

MÉTODOS COMPUTACIONALES ORIENTADOS A LA OBTENCIÓN DEL ÁRBOL EMST: UN APOORTE AL PROBLEMA DE STEINER

METHODS OF COMPUTER ORIENTED TO THE OBTAINING OF EMST TREE: A CONTRIBUTION TO THE STEINER PROBLEM

Pablo López¹, Carlos Yañez¹, Luzmila Pró, Jaime Alcalde²

RESUMEN

Un Árbol Recubridor Euclídeo Mínimo (EMST: Euclidean Minimum Spanning Tree) es un árbol o subgrafo sin ciclos, que permite interconectar todos los vértices y minimizar la longitud total de las aristas de un grafo determinado.

No obstante esta minimización, a través de la adición de nuevos vértices al conjunto original se logra efectuar una minimización más óptima. Estos árboles se denominan Árboles de Steiner y tienen grandes aplicaciones, particularmente en el diseño de redes de transportes y de comunicaciones.

Resultados esperados: Obtención del Árbol EMST.

Palabras clave: Árbol EMST, Árbol de Steiner, Algoritmo Genético.

ABSTRACT

A Euclidean Minimum Spanning Tree (EMST) is a tree or subgraph without cycles, that allows to interconnect all the vertices and to diminish the length overall of the edges of a certain graph.

Despite this minimization, through the addition of new vertices to the original set it is managed to carry out an optimal minimization but. These trees Trees of Steiner denominate themselves and have great applications, particularly in the design of networks of transports and communications.

Key words: EMST tree, Steiner tree, genetic algorithm.

1. INTRODUCCIÓN

Dados N puntos en el plano, construir un árbol tal que su longitud total sea la mínima posible y cuyos nodos sean los puntos dados. Los N puntos constituyen un grafo, por lo cual el problema consiste en determinar un árbol o subgrafo sin ciclos que contenga todos los nodos y cuya longitud sea mínima. Este tipo de árboles se denomina: Árbol Recubridor Euclídeo Mínimo (EMST: Euclidean Minimum Spanning Tree).

Sin embargo, el árbol EMST hallado puede no representar la solución óptima al problema, es decir, que pueden existir otros árboles que, conteniendo todos los vértices del problema original, proporcionen una longitud menor que la del EMST. A este tipo de árboles se les denomina árboles de Steiner (Steiner Trees) y son generados mediante la adición de nuevos vértices (puntos de Steiner) al conjunto original. Mediante estos vértices, se busca una minimización de la longitud proporcionada por los árboles EMST [2] [3]

¹ Docente del Departamento Académico de Ciencias de la Computación- FISI, Universidad Nacional Mayor de San Marcos.

² Docente del Departamento Académico de Investigación de Operaciones- FCM, Universidad Nacional Mayor de San Marcos.
E-mail: {plopezv, cyanezd, jalcalde}@unmsm.edu.pe

Existen muchas aplicaciones correspondientes a los Árboles EMST [7], [8], entre ellas podemos citar: Sistemas de Telecomunicaciones de Datos, Diseño de Redes de Sistema Eléctrico, Planificación de Transporte Urbano, etc. El EMST correspondiente permite que el coste de la instalación sea mínimo.

2. ÁRBOL RECUBRIDOR EUCLÍDEO MÍNIMO

Un Árbol Recubridor Euclídeo Mínimo (EMST: Euclidean Minimum Spanning Tree) es un subgrafo sin ciclos, conexo que minimiza la suma de los costos de las aristas de un grafo determinado [1].

La figura 1 muestra un grafo de 6 nodos con 10 aristas que representan costos de vincular los vértices asociados, y el árbol EMST obtenido que representa la optimización de los costos conteniendo todos los vértices.

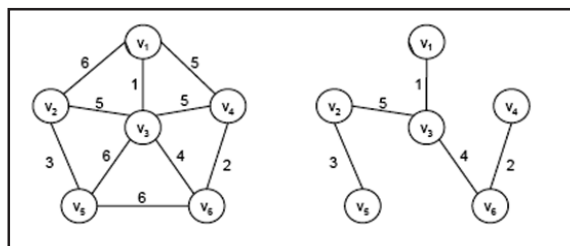


Figura N.º 1.

