UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA
DE LA COMPUTACIÓN

"ALGORITMOS PARA GRÁFICAS GEOMÉTRICAS CON AGENTES DE MEMORIA LIMITADA"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRO EN CIENCIAS
(COMPUTACIÓN)

P R E S E N T A

RAFAEL NORMAN SAUCEDO DELGADO

DIRECTOR DE TESIS: DR. JORGE URRUTIA GALICIA

México, D.F. Noviembre de 2005.

Índice general

Resumen

Las comunicaciones inalámbricas están transformando a los sistemas de cómputo en entornos altamente dinámicos que representan un reto para las ciencias de la computación. Actualmente uno de los más grandes retos es el diseño de redes *ad hoc* ó redes inalámbricas donde no existe una infraestructura fija para las comunicaciones. Algunos de los problemas que presentan estas redes son por un lado la movilidad de los elementos que no permite conocer de forma segura la forma de la red, y por el otro lado el hecho de que los dispositivos móviles ó portátiles cuentan con pocos recursos, tanto computacionales como de comunicación y energía.

Para estos sistemas existen dos tipos de soluciones: los métodos tradicionales adaptados a las circunstancias y las aproximaciones completamente nuevas. En este último destaca lo que en esta tesis se ha llamado la *propuesta geométrica*, cuya idea principal es obtener las posiciones de los elementos de la red en forma de coordenadas tipo (x, y), de manera que se puedan usar las relaciones geométricas entre vértices y aristas para resolver los problemas de la red. La propuesta geométrica ha demostrado ser eficiente y práctica por lo puede ser usada para aproximar soluciones a otros problemas dando lugar a lo que aquí se denomina *geometrización* de los problemas.

El objetivo de la tesis es continuar desarrollando resultados para este tipo de problemas en redes geométricas que modelan sistemas de comunicación inalámbrica *ad hoc*, mediante una colección de algoritmos diseñados bajo el paradigma de *agentes móviles*, lo que significa un menor tráfico en la red comparado con soluciones tipo cliente-servidor, además de otras ventajas. Estos algoritmos utilizan una cantidad de memoria constante con respecto al tamaño de la entrada, utilizan solo la información local disponible en cada punto, no alteran la gráfica y se pueden ejecutar en forma paralela, entre otras características que resultan muy importantes y de gran utilidad en sistemas de comunicación inalámbrica.

Como resultado se obtuvieron algoritmos que dan solución a los siguientes problemas: la simulación del barrido de línea, que es una técnica básica en geométrica computacional; el cálculo de la conexidad de la gráfica, lo que permite determinar los puntos vulnerables de la red y la fiabilidad de la misma; la coloración de la gráfica, fundamental para la asignación óptima de frecuencias; el árbol generador de peso mínimo, utilizado para reducir costos en las comunicaciones; ruteo, distancia máxima y centro de un árbol, todos estos para control de servicios en subestructuras de gráficas geométricas. En todos estos casos se resuelven problemas globales mediante el uso de información local, lo cual resulta muy interesante y útil sobre todo en cuestiones de escalabilidad.

Los resultados existentes y los que se alcanzaron durante el desarrollo de la tesis permiten asegurar que los sistemas inalámbricos tienen un gran futuro fundamentado en el apoyo de la tecnología a redes inalámbricas (Bluetooth), dispositivos de localización global (GPS, GALILEO, GLONASS) y dispositivos portátiles (teléfonos celulares, computadoras portátiles, etc). Además se puede prever la aplicación de la geometría a diversas áreas como parte de la optimización de recursos, la seguridad, la escalabilidad y el desempeño general de un sistema de cómputo.

Capítulo 1

Introducción

Actualmente los sistemas inalámbricos se han convertido en una realidad que invade nuestras vidas a una velocidad considerable. Las tecnologías inalámbricas plantean situaciones altamente dinámicas que plantean una serie de problemas interesantes y difíciles para las ciencias de la computación.

Los elementos de estos sistemas tienen libertad de movimiento y cuando esta libertad es utilizada de manera constante no es posible obtener información global acerca de la forma del sistema. En estos casos, la información en cada elemento es puramente local y acerca de los elementos que están cerca de él.

