UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE ESTUDIOS SUPERIORES ACATLÁN

Un algoritmo para coordinar el movimiento de un enjambre de robots conservando la conectividad en su gráfica de visibilidad

Tesis

QUE PARA OBTENER EL TITULO DE

Lic. en Matemáticas Aplicadas y Computación

PRESENTA

Elizabeth Arias Ramírez

Asesor: José Sebastián Bejos Mendoza

Noviembre 2014 Santa Cruz Acatlán, Estado de México

Dedicado a
mis padres

Agradecimientos

A mis padres por apoyarme en todo momento y porque a ellos les debo todo lo que he logrado. A mi asesor de tesis Sebastián Bejos quiero agradecerle toda la paciencia que tuvo en todo este proceso.

A la FES Acatlán por fomentar y apoyar este tipo de investigaciones mediante el Laboratorio de Algoritmos para la Robótica.

¡Muchas Gracias!

Índice general

Índice de figuras

Introducción

Una red *Ad hoc* es un tipo de red inalámbrica descentralizada, es decir, es una red que no utiliza puntos de acceso o routers, pues cada uno de los nodos que la conforman tienen la habilidad de transmitir, recibir y enrutar información.

Cada nodo tiene la capacidad de establecer enlaces de comunicación a cualquier otro nodo que se encuentre dentro de su alcance, a estos nodos se les conoce como *vecinos* dado que existe una comunicación directa. Esta infraestructura es comúnmente representada mediante una gráfica y cualquier par de nodos en la red puede establecer comunicación a través de otros siempre y cuando exista un camino entre ellos.

Existen redes ad hoc en las que los nodos que la forman poseen cierta movilidad, dichas redes son conocidas como MANET (por sus siglas en ingles Mobile Ad Hoc Network). Cada nodo en una MANET posee la capacidad de desplazarse independientemente hacia cualquier dirección, lo que conlleva a un cambio constante en los enlaces de comunicación en la red.

Teniendo en cuenta que cada nodo en una MANET solo cuenta con la información de sus vecinos y que estos pueden cambiar constantemente, uno de los principales retos es garantizar la conectividad de todos los nodos en la red.

Alejandro Cornejo en [7] expone un servicio distribuido que coordina el movimiento de los nodos en una MANET garantizando la conectividad global de la red. En dicha investigación se plantea un escenario en donde el alcance de transmisión de los nodos está dado por un radio y estos se encuentran en un espacio abierto libre de obstáculos.

Para este trabajo se ha planteado un escenario acotado por una región poligonal, en la cual, se encuentra un enjambre de robots móviles que forman una MANET. Se ha definido el alcance de transmisión de manera que existe comunicación entre dos agentes si no existe obstáculo alguno entre ellos. Además cada nodo en la red tiene definido una posición objetivo dentro del polígono la cual desea alcanzar.

El objetivo de este trabajo de investigación es determinar si es posible coordinar los movimientos de los agentes llevándolos hasta sus objetivos y garantizando la conectividad global en todo momento, de ser así dar un algoritmo distribuido y en caso contrario dar una prueba de imposibilidad.

En el Capítulo 1 se da una breve introducción a conceptos básicos que son indispensables para un claro entendimiento del tema de investigación, se aborda los siguientes temas: notación asintótica, teoría de gráficas y polígonos. El primer tema es la teoría necesaria para entender cómo se mide la complejidad de un algoritmo, y los temas restantes tratan la teoría de la representación gráfica y computacional de problemas geométricos. En el Capítulo 2 se expone el tema "Gráficas geométricas de proximidad" que aborda los principios geométricos y algoritmos que guían el control de la topología de la redes ad hoc. El Capítulo 3 presenta brevemente el algoritmo distribuido creado por Cornejo en [7]. Finalmente en el Capítulo 4 se muestra lo realizado en esta investigación y los resultados obtenidos.

Capítulo 1

Conceptos básicos

1.1. Notación Asintótica

La medida en la que crece el tiempo de ejecución de un algoritmo da una simple caracterización de su eficiencia, y también permite comparar su rendimiento contra otros algoritmos alternos. En general el tiempo de ejecución de un algoritmo crece de acuerdo al tamaño de la entrada.

La notación asintótica es utilizada para describir el tiempo de ejecución de un algoritmo, dicha notación está definida en términos de funciones cuyo dominio es el conjunto de los números naturales. Esta notación puede describir convenientemente el peor de los casos en tiempo de ejecución de una función $T(n)$, en donde generalmente la entrada es entera.

1.1.1. Notación Θ

Para una función dada $g(n)$, se denotará por $\Theta(g(n))$ al conjunto de funciones que satisfagan

$$\begin{aligned}\Theta(g(n)) = \{f(n) : &\text{ Existen constantes positivas } c_1, c_2, \text{y } n_0 \text{ tal que} \\ &0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ para todo } n \geq n_0\}.\end{aligned}$$

Una función $f(n)$ pertenece al conjunto $\Theta(g(n))$ si existe una constante positiva c_1 y c_2 tal que $f(n)$ puede ser "encerrada" entre las funciones $c_1 g(n)$ y $c_2 g(n)$, para una n suficientemente grande. Puesto que $\Theta(g(n))$ es un conjunto, se puede escribir $f(n) \in \Theta(g(n))$ para indicar que $f(n)$ es un miembro de $\Theta(g(n))$, en lugar de escribir $f(n) = \Theta(g(n))$ para expresar la misma idea. La Figura 1.1 da una intuición acerca de las funciones $f(n)$ y $g(n)$.

La definición de $\Theta(g(n))$ requiere de que cada miembro $f(n) \in \Theta(g(n))$ sea asintóticamente no negativo, es decir, que $f(n)$ no sea negativa cuando n es lo suficientemente grande, en consecuencia $g(n)$ también es una función no negativa. Esta suposición se mantiene para las otras notaciones asintóticas definidas más adelante.

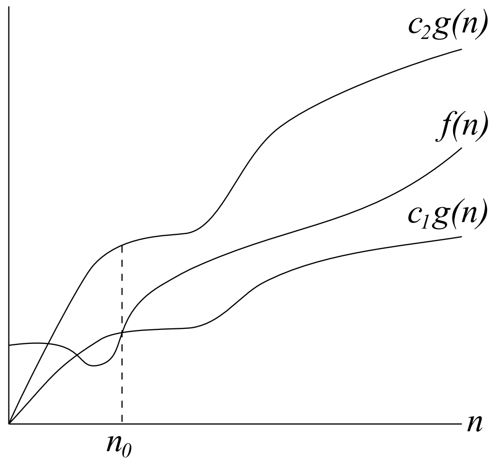


Figura 1.1: $f(n) \in \Theta(g(n))$

1.1.2. Notación O

La notación Θ acota la función $f(n)$ por arriba y por abajo. Cuando solo se tiene la *cota asintótica superior*, se utilizará la notación O. Para una función dada $g(n)$, denotaremos por $O(g(n))$ al conjunto de funciones tal que

$$O(g(n)) = \{f(n) : \text{Existen constantes positivas } c \text{ y } n_0 \text{ tal que } 0 \leq f(n) \leq cg(n) \text{ para todo } n \geq n_0\}.$$

Se utilizará la notación O para dar una cota superior de una función, dentro de un factor constante. La Figura 1.2 muestra la intuición de la notación O. Para todos los valores de n a la derecha de n_0, el valor de $f(n)$ esta por abajo de $cg(n)$.

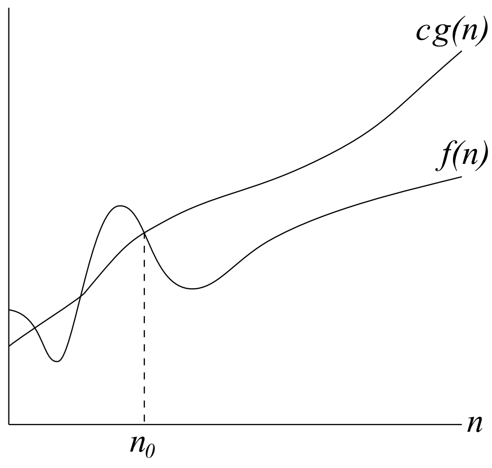


Figura 1.2: $f(n) \in O(g(n))$

Se debe notar que para $f(n) \in \Theta(g(n))$ implica que $f(n) \in O(g(n))$, puesto que la notación Θ es una noción mas fuerte que la notación O. Se puede escribir teóricamente que $\Theta(g(n)) \subseteq O(g(n))$.

1.1.3. Notación Ω

Tal como la notación O da una cota asintótica superior a una función, la notación Ω provee de una *cota asintótica inferior*. Dada una función $g(n)$, se denota por $\Omega(g(n))$ al conjunto de funciones

$$\Omega(g(n)) = \{f(n) : \text{Existen constantes positivas } c \text{ y } n_0 \text{ tal que } 0 \leq cg(n) \leq f(n) \text{ para todo } n \geq n_0\}.$$

La Figura 1.3 muestra la intuición de la notación Ω. Para todos los valores de n a la derecha de n_0, el valor de $f(n)$ está por arriba de $cg(n)$.

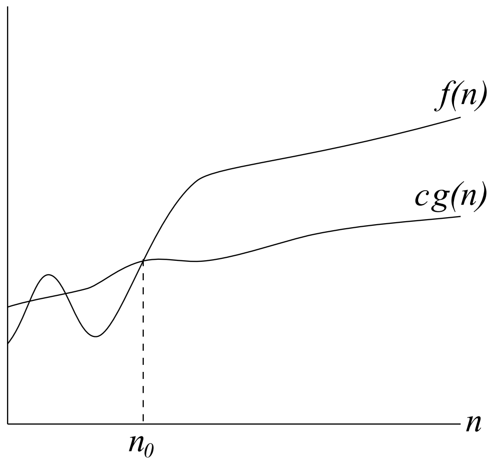


Figura 1.3: $f(n) \in \Omega(g(n))$

A partir de las definiciones de las notaciones asintóticas que se han visto hasta ahora, es fácil probar el siguiente teorema.

Teorema 1.1.1. *Para cualesquiera dos funciones $f(n)$ y $g(n)$, se tiene que $f(n) \in \Theta(g(n))$ si y solo si $f(n) \in O(g(n))$ y $f(n) \in \Omega(g(n))$.* □

1.2. Teoría de gráficas

1.2.1. Gráficas

Muchas situaciones de la vida real pueden ser convenientemente representadas por medio de un diagrama que consiste de un conjunto de puntos y un conjunto de líneas, las cuales unen a pares de puntos. Este diagrama es conocido como gráfica.

Definición 1.2.1 (Gráfica)**.** *Una gráfica es una tupla ordenada de conjuntos $G = (V, E)$, en donde V es el conjunto no vacío de los vértices (nodos), y E es el conjunto de pares no ordenados de vértices (no necesariamente distintos), los cuales representan una arista de la gráfica, es decir, una arista $e \in E$ está formada por dos vértices (u, v) tal que $u, v \in V$.*

Cada punto está representado por un vértice, y cada linea por una arista. Una arista e con vértices finales u y v también puede ser denotada como $e = uv$. En la Figura 1.4 se puede observar una gráfica.

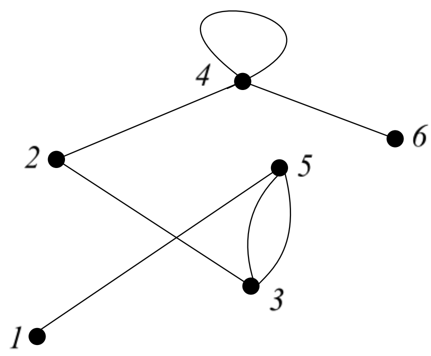


Figura 1.4: Ejemplo de una gráfica G con conjunto de vértices $V = \{1, 2, 3, 4, 5, 6\}$ y conjunto de aristas $E = \{(1, 5), (2, 3), (2, 4), (3, 5), (3, 5), (4, 4), (4, 6)\}$

Cuando los extremos de una arista son un mismo vértice, diremos que la arista es un *loop*. Cuando dos aristas tienen los mismos extremos las llamaremos *aristas paralelas*.

Definición 1.2.2 (Gráfica simple)**.** *Una gráfica simple es aquella que no contiene loops ni aristas paralelas.*

Se denotará al conjunto de vértices de una gráfica G como $V(G)$, y al conjunto de aristas como $E(G)$. Estas convenciones son independientes al nombre real de estos conjuntos, es decir, el conjunto de vértices de la gráfica $H = (W, F)$ es denotado como $V(H)$ y no como $W(H)$.

Definición 1.2.3 (Incidencia)**.** *Se dice que un vértice v es incidente con una arista e si $v \in e$.*

Definición 1.2.4 (Adyacencia)**.** *Dos vértices u, v de una gráfica son adyacentes o vecinos, si la arista (u, v) pertenece a la gráfica.*

Definición 1.2.5 (Gráfica completa)**.** *Si todos los pares de vértices en G son adyacentes, entonces se le conoce a G como una gráfica completa.*

Una gráfica completa con n vértices se representa como K_n. En la Figura 1.5 se muestran las gráficas completas K_3, K_4 y K_5.

Definición 1.2.6 (Gráfica plana)**.** *Se dice que G es un gráfica plana si existe una representación geométrica de manera que G puede ser dibujada en el plano tal que no existe un par de aristas que se intersecten.*

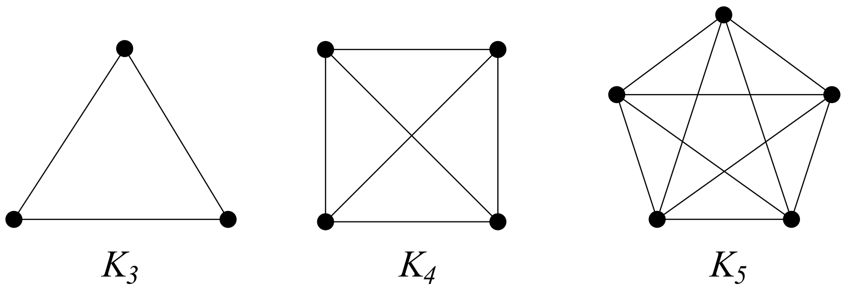


Figura 1.5: Gráficas completas K_3, K_4 y K_5

1.2.2. Subgráficas

Definición 1.2.7 (Subgráfica). *Sean las gráficas H y G, H es una subgráfica de G (representado como $H \subseteq G$) si $V(H) \subseteq V(G)$ y $E(H) \subseteq E(G)$.*

Cuando H es una subgráfica de G pero $H \neq G$ escribimos $H \subset G$ y decimos que H es una *gráfica propia* de G. Una *subgráfica de expansión* de G es una subgráfica H con $V(H) = V(G)$.

Sea la gráfica $G = (V, E)$ y sea V' un subconjunto no vacío de V. La subgráfica G' cuyo conjunto de vértices es V' y el conjunto de aristas está conformado por todas aquellas aristas de G que tienen como vértices finales a los contenidos en V', es llamada *subgráfica inducida* de G, y se denota $G[V']$

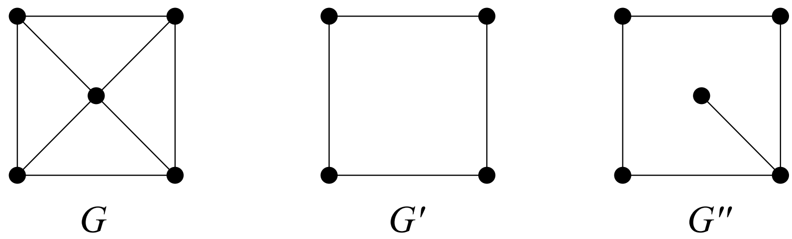


Figura 1.6: Una gráfica G con subgráficas G' y G''. G' es una subgráfica inducida de G y G'' es una subgráfica de expansión de G

1.2.3. Caminos y ciclos

Definición 1.2.8 (Camino). *Un camino de una gráfica G es una secuencia finita, no nula y alternada de vértices y aristas $C = v_0e_1v_1e_2v_2...e_kv_k$, de manera que para $1 \leq i \leq k$ los vértices finales de e_i son v_{i-1} y v_i.*

El entero k es la longitud del camino. En una gráfica simple un camino puede ser determinado solo por la secuencia sus de vértices $v_0v_1...v_k$. En la Figura 1.7 se muestra un ejemplo.

Definición 1.2.9 (Paseo). *Si no se repiten aristas en un camino, este es conocido como un paseo.*

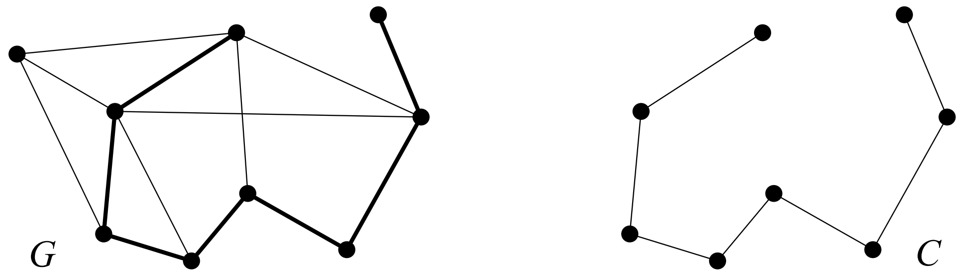


Figura 1.7: Una gráfica G sobre la cual se marca el camino C

Definición 1.2.10 (Trayectoria). *Es un camino en el cual no se repiten los vértices, y por lo tanto tampoco aristas.*

Definición 1.2.11 (Ciclo). *Un ciclo en una gráfica G es un camino cerrado que no repite vértices a excepción del primero y el último, v_0 y v_k respectivamente, en donde $v_0 = v_k$.*

1.2.4. Conectividad

Se dice que dos vértices u y v de una gráfica G son conectados si existe una trayectoria de u a v en la gráfica.

Definición 1.2.12 (Gráfica conectada). *Se dice que una gráfica G es conectada o conexa, si para cualquier par de vértices distintos $u, v \in V(G)$ existe una trayectoria entre u y v.*

Un conjunto S es *maximal* en relación a una determinada propiedad P si S satisface P y todo conjunto S' que contenga propiamente a S no satisface P. De manera análoga un conjunto *minimal*.

Definición 1.2.13 (Componente de una gráfica). *Sea la gráfica G, una componente de G es una subgráfica maximal conectada de G.*

Si una gráfica está conformada por más de una componente es una gráfica desconectada. En la Figura 1.8 podemos ver una gráfica con varias componentes.

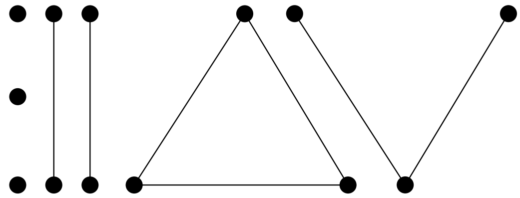


Figura 1.8: Una gráfica con siete componentes

Se denotará el número de componentes de una gráfica G como $\omega(G)$.

1.2.5. Árboles

Una gráfica acíclica es aquella que no contiene ningún ciclo.

Definición 1.2.14 (Árbol)**.** *Un árbol es una gráfica conectada acíclica.*

En la Figura 1.9 se muestra un árbol. Cualquier gráfica sin ciclos es un *bosque* de árboles.

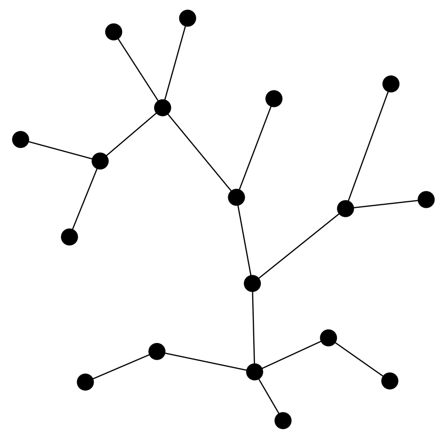


Figura 1.9: Un árbol

Teorema 1.2.1. *Sea la gráfica G, si G es un árbol, cualquier par de vértices distintos están conectados por una única trayectoria.*

Demostración. Por Contradicción. Sea G un árbol, se asume que para dos vértices distintos u y v existen dos trayectoria distintas que los conectan, P_1 y P_2 en G. Dado que $P_1 \neq P_2$, existe una arista xy en la trayectoria P_1 que no es arista de P_2. Claramente la gráfica $(P_1 \cup P_2) - xy$ permanece conectada, lo que significa que existe una trayectoria P de x a y, por lo tanto al agregar la arista xy a la trayectoria P (denotado como $P + xy$), existe un ciclo en G, lo cual una contradicción. □

Aristas de corte

Una *arista de corte* de una gráfica G es aquella tal que $\omega(G - e) > \omega(G)$. La Figura 1.10 muestra una gráfica que contiene tres aristas de corte.