3. GENERACIÓN DEL EMST

Existen muchas técnicas y/o métodos [1] asociadas a la obtención del árbol EMST a partir de un grafo de N vértices. Podemos citar entre ellas:

3.1. Método de PRIM

Es un método heurístico voraz que parte de un vértice seleccionado aleatoriamente, y va generando el árbol de recubrimiento mediante la incorporación de nuevos vértices en cada iteración, hasta llegar a cubrir todos los vértices del grafo.

Sea el grafo $G = (V, A)$, tal que:

- V : Conjunto de vértices
- A : Conjunto de aristas
- $x, y \in V$

En el grafo $G = (V, A)$ se identifican las siguientes particiones:

- U : Vértices enlazados por las aristas del árbol que se está generando

- $V - U$: Vértices que se hallan en V pero no en U

Además:

- T : Conjunto de aristas del árbol a generar

ALGORITMO DE PRIM

```

Procedure PRIM
  // Inicializar  $T$  sin aristas
   $T = \emptyset$ 
  // Inicializar  $U$  seleccionando aleatoriamente
  un vértice  $x_k \in V$ 
   $U = \{x_k\}$ 
  // Repetir mientras  $U \neq V$ 
  While  $U \neq V$  Do
    Escoger  $(x, y)$  de costo mínimo /  $x \in U, y \in V - U$ 
    Agregar  $(x, y)$  a  $T$ 
    Agregar  $x$  a  $U$ 
  EndWhile
EndProcedure

```

3.2. Método de KRUSKAL

Es un método heurístico voraz que comienza a partir de un *bosque* de árboles constituidos únicamente por un vértice cada uno, tal que en cada iteración se conecta un par de árboles a través de una arista, quedando en el bosque únicamente un árbol.

Este método evalúa en cada iteración la conexión de árboles a través de una arista. Si se detecta que la arista conecta dos vértices pertenecientes al mismo árbol, no se agrega para evitar la aparición de ciclos en el árbol de expansión. En caso contrario, se conecta ambas componentes en una única componente conexa.

Al final, si todos los vértices están en una componente conexa única, se habrá encontrado el árbol de expansión mínimo.

Sea el grafo $G = (V, A)$, tal que:

- V : Conjunto de vértices
- A : Conjunto de aristas

Identificamos las siguientes estructuras de datos:

- T : Conjunto de aristas del árbol a generar
- B : Tabla de enteros de tamaño N
- $a \in A$, una arista

Procedure Kruskal

```

//Inicializar T con los vértices de G, pero sin aristas
T = {V, ∅}
//Ordenar A crecientemente por el costo
Ordenar(A)
For i = 1 To N
    Bi = i
EndFor
i = 1;
While Cardinal(T) < N-1 Do
    a = SeleccionarArista()
    //Probar conexión de dos vértices de árboles distintos
    If BOrigen(a) ≠ BDestino(a) Then
        //Agregar a al conjunto de aristas T
        Agregar(a)
        //Unir en una única componente conexa
        Unir(B, Origen(a), Destino(a))
    EndIf
    i = i + 1
EndWhile
EndProcedure

```

3.3. Obtención del árbol EMST

Los métodos anteriores combinados con técnicas de búsqueda de caminos logran la implementación del siguiente procedimiento en VB, que permite obtener el árbol óptimo EMST a partir de un conjunto de vértices fuertemente conexo.