El ruteo en redes inalámbricas fue uno de los primeros problemas que se atacaron debido a su vital importancia para las redes de comunicación en general. Este problema sirvió de base para resolver otros problemas y plantear una nueva forma de *computación móvil.*

En los problemas que involucran movilidad solo se puede estar seguro de la información local que es particular a cada punto y muchos problemas son de naturaleza global, como por ejemplo encontrar la ruta entre cualesquiera dos puntos de la red.

A lo largo de la tesis se puede ver que es posible de manera práctica dar solución a problemas globales con información local, utilizando coordenadas geométricas (x, y) para los elementos de la red.

Otra de las características importantes del desarrollo de estos algoritmos es que las redes inalámbricas no solo plantean problemas por su topología altamente dinámica sino que también suelen ser entornos donde los recursos son escasos. Los participantes de las redes inalámbricas son dispositivos móviles ó portátiles que no cuentan con una fuente de energía constante. De igual manera un dispositivo portátil es mejor entre más pequeño sea y esto tiende a reducir las capacidades de cómputo.

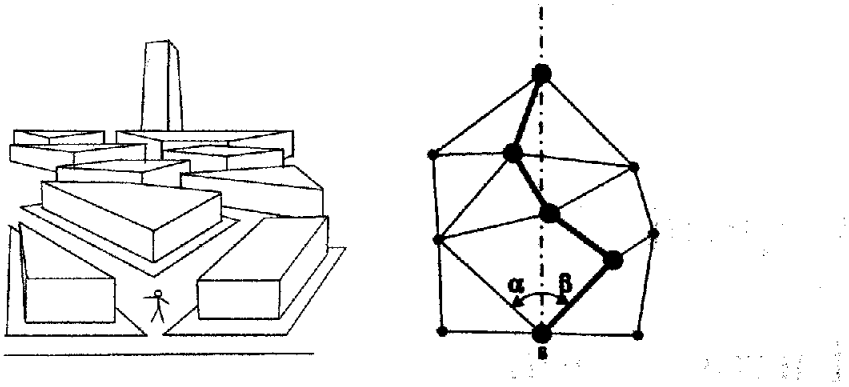

Figura 1.1: Vista de un turista en la ciudad de México y su gráfica correspondiente.

1.1. El modelo

Considérese a un turista paseando por primera vez en la Ciudad de México que desea visitar la Torre Latinoamericana, el edificio más grande del centro de la ciudad el cual se puede ver a gran distancia. El turista se encuentra en algún punto dentro de la ciudad desde donde se puede observar la torre, no tiene mapa y su lenguaje extranjero le impide obtener información como nombres de calles, avenidas e incluso preguntar por el camino.

Para poder llegar a su destino el turista no ejecutará un sofisticado esquema de ruteo ó cálculos complejos para encontrar su camino, tampoco hará un levantamiento topográfico y mucho menos dejará *marcas* en las calles para saber por donde ha pasado. La pregunta es ¿Podrá llegar a su destino? y ¿Que ruta tomará?. La mayoría de las personas al encontrarse en esta situación usarían el algoritmo 1 de forma intuitiva.

Algoritmo 1 Ruteo intuitivo

1: Mirar directamente a la Torre. (Ubicar la torre).
2: Observar las calles y avenidas que se pueden tomar en ese momento.
3: Tomar la avenida o calle cuyo ángulo con respecto a línea que se forma entre la torre y el observador sea mínimo.
4: Al llegar a la siguiente intersección regresar al paso 1 y terminar al llegar a la Torre.

La figura 1.1 muestra la vista general en algún punto del recorrido, la gráfica que modela la ciudad y el procedimiento a seguir. En cada paso el turista compara los ángulos hacia la derecha y hacia la izquierda, en la figura los ángulos α y β, y de los dos toma el más pequeño. A medida que se repiten los pasos en cada intersección se encuentra la ruta que lo lleva desde el punto de partida hasta el destino, sin usar un mapa con la información que le proporciona la vista, incluida la referencia de la torre, y sin dejar marcas.