Teorema 1.2.2. *Una arista e de G es una arista de corte si y solo si e no esta contenida en un ciclo de G.*

Demostración. Sea e una arista de corte de la gráfica G. Ya que $\omega(G-e) > \omega(G)$, entonces existen vértices u y v en G que están conectados en G pero no en $G - e$. Por lo tanto existe una trayectoria P de u a v en G, la cual necesariamente contiene a e. Supongamos que los vértices finales de e son x y y. Si e perteneciera a un ciclo C entonces x y y estarían conectadas en $G - e$ por la trayectoria $C - e$, por lo tanto u y v estarían conectados en $G - e$, lo cual es una contradicción. □

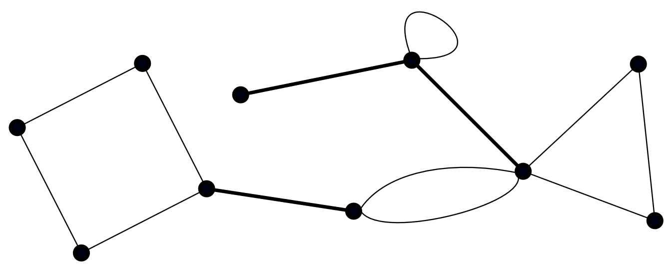


Figura 1.10: Aristas de corte en una gráfica

Teorema 1.2.3. *Una gráfica conectada es un árbol si y sólo si cada arista es una arista de corte.*

Demostración. Sea G un árbol y sea e una arista de G. Ya que G es acíclica entonces e no esta contenida en un ciclo y por el Teorema 1.2.2 e es una arista de corte. □

Definición 1.2.15. *Un árbol de expansión de una gráfica G es una subgráfica de G que es un árbol.*

Corolario 1.2.1. *Cada gráfica conectada contiene un árbol de expansión.*

Demostración. Sea G una gráfica conectada y sea T una subgráfica de expansión minimal conectada de G. Por definición $\omega(T) = 1$ y $\omega(T-e) > 1$ para cada arista e de T. Esto significa que cada arista de T es una arista de corte, y por lo tanto por el Teorema 1.2.3 T es conectada y es un árbol. □

En la Figura 1.11 se muestra una gráfica en donde se marca uno de sus árboles de expansión.

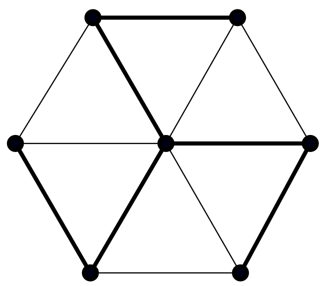


Figura 1.11: Gráfica en la cual se marca una árbol de expansión

Árbol de expansión mínima

Supóngase que está por construirse una red de ferrocarril que una a cierto número de ciudades. La construcción de la vías tiene cierto costo c_{ij} por conectar la ciudad i con la ciudad j. Se desea construir una red que minimice el costo total de la construcción de las vías.

Para este ejemplo, las ciudades son representadas por vértices en una gráfica ponderada con pesos en las aristas $w(v_iv_j) = c_{ij}$, el problema consiste en encontrar en la gráfica ponderada G, una subgráfica de expansión conectada con el mínimo peso, por lo que podemos asumir que dicha gráfica es un árbol.

Definición 1.2.16 (Árbol de expansión mínima)**.** *Un árbol de expansión mínima (MST por sus siglas en ingles) de una gráfica ponderada G, es un árbol de expansión de G cuya suma de los pesos de sus aristas es la mínima. El MST no es necesariamente único.*

El algoritmo de *PRIM* se basa en la construcción parcial de un MST, comienza con un vértice inicial, en cada paso del algoritmo se le agrega una arista a dicho árbol que conecta con un vértice vecino que aún no pertenece al árbol, el tamaño del árbol se va incrementando hasta que ya no quedan más vértices por agregar.

Algoritmo 1 Algoritmo de PRIM

Entrada: Una gráfica ponderada $G = (V, E)$
Salida: Un árbol de expansión mínima T

$T \leftarrow \emptyset$
Sea r un vértice seleccionado arbitrariamente de V
$U \leftarrow \{r\}$
While $|U| < |V|$ **Do**
    Encontrar $u \in U$ y $v \in (V - U)$ tal que la arista uv es la arista con el mínimo peso entre U y $V - U$
    $T \leftarrow T \cup \{uv\}$
    $U \leftarrow U \cup \{v\}$
End While

Los empates en el peso de las aristas son rotos arbitrariamente. En la Figura 1.12 podemos ver un ejemplo de la ejecución del Algoritmo 1.

Sea la gráfica G con n vértices y m aristas, un método sencillo para encontrar la arista de mínimo peso es buscar en la listas de adyacencia de los vértices en V, lo que tomaría un tiempo $O(m)$, y como esto se hace para cada uno de los vértices, el tiempo de ejecución del algoritmo es de $O(mn)$. Utilizando *Fibonacci heaps*[1] el algoritmo puede mejorar hasta $O(m + n \log n)$.

1.2.6. Digráficas

Definición 1.2.17 (Digráfica)**.** *Una digráfica (o gráfica dirigida) es una tupla ordenada que consta de dos conjuntos $D = (V, E)$, en donde $V(D)$ es el conjunto no vacío de los vértices, y $E(D)$ es el conjunto de pares ordenados de vértices (no necesariamente distintos), los cuales representan las aristas dirigidas de la gráfica.*

En la Figura 1.13 se muestra una gráfica y una digráfica. Cada concepto que es válido para gráficas automáticamente aplica para di-gráficas también. Sin

[1] Para más detalle acerca de este tema se puede consultar [5]

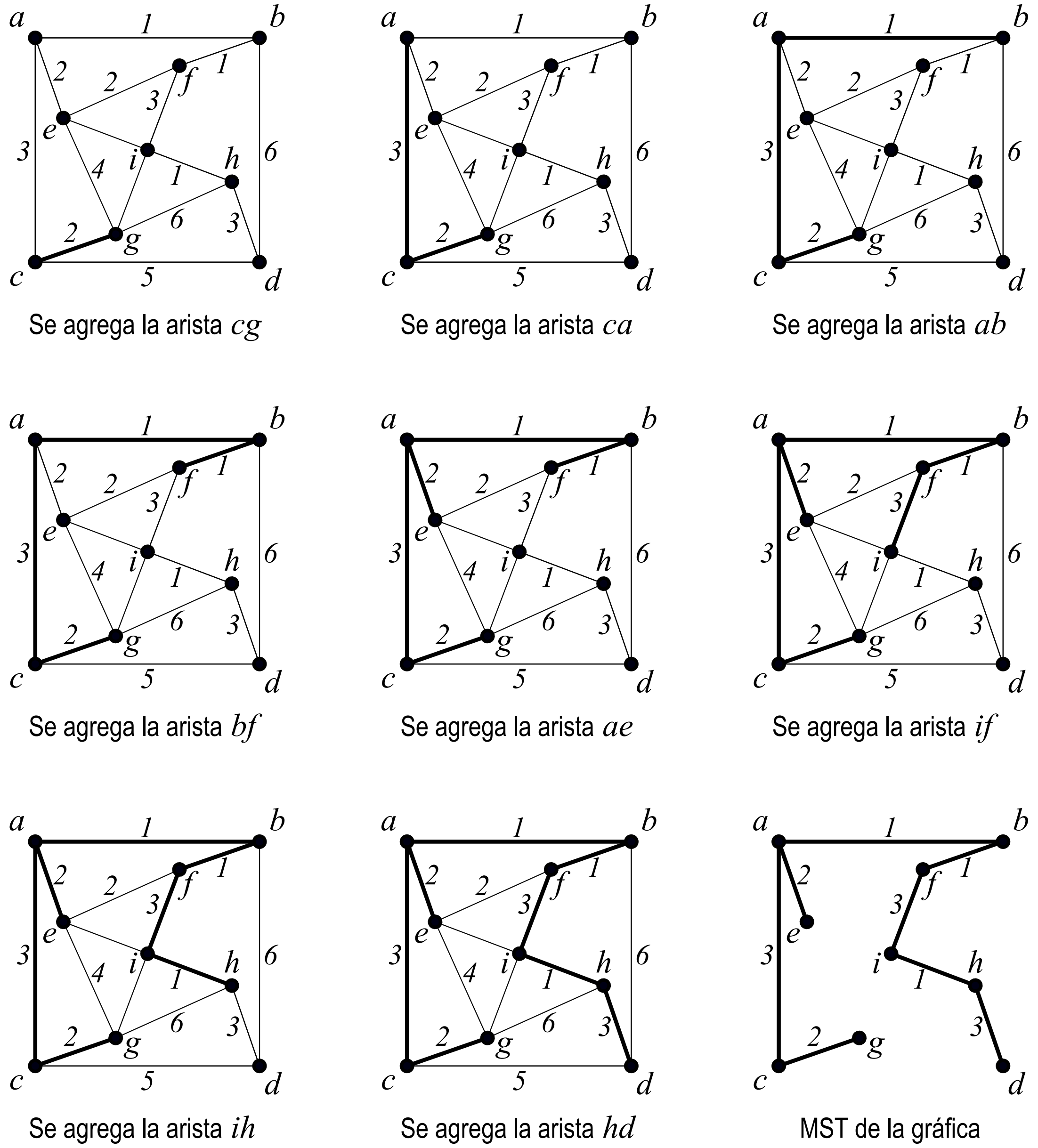


Figura 1.12: Ejemplo de la ejecución del algoritmo de PRIM en una gráfica

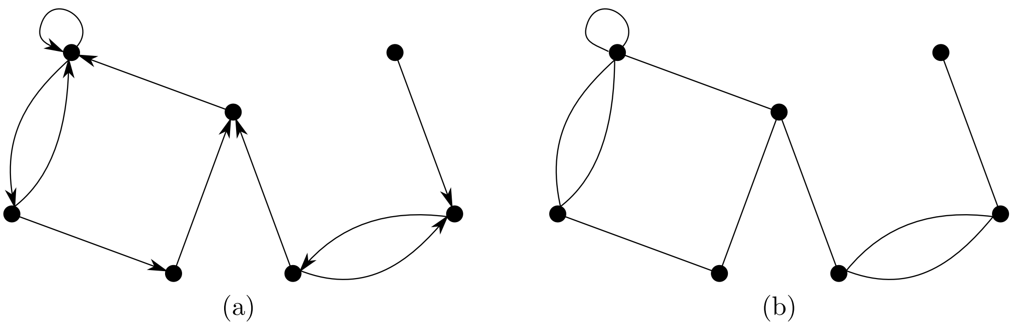


Figura 1.13: (a) Digráfica. (b) Gráfica

embargo existen conceptos que surgen a partir de la orientación de las aristas y por lo tanto aplican exclusivamente a las digráficas.

Definición 1.2.18 (Camino dirigido). *Un camino de una gráfica G es una secuencia finita, no nula y alternada de vértices y aristas $C = v_0 e_1 v_1 e_2 v_2 ... e_k v_k$, de manera que para $1 \leq i \leq k$ los vértices finales de e_i son v_{i-1} y v_i, en donde la dirección de e_i es de v_{i-1} a v_i.*

Definición 1.2.19 (Paseo dirigido). *Si no se repiten aristas en un camino dirigido, este es conocido como un paseo dirigido.*

Definición 1.2.20 (Trayectoria dirigida). *Es un camino dirigido en el cual no se repiten los vértices.*

Sea la gráfica D, si existe una trayectoria dirigida de un vértice u a un vértice v, se dice que el vértice v es *alcanzable* desde u. Dos vértices de G, u y v están conectados si existe una trayectoria dirigida de u a v y viceversa. D es conectada si tiene un solo componente.

1.3. Polígonos

Los polígonos son usualmente una precisa representación de muchos objetos de la vida real. Ejemplos de sus usos incluyen la representación de obstáculos a evitar en el entorno de un robot, etc.

Definición 1.3.1 (Polígono). *Un polígono P esta definido como una región cerrada R en el plano, limitada por un conjunto finito de segmentos (llamadas aristas de P). Los puntos extremos de una arista de P son llamados vértices de P.*[2]

Definición 1.3.2 (Vértice convexo). *Se dice que un vértice v de un polígono P es convexo si el angulo que se encuentra en la región interior de P formado por el par de aristas incidentes en v es de a lo más π.*

[2] A partir de este tema se hará referencia a los vértices de una gráfica como “nodos” para evitar confusión con los vértices de un polígono.

Definición 1.3.3 (Vértice cóncavo). *Se dice que un vértice v de un polígono P es cóncavo si el angulo que se encuentra en la región interior de P formado por el par de aristas incidentes en v es mayor a π.*

Definición 1.3.4 (Polígono simple). *Se dice que un polígono P es simple si existe una trayectoria entre cualesquiera dos puntos de R la cual no intersecte con alguna arista de P.*

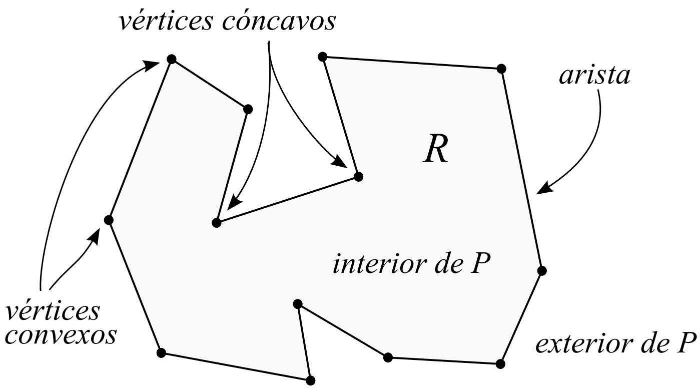


Figura 1.14: Polígono simple P

Un polígono simple P puede dividir al plano que lo contiene en dos regiones, la región interior al polígono R y la región exterior a él, como se muestra en la Figura 1.14.

1.3.1. Visibilidad

Definición 1.3.5 (Visibilidad). *Sean dos puntos p y q en el interior de un polígono simple P. Se dice que p y q son visibles si el segmento de línea pq no contiene algún punto del exterior de P, es decir, el segmento pq se encuentra totalmente contenido al interior de P.*

En la Figura 1.15 se muestra un ejemplo de visibilidad entre el punto p y q, y donde r no es visible a p ni a q.

Definición 1.3.6 (Polígono convexo). *Sea P un polígono simple. Se dice que además es convexo si para cualquier par de puntos p y q al interior de P, p y q son visibles.*

Definición 1.3.7 (Polígono estrella). *Se dice que un polígono simple P es un polígono estrella si existe un punto z al interior de P tal que todos los puntos de P son visibles desde z.*

Definición 1.3.8 (Núcleo de un polígono). *Sea P un polígono estrella, se define como su núcleo, denotado por $Kern(P)$, al conjunto de puntos en P desde donde son visibles todos los puntos de P, es decir, $Kern(P) = \{z \in P : \forall q \in P, q \text{ es visible con } z\}$*

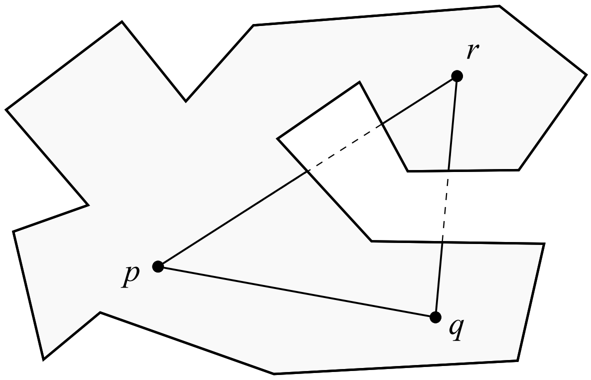


Figura 1.15: Visibilidad entre los puntos p y q. El punto r no es visible a p ni a q

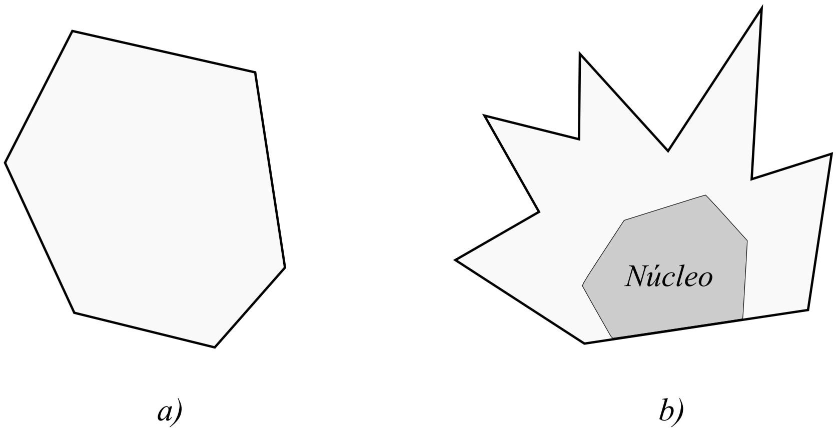


Figura 1.16: (a) Polígono convexo. (b) Polígono estrella

1.3.2. Triangulación

Definición 1.3.9 (Diagonal de P)**.** *Una diagonal en un polígono simple P es un segmento de línea que une a cualquiera dos vértices visibles de P y se encuentra contenida en el interior de P.*

Nótese que si un segmento une a dos vértices u y v de P y pasa a través de otro vértice w de P y el segmento se encuentra totalmente contenido en P, entonces los segmentos uw y vw con diagonales de P mientras que uv no lo es.

Definición 1.3.10 (Triangulación)**.** *Una triangulación de un polígono P es una partición de P en triángulos definidos por diagonales.*

La triangulación de un polígono no es única, ya que existen varios subconjuntos de diagonales que dan varias triangulaciones para el mismo polígono. La gráfica dual de una triangulación de P es una gráfica donde cada triángulo es representado como un nodo de la gráfica y dos nodos están conectados con una arista si y solo si los correspondientes triángulos comparten una diagonal. Ya que cada triángulo tiene tres lados el grado de cada nodo en la gráfica dual es de a lo más tres.

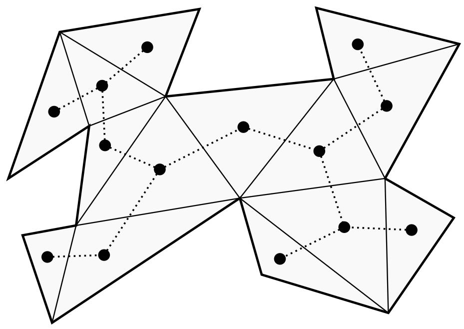


Figura 1.17: Triangulación de un polígono con su gráfica dual

Lema 1.3.1. *Todo polígono con más de tres vértices admite una diagonal.*

Demostración. Sea v el vértice más a la izquierda de P (en caso de empates, tomamos el vértice más inferior a la izquierda). Sean u y w vértices vecinos de v en el borde de P. Si el segmento uw se encuentra en el interior de P, se ha encontrado una diagonal. De otra manera, existen uno o mas vértices dentro del triángulo definido por v, u y w. De esos vértices, sea v' el vértice más alejado del segmento uw. El segmento que conecta a v' con v no puede intersectar una arista de P, por que tal arista tendría un vértice final más alejado del segmento uw contradiciendo la definición de v'. Por lo tanto vv' es una diagonal. □

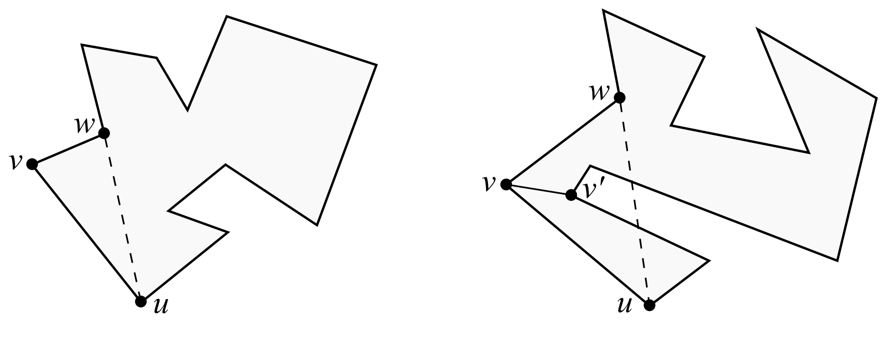


Figura 1.18: Prueba gráfica del Lema 1.3.1

Lema 1.3.2. *Todo polígono admite una triangulación.*

Demostración. Por inducción. Sea un polígono P con n vértices si $n = 3$ entonces el polígono es en sí mismo un triángulo y la prueba ha terminado. Para $n > 3$ por el Lema 1.3.1 el polígono P admite una diagonal que lo divide en dos subpolígonos P_1 y P_2, donde m_1 y m_2 es el número de vértices de cada polígono respectivamente. Ambos m_1 y m_2 son menores a n, aplicando la hipótesis de inducción a P_1 y P_2 estos admiten una diagonal y pueden ser triangulados. Por lo tanto P puede ser triangulado. □

Lema 1.3.3. *Cualquier triangulación de un polígono simple P con n vértices consiste de exactamente $n - 2$ triángulos.*

Demostración. Se considera una diagonal arbitraria de alguna triangulación T_P. Esta diagonal corta a P en dos subpolígonos con m_1 y m_2 vértices respectivamente. Cada vértice de P se encuentra en exactamente unos de los dos subpolígonos a excepción de los vértices que definen la diagonal, los cuales se encuentran en ambos subpolígonos. Por lo tanto $m_1 + m_2 = n + 2$. Por inducción, cualquier triangulación P_i consiste de $m_i - 2$ triángulos, lo que implica que T_P consiste de $(m_1 - 2) + (m_2 - 2) = n - 2$ triángulos. □

Corolario 1.3.1. *La suma de los ángulos internos de un polígono simple de n vértices es $(n-2)\pi$.*

El primer algoritmo de complejidad de tiempo $O(n \log n)$ para triangular un polígono P fue dado por Garey *et al.* [16]. El primer paso del algoritmo consiste en particionar el polígono P en subpolígonos *y-monótonos*. Se dice que un polígono es *y-monótono* si su límite puede ser dividido en dos cadenas de vértices tal que cada cadena de vértices tiene incremento en la coordenada y. Dicha partición puede calcularse en tiempo $O(n \log n)$ por el algoritmo de Lee y Preparata [13]. La complejidad en tiempo de calcular la triangulación de un polígono y-monótono es proporcional al número de vértices del polígono. Por lo tanto la complejidad de calcular la triangulación de un polígono es de $O(n \log n)$.