Private Sub cmdEMST_Click()

```

Dim i As Integer, j As Integer, k As Integer, A As Integer,
B As Integer
Dim L1 As Integer, L2 As Integer, H As Integer, TG As
Integer
Dim Sw As Boolean
cmdCAMINOS.Enabled = True: mnuCamino.Enabled = True
Call OrdenarAristas
TG = 0: P = 0: k = 1
Do While k <= T And TG < N - 1
    A = Arista(VVV(k), 1): B = Arista(VVV(k), 2)
    H = 0
    Sw = True
    i = 1
    Do While i <= P And Sw
        Mtx1 = Mtx_LoadString(Camino(i))
        L1 = LBound(Mtx1): L2 = UBound(Mtx1)
        HA = «0»: HB = «0»
        For j = L1 To L2
            If Mtx1(j) = A Then
                HA = «1»
            End If
            If Mtx1(j) = B Then
                HB = «1»
            End If
        Next j
        Select Case HA & HB
            Case «00»
                H = 0
            Case «11» ' Arita forma ciclos
                Sw = False
                H = 1
        End Select
    End While
    Loop
    For k = 1 To P
        Mtx1 = Mtx_LoadString(Camino(k))
        For i = LBound(Mtx1) To UBound(Mtx1) - 1
            A = Mtx1(i)
            B = Mtx1(i + 1)
            Call PintarArista(A, B, vbRed)
            j = 1
            Do While j <= T
                If (Arista(j, 1) = A And Arista(j, 2) = B) Or _
                    (Arista(j, 1) = B And Arista(j, 2) = A) Then
                    txtPeso(j).BackColor = 16777152
                j = j + 1
            End While
        Next i
    Next k

```

```

Case Else
    Sw = False
    H = 1
    If Mtx1(L1) = A Then
        AdicionarNodoCamino(-1,B)
        TG = TG + 1
    ElseIf Mtx1(L1) = B Then
        AdicionarNodoCamino(-1,A)
        TG = TG + 1
    ElseIf Mtx1(L2) = A Then
        AdicionarNodoCamino(1,B)
        TG = TG + 1
    ElseIf Mtx1(L2) = B Then
        AdicionarNodoCamino(1,A)
        TG = TG + 1
    Else
        Sw = True
        H = 0
    End If
End Select
i = i + 1
Loop
If H = 0 Then
    P = P + 1
    Camino(P) = CStr(A) & «.» & CStr(B)
    TG = TG + 1
End If
k = k + 1
Loop
For k = 1 To P
    Mtx1 = Mtx_LoadString(Camino(k))
    For i = LBound(Mtx1) To UBound(Mtx1) - 1
        A = Mtx1(i)
        B = Mtx1(i + 1)
        Call PintarArista(A, B, vbRed)
        j = 1
        Do While j <= T
            If (Arista(j, 1) = A And Arista(j, 2) = B) Or _
                (Arista(j, 1) = B And Arista(j, 2) = A) Then
                txtPeso(j).BackColor = 16777152
            j = j + 1
        End While
    Next i
Next k
End Sub

```

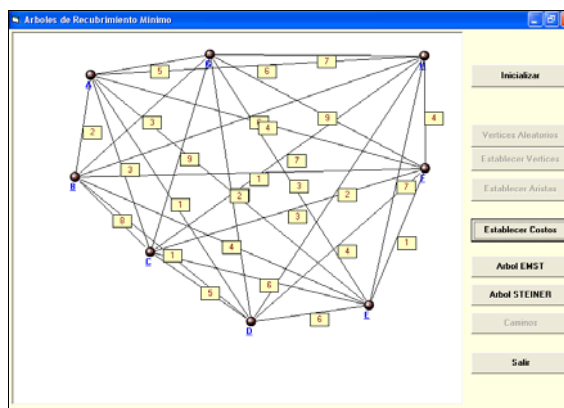


Figura N.º 2.

La figura N.º 2 muestra el grafo de entrada de 26 vértices fuertemente conexo, esto es, todos los vértices están conectados entre sí, y no existen ciclos en un mismo vértice.

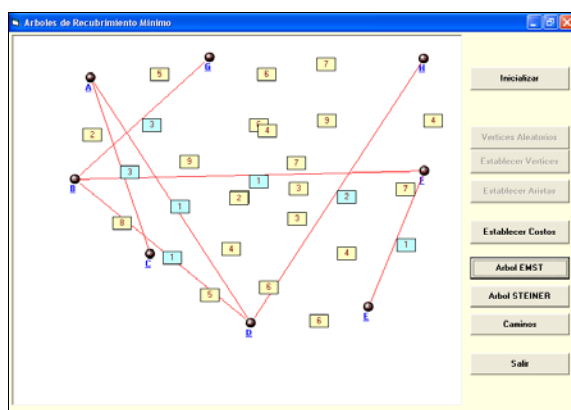


Figura N.º 3.