Este problema puede ser fácilmente modelado con una gráfica geométrica donde las calles de la ciudad son las aristas y las intersecciones los vértices. De esta forma el problema de llegar a la torre se convierte en el problema de *ruteo* en redes geométricas, es decir el

problema de encontrar un camino por el cual mandar mensajes de un punto a otro sobre la gráfica geométrica.

En [12] se encuentra un ejemplo similar y allí mismo se plantean los algoritmos de ruteo que dieron origen a una revolución en cuanto a las soluciones de ruteo en diversos tipos de redes susceptibles a ser modeladas como redes geométricas, como las redes inalámbricas. El algoritmo 1 es la base de lo que se conoce como el algoritmo de ruteo por brújula, propuesto precisamente en [12].

Este ejemplo muestra claramente las condiciones que enmarcan y definen el área de estudio de esta tesis las cuales se enlistan a continuación:

1. En cada momento se conoce la posición (x, y) del vértice.
2. Estando en un vértice dado se conocen las posiciones de los vértices adyacentes a este.
3. La cantidad de memoria disponible es constante.
4. No se puede guardar información en los vértices.

Bajo estas condiciones se busca desarrollar algoritmos para recorrido, coloración, detección de conexidad, árboles generadores de peso mínimo, etc. Es importante observar que las restricciones se derivan directamente del tipo de tecnología disponible en los problemas actuales de comunicación.

1.2. Ejemplos

Esta sección presenta una serie de casos prácticos que surgen en varias áreas de la computación. Estos casos permiten entender la problemática que originó el estudio del tema de esta tesis y al mismo tiempo la formulación del modelo y las soluciones propuestas. Los ejemplos van desde aplicaciones militares como redes de sensores, hasta aplicaciones civiles como las redes de area personal y la robótica.

1.2.1. Exploración

Los robots son herramientas que tienen muchas ventajas debido a su diversidad de formas y materiales así como por su carácter no biológico que les permite realizar tareas que para un ser vivo serían imposibles. Desde robots miniatura dentro del cuerpo humano hasta los exploradores de Marte, los robots son los exploradores ideales, ver figura 1.2.

En los problemas de exploración se desconoce la geografía o topología del lugar, parcial o completamente. Para realizar su tarea el robot es equipado con sensores que le permiten obtener información de su entorno a medida que avanza en él. Si se trata de un problema con

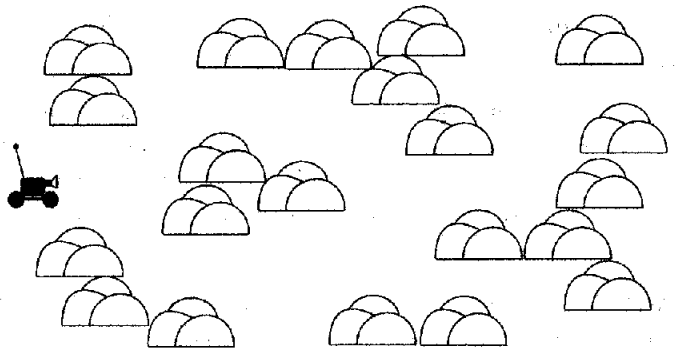

Figura 1.2: Robot explorador.

navegación autónoma los datos del entorno son procesados para poder tomar una decisión respecto a la ruta que debe de seguir sin la intervención humana.

Los robots autónomos tienen problemas adicionales como el abastecimiento de energía. En general tienen problemas debido a que necesitan hacer un uso eficiente de los recursos de los que disponen y que generalmente son reducidos. Algunos recursos importantes son la energía, la memoria, la comunicación y los sensores, un ejemplo concreto se encuentra en [22].

Retomando el problema de la exploración, una cualidad deseable para un robot explorador es la orientación, es decir que sea capaz de determinar su ubicación dentro de un entorno o marco de referencia para llevar a cabo sus tareas o simplemente para que no se pierda. Esta orientación la puede llevar a cabo por él mismo si es que el robot cuenta con un dispositivo de localización que le permite conocer sus coordenadas en algún sistema de referencia geométrico. En este caso el problema de exploración se puede modelar con gráficas geométricas. Cabe mencionar que la mayoría de los dispositivos de localización utilizan una referencia externa ya sea una antena o un satélite en el caso de localización global.

