Algoritmos locales para detectar conjuntos de corte

D. Flores-Penaloza[†], J. Rivera[*], M.Soriano[*], J. Urrutia[*], C. Velarde[‡]
[*] Instituto de Matemáticas, UNAM, México.
[†] Departamento de Matemáticas, Facultad de Ciencias, UNAM, México.
[‡] Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, UNAM, México.

***Abstract*—Consideramos un problema clásico de teoría de gráficas. Dada una gráfica simple plana conexa G de n vértices, se desean conocer los conjuntos de aristas de cardinalidad mínima que desconecten a G. Existen diversos algoritmos que dan solución a este problema, sin embargo, todas estas soluciones requieren de que se conozca en todo momento la topología de la gráfica. Estas soluciones globales (denominadas así porque es necesario conocer la gráfica en su totalidad) son deficientes cuando se intentan modelar redes dinámicas (redes donde los nodos están en constante movimiento), en su lugar se utilizan algoritmos donde no es necesario tener el conocimiento pleno de la topología de la red (denominados algoritmos locales). En este trabajo presentamos una solución local al problema de la identificación de los conjuntos de corte de cardinalidad mínima, el cual presenta una complejidad de $O(n^6)$. También presentamos un algoritmo de enumeración efectiva, este algoritmo local reporta cada vértice y arista de G en tiempo $O(n \log n)$.**

I. Introducción

En años recientes se ha observado que las gráficas geométricas son ideales para modelar redes de computadoras en las que la posición es conocida, tal como redes *ad-hoc*, redes celulares y redes de sensores [6].

Estas redes se caracterizan por tener un alto grado de movilidad, por lo que normalmente se catalogan como redes naturalmente dinámicas. Como consecuencia, la aplicación de algoritmos con conocimiento global de la red empiezan a ser ineficientes para este tipo de escenarios [6]. Muchos algoritmos trabajan bajo la suposición de que se cuenta con información global y centralizada sobre la topología de una red [7], lo cual no es conveniente al tratarse de redes dinámicas, ya que el uso de memoria y el mantenimiento de información es muy costoso. La construcción y mantenimiento distribuido (local por nodo) es preferible debido a los recursos limitados (redes de sensores) y a la movilidad de los nodos [7]

Un algoritmo local se modela como un agente con memoria constante, que tiene la capacidad de migrar de un nodo a otro, y que toma decisiones basado sólo en la información que contiene en su memoria, y la ubicación y conectividad del nodo en el que se encuentra. Los algoritmos locales son algoritmos distribuidos, los cuales son altamente escalables, además de ser tolerantes a fallas y cambios sobre la topología de la red [5].

En esta investigación presentamos una propuesta de solución al problema de detección de conjuntos de aristas de corte en una gráfica, mediante la utilización de algoritmos locales. Con ésto se intenta dar una solución distribuida y altamente escalable a este problema tan bien conocido. Los resultados de esta investigación están restringidos para gráficas (redes) geométricas planas simples y conexas.

II. Preliminares

Un algoritmo local se modela como un agente con memoria constante, que tiene la capacidad de migrar de un nodo a otro, y que toma decisiones basado sólo en la información que contiene en su memoria, y la ubicación y conectividad del nodo en el que se encuentra. Por lo tanto, todo nodo o vértice de una gráfica sólo tiene información de vértices que se encuentran a un salto de distancia de él [6].

En [6], se plantea el modelo de algoritmos locales en redes de la siguiente manera: sea G una gráfica plana. Suponga que cada vértice v en G conoce su posición en la gráfica (sus coordenadas) y la posición de sus vecinos. Existe un algoritmo determinístico que permitirá a un agente A colocado en un vértice u viajar a un vértice v bajo las siguientes condiciones:

- A tiene memoria constante. En todo momento A conoce sólo la posición de u y v, y la posición de un número constante de nodos.
- Cuando el agente visita un vértice w, este puede usar la lista de vértices adyacentes a él (sus posiciones).
- A no deja marcas en su recorrido.

La primera restricción trae como consecuencia que el agente A no tenga conocimiento pleno de la topología de G [6]. Podría parecer que algunas de las restricciones anteriores son innecesarias, sin embargo, si consideramos el comportamiento de redes tan grandes como Internet, entonces éstas se convierten claramente en condiciones relevantes. Bajo este modelo, muchos problemas clásicos de teoría de gráficas han sido resueltos, tales como la construcción de gráficas planas [8], problemas de ruteo [3], construcción de árboles generadores [1], así como problemas de elección de líder [2].