La figura N.º 3 visualiza el árbol EMST asociado, lo cual proporciona el costo mínimo asociado a las aristas.

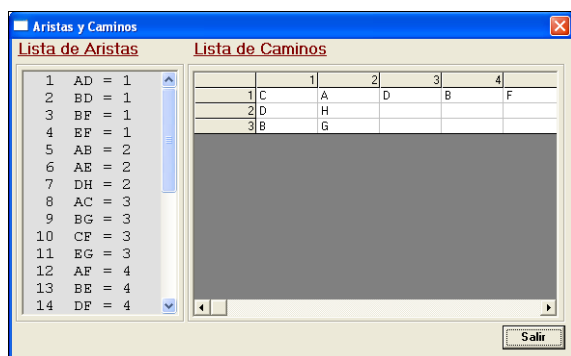


Figura N.º 4.

La figura N.º 4 determina los caminos correspondientes al árbol EMST.

4. ÁRBOLES DE STEINER

Representan una solución más óptima que los árboles EMST. Son generados a partir de los árboles EMST mediante el agregado de vértices adicionales al conjunto original de vértices. Estos vértices adicionales se denominan *Vértices o Nodos de Steiner* [2], [3], [4]. La figura N.º 5 muestra la generación del árbol de Steiner a partir del árbol EMST

Mediante estos vértices se busca efectuar una minimización más óptima de los árboles de recubrimiento.

El árbol de Steiner satisface los siguientes criterios:

- Las aristas adyacentes concurrentes en un vértice común, forman un ángulo por lo menos de 120 grados
- No debe existir cruces de aristas.
- Cada punto de Steiner tiene grado tres, es decir, existen tres aristas que concurren a él.

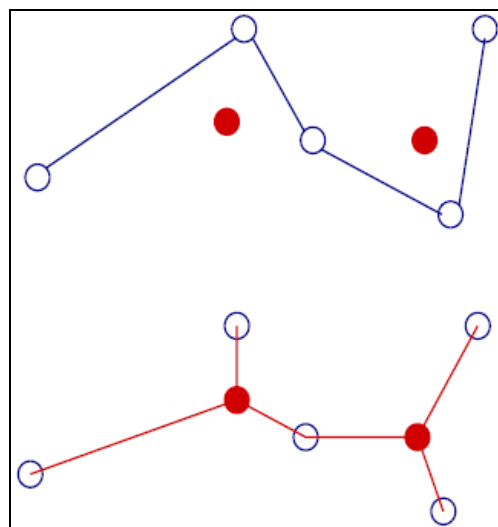


Figura N.º 5.

4.1. Generación del árbol de Steiner-Algoritmos genéticos

La generación de Árboles de Steiner representa un tipo de problemas en el cual el costo computacional es elevado[5], [6]. Se proporciona una metodología orientada a su generación a través de técnicas evolutivas. Los algoritmos genéticos efectúan una simulación del comportamiento de las especies de la naturaleza, donde prevalece el más fuerte. Utilizando una representación adecuada para el problema, y mediante la actuación de unos cuantos operadores genéticos de selección, cruce y mutación, se logra obtener una buena aproximación

Cada elemento en el sistema que va a evolucionar bajo los AGs, es con frecuencia designado por cromosoma y posee un conjunto de parámetros que identifican una potencial respuesta. Un cromosoma deberá poseer la información identificadora de los puntos de Steiner necesarios para obtener la respuesta, una vez añadidos a los puntos iniciales, que son fijos para cada caso.

CONSIDERACIONES

- Para un conjunto de N vértices, se tiene como máximo N-2 vértices de Steiner. Una particularidad de los algoritmos genéticos es la iniciación del conjunto de cromosomas, designado por población, realizado de forma totalmente aleatoria. Existe la posibilidad de que algunos cromosomas se sitúen fuera de la envolvente convexa que limita los puntos iniciales, presentando, de esa forma, una pérdida de esfuerzo computacional.