La información geométrica del lugar que se está explorando puede ser de mucha utilidad pues no solo le servirá para ubicarse y ubicar sus objetivos sino también para desplazarse adecuadamente por el entorno. De esta manera el uso de un dispositivo de posicionamiento le permite al robot explorar un entorno desconocido mediante una gráfica geométrica lo cual hace posible la aplicación de resultados en el área de la geométrica y las gráficas directamente a la robótica.

1.2.2. Redes *ad hoc*

Una red *ad hoc* es aquella red de dispositivos para los cuales no existe una infraestructura fija de comunicación entre ellos como podrían ser los cables ó estaciones y centrales retransmisoras. Una red *ad hoc* utiliza tecnologías de comunicación inalámbrica como ondas de radio y luz infra roja para comunicarse. Este tipo de redes son necesarias cuando la

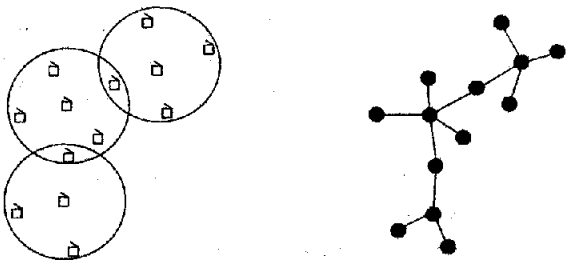

Figura 1.3: Tres piconet y la gráfica que las modela.

instalación y mantenimiento de la infraestructura son costosos o imposibles.

Debido a las ventajas de las redes *ad hoc* es muy probable que el futuro de las redes se encuentre en una combinación balanceada entre redes *ad-hoc* y redes estáticas clásicas. Esta tendencia está confirmada por la relación que actualmente existe entre teléfonos celulares y sistemas de cómputo, así como por el apoyo de tecnologías inalámbricas como Bluetooth [5].

Bluetooth es un estándar de comunicación por radio frecuencia (RF) de corto alcance que puede ser usado por equipos electrónicos como teléfonos celulares, computadoras portátiles, agendas de bolsillo e incluso aparatos electrodomésticos. Los aparatos que cuentan con esta tecnología solo pueden comunicarse entre sí cuando se encuentran a una distancia menor de 10 metros aproximadamente. Ciertamente es una distancia bastante corta pero en un principio el objetivo es trabajar con redes de área personal formadas por dispositivos propiedad de una sola persona y diseñados para realizar tareas pequeñas.

Sin embargo las posibilidades que ofrece este esquema de comunicación *ad hoc* van más allá de simples tareas. Cuando se dispone de una concentración de dispositivos lo suficientemente grande es posible establecer comunicación con dispositivos que se encuentran fuera del alcance de transmisión. Esto se logra de la misma manera en que una computadora conectada a Internet manda mensajes al otro lado del mundo sin que exista un cable o enlace satelital directo, es decir mediante *saltos* a computadoras intermedias. La idea es que los dispositivos intermedios retransmitan la señal hasta llegar al destino. La diferencia con las redes fijas es que un dispositivo con Bluetooth, u otra tecnología similar, no tendrá cables y el punto en que se conecta a la red de dispositivos puede cambiar constantemente sin problemas.

Una de las ventajas que esto supone es que si una tarea es demasiado grande para ser realizada por un solo dispositivo se puede dividir en tareas más pequeñas y enviarse a otros dispositivos para que las resuelvan y posteriormente unir las respuestas para dar una respuesta final. Este esquema de computación distribuida móvil es posible gracias a la comunicación que existe entre los dispositivos de la red *ad hoc*.

La movilidad y el dinamismo de las redes *ad hoc* es uno de los aspectos más interesantes y complicados porque además de no contar con una posición fija es posible que se integren y se separen de la red en cualquier momento.

Para empezar a entender este tipo de redes se utilizan modelos sencillos que en un principio suponen que al menos durante un instante breve de tiempo la red no cambia[35]. Uno de estos modelos de comunicación lo ofrece precisamente Bluetooth y se trata de agrupaciones de dispositivos llamadas *piconet*.

