con el contenido del algoritmo de reducción en caso de tener que utilizarlo, también el algoritmo de búsqueda voraz – Estructura de camino, y la definición de este último, pero con estructura en ciclo. Los primeros dos contenían su definición y los pasos a seguir, al igual que el pseudocódigo.
- Archivo TXT con el contenido de una tabla la cual se usó para obtener los polígonos dentro del grafo por medio de coordenadas.
- Archivo nb (Mathematica) para usarlo como ejemplo del lenguaje, y así facilitar la adaptación a él.

RESULTADOS

Después de todo el proceso realizado durante todo este proyecto se llegó a la solución óptima, con un tiempo de procesamiento muy aceptable, por medio del algoritmo de búsqueda voraz – Estructura de camino, que la única diferencia que guarda con el de ciclo es que el último vértice se mantiene conectado con el primero, en la figura 6 se puede observar el resultado obtenido de ambos algoritmos (La línea de color rojo marca el camino de la solución).

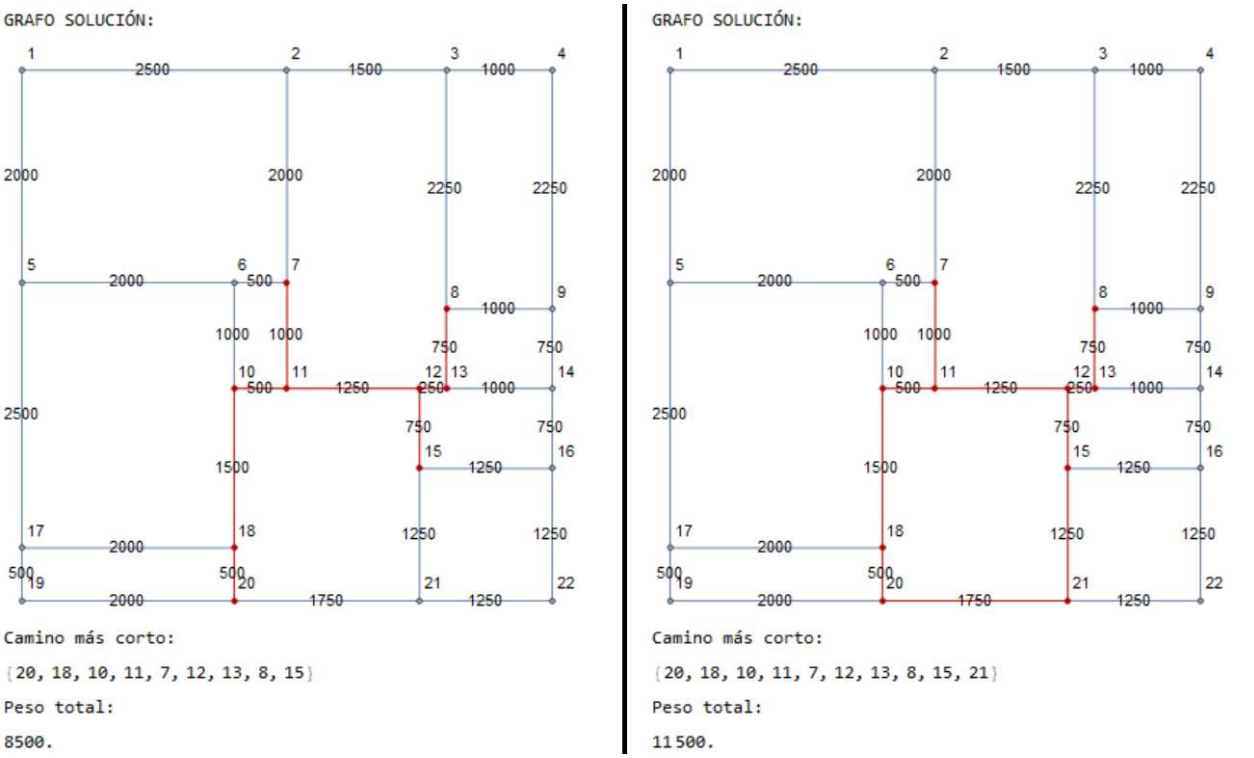


Figura 6. Camino de la solución óptima con el algoritmo de búsqueda voraz - Estructura de camino (izquierda) y algoritmo de búsqueda voraz – Estructura de ciclo (derecha).

CONCLUSIONES

Los algoritmos para la obtención de un árbol o camino de extensión mínima a partir de un grafo con forma poligonal, han sido desarrollados de manera satisfactoria, cumpliendo con las condiciones iniciales planteadas por el investigador, consumiendo un tiempo de ejecución bastante aceptable y entregando una aproximación óptima a la solución del problema. A pesar de no tener como objetivo una aplicación de estos algoritmos, quedan como herramientas de optimización que podrán ser utilizadas en los ámbitos que sean requeridos.

La investigación ha sido llevada a cabo con gran motivación y satisfacción, ya que durante el transcurso de ella se han encontrado varias problemáticas que han sido resueltas sin muchas complicaciones, unas más que otras, pero el hecho de comprender y usar un lenguaje de programación desconocido, fue de gran provecho, pues se ha podido contar con la introducción a él por parte del investigador. Es completamente enriquecedor poder trabajar con algo nuevo, resulta muy gratificante el poder realizar un proyecto desde cero por medio de un mundo desconocido.

BIBLIOGRAFÍA

CAIRÓ, O., (2006). “Metodología de la programación. Algoritmos, diagramas de flujos y programas”. México D.F., México: ALFAOMEGA GRUPO EDITOR, S.A. DE C.V.

ÁLVAREZ, M., (1994). “Aplicación de la teoría de grafos a la planificación y programación de proyectos” en *El Instituto Eduardo Torroja - CSIC,* pp. 49-60. Consultada en http://digital.csic.es/bitstream/10261/54982/1/1215.pdf (fecha de consulta 22-07-2018).