- Usaremos la propiedad de la envolvente convexa:

Sean $P_1, P_2, P_3, \dots, P_{N-1}, P_N$ puntos de R^2 .
 Tal que
 $0 \leq x_i \leq 1$
 $x_1 + x_2 + x_3 + \dots + x_{N-1} + x_N = 1$
 El conjunto de todas las combinaciones lineales
 $x_1 P_1 + x_2 P_2 + x_3 P_3 + \dots + x_{N-1} P_{N-1} + x_N P_N$
 es un conjunto convexo

OPTIMIZACIÓN

Existen dos metas: Aceleración de la convergencia del algoritmo y aumento de la fiabilidad de la respuesta obtenida.

Área de supervivencia

Para cada punto, se debe delimitar un área mínima donde éste ejerce sin oposición su influencia.

La eliminación directa de los demás puntos en dicha región de supervivencia de un cromosoma se justifica porque dos puntos de Steiner muy «próximos» no aportan beneficio directo

Verificación de los árboles de Steiner previos

Una vez obtenidos los árboles de Steiner previos, se repasan los puntos de Steiner obtenidos y se analizan sus valencias.

La heurística correspondiente está compuesta por las siguientes fases:

- Eliminar los vértices que presentan valencia 1.
- Eliminar los vértices con valencia 2, y unir los adyacentes mediante una nueva arista. Esto se ilustra en la Figura N.º 6.

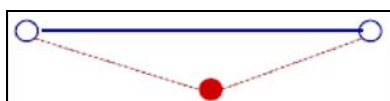


Figura N.º 6.

- Para vértices de valencia 3, el punto es sustituido usando la técnica geométrica de Torricelli. Mostrada en la Figura N.º 7.

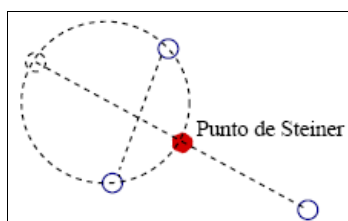


Figura N.º 7.

5. CONCLUSIONES

La investigación correspondiente a los árboles EMST permite señalar que el procesamiento de los árboles de Steiner implica la adopción de métodos heurísticos asociados a técnicas evolutivas.

El procesamiento árboles de Steiner es un problema NP duro, que no puede ser implementado en tiempo polinomial, porque implica un elevado costo computacional.

6. TRABAJO FUTURO

Esta investigación establece las bases para diseñar e implementar —en un futuro próximo— los métodos heurísticos correspondientes a técnicas evolutivas, que permitan llegar a la optimización de un conjunto de N vértices fuertemente conexo, más allá del obtenido por los árboles EMST, permitiendo obtener inclusive la solución del Problema de Steiner Generalizado.

7. REFERENCIAS BIBLIOGRÁFICAS

- [1] Altamirano, Néstor Daniel. «Algoritmos eficientes para la generación de árboles de recubrimiento mínimo», 33.ª Jornadas Argentinas de Informática e Investigación Operativa, 2002.
- [2] Mário Jesus, Sérgio Jesus y Alberto Márquez, «Optimización de Algoritmos Genéticos en el Procesamiento de Árboles de Steiner».
- [3] F. Hwang, D. Richards and P. Winter. «The Steiner problem», en Elsevier Science Publishers B.V., 1992.
- [4] Dietmar, Cieslik (1998), *Steiner minimal trees*. Kluwer Academic Publishers.
- [5] M. Garey, R. Graham and D. Johnson, «The complexity of computing Steiner minimal trees». en SIAM Journal Applied Mathematics, vol. 32, pp. 835-850, 1997.
- [6] L. Kou, G. Markowsky and L. Berman (1981), «A fast algorithm for Steiner trees», en Acta Informática vol. 15, pp. 141-145, 1981
- [7] Nesmachnow, Sergio; Héctor, Cancela y Enrique Alba (2003). *Técnicas evolutivas aplicadas al diseño de redes confiables*. Instituto de Computación—Universidad de la República, Uruguay.
- [8] Shin'ichi Wakabayashi, «A Genetic Algorithm for Generating a set of rectilinear Steiner Trees» en VLSI Interconnect Layout, Faculty of Engineerin, Hiroshima University, 2001.