1.3.3. Polígonos de visibilidad

Definición 1.3.11 (Polígono de visibilidad de un punto)**.** *El polígono de visibilidad $Vis(q)$ de un punto q en un polígono simple P está dado por el conjunto de todos los puntos de P que son visibles desde q. En otras palabras, $Vis(q) = \{p \in P : q \text{ es visible con } p\}$*

Se puede calcular el polígono de visibilidad de un punto en tiempo lineal $O(n)$ con el algoritmo desarrollado por Simpson [12], dicho algoritmo se describe en la Sección 1.3.3. La Figura 1.19 muestra el polígono de visibilidad $Vis(q)$ de un polígono simple P.

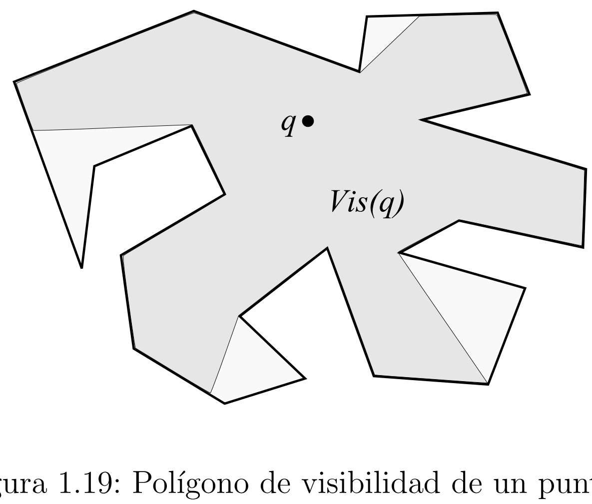


Figura 1.19: Polígono de visibilidad de un punto q

Por definición, cualquier polígono $Vis(q)$ es un *polígono estrella* y q se encuentra en el *núcleo* de P.

Ahora consideraremos una variación y se definirá la visibilidad desde una arista v_iv_{i+1} de polígono simple P en tres diferentes maneras. Fig. 1.20:

I) Se dice que el polígono P es *completamente visible* desde la arista v_iv_{i+1} si cada punto $z \in P$ es visible a todo punto $w \in v_iv_{i+1}$.

II) Se dice que el polígono P es *fuertemente visible* desde la arista v_iv_{i+1} si existe un punto $w \in v_iv_{i+1}$ tal que para cada punto $z \in P$, w y z son visibles.

III) Se dice que el polígono P es *débilmente visible* desde la arista v_iv_{i+1} si para cada punto $z \in P$ existe un punto $w \in v_iv_{i+1}$ tal que w y z son visibles.

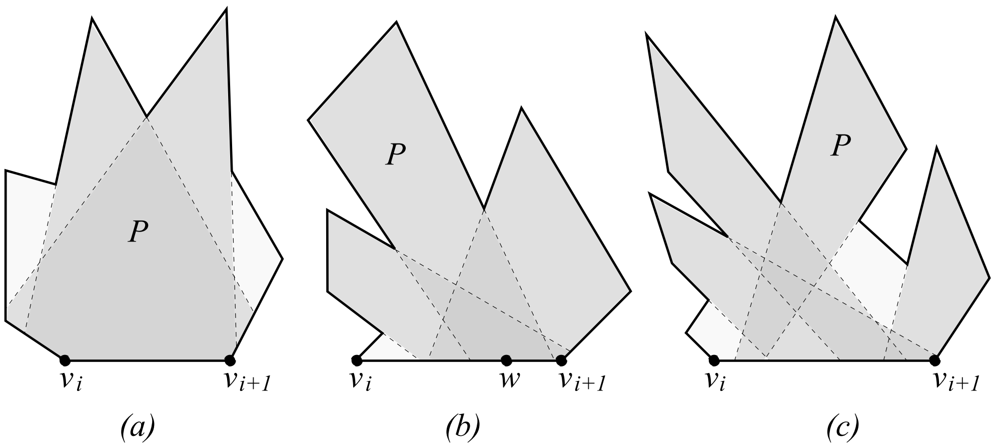


Figura 1.20: P es un polígono (a) completamente, (b) fuertemente y (c) débilmente visible desde la arista v_iv_{i+1}

Se dice que un punto $z \in P$ es *completamente visible* desde la arista v_iv_{i+1} si este es visible a cada punto de v_iv_{i+1}. De manera similar, se dice que un punto $z \in P$ es *débilmente visible* desde la arista v_iv_{i+1} si este es visible a algún punto de v_iv_{i+1}

Definición 1.3.12 (Polígono de visibilidad completa desde una arista)**.** *Sea P un polígono simple. El polígono $VisC(v_iv_{i+1})$ formado por el conjunto de todos los puntos $z \in P$ que son completamente visibles desde la arista v_iv_{i+1} se le llama polígono de visibilidad completa de la arista v_iv_{i+1}.*

Definición 1.3.13 (Polígono de visibilidad débil desde una arista)**.** *Sea P un polígono simple. El polígono $VisD(v_iv_{i+1})$ formado por el conjunto de todos los puntos $z \in P$ que son débilmente visibles desde la arista v_iv_{i+1} se le llama polígono de visibilidad débil de la arista v_iv_{i+1}.*

Una noción mas general de los polígonos de visibilidad débil y completa permite tomar en cuenta un segmento interno pq y no necesariamente una arista v_iv_{i+1} como se puede ver en la Figura 1.22.

Se considera ahora el problema de calcular el polígono de visibilidad completa desde un segmento de línea pq en un polígono P. El siguiente lema sugiere una manera de calcular el polígono de visibilidad completa desde un segmento pq.

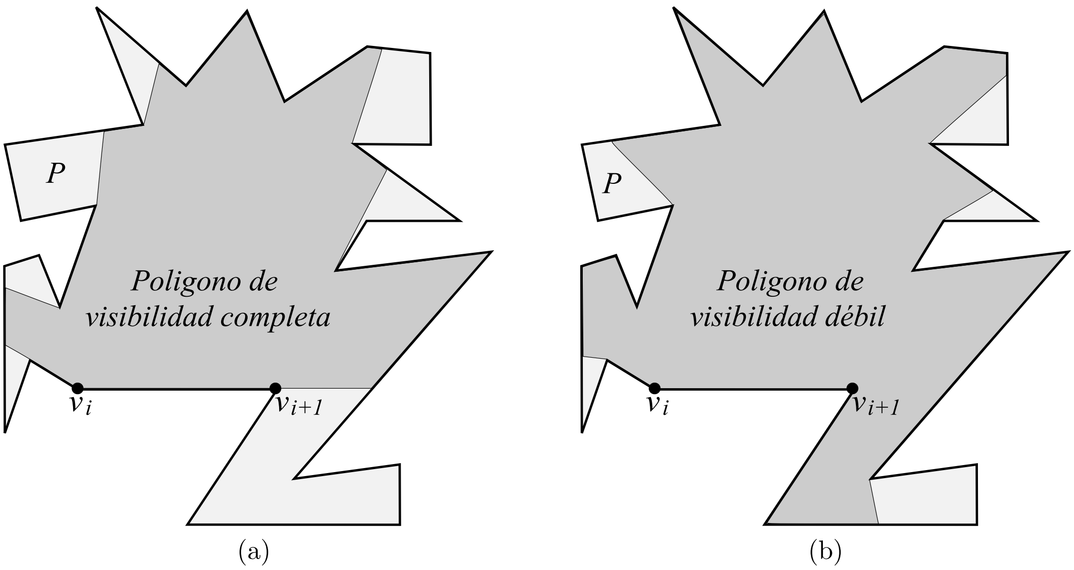


Figura 1.21: El (a) polígono de visibilidad completa y (b) visibilidad débil de P desde la arista v_iv_{i+1}

Lema 1.3.4. *El polígono de visibilidad completa $VisC(pq)$ de un segmento pq en un polígono P, es el polígono visibilidad completa de q calculado dentro del polígono de visibilidad completa de p, calculado dentro de P.*

Demostración. Sea $Vis(p)$ el polígono de visibilidad de p en P, y $Vis(q)$ el polígono de visibilidad de q en P. Sea $z \in P$ un punto en el plano visible desde p. Dado que z es visible desde p, entonces $z \in Vis(p)$. Por definición, para que z sea completamente visible desde la línea pq, z también tiene que ser visible desde q, lo que significa que $z \in Vis(q)$. Por lo tanto el polígono de visibilidad completa de pq en P está dado por el conjunto de puntos $z \in (Vis(p) \cap Vis(q))$, lo que es equivalente a obtener el polígono $Vis(q)'$ dentro del polígono $Vis(p)$. □

El Lema 1.3.4 sugiere calcular primero $Vis(p)$ dentro del polígono P, lo que se puede hacer en tiempo $O(n)$. Posteriormente calcular $Vis(q)'$ dentro del polígono $Vis(p)$ lo que también se obtiene en tiempo $O(n)$.

Algoritmo para calcular el polígono de visibilidad de un punto

Para la ejecución del algoritmo se dispone de las siguientes precondiciones.

- Los vértices del polígono $v_0, v_1, ..., v_n = v_0$ deben encontrarse ordenados en sentido contrario a las manecillas del reloj, es decir, deben poder recorrerse hacia la izquierda.
- x debe encontrarse a la izquierda del primer vértice del polígono.

Para cada vértice v_i de P, se define su ángulo $\alpha(v_i)$ alrededor de x como sigue:

I) $\alpha(v_0) = 0$

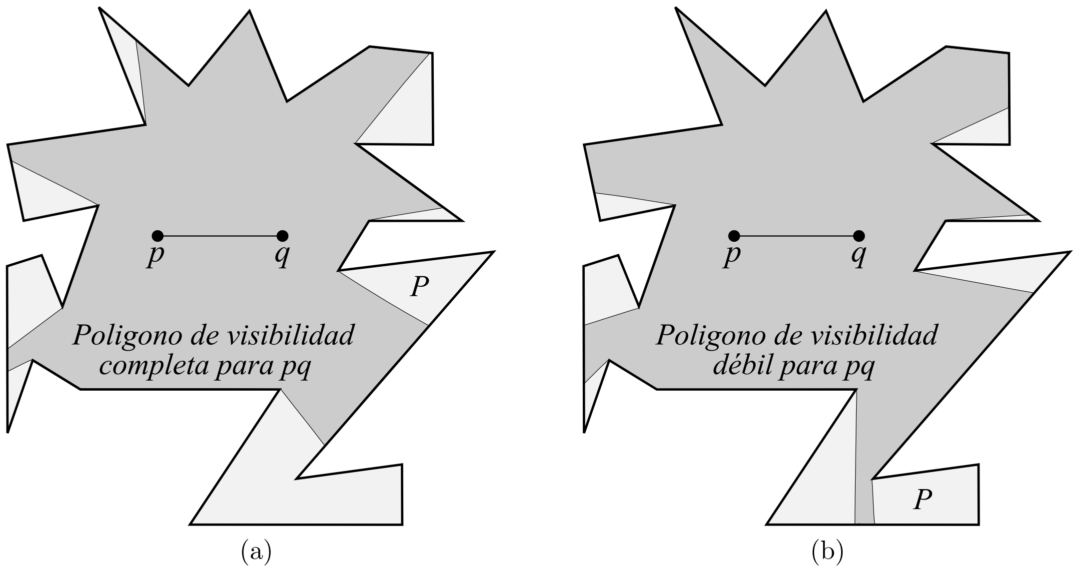


Figura 1.22: El (a) polígono de visibilidad completa y (b) visibilidad débil de P desde un segmento pq

II) $\alpha(v_i) = \alpha(v_{i-1}) + \sigma \cdot angle(v_{i-1}xv_i)$

En donde $\sigma = 1$ si $xv_{i-1}v_i$ es un giro a la izquierda, $\sigma = -1$ si $xv_{i-1}v_i$ es un giro a la derecha, y $\sigma = 0$ si $xv_{i-1}v_i$ no realiza giro. Por lo tanto si $\alpha(v_i) > 2\pi$ se ha "dado una vuelta alrededor" de x, de v_0 a v_i. Está claro que solo los vértices v con $0 \leq \alpha(v) \leq 2\pi$ son candidatos para el polígono de visibilidad $Vis(x)$.

El algoritmo consiste de tres procedimientos: *Push*, *Pop* y *Wait*. *Push* agrega un nuevo vértice visible arriba de la pila. *Pop* elimina uno o más vértices de la pila hasta que la interferencia es eliminada. Y *Wait* atraviesa una porción del polígono que no se encuentra visible a x, esperando al siguiente vértice visible.

Ahora se describirá con más detalle cada procedimiento. Sea la arista actual v_iv_{i+1} en el proceso, y sean los vértices en la pila $s_0, ..., s_t$.

Push

Este procedimiento es ejecutado cuando se cumple que $\alpha(v_{i+1}) \geq \alpha(s_t)$ y $\alpha(v_{i+1}) \geq \alpha(v_i)$. Se distinguen dos casos.

Caso a ($\alpha(v_{i+1}) \leq 2\pi$). Este es el caso "normal". Se agrega v_{i+1} a la pila y se incrementa i. La siguiente acción es determinada por la nueva arista v_iv_{i+1} como sigue (nota que ahora $s_t = v_i$). Si $i = n$ el algoritmo termina. Si $\alpha(v_{i+1}) \geq \alpha(v_i)$, entonces *Push* es ejecutado otra vez. Si $\alpha(v_i) > \alpha(v_{i+1})$, entonces verificamos si hay un giro a la izquierda en v_i, *Pop* es ejecutado (Figura 1.23a); y si hay un giro a la derecha, entonces ejecutamos el procedimiento *Wait* con $W = s_t\infty$ (Figura 1.23b).

Caso b ($\alpha(v_{i+1}) > 2\pi$). Entonces la intersección del rayo xv_0 (cuyo ángulo es $0 = 2\pi$) con v_iv_{i+1} es agregada a la pila, y *Wait* es ejecutado con $W = v_0s_t$

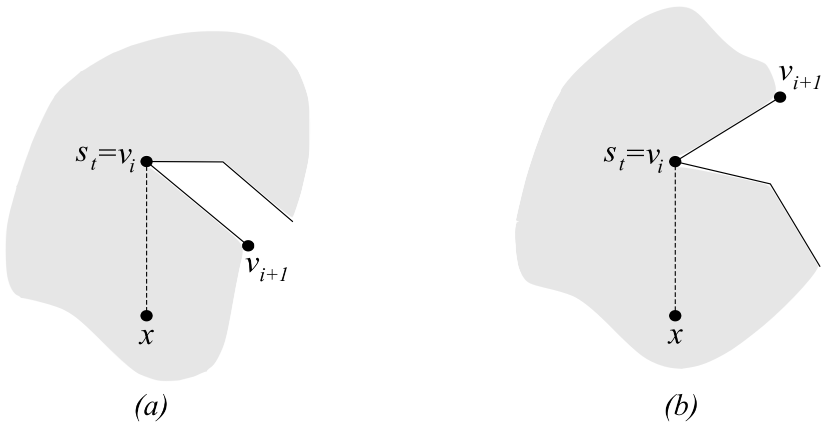


Figura 1.23: (a) *Pop* es llamado. (b) *Wait* es llamado.

Caso b ($\alpha(v_{i+1}) > 2\pi$). Entonces la intersección de el rayo xv_0 (cuyo ángulo es $0 = 2\pi$) con v_iv_{i+1} es agregada a la pila, y *Wait* es ejecutado con $W = v_0s_t$

Pop

Los vértices de la pila son eliminados hasta que s_j cumple:

(a) $\alpha(s_{j+1}) \geq \alpha(v_{i+1}) > \alpha(s_j)$ (Figura 1.24a), o

(b) $\alpha(s_{j+1}) = \alpha(s_j) \geq \alpha(v_{i+1})$ y el punto de intersección y se encuentra entre s_j y s_{j+1} (Figura 1.24b).

Caso a. Se agrega y al tope de la pila e i es incrementada. La siguiente acción es determinada por la nueva arista, similar al *Caso a* de *Push*. Si $i = n$ se terminar la ejecución. Si $\alpha(v_i) > \alpha(v_{i+1})$, entonces *Pop* es ejecutado nuevamente. Si $\alpha(v_{i+1}) \geq \alpha(v_i)$, entonces si v_i gira a la derecha se llama el procedimiento *Push*, y si se gira a la izquierda se llama el procedimiento *Wait* con $W = v_is_t$.

Caso b. Ignorando el caso degenerado cuando s_j, v_{i+1} y s_{j+1} con colineales, el procedimiento *Wait* es llamado con $W = s_jy$.

Wait

i es incrementada hasta que v_iv_{i+1} intersecta a W en el punto y desde la correcta dirección. Cuando esto ocurre, y es agregado a la pila, y *Push* o *Pop* son llamados dependiendo si $\alpha(v_{i+1}) \geq \alpha(v_i)$ o viceversa, respectivamente.

Se muestra un ejemplo en la Figura 1.25. *Push* avanza al vértice 3, cuando $S = 0\ 1\ 2\ 3$. Ya que $\alpha(3) > \alpha(4)$ y 3 gira a la derecha, *Wait* es llamado con W como se ilustra. *Wait* detecta que 8 emerge a través de W, se agrega $7'$ a la pila y se llama el procedimiento *Push*. $S = 0\ 1\ 2\ 3\ 7'\ 8$. Ya que $\alpha(8) > \alpha(9)$ y 8 gira

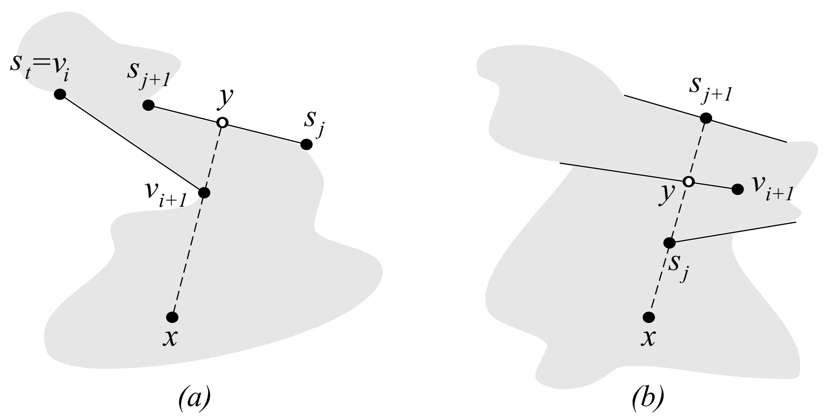


Figura 1.24: Dos casos del procedimiento *Pop*.

a la izquierda el procedimiento *Pop* es llamado. Todos los vértices son eliminados de la pila hasta el vértice 1, $1'$ y 9 son agregados a la pila quedando $S = 0\ 1\ 1'\ 9$. Finalmente *Push* avanza hasta que encuentra a 0 otra vez y $S = 0\ 1\ 1'\ 9\ 10\ 11$ que es de hecho $Vis(x)$.

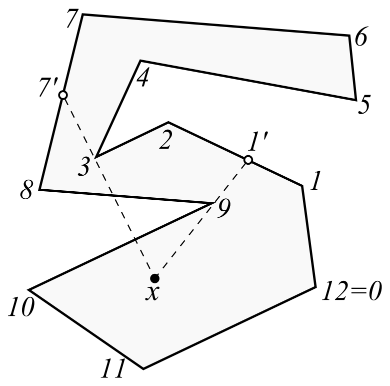


Figura 1.25: Ejemplo de un polígono de visibilidad: $Vis(x) = 0\ 1\ 1'\ 9\ 10\ 11$.