Uno de los algorirmos referencia para el recorrido de gráficas planas es el algoritmo de recorrido de caras siguiendo la regla de la mano derecha [4], el cual fue inspirado en el

[†] Investigación parcialmente apoyada por los proyectos 168277 (CONACYT, Mexico) y IA102513 (PAPIIT, UNAM, Mexico).
[§] Autor responsable de la correspondencia: joshep.milo@gmail.com
.

recorrido de laberintos. El algoritmo es simple, indica que si un jugador en un laberinto camina por este colocando su mano derecha sobre el muro siempre cuidando de no despegarla, entonces el jugador eventualmente visitará cada muro del laberinto, si el laberinto es la cara de una gráfica plana, el jugador visitará cada arista y cada vértice de la cara.

III. Algoritmo de Enumeración Efectiva

Una gráfica geométrica plana conexa divide el plano en cierto número de regiones conexas, las cerraduras de estas regiones se denominan caras. Si se elimina una de las aristas frontera de una de las regiones, esta se mezclará y formará una nueva región (cara) con la región colindante a la arista que se eliminó. Si este procedimiento se realiza con todas las caras internas, al final quedará sólo una cara en la gráfica, que será la externa. Además, si durante el procedimiento se cuida de no romper la conexidad de la gráfica (no formar más componentes conexas) entonces al final del procedimiento la gráfica resultante será un árbol. Por tanto, para reportar todos los elementos de una gráfica basta con hacer un recorrido sobre ese árbol y mediante un conjunto de restricciones sencillas decidir cuando reportar un elemento.

El Algoritmo 1, al ejecutarse sobre una gráfica, reporta exáctamente una vez cada vértice y cada arista de G. A continuación presentamos algunas definiciones relevantes a este algoritmo.

Definición 1: Diremos que una arista $e = \{u, v\}$ es dominante sobre el vértice u si, al trazar una semirecta vertical a partir de u en la dirección $(0, 1)$, e forma el ángulo menor con esta semirecta.

Definición 2: Sea $e = \{u, v\}$ con $e \in E$ y $u, v \in V$, cuyas coordenadas de los vértices en el plano son $u = (x_1, y_1)$ y $v = (x_2, y_2)$. Se dirá que la arista e esta siendo revisada por la izquierda si durante el algoritmo, al estar posicionado el agente sobre el vértice u, $x_2 > x_1$. Para el caso donde $x_1 = x_2$, se dirá que la arista e esta siendo revisada por la izquierda si $y_2 > y_1$.

Definición 3: Sea f una cara interna en G y u el vértice con abscisa mayor en f (en caso de empate, u tendrá la ordenada menor de los vértices empatados). La arista de entrada en f será aquella arista de f que sea adyacente a u y que, al trazar una semirecta vertical a partir de u en la dirección $(0, -1)$ y medir los ángulos en sentido negativo, forme el ángulo menor en valor absoluto con la semirecta.

Sea $G = (V, E)$ una gráfica geométrica plana conexa, donde V denota el conjunto de los vértices pertenecientes a G y E el conjunto de las aristas. El algoritmo 1 describe el procedimiento para reportar cada arista y vértice de G exactamente una vez, a través de un recorrido completo de la gráfica.

Al realizar el recorrido de la gráfica se evita recorrer todas aquellas aristas que sean de entrada, por lo que el recorrido realizado sobre G será equivalente al recorrido sobre un árbol generador en la gráfica. Por ésto, cada arista será revisada dos veces, una en cada sentido, incluso las aristas de entrada, ya que antes de ser descartadas en el recorrido tendrán que ser evaluadas para ver si son aristas dominantes de algún vértice o verificar si deben ser reportadas o no.

Algoritmo 1 Algoritmo de enumeración efectiva

Entrada: Gráfica geométrica plana conexa $G = (V, E)$, u tal que $u \in V$
Salida: Reporta una vez cada vértice y arista de G

```
Sea e_u = {u, v} la arista dominante de u
Sea a = (u, v) una dirección sobre e_u
La arista de inicio del recorrido será e_u
La dirección de inicio del recorrido será a
repetir
    Sea a = (u, v)
    si e_u es la arista dominante de u entonces
        Reportar el vértice u
    fin si
    si e_u está siendo revisada por la izquierda entonces
        Reportar la arista e_u
    fin si
    si e_u es una arista de entrada entonces
        Ahora e_u será la arista siguiente en el recorrido de
        la cara siguiendo la regla de la mano derecha y sin
        tomar en cuenta la arista {u, v}
        Sea e_u = {u, w}, ahora a = (u, w)
    si no
        Ahora e_u será la arista siguiente en el recorrido de la
        cara siguiendo la regla de la mano derecha
        Sea e_u = {v, w}, ahora a = (v, w)
    fin si
hasta que e_u y a sean los valores de inicio
```

Un vértice será reportado sólo cuando él ha sido elegido como vértice u y la arista que se está revisando (e_u) es su arista dominante, lo cual sólo pasa una vez. Por otro lado, una arista sólo será reportada cuando sea revisada en un sentido, en particular, cuando u sea el vértice con coordenada menor.