1.3.4. El camino más corto euclidiano

Definición 1.3.14 (El camino más corto euclidiano)**.** *El camino más corto euclidiano entre dos puntos s y t en P (denotado como $SP(s,t)$) es el camino que conecta s a t tal que (i) el camino completo se encuentra totalmente contenido en P y (ii) la longitud del camino es la más corta en distancia euclidiana que cualquier otro camino que conecte s con t, ver Figura 1.26.*

Lema 1.3.5. *$SP(s,t)$ es un camino simple en P, es decir, dicho camino no contiene ciclos.*

Demostración. Si $SP(s,t)$ se intersecta a si mismo en algún punto u, eliminando el sub-camino de u a si mismo (es decir, el ciclo de u) un $SP(s,t)$ puede ser obtenido, lo cual es una contradicción. □

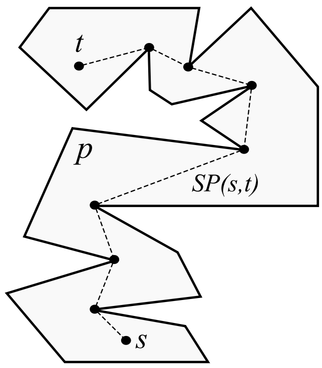


Figura 1.26: El camino más corto euclidiano entre los puntos s y t

Lema 1.3.6. *Sea $SP(s,t) = (s, ..., u, ..., v, ..., t)$. Entonces, $SP(u,t)$ y $SP(s,v)$ pasan a través de u y v respectivamente.*

Demostración. Si $SP(u,t)$ no pasa a través de v, entonces existe otro $SP(s,t)'$ que consiste de $SP(s,u)$ y $SP(u,t)$, lo que contradice la suposición de que $SP(s,t)$ es el camino más corto euclidiano. De manera análoga se argumenta para mostrar que $SP(s,v)$ pasa a través de u. □

Lema 1.3.7. *$SP(s,t)$ gira solo en los vértices de P.*

Demostración. Se asumirá que $SP(s,t)$ hace un giro en algún punto u, donde u no es un vértice de P. Se consideran dos puntos v y w en $SP(s,t)$ tal que (i) se encuentran arbitrariamente cerca de u, (ii) v está en $SP(s,u)$ y (iii) w está en $SP(u,t)$. Si el segmento vw no se encuentra contenido en P para cualquier selección de v y u, entonces u es un vértice de P lo que contradice la suposición de que u no es vértice de P. Entonces asumimos que el segmento vw se encuentra contenido en P. Por desigualdad del triángulo la longitud de vw es más corta que la suma de la longitud de $SP(v,u)$ y $SP(u,w)$. Entonces existe un camino más corto entre s y t que consiste de $SP(s,v)$, vw y $SP(w,t)$. Por lo tanto, si $SP(s,t)$ gira en un punto u,este debe se un vértice de P. □

Corolario 1.3.2. *El ángulo que se encuentra mirando hacia el exterior de un polígono P en un $SP(s,t)$ es convexo.*

Lema 1.3.8. *El $SP(s,t)$ es único en un polígono P.*

Demostración. Se asume que existen dos caminos entre s y t dentro de P que tienen distancia mínima. Ya que ambos son los caminos más cortos euclidianos entre s y t, satisfacen el Lema 1.3.5 y el Lema 1.3.7. Sin pérdida de generalidad, se asume que ambos caminos se encuentran solo en s y t. Ya que los dos caminos se encuentran dentro de P, los dos caminos forman un polígono simple P'. Sea m el número de vértices de P'. Por el Corolario 1.3.2, el ángulo que mira al interior del polígono P' de cada uno de sus vértices es cóncavo, a excepción de s y t. Por lo tanto, la suma de dichos ángulos internos de P' es más de $(m-2)\pi$, lo que contradice el hecho de que la suma interna de los ángulos de un polígono simple de m vértices es $(m-2)\pi$. □

Calculando el camino más corto euclidiano

Consideremos el problema de calcular *el camino más corto* entre dos puntos s y t dentro de un polígono simple P. Sean T_s y T_t los triángulos en $T(P)$ que contienen a los puntos s y t, respectivamente. Sea p_{st} el camino de T_s a T_t en la gráfica dual de $T(P)$. Se construye el subpolígono P' de P que consiste solo de estos triángulos de $T(P)$ que están en p_{st}. Ya que $SP(s,t)$ se encuentra dentro de P', P' puede ser considerado para calcular $SP(s,t)$ en lugar de P. Ya que en la gráfica dual de $T(P')$ es una trayectoria, el cálculo del $SP(s,t)$ puede obtenerse en tiempo $O(n)$, como se describe en *Lee y Preparata*[14].

Capítulo 2

Gráficas geométricas de proximidad

Las redes *Ad hoc* son sistemas de redes que facilitan la comunicación entre dispositivos que no cuentan con alguna infraestructura preestablecida (tales como routers, puntos de acceso, etc.)

Los protocolos y algoritmos para redes ad hoc deben permitir la comunicación global entre los dispositivos únicamente haciendo uso de recursos locales, es decir, cada dispositivo independientemente explora y establece conexiones con otros dispositivos que se encuentren dentro de su radio de transmisión. Esta importante propiedad se denomina como *localidad*.

Se le conoce como topología de la red a la estructura de enlaces que conectan pares de nodos en la red. La tarea principal del control de la topología es seleccionar un subconjunto de enlaces tal que todos los nodos de la red se encuentran conectados. El principal objetivo al reducir el número de enlaces es hacer del envío de datos una tarea más rápida y fácil.

En este capítulo se abordarán los principios geométricos que guían la comunicación dentro de las redes ad hoc y los algoritmos locales para el control de la topología.

2.1. Proximidad geométrica

Un requerimiento esencial para la propagación de la información en las redes ad hoc es que cada dispositivo debe poder identificarse por medio de un id, debe estar equipado con un dispositivo GPS para tener conocimiento de su posición y tiene un rango de transmisión de distancia r.

Una red ad hoc puede ser claramente descrita mediante una gráfica, como se define a continuación.

Definición 2.1.1 (Gráfica de disco unitario (UDG)). *Sea la gráfica $G = (V, E)$, donde V consiste de un conjunto de puntos en el plano los cuales representan dispositivos en la red y existe una arista $(v, u) \in E$ si y solo si la distancia euclidiana entre los nodos $v, u \in V$ es menor a r.*

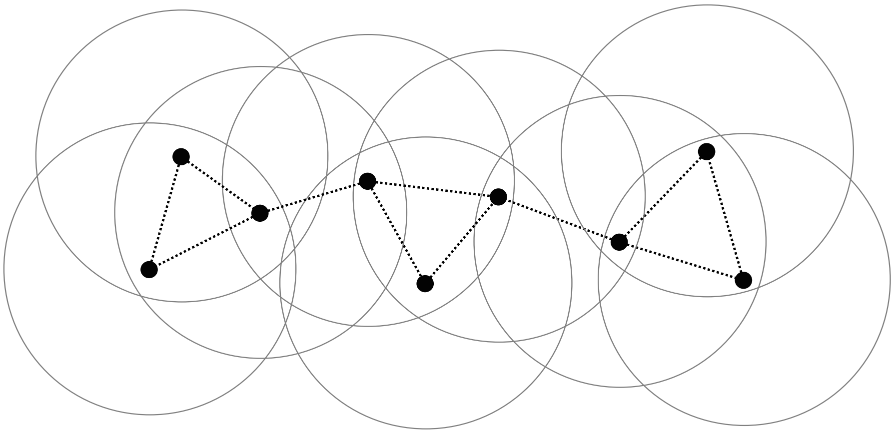


Figura 2.1: Gráfica de disco unitario

2.1.1. Gráfica del vecino más cercano (NNG)

Las aristas de una NNG son determinadas por la mínima distancia. Más precisamente, para dos nodos p y q existe una arista dirigida $(p, q) \in E \Leftrightarrow q$ es el vecino más cercano a p. NNG es dirigida y puede ser desconectada. Una generalización de NNG es la k-NNG, para alguna $k > 1$. En este caso existe una arista dirigida $(p, q) \in E \Leftrightarrow q$ es el k-ésimo vecino más cercano a p.

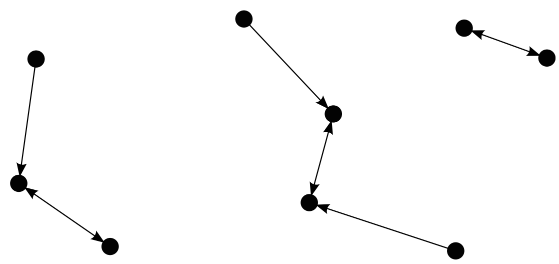


Figura 2.2: Gráfica del vecino más cercano

2.1.2. Gráfica de vecindad relativa (RNG)

La vecindad asociada a la RNG es determinada por una luna. Formalmente, la *luna* $L_{p,q}$ de dos nodos p y q es la intersección de los discos abiertos con radio igual a la distancia $d(p, q)$ entre p y q respectivamente, ver Figura 2.3. El conjunto de aristas está definido por $pq \in E \Leftrightarrow$ la luna $L_{p,q}$ no contiene algún otro nodo además de p y q.

2.1.3. Gráfica de Gabriel (GG)

La GG fue introducida por *Gabriel y Sokal* en 1969, y ha sido utilizada para el análisis geométrico y el reconocimiento de patrones. La región de asociación entre dos nodos p y q está especificada por el disco cuyo diámetro queda determinado

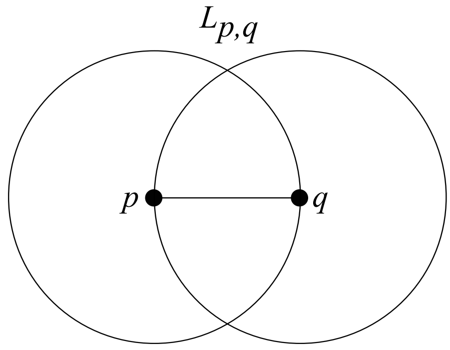


Figura 2.3: Gráfica del vecino más cercano

por p y q como se muestra en la Figura 2.4. Formalmente el conjunto de aristas de GG esta definido por $pq \in E \Leftrightarrow$ el disco centrado en $\frac{p+q}{2}$ y de radio $\frac{d(p,q)}{2}$ no contiene en su interior algún otro nodo además de p y q.

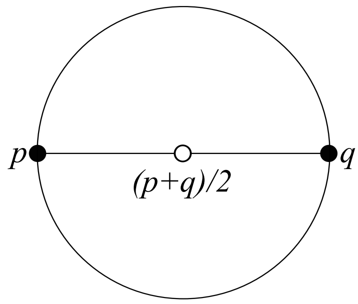


Figura 2.4: Gráfica de Gabriel

Lema 2.1.1. *GG es una gráfica plana.*

Demostración. Sea AB es una arista de GG. Es fácil ver que un nodo X sé encuentra dentro del círculo con diámetro AB si y solo si el ángulo $\angle AXB$ es mayor a $\pi/2$, ver Figura 2.5. Sean A, B, C y D nodos en una GG. Asumimos que AC y BD se intersectan. A, B, C y D deben poder formar un polígono convexo de cuatro vértices, cuya suma de los ángulos interiores de dicho polígono es 2π. Ya que AC (respectivamente BD) es una arista de GG, ninguno de los nodos B y D (respectivamente A y C) puede encontrarse dentro del círculo con diámetro AC (respectivamente BD). Por lo tanto $\angle ABC$, $\angle BCD$, $\angle CDA$ y $\angle DAB < \pi/2$, lo cual contradice el hecho de que $\angle ABC + \angle BCD + \angle CDA + \angle DAB = 2\pi$. □

2.1.4. Triangulación de Delaunay (DT)

La triangulación de Delaunay es otra forma de gráfica de proximidad. Difiere de las anteriores en donde la relación de vecindad está definida sobre triangulaciones definidas por tripletas de nodos. Considera la triangulación $\mathcal{T}$ de un conjunto de nodos P. Para cualquier tripleta de nodos (p, q, r) sea $S_{p,q,r}$ el círculo formado

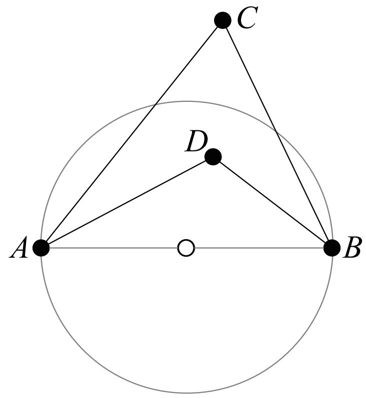


Figura 2.5: $\angle ADB > \pi/2$
$\angle ACB < \pi/2$

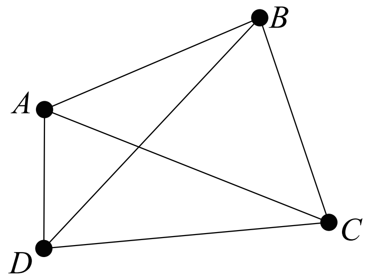


Figura 2.6: Prueba gráfica del Lema 2.1.1

por los puntos p, q y r, ver Figura 2.7. $\mathcal{T}$ es una DT de un conjunto de nodos P si para cualquier triángulo (p, q, r) de la triangulación se cumple que $S_{p,q,r}$ no contiene algún otro nodo de P además de p, q y r.

2.2. Control de la topología

Consideremos una gráfica simple $G = (V, E)$, donde el conjunto de nodos V consiste de los nodos en la red, y el conjunto de las aristas E es la representación de los enlaces de comunicación entre pares de nodos.

La tarea del control de la topología es calcular una subgráfica $G' = (V, E')$, $E' \subseteq E$ y G' es conectada.

2.2.1. Test de Gabriel

Uno de los más importantes test para eliminar enlaces en una red ad hoc es el *test de Gabriel*, el cual se aplica a cada enlace de la red. Se asume que todos los nodos tienen el mismo rango de transmisión r en la red. En el test de Gabriel, si no existe algún nodo dentro del círculo con diámetro AB entonces el enlace entre A y B se conserva. Si por el contrario existe un nodo C dentro del círculo con diámetro AB, el enlace entre A y B es eliminado, como se muestra en Fig 2.8.

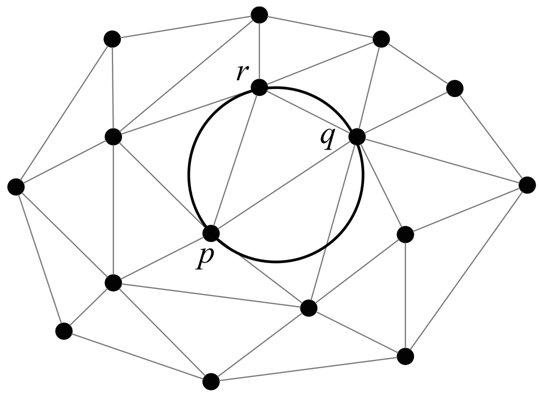


Figura 2.7: Triangulación de Delaunay

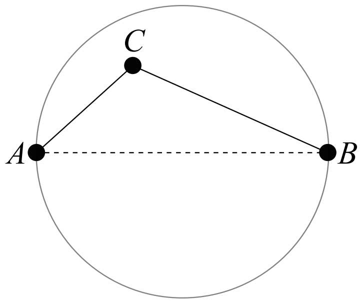


Figura 2.8: Test de Gabriel

Teorema 2.2.1. *Si la red original es conectada, entonces el Test de Gabriel genera una gráfica plana conectada.*

Demostración. Sea G la gráfica conectada de una red ad hoc. Sea $MST(G)$ el árbol de expansión mínima de G. Por el Teorema 2.4.1 se conoce que $MST \subset GG$, en consecuencia GG de G es conectada. Por el Lema 2.1.1 G es plana. □

En general las gráficas de proximidad definidas en la sección anterior pueden ser utilizadas a manera de Test para la construcción de gráficas de expansión en redes ad hoc.

2.3. Árbol de Expansión Mínima Local (LMST)

En esta sección se describirá el algoritmo para el control de la topología en redes wireless multi-hop llamado "Árbol de Expansión Mínima Local" que se encuentra en [15].

En general en el algoritmo, cada nodo construye su árbol de expansión mínima independientemente, y solo mantiene las aristas que inciden en él y por tanto la comunicación con dichos nodos.

Para facilitar la discusión acerca del algoritmo propuesto, primero definiremos algunos términos. Se denota la topología de la red como una gráfica de disco unitario $G = (V, E)$ con máximo radio de transmisión d_{max}, donde V es el conjunto de nodos en la red y $E = \{(u, v) : d(u, v) \leq d_{max}\ u, v \in V\}$ es el conjunto de aristas de G. Cada nodo tiene un identificador único que por simplicidad denotamos como $id(v_i) = i$. También se define la *Vecindad Visible* $NV_u(G)$ del nodo u como sigue.

2.3.1. El Algoritmo LMST

Definición 2.3.1 (Vecindad Visible)**.** *La vecindad visible $NV_u(G)$ es el conjunto de nodos a los cuales el nodo u puede alcanzar utilizando su máximo poder de transmisión, es decir, $NV_u(G) = \{v \in V(G) : d(u, v) \leq d_{max}\}$. Para cada nodo $u \in V(G)$ sea $G_u = (V_u, E_u)$ la gráfica inducida de G tal que $V_u = NV_u$.*

El algoritmo propuesto está compuesto por tres fases: fase de recolección de la información, construcción de la topología y determinación del poder de transmisión, y una fase opcional acerca de la construcción de la topología solo con aristas bidireccionales.

i) *Intercambio de la Información:* La información necesaria por cada nodo u en el proceso de construcción de la topología es la información de todos los nodos en $NV_u(G)$. Esta información se puede obtener a partir de que cada nodo envíe un mensaje "Hola"periódicamente utilizando su máximo poder de transmisión. La información contenida en el mensaje debe incluir el id del nodo y su posición.

ii) *Construcción de la Topología* Después de obtener la información de su vecindad visible $NV_u(G)$, cada nodo u aplica el algoritmo de Prim independientemente, para obtener su árbol de expansión mínima localmente $T_u = (V(T_u), E(T_u))$ de G_u. Nota que la complejidad del algoritmo de Prim es de $O(e + n \log n)$, donde n es el número de nodos y e es el número de aristas de G_u, usando *Fibonacci Heaps*.

    El árbol de expansión mínima resultado del algoritmo de Prim podría no ser único si existen múltiples aristas con el mismo peso, por ello se define una función peso para eliminar los empates entre tales aristas.

    Definición 2.3.2 (Función Peso)**.** *: Dadas dos aristas $(u_1, v_1), (u_2, v_2) \in E(G)$ y la sea la distancia euclidiana $d(*, *)$, la función peso $w : E \to R$ satisface:*
    $w(u_1, v_1) > w(u_2, v_2) \iff d(u_1, v_1) > d(u_2, v_2)$ *ó*
    $(d(u_1, v_1) = d(u_2, v_2) \&\& max\{id(u_1), id(v_1)\} > max\{id(u_2), id(v_2)\})$ *ó*
    $(d(u_1, v_1) = d(u_2, v_2) \&\& max\{id(u_1), id(v_1)\} = max\{id(u_2), id(v_2)\}$ *y*
    $min\{id(u_1), id(v_1)\} > min\{id(u_1), id(v_1)\}$

    La función de peso w garantiza que cada paso del algoritmo de Prim, la selección de la arista de menor peso sea única, así que el árbol de expansión mínima T_u construido por el nodo u, es único.

    Posterior a que el nodo u construye el árbol de expansión mínima de su vecindad visible, tiene que determinar con cuales vecinos conservará conectividad. Para facilitar esta discusión, definimos la *relación Vecino* y el *conjunto Vecino*:

Definición 2.3.3 (Relación Vecino y Conjunto Vecino). *Se denota la relación vecino entre dos nodos u y v como $u \to v$, en donde el nodo v es vecino del nodo u si y solo si existe una arista $(u, v) \in E(T_u)$. El conjunto vecino $N(u)$ del nodo u es $N(u) = \{v \in V(G_u) : u \to v\}$. $u \leftrightarrow v$ si y solo si $u \to v$ y $v \to u$.*

La relación vecino definida arriba no es simétrica, es decir, $u \to v$ no necesariamente implica $v \to u$. La Figura 2.9 muestra tal ejemplo. Supongamos los siguientes seis nodos $V = \{u, v, w_1, w_2, w_3, w_4\}$, donde $d(u, v) = d < d_{max}$, $d(u, w_4) < d_{max}$, $d(u, w_i) < d_{max}, i = 1, 2, 3$ y $d(v, w_j) < d_{max}, j = 1, 2, 3, 4$. Ya que $NV_u\{u, v, w_4\}$, podemos obtener de T_u que $u \to v$ y $u \to w_4$, por otro lado $NV_v = \{u, v, w_1, w_2, w_3, w_4\}$, por lo que $v \to w_1$. En este ejemplo se tiene que $u \to v$ pero no $v \nrightarrow u$.

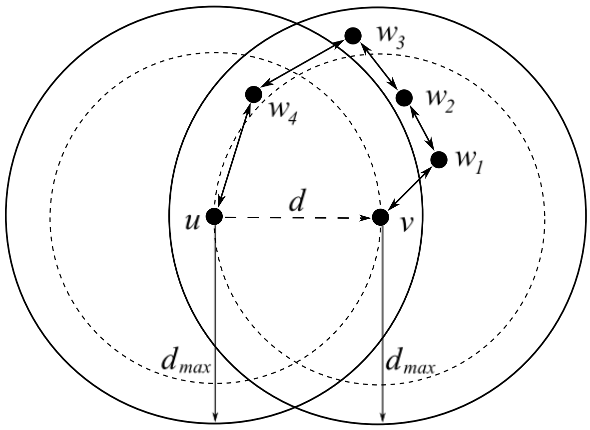


Figura 2.9: Ejemplo de que las aristas derivadas de la topología LMST pueden tener una sola dirección

Definición 2.3.4 (Topología G_0). *La topología generada bajo el LMST es una gráfica dirigida $G_0 = (V_0, E_0)$, donde $V_0 = V$ y $E_0 = \{(u, v) : u \to v,\ u, v \in V(G)\}$*

III) *Construcción de la topología solo con aristas bidireccionales*: Como se ilustró en la Figura 2.9 algunas aristas en G_0 tienen solo una dirección. Pero como se mencionó anteriormente es deseable que la topología de la red consista solo de aristas bidireccionales. Existen dos posibles soluciones: 1) forzar a todas la aristas que tienen una sola dirección a convertirse en bidireccionales; o 2) eliminar todas las aristas con una sola dirección. Llamamos a las dos nuevas topologías G_0^+ y G_0^- respectivamente.

Definición 2.3.5 (Topología G_0^+). *La topología G_0^+ es una gráfica no dirigida $G_0^+ = (V_0^+, E_0^+)$, donde $V_0^+ = V_0$, y $E_0^+ = \{(u, v) : (u, v) \in E(G_0)$ o $(v, u) \in E(G_0)\}$.*

Definición 2.3.6 (Topología G_0^-). *La topología G_0^- es una gráfica no dirigida $G_0^- = (V_0^-, E_0^-)$, donde $V_0^- = V_0$, y $E_0^- = \{(u,v) : (u,v) \in E(G_0)$ y $(v,u) \in E(G_0)\}$.*

Para convertir G_0 a G_0^+ o G_0^-, cada nodo u puede probar cada uno de sus vecinos en el conjunto de vecino $N(u)$ para averiguar si la arista correspondiente es uni-direccional, y si es el caso eliminar la arista (G_0^-) o notificar a su vecino para que agregue la arista de reversa (G_0^+).

2.3.2. Conectividad en la red

Se probará que la topología G_0 derivada de LMST conserva la conectividad de G. Para cualesquiera dos nodos $u, v \in V(G_0)$, se dice que el nodo u está conectado al nodo v (denotado como $u \Leftrightarrow v$) si y solo si existe un camino $(q_0 = u, q_1, \ldots, q_{m-1}, q_m = v)$ tal que $q_j \leftrightarrow q_{j+1}$, $j = 0, 1, \ldots, m-1$, donde $q_k \in V(G_0)$, $k = 1, 0, \ldots, m$.

Lema 2.3.1. *Para cualquier par de nodos $[u,v]$, $u, v \in V(G_0)$, si $w(u,v) \leq d_{max}$, entonces $u \Leftrightarrow v$*

Demostración. Para cualquier par de nodos $[u,v]$ que satisfacen que $w(u,v) \leq d_{max}$ y $u, v \in V(G_0)$, se ordenarán los pesos $w(u,v)$ de forma ascendente, es decir, $w(u_1, v_1) < w(u_2, v_2) < \cdots < w(u_l, v_l)$. Se probará por inducción en el rango de los pares de nodos en el orden.

I) Para $k = 1$. Sea el par $[u_1, v_1]$ el primero en el orden, se satisface que $w(u_1, v_1) = min_{(u,v) \in V(G_0)}\{w(u,v)\}$ y $w(u_1, v_1) \leq d_{max}$. Entonces $u \leftrightarrow v$, lo que significa $u \Leftrightarrow v$.

II) *Hipótesis de Inducción* Se asume que el Lema 2.3.1 se mantiene para los pares $[u_i, v_i]$, $i = 1, 2, ..., k-1$. Ahora se probará que el Lema 2.3.1 también se mantiene para el caso del par $[u_k, v_k]$.

III) *Tesis de Inducción:* Se consideran dos casos:

- *Caso 1:* $u_k \leftrightarrow v_k$, lo que implica $u_k \Leftrightarrow v_k$.
- *Caso 2:* Cualquiera $u_k \nleftrightarrow v_k$, $v_k \nleftrightarrow u_k$ o ambas.

Sin pérdida de generalidad, se asume $u_k \nleftrightarrow v_k$. Ya que $v_k \in NV_{u_k}$, existe un único camino $p = (q_0 = u_k, q_1, q_2, \cdots, q_{m-1}, q_m = v_k)$ del nodo u_k al nodo v_k, donde $(q_i, q_{i+1}) \in E(T_{u_k})$, $i = 0, 1, ..., m-1$. Ya que T_{u_k} es el único MST de G_{u_k}, tenemos que $w(q_i, q_{i+1}) < w(u_k, v_k)$. Aplicando la hipótesis de inducción a cada par $[q_i, q_{i+1}]$, $i = 0, 1, ..., m-1$, se tiene que $q_i \Leftrightarrow q_{i+1}$, por lo tanto $u_k \Leftrightarrow v_k$.

□

Teorema 2.3.1 (Conectividad). *G_0 conserva la conectividad de G, es decir, G_0 es conectada si G es conectada.*

Demostración. Se supondrá que G es conectada. Para cualesquiera dos nodos $u, v \in V(G)$, existe al menos un camino $p = (q_0 = u, q_1, q_2, \cdots, q_{m-1}, q_m = v)$ de u a v, donde $(q_i, q_{i+1}) \in E(G)$, $i = 0, 1, ..., m-1$ y $w(q_i, q_{i+1}) < d_{max}$. Dado que $q_i \Leftrightarrow q_{i+1}$ por el Lema 2.3.1, tenemos que $u \Leftrightarrow v$ $\square$

Teorema 2.3.2. *G_0^- conserva la conectividad de G, es decir, G_0^- es conectada si G es conectada.*

Demostración. Si el par de nodos $[u, v]$, $u, v \in V(G_0)$ satisface que $w(u, v) < d_{max}$, por el Lema 2.3.1, existe un camino $p = (q_0 = u, q-1, q_2, \cdots, q_{m-1}, q_m = v)$ tal que $q_j \leftrightarrow q_{j+1}$, $j = 0, 1, \cdots, m-1$, donde $q_k \in V(G_0)$, $k = 0, 1, \cdots, m$. El mismo resultado se mantiene para G_0^- dado que todas las arista en p son bidireccionales y el quitar las aristas de una sola dirección no afecta la existencia de dicho camino. Siguiendo el mismo argumento que se presentó en el teorema anterior, se puede probar que G_0^- mantiene la conectividad de G. $\square$

2.4. Relación entre las gráficas de proximidad

Se puede probar el siguiente teorema.

Teorema 2.4.1. *Se satisfacen las siguientes inclusiones por las gráficas descritas anteriormente*

$$MST \subset LMST \subset RNG \subset GG \subset DT,$$

donde MST denota al árbol de expansión mínima.

Demostración. La Figura 2.9 muestra claramente que $MST \subset LMST$. La Figura 2.10 esboza la prueba de que LMST es un subconjunto de RNG. La Figura 2.11 esboza la prueba de que RNG es un subconjunto de GG. La prueba de que GG es subconjunto de DT es representada en la Figura 2.12 y se utiliza el hecho de que DT es la gráfica dual de un diagrama de Voronoi [1] $\square$

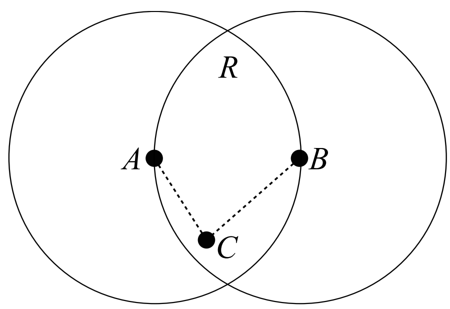


Figura 2.10: $LMST \subset RNG$

[1]Ver *Preparata y Shamos* [19] para mas detalles.

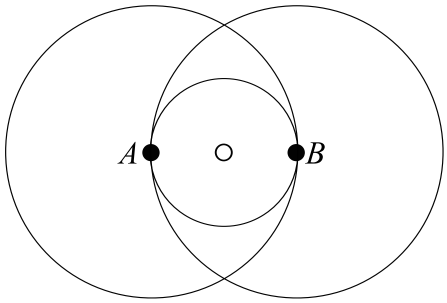


Figura 2.11: $RNG \subset GG$

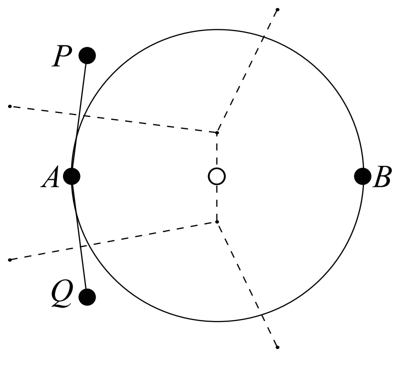


Figura 2.12: $GG \subset DT$

Capítulo 3

Conservando la conectividad por proximidad en espacios libres

En una red móvil ad hoc, el movimiento de los agentes provoca cambios en la topología de la red, en este capítulo se presentará un servicio distribuido que coordina el movimiento de dichos agentes y garantiza que la red de comunicación permanecerá conectada en todo momento. *Cornejo* [7].

3.1. Definiciones

Al igual que en capítulos anteriores, se asumirá que los dispositivos en la red ad hoc móvil cuentan con un GPS y un dispositivo de comunicación de radio r. Dichos dispositivos se encuentran en un espacio abierto y libre de obstáculos.

El servicio es distribuido, es decir, cada agente ejecuta el algoritmo, y por cada ronda se produce una trayectoria que mantiene la conectividad, y cuando es posible, acerca al agente a su objetivo.

Definición 3.1.1 (Disco abierto). *El disco abierto centrado en un punto p con radio r está formado por el conjunto de puntos que se encuentran a una distancia menor a r de p. Formalmente* $\mathrm{disk}_r(p) = \{q : \| p - q \| < r\}$.

El disco unitario para un punto p se denota como $\mathrm{disk}_1(p)$.

Definición 3.1.2 (Círculo). *El círculo centrado en un punto p con radio r está formado por el conjunto de puntos que se encuentran a distancia r de p. Formalmente* $\mathrm{circle}_r(p) = \{q : \| p - q \| = r\}$.

Definición 3.1.3 (Disco cerrado). *El disco cerrado centrado en un punto p con radio r está formado por el conjunto de puntos que se encuentran a distancia de a lo más r de p. Formalmente* $\overline{\mathrm{disk}}_r(p) = \mathrm{circle}_r(p) \cup \mathrm{disk}(p) = \{q : \| p - q \| \leq r\}$.

Definición 3.1.4 (Trayectoria). *Un agente i se mueve desde una posición s_i a una posición p_i siguiendo una trayectoria γ_i, donde $\gamma_i : [0, 1] \to \mathbb{R}^2$, $\gamma_i(0) = s_i$ y $\gamma_i(1) = p_i$.*

Definición 3.1.5 (Gráfica de configuración)**.** *Se dice que* $C = \langle I, F \rangle$ *es una gráfica de configuración si* G *es una gráfica de disco unitario donde cada agente* $i \in I$ *tiene coordenada origen* $s_i \in \mathbb{R}^2$ *y coordenada objetivo* $t_i \in \mathbb{R}^2$ *con distancia* $d_i = \parallel s_i - t_i \parallel$ *y radio de comunicación* r*.*

Se dice que un conjunto de agentes I es conectado, si y solo si su gráfica de configuración es conectada.

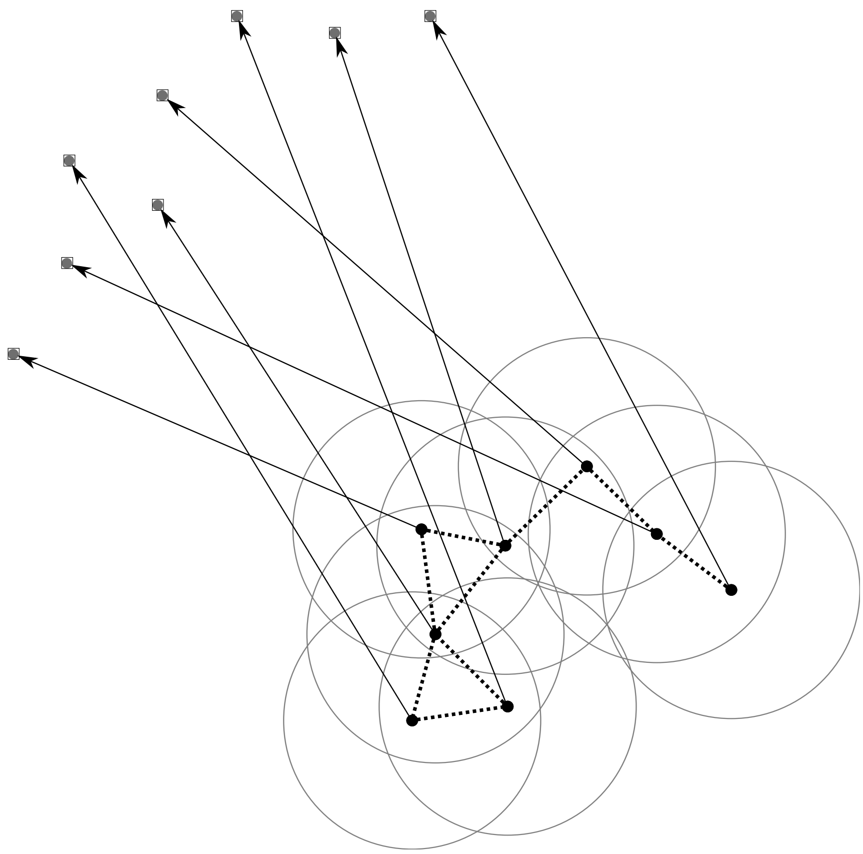


Figura 3.1: Gráfica de configuración

3.2. Servicio distribuido de conectividad

El servicio de conectividad solo necesita tener conocimiento de la posición actual del agente y la posición de su objetivo. Dicho servicio genera una trayectoria recta en cada ronda del algoritmo, la trayectoria compuesta por la ejecución de varias rondas no necesariamente es recta. La trayectoria generada cada ronda es calculada de manera que se conserva la conectividad en la gráfica de configuración incluso si algún agente no hace el recorrido completo de la trayectoria calculada, lo que permite a los agentes parar y decidir a la velocidad a la que se mueven.

El algoritmo es parametrizado por una función Filtro que determina un subconjunto suficiente de vecinos de tal manera que se conserve la conectividad entre esos vecinos, y así mantener la conectividad global. Es decir se requiere una función que minimice el número de aristas en la gráfica de configuración.

3.2.1. Función Filtro

Se asume que la gráfica de configuración es conectada. El objetivo de la función filtro es obtener una subgráfica de expansión de la gráfica de configuración, a manera de minimizar el número de enlaces existentes en la red garantizando la conectividad, en otras palabras, se requiere del control de la topología en la gráfica de configuración.

Definición 3.2.1 (Función FILTRO(N_i, s_i)). *Sea N_i el conjunto de vecinos del agente s_i. La función* FILTRO(N_i, s_i) *regresa un subconjunto de vecinos $N_i' \subseteq N_i$, tal que preservando los enlaces de s_i con los agentes del subconjunto N_i' es suficiente para garantizar la conectividad de la gráfica de configuración.*

Una selección natural para la función Filtro sería calcular el árbol de expansión mínima (MST) de la gráfica de configuración, sin embargo se debe recordar que la información con la que cuenta cada agente es local, y por lo tanto esta no es una opción óptima.

Existe algoritmos bien conocidos que calculan una subgráfica de expansión conectada de manera local, los cuales se abordaron en el Capítulo 2, entre ellos se encuentra la gráfica de Gabriel (GG), la gráfica de vecindad relativa (RNG) y el árbol de expansión mínima local (LMST). Todas estas estructuras son conectadas y se pueden calcular de manera local.

Ya que se busca eliminar el mayor número de aristas en la gráfica de configuración, y por el Teorema 2.4.1 se sabe que $MST \subset LMST \subset RNG \subset GG \subset DT$, la mejor selección es el $LMST$.

La subgráfica de expansión que regresa el LMST va a depender de la posición de los agentes, la cual varía en dada ronda, por lo tanto el uso de la función filtro permite la conectividad sin tener que conservar un conjunto fijo de vecinos durante toda la ejecución. Es posible que ninguna de las aristas originales aparezcan el la gráfica final.

3.2.2. El algoritmo

El algoritmo consta de tres fases, una primera de recolección, una de propuesta y una última de ajuste.

En la *fase de recolección* cada agente envía información a sus agente vecinos acerca de su ubicación actual y la posición de su objetivo (s_i y t_i respectivamente), a su vez el agente recolecta la misma información de sus vecinos.

En la *fase de propuesta* se hace uso de la función FILTRO, la cual deja a cada agente con un subconjunto de vecinos. Posteriormente se calcula una propuesta de objetivo óptimo p_i (el objetivo es óptimo en el sentido de que el movimiento del agente de s_i a t_i no desconecta la red). La posición de la propuesta es enviada al mismo tiempo que se recolecta la información de las propuestas de los agentes vecinos.

Finalmente en la *fase de ajuste*, cada agente checa si los vecinos que conservó después de la función FILTRO serán alcanzables después de que cada agente se mueva a su objetivo propuesto. Si todos los vecinos serán alcanzables, entonces el agente se mueve a p_i, de otra manera solo se moverá la mitad de la distancia a p_i. Con esto se asegura que se mantiene conectividad.

Algoritmo 2 Algoritmo del Servicio de Conectividad para un agente i

FASE DE RECOLECCIÓN:
$s_i \leftarrow$ posición del agente()
$t_i \leftarrow$ posición del objetivo()
enviar s_i a todos los vecinos
$N_i \leftarrow \{s_i|$ para cada s_j recibido$\}$

FASE DE PROPUESTA:
$N'_i \leftarrow Filtro(N_i, s_i)$
$R_i \leftarrow \bigcap_{s_j \in N'_i} disk_1(s_j)$
$p_i \leftarrow$ punto dentro de R_i más cercano a t_i
enviar p_i a todos los vecinos.
$P_i \leftarrow \{p_j|$ para cada p_j recibido$\}$

FASE DE AJUSTE:
If $\forall s_j \in N'_i \ \|p_j - p_i\| \leq r$ **Then**
  return trayectoria de s_i a p_i
Else if
  return trayectoria de s_i a $s_i + \frac{1}{2}(p_i - s_i)$
End If

3.3. Conservando la conectividad

En esta sección se probará la conectividad de la red para cualquier función Filtro válida.

Observe en el Algoritmo 2 que R_i es la intersección de un conjunto de discos que contienen dentro a s_i, se sigue que R_i es una región convexa y contiene en su interior a s_i. Por construcción también $p_i \in R_i$ y por lo tanto, bajo la definición de convexidad, la trayectoria lineal entre s_i y p_i está totalmente contenida dentro de R_i. Dado lo anterior se entiende que la gráfica quedaría conectada si el agente i se mueve de s_i a p_i y el resto de los agentes permanecieran en su posición.

En el siguiente teorema se muestra que los agentes permanecerán conectados independientemente de su velocidad de movimiento o inclusive si alguno de ellos para abruptamente en cualquier punto de su trayectoria.

Lema 3.3.1 (Ajuste)**.** *Las propuestas ajustadas de los agentes son conectadas.*

Demostración. Sean $p'_i = s_i + \frac{1}{2}(p_i - s_i)$ y $p'_j = s_j + \frac{1}{2}(p_j - s_j)$ las propuestas ajustadas de los agentes i y j respectivamente. Por construcción $\| s_i - p_j \| \leq r$ y

$\| s_j - p_i \| \leq r$, entonces las propuestas ajustadas son conectadas:

$$\begin{aligned}
\| p'_i - p'_j \| &= \| s_i - s_j + \frac{1}{2}(p_i - p_j + s_j - s_i) \| \\
&= \| \frac{1}{2}(s_i - s_j + p_i - p_j) \| \\
&\leq \frac{1}{2}(\| s_i - p_j \| + \| s_j - p_i \|) \\
&\leq r
\end{aligned}$$

□

Teorema 3.3.1 (Teorema de seguridad)**.** *Si la función* FILTRO *es válida, el servicio mantiene la conectividad de la gráfica.*

Demostración. Sean los agentes vecinos i y j. Si $\| p_i - p_j \| > r$, ambos agentes ajustan sus propuestas y permanecen conectados por el Lema 3.3.1. Si $\| p_i - p_j \| \leq r$ y ninguno ajusta, entonces ambos están trivialmente conectados. Si $\| p_i - p_j \| \leq r$ pero (sin pérdida de generalidad) i ajusta su propuesta y j no, entonces $s_i, p_i \in \text{disk}_1(p_j)$ y por convexidad $p'_i \in \text{disk}_1(p_j)$ donde $\| p'_i - p_j \| \leq r$. □

Teorema 3.3.2 (Teorema de robustez)**.** *Las trayectorias lineales que siguen agentes vecinos, son robustas.*

Demostración. Sean i y j vecinos fijos. Ya que cada trayectoria propuesta por los agentes es lineal, se probará que todos los puntos intermedios contenidos en dichas trayectorias mantienen la conectividad. Sean los puntos fijos intermedios $q_i \in \gamma_i$ y $q_j \in \gamma_j$ de dichas trayectorias. $s_i, p_i \in disk_1(s_j) \cap disk_1(p_j)$ y por convexidad $q_i \in disk_1(s_j) \cap disk_1(p_j)$. Similarmente $s_j, p_j \in disk_1(s_i) \cap disk_1(p_i)$ y por convexidad $q_j \in disk_1(s_i) \cap disk_1(p_i)$, por lo tanto $\| q_i - q_j \| \leq r$. □

3.4. Asegurando el progreso

Para que el algoritmo sea útil, además de conservar la conectividad como se probó en la sección anterior, debe también garantizar que los agentes tienen cierto avance (progreso) y eventualmente alcanzan sus metas.