Se puede demostrar que el algoritmo de enumeración efectiva (Algoritmo 1) tiene un costo total de tiempo $O(n \log n)$.

IV. Detección de conjuntos de corte de cardinalidad m

Una arista e es una arista de corte en G si al eliminarla, la subgráfica obtenida tiene más componentes conexas que G. Así mismo, un conjunto de corte c es un conjunto de aristas de G tal que $G \backslash c$ incrementa el número de componentes conexas en la subgráfica restante, mientras que la eliminación de cualquier subconjunto propio no lo hace.

Siguiendo estas definiciones, podemos obtener los siguientes lemas.

Lema 1: Una arista $e \in G$ es de corte si pertenece sólo a una cara.

Prueba 1: Si e perteneciera a dos caras en G, por tratarse de gráficas planas, por lo menos una de las caras sería un ciclo, por lo tanto, al eliminar a e de G exístirá por lo menos una trayectoria que una a los dos vértices pertenecientes a e, contradiciendo nuestra definición de aristas de corte. Por lo tanto, e sólo puede pertenecer a una cara en G.

Lema 2: Sea G una gráfica plana conexa, c un conjunto de corte de cardinalidad $k \geq 2$ y sea e_j un elemento de c. Al realizar $G \backslash e_j$, en la gráfica resultante, $c \backslash e_j$ sigue siendo un conjunto de corte de cardinalidad $k-1$ en $G \backslash e_j$.

Prueba 2: Por definición de conjunto de corte, para que aumenten el número de componentes conexas en G se deben eliminar todas las aristas que pertenecen al conjunto c, sin importar el orden, por lo tanto, al eliminar una arista, sin importar cual sea, aún se deben eliminar todas las aristas restantes en c para que la gráfica aumente el número de sus componentes conexas, por lo tanto el conjunto $c' = c \backslash e$ cumple con la definición de conjunto de corte, ahora de cardinalidad $k-1$.

Lema 3: Sea c un conjunto de corte y c_f el conjunto de caras a las que pertenecen las aristas de c. El conjunto c y el conjunto c_f tienen la misma cardinalidad.

Lema 4: Sea c un conjunto de corte de cardinalidad $k \geq 2$ y c_f el conjunto de las caras a las que pertenece c. En la gráfica dual, los elementos de c_f forman un ciclo.

Por el lema 4 sabemos que, para encontrar un conjunto de aristas de corte de tamaño m es necesario buscar una secuencia de m aristas que formen un ciclo de tamaño m con las caras a las que pertenecen.

El algoritmo 2 reporta dada una arista e, todos los conjuntos de aristas de corte de cardinalidad m a los que pertenece e. Es fácil ver que, según el tamaño del conjunto de corte es el número de caras que se deben recorrer, pues un conjunto de corte de tamaño m forma un ciclo de tamaño m en la gráfica dual. Por lo tanto la complejidad del algoritmo estará dada por el tamaño del conjunto de corte que se desea obtener.

Así como ejemplo tenemos que, dada una arista e, para encontrar un conjunto de aristas de corte de cardinalidad 2 a los que pertenece e sólo es necesario encontrar a que caras pertenece e, y hacer un recorrido sobre una de esas caras y por cada arista diferente de e revisar a que caras pertenece. Si se encuentra una arista que pertenezca a las mismas caras que e entonces e y esa arista forman un conjunto de corte de tamaño 2. La complejidad temporal del algoritmo para encontrar un conjunto de corte de cardinalidad m es $O(n^m)$.

V. Detección de conjuntos de corte de cardinalidad mínima

El algoritmo tomará como entrada una gráfica geométrica plana conexa y para cada una de las aristas de la gráfica revisará si son aristas de corte, si es así, las aristas de corte serán reportadas y el algoritmo terminará. De lo contrario, para cada una de las aristas de la gráfica se reportarán (si es que existen) los conjuntos de corte de cardinalidad 2 a los que pertenecen, si ningún conjunto de cardinalidad 2 es encontrado, entonces se realizará la búsqueda de conjuntos de corte de cardinalidad 3. Este procedimiento se seguirá hasta encontrar por lo menos un conjunto de corte, que para el peor de los casos, será un conjunto de corte de cardinalidad 5 (por tratarse de gráficas planas).

Algoritmo 2 Reportar los conjuntos de corte de cardinalidad m a los que pertenece una arista

Entrada: Gráfica geométrica plana conexa $G = (V, E)$, arista $e = \{u, v\}$ tal que $e \in E$.
Salida: Reporta todos los conjuntos de corte de cardinalidad m que forma e con las demás aristas de la gráfica.

1: Sean $Cara_e$ y $Cara_{inv(e)}$ las caras a las que pertenece e
2: Realizar el recorrido de una de las caras, en este caso el recorrido se hará sobre $Cara_e$
3: **para** cada arista de $Cara_e$ diferente de e **hacer**
4: La arista se nombrará $candidato1$
5: Sean $Cara_{c1}$ y $Cara_{inv(c1)}$ las caras a las que pertenece $candidato1$, con $Cara_{c1} = Cara_e$
6: **si** las caras $\{Cara_e, Cara_{inv(c1)}\}$ son diferentes y $Cara_{inv(c1)} = Cara_{inv(e)}$ **entonces**
7: **devolver** e y $candidato1$ como un conjunto de corte de cardinalidad m
8: **fin si**
9: **fin para**

Para verificar los conjuntos de corte de cardinalidad k en la gráfica, se debe realizar un recorrido completo de la misma. El recorrido de la gráfica tiene una complejidad de $O(n \log n)$. En el peor de los casos, se deberán buscar conjuntos de corte de cardinalidad 1, 2, 3, 4 y 5 para cada arista. Por lo tanto, la complejidad total del algoritmo queda de la siguiente forma

$$O(n \log n) + n(O(n)) + n(O(n^2)) + \ldots + n(O(n^5))$$

lo cual resulta en la complejidad de $O(n^6)$.

VI. Conclusiones

A lo largo de la presente investigación, se estudió un problema muy conocido en gráficas como es la identificación de conjuntos de corte, y se ha presentado una propuesta de solución al problema de encontrar el o los conjuntos de corte de cardinalidad mínima en una gráfica geométrica plana conexa, de manera local. Como una propuesta de trabajo a futuro, se tiene la implementación de estos algoritmos locales en redes físicas que necesiten de algoritmos de este tipo, como las redes de sensores o las redes *ad-hoc*. Otra propuesta es la mejora en la complejidad temporal de los algoritmos, o en su defecto, la demostración de que estas cotas son justas para este problema en particular.

Algoritmo 3 Algoritmo que reporta los conjuntos de corte de cardinalidad mínima

Entrada: Gráfica geométrica plana conexa $G = (V, E)$, u tal que $u \in V$.

Salida: Reporta el o los conjuntos de corte de cardinalidad mínima en G.

$m \leftarrow 0$
repetir
  Sea e_u la arista dominante del vértice u
  Sea $a = (u, v)$ una dirección sobre e_u
  e_u será la arista de inicio
  a será la dirección de inicio
  $m \leftarrow m + 1$
  repetir
    si e_u está siendo revisada por la izquierda **entonces**
      si e_u forma conjuntos de corte de cardinalidad m **entonces**
        Reportar los conjuntos de corte que forma e_u
        Terminar el algoritmo
      fin si
    fin si
    si e_u es una arista de entrada **entonces**
      Ahora e_u será la arista siguiente en el recorrido de la cara siguiendo la regla de la mano derecha y sin tomar en cuenta la arista $\{u, v\}$
      Sea $e_u = \{u, w\}$, ahora $a = (u, w)$
    si no
      Ahora e_u será la arista siguiente en el recorrido de la cara siguiendo la regla de la mano derecha
      Sea $e_u = \{v, w\}$, ahora $a = (v, w)$
    fin si
  hasta que e_u y a sean los valores de inicio
hasta que sea reportado algún conjunto de corte

REFERENCES

[1] M. Berg, M.V. Kreveld, R.V. Oostrum, and M. Overmars. Simple traversal of a subdivision without extra storage. *International Journal of Geographical Information Science*, pages 359–373, 1997.

[2] D.S. Hirschberg and J.B. Sinclair. Decentralized extrema-finding in circular configurations of processors. *Technical Note Operating Systems, Communications of the ACM*, 1980.

[3] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. *IN PROC. 11 TH Canadian Conference on Computational Geometry*, pages 51–54, 1999.

[4] F.P. Preparata and M.I. Shamos. *Computational Geometry, An introduction*. Springer-Verlag, 1985.

[5] J. Suomela. Survey of local algorithms. *ACM Computing Surveys (CSUR)*, 2013.

[6] J. Urrutia. Local solutions for global problems in wireless networks. *Instituto de Matemáticas, Universidad Nacional Autónoma de México*, 2006.

[7] D. Wagner and R. Wattenhofer. *Algorithms for Sensor and Ad HocNetworks*. Springer, 2007.

[8] D. Wagner and R. Wattenhofer. Local algorithms. *Springer*, 2007.