Antes de probar progreso se identifican varias condiciones sutiles sin las cuales ningún algoritmo local puede garantizar a la vez conectividad y progreso.

3.4.1. Ciclos

Se considera una configuración donde los nodos se encuentran en un ciclo, dos vecinos desean moverse y romper el ciclo y los nodos restantes quieren permanecer en sus posiciones, claramente ningún algoritmo local puede garantizar el progreso. Dado que no se cuenta con información global los agentes no pueden distinguir si se encuentran dentro de un ciclo o en una cadena, y en el último caso cualquier movimiento de los agentes atentaría contra la conectividad.

Para demostrar el progreso, en el resto del capítulo se asume que no hay ciclos en la gráfica filtrada.

Si las metas son desconectadas, claramente no se puede obtener progreso sin violar la conectividad, por lo cual es necesario asumir que la gráfica unitaria de las metas es conectada.

3.4.2. Gráficas de dependencia

Se definirá una región $\text{region}(S) = \bigcap_{s \in S} \text{disk}_1(s)$ y un punto propuesta como $\text{proposal}(S, t) = \text{argmin}_{p \in \text{region}(S)} \parallel p - t \parallel$.

Un nodo con conjunto de vecinos filtrado N' y meta t depende de k vecinos (tiene dependencia k) si existe un subconjunto $S \subseteq N'$ de tamaño $|S| = k$ tal que la propuesta $\text{prosal}(S, t) = \text{prosal}(N', t)$, pero $\text{prosal}(S', t) \neq \text{prosal}(N', t)$, para cualquier subconjunto $S' \subseteq N'$ de menor tamaño $|S'| < k$.

El siguiente Lema da una cota de dependencia.

Lema 3.4.1. *Cada agente depende de a lo más dos vecinos.*

Demostración. Sea el agente i con vecindad filtrada N', meta t, y $R = \text{region}(N')$. Si $t \in R$ entonces $\text{proposal}(N', t) = \text{proposal}(\varnothing, t) = t$ y el agente i no depende de ningún vecino. Si $t \notin R$ entonces $\text{prosal}(N', t)$ regresa un punto p en el límite de la región R. Ya que R es la intersección de un conjunto finito de discos se sigue que p se encuentra en el límite de un solo disco para el caso en el que i dependa de un solo vecino, o si p se encuentra en la intersección de dos discos en ese caso i depende de a lo más dos vecinos. □

Dado el Lema 3.4.1, para cualquier configuración $C = \langle I, F \rangle$ podemos considerar su gráfica de dependencia $D = \langle I, E \rangle$ en donde existe una arista dirigida $(u, v) \in E$ si el nodo u depende del nodo v. Por lo tanto D es una subgráfica dirigida de C con grado máximo de dos. Para el objetivo de demostrar progreso asumiremos que no existen ciclos en C, dado que ningún servicio local puede manejarlo. Esto implica que en D solo existen ciclos dirigidos simples de longitud dos, se hará referencia a dichas gráficas de dependencias como *nice graphs*.

Definición 3.4.1 (Pre-cadena). *Una pre-cadena H es una secuencia de vértices $\langle v_i \rangle_{i \in 1..n}$ tal que existe un ciclo simple entre v_i, v_{i+1} $(i \in 1..n-1)$.*

Se debe observar que un nodo v es considerado una pre-cadena. Enseguida se demostrará que cualquier gráfica de dependencia *nice graph* contiene una pre-cadena no vacía H la cual no tiene aristas de salida.

Teorema 3.4.1. *Cada nice graph finita $G = \langle V, E \rangle$ contiene una pre-cadena no vacía $H \subseteq V$ que no contiene aristas de salida.*

Demostración. Sea la gráfica $G = \langle V, E \rangle$ se considera la gráfica G' que resulta de contraer iterativamente los vértices $u, v \in V$ si $(u, v) \in E$ y $(v, u) \in E$. Claramente G' es una *nice graph* finita, y cualquier vértice v' en G' es una pre-cadena de G, sin embargo G' no contiene ningún ciclo dirigido. Se sigue que cualquier camino dirigido en G' empieza en algún vértice arbitrario u', y ya que la gráfica es finita y no contiene ciclos, eventualmente se alcanzara un vértice v' que no tenga aristas de salida, tal vértice es una pre-cadena que no contiene aristas de salida. $\square$

Por el Teorema 3.4.1 cualquier límite inferior para el progreso en cadenas también se mantiene para configuraciones generales. En el Articulo [7] se explica con mucho más detalle la prueba de progreso del Algoritmo 2.

Capítulo 4

Conservando la conectividad por visibilidad en espacios acotados

En el capítulo anterior se presentó un servicio distribuido para agentes que se encuentran en espacios abiertos, es decir, sin algún obstáculo que pudiera acotar su radio de comunicación y con ello impedir que se cumpla con el objetivo de conectividad.

En este capítulo se presentará un algoritmo distribuido que conserva la conectividad de agentes móviles dentro de espacios acotados por una región poligonal. Se asume que cada agente cuenta con un sistema GPS y un dispositivo de comunicación que le permite el intercambio de datos con otros agentes visibles.

4.1. Definiciones

Definición 4.1.1 (Trayectoria rectilínea)**.** *Un agente i se mueve desde una posición s_i a una posición p_i siguiendo el segmento γ_i, donde $\gamma_i : [0,1] \to \mathbb{R}^2$, $\gamma_i(0) = s_i$ y $\gamma_i(1) = p_i$.*

Definición 4.1.2 (Gráfica de configuración)**.** *Sea un polígono simple P. Se dice que $G = \langle I, F \rangle$ es una gráfica de configuración en donde el conjunto de vértices I representa al conjunto de agentes al interior de P, y existe una arista $(u, v) \in F$ si u y v son mutuamente visibles dentro de P. Cada agente $i \in I$ tiene coordenada origen $s_i \in \mathbb{R}^2$ y coordenada objetivo $t_i \in \mathbb{R}^2$, tal que s_i y t_i se encuentran en P. Además $SP_i(s_i, t_i)$ es el trayecto más corto dentro de P del punto s_i a t_i, para el agente i.*

4.2. Algoritmo distribuido de conectividad para agentes dentro de una región poligonal

El algoritmo que se presenta a continuación produce en cada ronda de ejecución una trayectoria rectilínea γ_i para cada agente i basándose el la trayectoria más corta $SP(s_i, t_i)$, dicha trayectoria rectilínea γ_i mantiene la conectividad y cuando es posible acerca al agente a su objetivo.

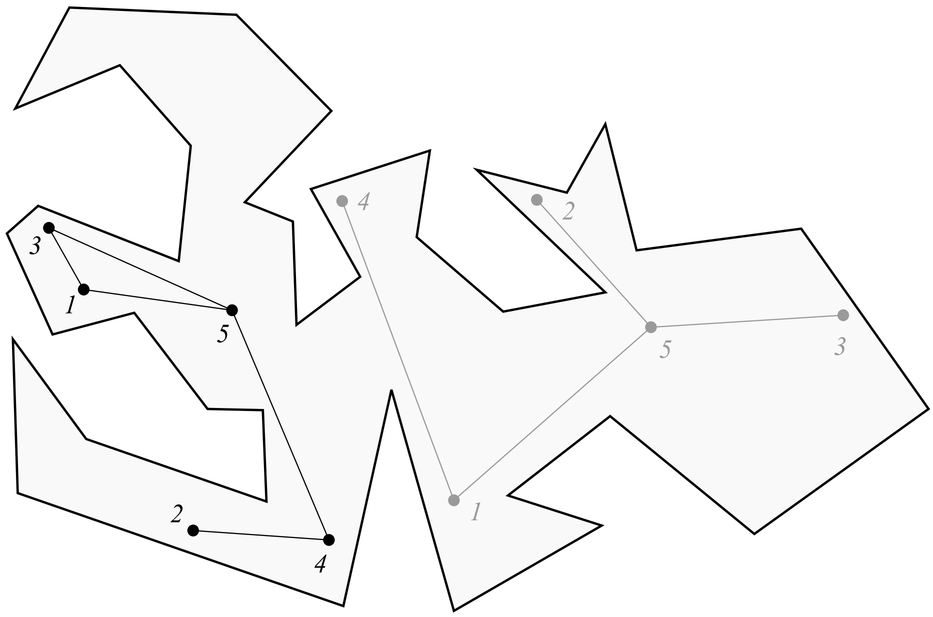


Figura 4.1: Gráfica de configuración

Al igual que en el algoritmo presentado en el capítulo anterior, este también es parametrizado por una función Filtro, dicha función cumple los mismos objetivos.

El algoritmo consiste de dos fases principalmente, una primera fase dedicada a la recolección de información local y la segunda fase dedicada a generar una propuesta de movimiento para cada agente.

En la *fase de recolección* cada agente obtiene la información acerca de la ubicación actual dentro del polígono de los agentes que le son visibles, y envía la información de su posición actual a esos mismos agentes.

En la *fase de propuesta* se hace uso de la función Filtro para obtener un subconjunto de vecinos suficiente para mantener la comunicación global de la gráfica de configuración. Posteriormente se utiliza una función a la que llamaremos Macs, dicha hace calculó de una región convexa R_i donde cada agente i pueda moverse libremente sin perder comunicación con el subconjunto de agentes que se calculo en la función Filtro. Para finalizar se calcula un objetivo óptimo $p_i \in R_i$, $p_i \in SP(s_i, t_i)$.

Por último solo resta que el agente se traslade a p_i y ejecute una nueva ronda hasta alcanzar a t_i.

4.3. Función *Filtro*

El algoritmo hace uso de una función Filtro, que determina un subconjunto suficiente de vecinos de tal manera que se mantenga la gráfica de configuración conectada, es decir, esta función debe minimizar el número de aristas de la gráfica de configuración, tal como se realiza en el algoritmo del capítulo anterior.

Definición 4.3.1. *Sea N_i el conjunto de vecinos del agente s_i. La función*

Algoritmo 3 Algoritmo de conectividad para el agente i

FASE DE RECOLECCIÓN:
$s_i \leftarrow \textit{posicion\_actual}$
$t_i \leftarrow \textit{posicion\_objetivo}$
Envía s_i a todos sus vecinos
$N_i \leftarrow \{s_j |$ por cada s_j recibido $\}$
FASE DE PROPUESTA:
$N'_i \leftarrow$ Filtro (N_i, s_i)
$R_i \leftarrow \bigcap_{s_j \in N'_i} \text{Macs}(s_i s_j)$
$p_i \leftarrow$ punto en de R_i más cercano a t_i
RETURN Trayectoria rectilínea γ_i de s_i a p_i

Filtro(N_i, s_i) *regresa un subconjunto de vecinos* $N'_i \subseteq N_i$*, tal que conservando la conectividad entre* s_i *con los agentes del subconjunto* N'_i *es suficiente para garantizar la conexidad de la gráfica de configuración, y por lo tanto la conectividad global.*

A esto se le conoce como control de la topología de la red, y su principal objetivo es reducir el número de enlaces existentes en la red, de manera que el enrutamiento sea fácil y rápido.

Como se estudió en el Capítulo 2 y se analizó en el Capítulo 3 existen algoritmos que calculan de manera local una subgráfica de expansión de una gráfica de configuración en espacios libres de obstáculos. Sin embargo, para el escenario que se ha descrito, no todos los algoritmos cumplen con el objetivo de obtener una subgráfica de expansión conectada, ya que la mayoría de ellos se basan en un modelo de gráfica de disco unitario, y nuestro modelo se basa en una gráfica de visibilidad. Ver Figura 4.2.

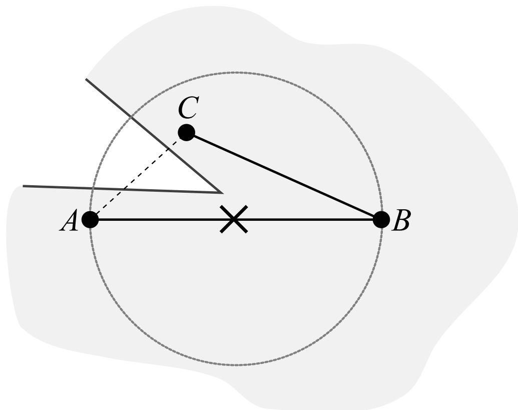


Figura 4.2: El Test de Gabriel deja al agente A desconectado.

Para lograr el objetivo se ha hecho una adaptación del LMST presentado en el Capítulo 2, la diferencia principal es la definición de la gráfica sobre la cual se aplica el LMST.

4.3.1. Árbol de expansión mínima local para una gráfica de visibilidad

En esta sección retomaremos el algoritmo para calcular el árbol de expansión mínima local que se encuentra en la sección 2.3.1.

Se denotará la gráfica de la red construida bajo visibilidad como una gráfica no dirigida $G = (V, E)$ en el plano, donde V es el conjunto de vértices de la red y $E = \{(u, v) : v\ es\ visible\ con\ u\ u, v \in V\}$ es el conjunto de aristas de G. Cada agente tiene un único identificador, por simplicidad denotamos $id(v_i) = i$. Definiremos la *Vecindad Visible* NV_u del vértice u como sigue:

Definición 4.3.2 (Vecindad Visible)**.** *La vecindad visible de un vértice u, $NV_u(G)$, es el conjunto de nodos a los cuales el nodo u puede ver desde su ubicación, es decir, $NV_u(G) = \{v \in V(G) : u\ es\ visible\ con\ v\}$. Para cada nodo $u \in V(G)$ sea $G_u = (V_u, E_u)$ la gráfica inducida de G tal que $V_u = NV_u$.*

Asumimos que la comunicación es simétrica, y cada nodo está equipado con la habilidad de reunir toda la información de cada vecino visible por ese medio.

I) *Intercambio de la Información:* La información necesaria por cada nodo u en el proceso de construcción de la topología es la información de todos los nodos en $NV_u(G)$. Esta información se puede obtener a partir de que cada nodo envíe un mensaje "Hola"periódicamente. La información contenida en el mensaje debe incluir el id del nodo y su posición

II) *Construcción de la Topología:* Después de obtener la información de su vecindad visible $NV_u(G)$, cada nodo u aplica el algoritmo de Prim independientemente, para obtener su árbol de expansión mínima localmente $T_u = (V(T_u), E(T_u))$ de G_u. Nótese que la complejidad del algoritmo de Prim es de $O(e + n \log n)$, donde n es el número de nodos y e es el número de aristas de G_u, usando *Fibonacci Heaps.*

El árbol de expansión mínima resultado del algoritmo de Prim podría no ser único si existen múltiples aristas con el mismo peso, por ello se define una función peso:

Definición 4.3.3 (Función Peso)**.** *: Dadas dos aristas $(u_1, v_1), (u_2, v_2) \in E(G)$ y la sea la distancia euclidiana $d(*, *)$, la función peso $w : E \to R$ satisface:*
$w(u_1, v_1) > w(u_2, v_2) \iff d(u_1, v_1) > d(u_2, v_2) or$
$(d(u_1, v_1) = d(u_2, v_2)\&\&max\{id(u_1), id(v_1)\} > max\{id(u_2), id(v_2)\}) or$
$(d(u_1, v_1) = d(u_2, v_2)\&\&max\{id(u_1), id(v_1)\} = max\{id(u_2), id(v_2)\}\&\&$
$min\{id(u_1), id(v_1)\} > min\{id(u_1), id(v_1)\}$

La función de peso w garantiza que cada paso del algoritmo de Prim, la selección de la arista de menor peso sea única, así que el árbol de expansión mínima T_u construido por el nodo u, es único.

Posterior a que el nodo u construye el árbol de expansión mínima de su vecindad visible, tiene que determinar con cuales vecinos conservará conectividad. Para facilitar esta discusión, definimos la *relación Vecino* y el *conjunto Vecino*:

Definición 4.3.4 (Relación Vecino y Conjunto Vecino)**.** *Denotamos la relación vecino entre dos nodos u y v como $u \to v$, en donde el nodo v es vecino del nodo u si y solo si existe una arista $(u, v) \in E(T_u)$. El conjunto vecino $N(u)$ del nodo u es $N(u) = \{v \in V(G_u) : u \to v\}$. $u \leftrightarrow v$ si y solo si $u \to v$ y $v \to u$.*

Definición 4.3.5 (Topología G_0)**.** *La topología generada bajo el LMST es una gráfica dirigida $G_0 = (V_0, E_0)$, donde $V_0 = V$ y $E_0 = \{(u, v) : u \to v,\ u, v \in V(G)\}$*

III) *Construcción de la topología solo con arista bidireccionales*: Algunas aristas en G_0 son solo hacia una sola dirección. Pero como se mencionó antes, es deseable que la topología de la red consista solo de arista bidireccionales. Existen dos posibles soluciones: (i) forzar a todas la aristas que tienen una sola dirección a convertirse en bidireccionales; o (ii) eliminar todas las aristas con una sola dirección. Llamamos a las dos nuevas topologías G_0^+ y G_0^- respectivamente.

Definición 4.3.6 (Topología G_0^+)**.** *La topología G_0^+ es una gráfica no dirigida $G_0^+ = (V_0^+, E_0^+)$, donde $V_0^+ = V_0$, y $E_0^+ = \{(u, v) : (u, v) \in E(G_0)$ o $(v, u) \in E(G_0)\}$.*

Definición 4.3.7 (Topología G_0^-)**.** *La topología G_0^- es una gráfica no dirigida $G_0^- = (V_0^-, E_0^-)$, donde $V_0^- = V_0$, y $E_0^- = \{(u, v) : (u, v) \in E(G_0)$ y $(v, u) \in E(G_0)\}$.*

Para convertir G_0 a G_0^+ o G_0^-, cada nodo u puede probar cada uno de sus vecinos en el conjunto de vecino $N(u)$ para averiguar si la arista correspondiente es uni-direccional, y si es el caso eliminar la arista (G_0^-) o notificar a su vecino para que agregue la arista de reversa (G_0^+).

En este caso nos interesa que la topología tenga el menor número de aristas posibles, por lo que optaremos por convertir G_0 a G_0^-.

4.3.2. Conectividad en la red

Probaremos que la topología G_0 derivada de LMST conserva la conectividad de G. Para cualesquiera dos nodos $u, v \in V(G_0)$, se dice que el nodo u está conectado al nodo v (denotado como $u \Leftrightarrow v$) si y solo si existe un camino $(q_0 = u, q_1, \ldots, q_{m-1}, q_m = v)$ tal que $q_j \leftrightarrow q_{j+1}$, $j = 0, 1, \ldots, m-1$, donde $q_k \in V(G_0)$, $k = 1, 0, \ldots, m$.

Lema 4.3.1. *Para cualquier par de nodos $[u, v]$, $u, v \in V(G_0)$, si u es visible con v, entonces $u \Leftrightarrow v$*

Demostración. Para cualquier par de nodos $[u, v]$ que satisfacen que u es visible con v y $u, v \in V(G_0)$, ordenaremos los pesos $w(u, v)$ en orden ascendente, es decir, $w(u_1, v_1) < w(u_2, v_2) < \cdots < w(u_l, v_l)$. Probaremos por inducción en el rango de los pares de nodos en el orden.

I) Sea el par $[u_1, v_1]$ el primero en el orden. Se satisface que $w(u_1, v_1) = min_{(u,v) \in V(G_0)}\{w(u, v)\}$ y u_1 es visible con v_1. En consecuencia, $u \leftrightarrow v$, lo que significa $u \Leftrightarrow v$.

II) *Hipótesis de inducción* Asumimos que el lema 2 se mantiene para los pares $[u_i, v_i]$, $i = 1, 2, ..., k - 1$. Ahora probaremos que el lema 2 también se mantiene para el caso del par $[u_k, v_k]$.

III) *Tesis de Inducción:* Consideramos dos casos:

- *Caso 1:* $u_k \leftrightarrow v_k$, lo que implica $u_k \Leftrightarrow v_k$.
- *Caso 2:* Cualquiera $u_k \nrightarrow v_k$, $v_k \nrightarrow u_k$ o ambas.

Sin pérdida de generalidad, asumimos $u_k \nrightarrow v_k$. Ya que $v_k \in NV_{u_k}$, existe un único camino $p = (q_0 = u_k, q_1, q_2, \cdots, q_{m-1}, q_m = v_k)$ del nodo u_k al nodo v_k, donde $(q_i, q_{i+1}) \in E(T_{u_k})$, $i = 0, 1, ..., m - 1$. Ya que T_{u_k} es el único MST de G_{u_k}, tenemos que $w(q_i, q_{i+1}) < w(u_k, v_k)$. Aplicando la hipótesis de inducción a cada par $[q_i, q_{i+1}]$, $i = 0, 1, ..., m - 1$, tenemos que $q_i \Leftrightarrow q_{i+1}$, por lo tanto $u_k \Leftrightarrow v_k$.

□

Teorema 4.3.1 (Conectividad)**.** *G_0 conserva la conectividad de G, es decir, G_0 es conectada si G es conectada.*

Demostración. Supongamos que G es conectada. Para cualquiera dos nodos $u, v \in V(G)$, existe al menos un camino $p = (q_0 = u, q_1, q_2, \cdots, q_{m-1}, q_m = v)$ de u a v, donde $(q_i, q_{i+1}) \in E(G)$, $i = 0, 1, ..., m - 1$ y q_i es visible con q_{i+1}. Dado que $q_i \Leftrightarrow q_{i+1}$ por el lema número 2, tenemos que $u \Leftrightarrow v$ □

Teorema 4.3.2. *G_0^- conserva la conectividad de G, es decir, G_0^- es conectada si G es conectada.*

Demostración. Si el par de nodos $[u, v]$, $u, v \in V(G_0)$ satisface que u es visible con v, por el lema 2, existe un camino $p = (q_0 = u, q - 1, q_2, \cdots, q_{m-1}, q_m = v)$ tal que $q_j \leftrightarrow q_{j+1}$, $j = 0, 1, \cdots, m - 1$, donde $q_k \in V(G_0)$, $k = 0, 1, \cdots, m$. El mismo resultado se mantiene para G_0^- dado que todas las arista en p son bidireccionales y el quitar las aristas unidireccionales no afecta la existencia de dicho camino. Siguiendo el mismo argumento que se presentó en el teorema anterior, podemos probar que G_0^- mantiene la conectividad de G. □

4.4. Función *Macs*

Definición 4.4.1. *Dada una recta r sobre el plano $\mathcal{P}$, el conjunto $\mathcal{P} - r$ es la unión de dos conjuntos no vacíos y disjuntos entre sí, llamados semiplanos abiertos. Si a un semiplano le añadimos r, hablaremos de un semiplano cerrado.*

Definición 4.4.2. *Una cuerda maximal dentro de un polígono P es el segmento maximal totalmente contenido en P.*

Definición 4.4.3. *Decimos que una cuerda maximal es extremal si contiene una arista de P que incide en un vértice cóncavo.*

Sean $C_1, C_2, ..., C_m$ cuerdas maximales de P con $m \leq k$, donde k es el número de vértices cóncavos en P y tal que C_i pasa a través de uno o más vértices cóncavos de P. Se asocia a cada cuerda maximal C_i el semiplano cerrado C_i^+ definido por C_i.

Se clasificarán las cuerdas maximales en pivote-simple si contienen solo un vértice cóncavo, y pivote doble incluyen dos vértices cóncavos distinto.

4.4.1. El algoritmo

El siguiente paso después de que se ha aplicado la función Filtro es calcular un objetivo inmediato p_i. Este objetivo tiene que ser óptimo en el sentido de que el movimiento de s_i a p_i no desconecta la red y p_i esté lo mas cercano a t_i. Tomando en cuenta lo anterior, vamos a calcular una región R_i donde aseguremos que s_i puede moverse sin perder comunicación con sus vecinos, y tomaremos el punto $p_i \in R_i$ más cercano a t_i.

La función Macs es un algoritmo que dado un polígono simple no convexo P y que además es un polígono star-shaped, calcula una aproximación a la máxima área convexa dentro de P. Este algoritmo es presentado en [4].

Para los fines del Algoritmo 3 se agregaron unos pasos previos al algoritmo original de la función Macs.

La función Macs recibe por entrada un segmento ij que se encuentra al interior de un polígono simple P y regresa un subpolígono convexo P_j.

Definición 4.4.4. *Sea $N_i^\prime$ el conjunto de vecinos del agente s_i y sea $s_j \in N_i^\prime$. Sea el polígono P en donde se encuentran s_i y s_j. La función MACS($s_i s_j$) regresa un subconjunto $P_j \subseteq P$ tal que P_j es una región convexa y el segmento $s_i s_j$ está totalmente contenido en P_j.*

Como paso inicial de la función Macs se calcula el polígono de visibilidad completa $VisC(ij)$ del segmento ij como se sugiere por el Lema 1.3.4. Se sabe por definición que $VisC(ij)$ es un polígono star-shape y que el segmento ij se encuentra totalmente contenido en el núcleo de $VisC(ij)$.

Posteriormente al polígono $VisC(ij)$ se le calcula una aproximación al subconjunto de mayor área convexo.

Proposición 4.4.1. *El kernel de un polígono P está dado por la intersección entre P y los planos medios c_i^+ definidos por todas las cuerdas extremales c_i asociadas a todos los vértices cóncavos.*

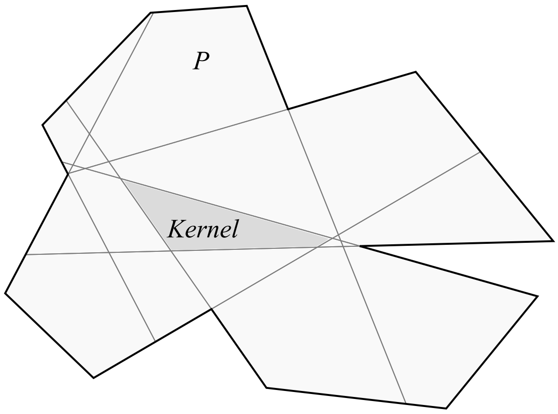


Figura 4.3: Kernel de un polígono

Teorema 4.4.1. *Sea P un polígono star-shape, entonces el kernel es un subconjunto de la máxima área convexa de P.*

Demostración. Sea c_i la cuerda óptima asociada al vértice cóncavo r_i de P. Se considera el espacio cerrado K definido por la intersección entre P y las dos cuerdas extremales de r_i. Si c_i es una cuerda extremal, es claro que $K \subseteq (c_i^+ \cap P)$. Si c_i es una cuerda-simple o una cuerda-doble balanceada, la pendiente de c_i es estrictamente delimitada por las pendientes de las dos cuerdas extremales. Además, ya que todos los planos medios tienen la misma orientación de acuerdo con P y r_i, también se tiene que $K \subseteq (c_i^+ \cap P)$. Finalmente, ya que las dos cuerdas extremales siempre definen un subconjunto de la cuerda óptima asociada, la intersección de todas las cuerdas extremales es un subconjunto de la máxima área convexa de P. Con la proposicion 4.4.1, el kernel de P es un subconjunto convexo de la máxima área convexa dentro del polígono. $\square$

En otras palabras, existe una deformación continua que transforma el kernel a la máxima área convexa dentro de un polígono. La estrategia que se utiliza para aproximar esta área es considerar que la deformación es como una dilatación euclidiana del kernel. Basados en esta heurística, muchas observaciones pueden ser hechas: los vértices deben ser tomados en el orden en el cual son alcanzados por el frente de onda de la dilatación. Más formalmente consideremos una lista $\mathcal{O}$ de vértices cóncavos tal que los puntos son ordenados de acuerdo a la distancia que existe del polígono al kernel. Cuando un vértice cóncavo es analizado, fijamos una posible cuerda e.

Las cuerdas pueden ser clasificadas como sigue:

- La cuerda puede ser extremal
- La cuerda puede ser pivote-simple, tal que su pendiente es tangente a el frente de onda
- La cuerda puede se pivote-doble. En este caso el segundo vértice cóncavo que pertenece a la cuerda es necesario. Debe de corresponder a el siguiente vértice cóncavo en la lista ordenada $\mathcal{O}$

Cuando un vértice cóncavo es analizado, se selecciona una cuerda de esta lista, de manera que maximice el área resultante del polígono. Si se denota por P' el polígono dado por la intersección entre P y el plano-medio asociado a la cuerda seleccionada, la cuerda debe maximizar el área de P'. En el algoritmo, esto equivale a minimizar el área de las partes removidas P/P'.

Algoritmo 4 MACS($s_i s_j$) - Cálculo del polígono convexo de máxima área

Calcular $VisC(s_i s_j)$
Calcular el Kernel de $VisC(s_i s_j)$
Calcular una lista ordenada de vértices cóncavos $\mathcal{O}$
Extraer el primer punto r_1 de $\mathcal{O}$
While $\mathcal{O}$ no sea vacia **Do**
  Extraer el siguiente vértice r_2 de $\mathcal{O}$
  Seleccionar la mejor cuerda que maximice el área del polígono resultante, tomando en cuenta (r_1, r_2)
  Modificar el polígono $VisC(s_i s_j)$
  Actualizar la lista $\mathcal{O}$ eliminando los vértices excluidos por la cuerda
  $r_1 \leftarrow r_2$
End While

El kernel de un polígono puede ser construido en $O(n)$. La complejidad de calcular las distancias entre los vértices cóncavos y el kernel de P para crear la lista ordenada $\mathcal{O}$ tiene complejidad $O(n + k \log k)$, donde k es el número de vértices cóncavos de P.

Algoritmo 5 Cálculo de la distancia al kernel

Encontrar la arista e_j del $Kern(P)$ más cercana al punto $v_0 \in P$
For i from 1 to n **Do**
  While $d(v_i, e_j) > d(v_i, e_{j+1(mod|kern(p)|)})$ **Do**
    $j := j + 1(mod|Kern(P)|)$
  End While
  Almacenar la distancia $d(v_i, e_j)$ a v_i
End For

4.5. Cálculo de R_i como la intersección de polígonos convexos

$$R_i \leftarrow \bigcap_{s_j \in N_i'} \textsc{Macs}(s_i s_j)$$

Por cada agente i se calcula un polígono convexo R_i resultado de la intersección de todos los polígonos convexos calculados por la función $\textsc{Macs}(s_i s_j)$, posteriormente se propone un punto p_i dentro de esta área, lo más cercano a t_i.

Existen varios algoritmos que calculan la intersección entre dos polígonos convexos, el mejor de ellos hace el cálculo en tiempo lineal. En el escenario que se ha descrito tenemos tantos polígonos convexos como agentes vecinos, y se quiere calcular la intersección entre todos ellos. Se puede notar que R_i nunca será vacío, aprovechando esta característica se ha desarrollado un algoritmo que calcule la intersección de n polígonos convexos, el cual se describirá a continuación.

4.5.1. Algoritmo para calcular la intersección entre n polígonos convexos

Se describe el algoritmo que a partir de un conjunto de polígonos convexos $\mathcal{P} = \{P_1, ..., P_n\}$ calcula el polígono $\mathcal{Q}$ tal que:

$$\mathcal{Q} \leftarrow \bigcap_{P_i \in \mathcal{P}} P_i, \ \textit{en donde } Q \neq \emptyset$$

El algoritmo trabaja en tres pasos.

I) Como primer paso se encuentran todos los puntos de intersección que existen entre las aristas del los polígonos en $\mathcal{P}$

II) En la segunda parte del algoritmo modificamos las propiedades de navegación de las aristas que intersectaron con al menos un punto.

III) Para finalizar se hace un recorrido entre las aristas de los polígonos hasta encontrar un ciclo entre las aristas, dicho ciclo es el límite del polígono $\mathcal{Q}$.

Asumimos que las aristas de los polígonos en $\mathcal{P}$ se encuentran en sentido contrario a las manecillas del reloj como es usual.

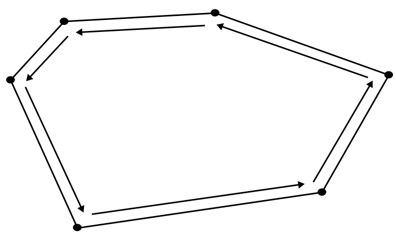


Figura 4.4: Dirección de las aristas de un polígono

Estructura de datos

Cada polígono convexo $P \in \mathcal{P}$ estará almacenado en un arreglo de datos. Este arreglo de datos está conformado por un conjunto de estructuras llamadas aristas; cada arista $\vec{e}$ tiene las siguientes propiedades: Un apuntador $Origen(\vec{e})$ hacia un vértice u, un apuntador $Destino(\vec{e})$ hacia un vértice v, de manera que el polígono quede del lado izquierdo de la arista, un apuntador $siguiente(\vec{e})$ hacia la siguiente arista del polígono P que se encuentra en sentido contrario a las manecillas del reloj y una propiedad $Padre(\vec{e})$ la cual es un identificador del polígono al que pertenece dicha arista. A su vez cada vértice v tiene una variable llamada $Coordenada(v)$ la cual almacena las coordenadas geométricas del vértice.

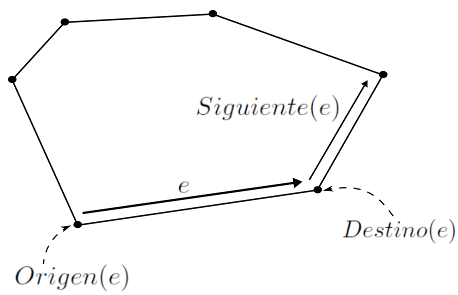


Figura 4.5: Estructura de un polígono

Calculando los puntos de intersección

Para encontrar los puntos de intersección entre las aristas de los polígonos en $\mathcal{P}$ se utiliza el algoritmo *FindIntersections(S)* que se presenta en [9], el cual recibe de entrada un conjunto de segmentos S (en este caso las aristas de los polígonos), y regresa un conjunto de puntos que a su vez contienen una lista de aristas que lo intersectaron.

Para el caso de los segmentos que se encuentran sobrepuestos, dicho algoritmo maneja como puntos de intersección los puntos iniciales o finales de los segmentos, según sea el caso. Las intersecciones entre pares de aristas pertenecientes al mismo polígono no son tomadas en cuenta.

Modificando las propiedades de navegación

En esta parte del algoritmo, se analiza cada uno de los puntos de intersección que se obtuvieron del paso anterior. Cada punto de intersección v cuenta con una lista de referencias a las aristas que lo intersectan; dicha lista de aristas se encuentra clasificada en tres subconjuntos diferentes:

i) Subconjunto $O(v)$, se encuentran todas las aristas e que cumplen que $Origen(\vec{e}) = v$.

ii) Subconjunto $D(v)$, son todas las aristas $\vec{e}$ que cumplen que $Destino(\vec{e}) = v$.

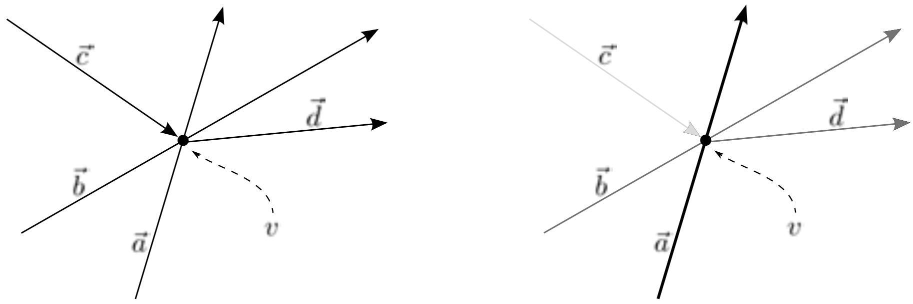
iii) Subconjunto $C(v)$, en este subconjunto se encuentran las aristas que contiene en su interior al punto v

Para el ejemplo que se muestra en la figura 4.6a los subconjuntos quedarían como sigue: $O(v) \leftarrow \{\vec{d}\}$, $D(v) \leftarrow \{\vec{c}\}$ y $C(v) \leftarrow \{\vec{a}, \vec{b}\}$

El siguiente paso es buscar la arista $\vec{a} \in (C(v) \cup O(v))$ cuyo vértice $Destino(\vec{a})$ se encuentre más a la izquierda que el de cualquier otra arista en el conjunto, es decir:

$$Destino(\vec{a}) \in H(\vec{x}), \forall \vec{x} \in (C(v) \cup O(v)), \vec{x} \neq \vec{a}$$

Donde $H(\vec{x})$ es el semiplano abierto a la izquierda de la arista dirigida $\vec{x}$. En caso de que existan aristas paralelas cuyos vértices finales cumplan la condición se tomará la arista cuya distancia entre v y su vértice destino sea la menor; si los vértices destino son iguales se toma una arista de manera arbitraria.



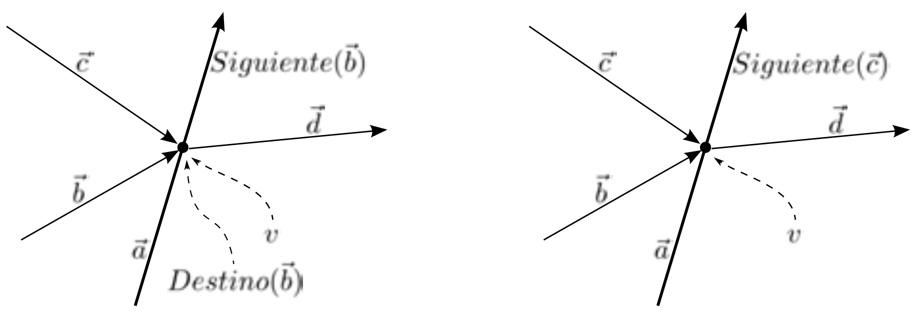
(a) (b)

Figura 4.6: (a) Ejemplo de un punto de intersección v con las respectivas aristas que lo intersectan. (b) La arista a cumple con la condición $Destino(\vec{a}) \in H(\vec{x}), \forall \vec{x} \in \{\vec{a}, \vec{b}, \vec{d}\}, \vec{x} \neq \vec{a}$.

Una vez que hemos encontrado a la arista $\vec{a}$ que cumple con las condiciones anteriores, entonces procedemos a modificar las propiedades de navegación de las aristas que se encuentran en $D(v) \cup C(v)$ de la siguiente manera:

- Para toda arista $\vec{e}$ hacer $Destino(\vec{e}) \leftarrow v$, tal que $\vec{e} \in C(v)$, $\vec{e} \neq \vec{a}$. Figura 4.7a.

- Para toda arista $\vec{e}$ hacer $Siguiente(\vec{e}) \leftarrow \vec{a}$, tal que $\vec{e} \in (D(v) \cup C(v))$, $\vec{e} \neq \vec{a}$. Figura 4.7.



(a) (b)

Figura 4.7: (a) Se modifica $Destino(\vec{b}) \leftarrow v$ y $Siguiente(\vec{b}) \leftarrow \vec{a}$. (b) Se modifica $Siguiente(\vec{c}) \leftarrow \vec{a}$.

Obteniendo $\mathcal{Q}$

Para finalizar en el algoritmo se hace un recorrido por las aristas de los polígonos, las cuales pasaron anteriormente por un proceso que modificó sus propiedades navegación, y lo cual nos permitirá hacer un recorrido que nos lleve a encontrar un ciclo, que es el límite de $\mathcal{Q}$

El algoritmo

En el algoritmo se creará una arreglo de datos I, el cual estará conformado por un conjunto de estructuras de tipo arista, como se describió anteriormente, las cuales cuentan con una propiedad más: un booleano $Visitada(\vec{e})$ que de inicio sera $false$.

Estimación de la complejidad

La primera parte del algoritmo que encuentra los puntos de intersección tiene una complejidad de $O((n + I)\log n)$, en donde n es el número total de aristas de todos los polígonos $P \in \mathcal{P}$, e I es el número total de intersecciones. En la segunda parte se analizan todas las intersecciones $O(I)$ y en el peor de los casos se recorren todas las arista para encontrar $\mathcal{Q}$, lo que resulta en $O(n)$.

4.6. Conectividad

En esta sección probaremos que las trayectorias rectilíneas generadas por el Algoritmo 3 mantienen la conectividad en todo momento.

Observe que, dado que R_i esta formada por la intersección de polígonos convexos que contienen a s_i, se sigue R_i es convexo y contiene a s_i, y por construcción la trayectoria rectilínea de s_i a cualquier punto $p \in R_i$ está totalmente contenida en R_i, por lo tanto la gráfica permanecerá conectada si el agente i se mueve de s_i a p y los demás agentes permanecen en su lugar. El siguiente teorema muestra una propiedad más fuerte.

Teorema 4.6.1. *Sean los agentes vecinos i y j, las trayectorias $\gamma_i \in R_i$ y $\gamma_j \in R_j$. Para cualesquiera puntos $q_i \in \gamma_i$ y $q_j \in \gamma_j$ se mantiene la adyacencia en la gráfica de configuración.*

Demostración. Sea P_j el resultado de la función Macs(ij), se sabe que P_j es un polígono convexo, por lo tanto cualquier par de puntos p_i y p_j que se encuentren contenidos en P_j son visibles entre sí, por lo tanto cualquier propuesta para los agentes i y j que se encuentre dentro de P_j conservará la adyacencia entre estos agentes. Siguiendo esta noción y sabiendo que γ_i y γ_j son trayectorias lineales contenidas en P_j podemos llegar a la conclusión de que cualesquiera dos puntos $q_i \in \gamma_i$ y $q_j \in \gamma_j$ están conectados, es decir, son visibles entre sí. $\square$

Algoritmo 6 Algoritmo para calcular la intersección entre n polígonos convexos

Entrada: Conjunto de polígonos $\mathcal{P} = \{P_1, ..., P_n\}$
Salida: Polígono convexo Q

$I \leftarrow \emptyset$
Almacenar las aristas de todos los polígonos $P \in \mathcal{P}$ en I
CALCULANDO LOS PUNTOS DE INTERSECCIÓN
FindIntersections(I)
MODIFICANDO LA NAVEGACIÓN
For each punto de intersección v encontrado **Do**
  Tomar alguna arista $\vec{q} \in (C(v) \cup O(v))$
  For each arista $\vec{e} \in (C(v) \cup O(v))$ $\vec{e} \neq \vec{q}$ **Do**
    If $Destino(\vec{e}) \in H(\vec{q})$ **o** $(Destino(\vec{q}) \notin H(\vec{e})$ **y** $dist(v, Destino(\vec{e})) < dist(v, Destino(\vec{q})))$ **Then**
      $\vec{q} \leftarrow \vec{e}$
    End If
  End For
  For each arista $\vec{e} \in C(v)$ $\vec{e} \neq \vec{q}$ **Do**
    $Destino(\vec{e}) \leftarrow v$
  End For
  For each arista $\vec{e} \in (D(v) \cup C(v))$ $\vec{e} \neq \vec{q}$ **Do**
    $Siguiente(\vec{e}) \leftarrow \vec{q}$
  End For
End For
OBTENIENDO A $\mathcal{Q}$
$\mathcal{Q} \leftarrow \emptyset$
Tomar una arista aleatorio $\vec{t}$ de I
While $Visit(\vec{t}) = false$ **Do**
  $Visit(\vec{t}) \leftarrow true$
  $\vec{t} \leftarrow Siguiente(\vec{t})$
End While
While $Visit(\vec{t}) = true$ **Do**
  $Visit(\vec{t}) \leftarrow false$
  $Origen(\vec{r}) \leftarrow Destino(\vec{t})$
  $\vec{t} \leftarrow Siguiente(\vec{t})$
  $Destino(\vec{r}) \leftarrow Destino(\vec{t})$
  Almacenar r en $\mathcal{Q}$
  If $\mathcal{Q} \neq \emptyset$ **Then**
    $Siguiente(\vec{a}) \leftarrow \vec{r}$
  End If
  $\vec{a} \leftarrow \vec{r}$
End While
$Siguiente(\vec{a}) \leftarrow \mathcal{Q}[0]$
Return $\mathcal{Q}$

4.7. Estimación de la complejidad

Sea el polígono P, E el número de aristas de P, V en número de vértices de P, K el número de vértices cóncavos de P, N el número de agentes en P y M el número de aristas de una gráfica completa de N nodos.

El cálculo del $SP(i,j)$ tiene complejidad de tiempo de $O(V \log V)$, la función Filtro tiene complejidad de $O(M + N \log N)$, calcular $VisC(ij)$ toma tiempo lineal lo que nos da una complejidad total de a lo más $O(NV)$. La función Macs(ij) tiene una complejidad de $O(V + K \log K)$ y sabemos que se calcula a lo más N veces. El cálculo de la intersección se puede hacer en tiempo $O((NE+I)\log NE)$.

$$O(V \log V)+O(M+N \log N)+O(NV)+O(NV+NK \log K)+O((NE+I)\log NE)$$

$$\begin{aligned}
&= V \log V + M + N \log N + 2NV + NK \log K + (NE+I)\log NE \\
& \text{Para un } A > NV, M, (NE+I). \\
&= A \log A + A + A \log A + A + A \log A + A \log A \\
&= A + A \log A
\end{aligned}$$

La complejidad del Algoritmo 3 es $O(A + A \log A)$.

4.8. Progreso

Para que el algoritmo sea útil, además de conservar la conectividad como se probó en la sección anterior, también debe garantizar que los agentes tienen un progreso en cada ronda y eventualmente alcanzan sus metas. Sin embargo se deben tomar algunas consideraciones en cuenta, que se explican a continuación.

Características de la gráfica de configuración

Si las metas son desconectadas, claramente no se puede obtener progreso sin violar la condición de conectividad, por lo que es necesario asumir que las metas se encuentran en una configuración conectada.

Ciclos

Supóngase una configuración en donde todos agentes se encuentran dentro de un ciclo posterior a haber aplicado la Función *Filtro* a la gráfica de configuración, dos agentes i y j desean moverse y romper el ciclo mientras que los agentes restantes desean conservar sus posiciones. Claramente ningún algoritmo local es capaz de reconocer si se encuentra dentro de un ciclo o una cadena, y en último caso cualquier movimiento de los agentes atentaría contra la conectividad.

Lo anterior provoca que i y j no tengan posibilidad alguna de realizar progreso hacia sus metas, por lo que se asumirá que no existen ciclos en la gráfica filtrada de configuración.

Dependencia de agentes

Sea el polígono P y sean los agentes vecinos p y q que son colineales a uno o más vértices de P, es fácil ver que q inmoviliza a p, es decir, p queda imposibilitado de realizar progreso sobre su trayectoria, si el $SP(s_p, t_p)$ se encuentra totalmente contenido en $P - VisC(pq)$).

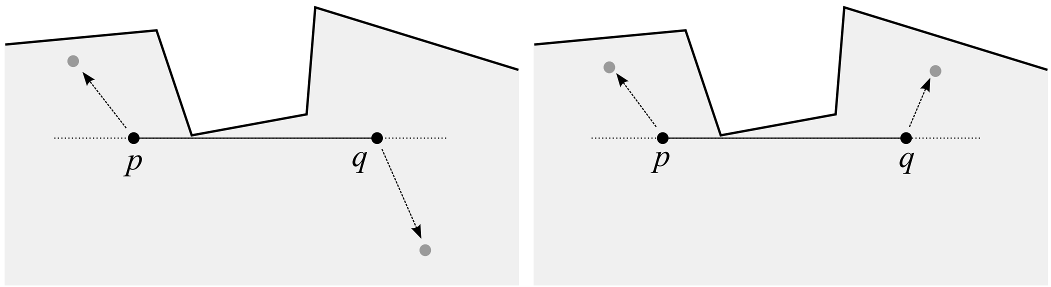


Figura 4.8: q inmoviliza a p

Si en la ejecución de una ronda del Algoritmo 3 un agente p, posterior a la Función *Filtro*, se queda con un vecino q que lo inmoviliza, p dependerá de que q cambie de posición y/o perder el enlace pq en la siguiente ejecución del algoritmo, para eventualmente realizar progreso en su trayectoria.

En las siguientes Figuras 4.9 y 4.10 se muestran ejemplos para los cuales el servicio de conectividad no garantiza el progreso de los agentes dado que todos se encuentran inmovilizados.

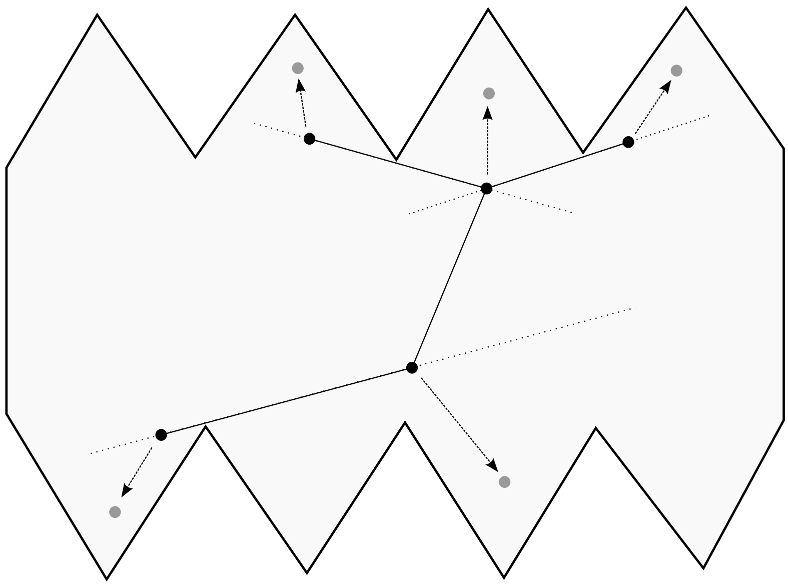


Figura 4.9: Configuración en la cual los agentes se encuentran imposibilitados de realizar progreso sobre sus trayectorias.

Una forma de eliminar estos casos es evitar que posterior a la Función *Filtro* los agentes conserven vecinos que los inhabilitan para realizar progreso como se

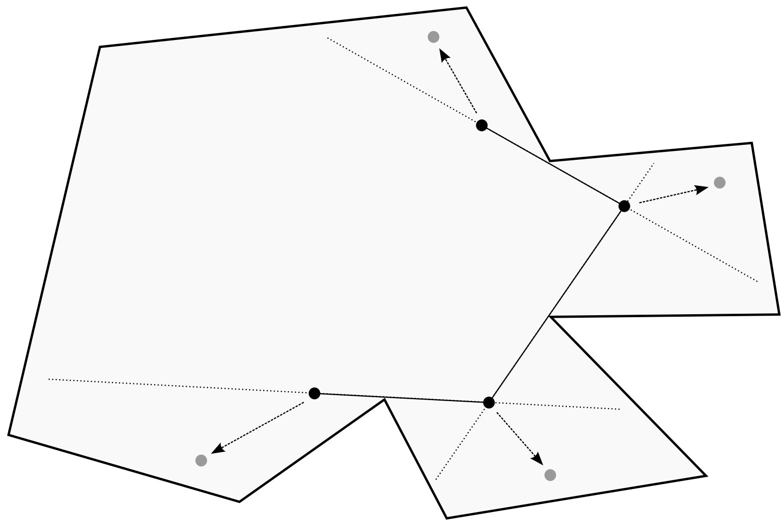


Figura 4.10: Configuración en la cual los agentes se encuentran imposibilitados de realizar progreso sobre sus trayectorias.

ya se mostró, para ello modificaremos la manera de asignar pesos a las aristas de la gráfica de configuración.

Sea el polígono P, y sean los agentes q y p dentro de P y que además son visibles entre sí, si q y p no están en posición general con los vértices de P entonces consideraremos $d(q,p) = \infty$, y la función peso se sigue aplicando con las mismas reglas ya establecidas. Dicha modificación no afecta la prueba de conectividad del algoritmo, y permite a los agentes de los ejemplos antes mostrados alcanzar sus objetivos.

Configuraciones degeneradas

En la Figura 4.13 se muestra una configuración para la que ningún algoritmo local puede garantizar el progreso, dado que cada agente es dependiente de otro y cualquier movimiento atentaría contra la conectividad.

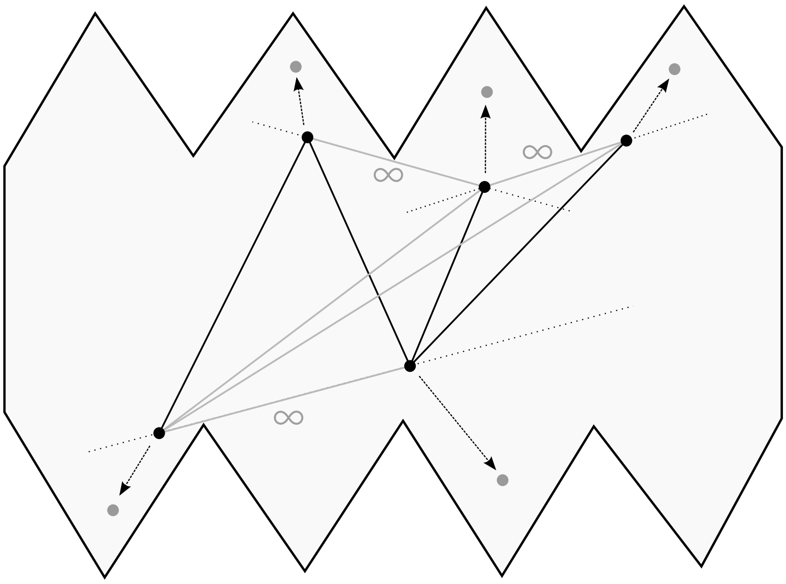


Figura 4.11: Ejemplo de la Figura 4.9 en donde se ha modificado la asignación de peso a las aristas de la gráfica de configuración.

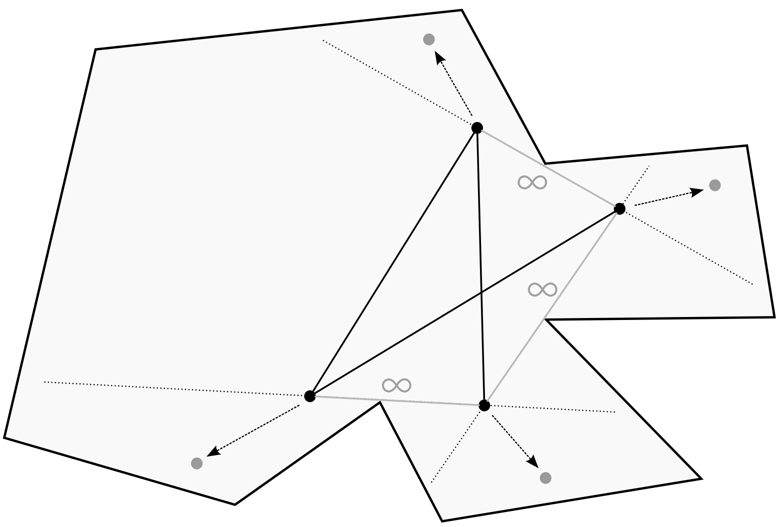


Figura 4.12: Ejemplo de la Figura 4.10 en donde se ha modificado la asignación de peso a las aristas de la gráfica de configuración.

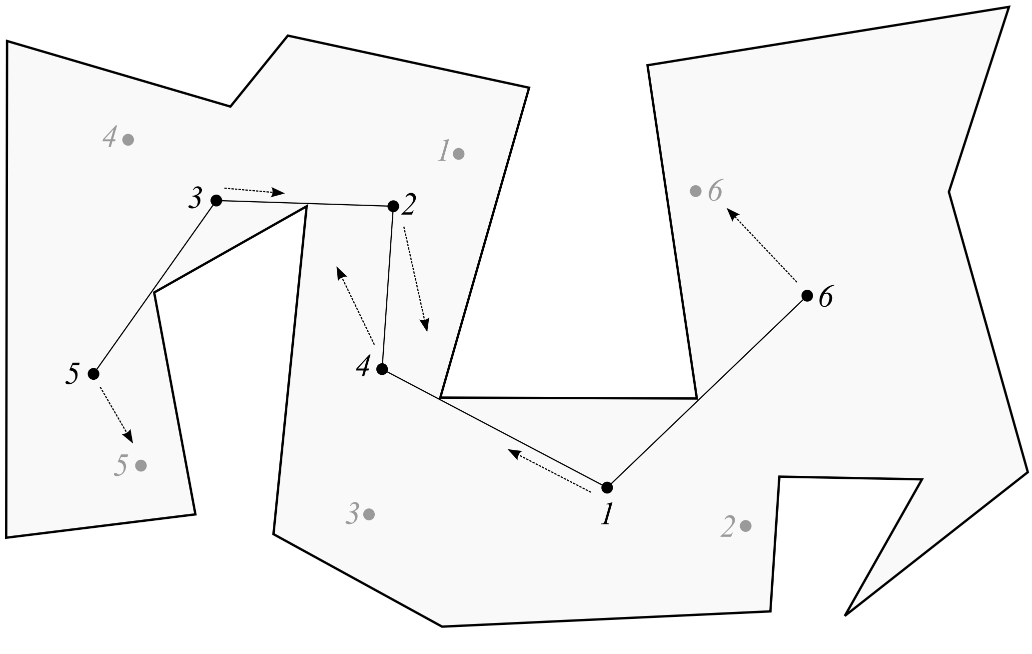


Figura 4.13: Configuración degenerada.

Conclusión

Está investigación se enfoca en determinar si es posible desplazar un enjambre de robots móviles dentro de una región poligonal manteniendo la conectividad global en la gráfica de visibilidad, de los agentes, en todo momento.

Se expone un algoritmo distribuido que en tiempo logarítmico planifica el movimiento de un enjambre de robots utilizando solo información local.

Se demostró que dicho algoritmo mantiene la conectividad global de la gráfica de visibilidad en todo momento. Sin embargo cuando se intentó provar que el algoritmo garantiza que todos los agentes llegaran a sus metas, nos encontramos con configuraciones para las cuales no se puede hacer dicha garantia sin atentar contra la conectividad, es decir, los agentes quedan en una posición en la donde cualquier movimiento que intenten realizar para alcanzar su objetivo los dejaría desconectados de la gráfica de visibilidad.

El resultado principal de está investigación ha sido probar que no existe algoritmo local que garantice conectividad y progreso al mismo tiempo.

Como trabajo futuro sería interesante poder caracterizar los polígonos y configuraciones de agentes en donde los robots no alcanzan sus objetivos y de esta manera acotar los escenarios para los que el algoritmo se ejecuta con éxito.

En lo personal considero que la carrera de Matemáticas Aplicadas y Cómputación da los conocimientos necesarios para abarcar y profundizar en este tipo de temas, y el Laboratorio de Algoritmos para la Robótica ha sido un gran acierto, permitiendo a sus estudiantes enfocar sus estudios en esta rama del conocimiento.

Bibliografía

[1] M. Barbeau and E. Kranakis. *Principles of Ad-hoc Networking*. Wiley, 2007.

[2] J.A. Bondy and U.S.R. Murty. *Graph Theory with Applications*. North Holland, 1976.

[3] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with guaranteed delivery in ad hoc wireless networks. In *Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*, DIALM '99, pages 48–55, New York, NY, USA, 1999. ACM.

[4] David Coeurjolly and Jean-Marc Chassery. Fast approximation of the maximum area convex subset for star-shaped polygons. Technical Report RR-LIRIS-2004-006, LIRIS UMR 5205 CNRS/INSA de Lyon/Université Claude Bernard Lyon 1/Université Lumière Lyon 2/École Centrale de Lyon, February 2004.

[5] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[6] Alejandro Cornejo. Research. http://people.csail.mit.edu/acornejo/Research, May 2013.

[7] Alejandro Cornejo, Fabian Kuhn, Ruy Ley-Wild, and Nancy Lynch. Keeping mobile robot swarms connected. In *Proceedings of the 23rd international conference on Distributed computing*, DISC'09, pages 496–511, Berlin, Heidelberg, 2009. Springer-Verlag.

[8] Alejandro Cornejo and Nancy Lynch. Connectivity service for mobile ad-hoc networks. In *Proceedings of the 2008 Second IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshops*, SASOW '08, pages 292–297, Washington, DC, USA, 2008. IEEE Computer Society.

[9] M. de Berg, O. Cheong, M. van Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer, 2008.

[10] R. Diestel. *Graph theory*. Graduate Texts in Mathematics Series. Springer London, Limited, 2005.

[11] S.K. Ghosh. *Visibility Algorithms in the Plane.* Cambridge University Press, 2007.

[12] B. Joe and R. B. Simpson. Corrections to lee's visibility polygon algorithm. *BIT*, 27(4):458–473, October 1987.

[13] D. T. Lee and F. P. Preparata. Location of a point in a planar subdivision and its applications. In *Proceedings of the eighth annual ACM symposium on Theory of computing*, STOC '76, pages 231–235, New York, NY, USA, 1976. ACM.

[14] Der-Tsai Lee and Franco P. Preparata. Euclidean shortest paths in the presence of rectilinear barriers. *Networks*, 14(3):393–415, 1984.

[15] N. Li, J.C. Hou, and L. Sha. Design and analysis of an mst-based topology control algorithm. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, volume 3, pages 1702 – 1712 vol.3, march-3 april 2003.

[16] F. P. Preparata M. R. Garey, D. S. Johnson and R. E. Tarjan. Triangulating a simple polygon. In *Information Processing Letters*, pages 7:175–179, 1978.

[17] J. O'Rourke. *Art Gallery Theorems and Algorithms.* International Series of Monographs on Computer Science, No 3. Oxford University Press, 1987.

[18] J. O'Rourke. *Computational Geometry in C.* Cambridge University Press, 1998.

[19] F.P. Preparata and M.I. Shamos. *Computational geometry: an introduction.* Texts and monographs in computer science. Springer-Verlag, 1988.

[20] D. Wagner and R. Wattenhofer. *Algorithms for Sensor and Ad Hoc Networks: Advanced Lectures.* LNCS sublibrary. SL 1, Theoretical computer science and general issues. Springer London, Limited, 2007.

[21] B.Y. Wu and K.M. Chao. *Spanning Trees and Optimization Problems.* Discrete Mathematics and Its Applications. Taylor & Francis, 2004.