Todos los dispositivos dentro del rango de comunicación de un dispositivo forman una *piconet* y un dispositivo puede ser miembro de una o más piconets. Si es miembro de dos diferentes piconets servirá de enlace entre ellas y a través de él se podrán comunicar dispositivos que estén a distancias mayores de 10 metros. De esta forma el conjunto de piconets puede incluso tener acceso a Internet mediante estaciones o puntos de acceso especiales.

En general las redes *ad hoc* son un reto para todos los niveles de computación, desde la seguridad de los usuarios y la calidad del servicio hasta la distribución de frecuencias y uso óptimo de baterías. Uno de estos retos es encontrar protocolos de comunicación que permitan mandar mensajes entre cualesquiera dos dispositivos de la red e incluso a Internet. En la figura 1.3 se pueden observar tres piconets, su interacción y la gráfica que las modela.

Actualmente ya existen soluciones para la comunicación en estas redes pero la mayoría son extensiones de las soluciones para redes estáticas basadas en IP, el protocolo de Internet. La facilidad para rutear en redes geométricas ha puesto de manifiesto la posibilidad de incluir dispositivos de posicionamiento que les permitan a los participantes de la red conocer su ubicación geográfica. De esta manera la red se volvería una red geométrica con las ventajas que esto supone, ventajas que se derivan de algoritmos como los que se exponen en esta tesis. Algunas de las ventajas de estos algoritmos geométricos son el uso de memoria limitada, información local que por definición implica no usar tablas de ruteo que hagan referencia a cada parte de la red, así como la ventaja de que no se alteran o marcan los dispositivos.

1.2.3. Redes de sensores

Las redes de sensores inalámbricas (Wireless Sensor Networks - WSN) [21, 9], son redes *ad hoc* con la característica de que están compuestas por pequeños dispositivos electrónicos de bajo consumo de energía con la capacidad de sensar una o varias variables físicas, es decir que son capaces de convertir un atributo físico como la temperatura o las vibraciones en información clara para un sistema de cómputo.

Los sensores están diseñados con el más estricto sentido minimalista como lo demuestra el famoso proyecto SmartDust [21]. Debido a estas restricciones solo pueden comunicarse a distancias muy cortas y son diseños muy específicos usados generalmente en el monitoreo del clima, investigaciones sobre terrenos, simulaciones e incluso para el monitoreo de actividad en campos enemigos en aplicaciones militares.

Para darse una idea de las severas restricciones el proyecto MANTIS [19] cuenta con alrededor de 500 bytes de memoria para poder realizar todas sus actividades entre las que destacan la ejecución de múltiples procesos, la administración de los datos de entrada y salida y las comunicaciones inalámbricas cuyos paquetes de información son de solo 64 bytes.

Otra diferencia sustancial es el hecho de que una vez colocados los sensores su posición

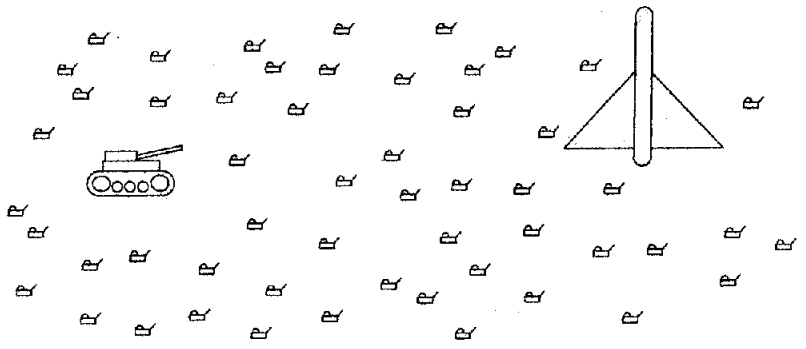

Figura 1.4: Red de sensores para uso militar.

cambiará en muy pocas ocasiones, es decir que se pueden considerar estáticos aunque la topología es espontánea pues no hay manera de saber como quedarán esparcidos.

Un ejemplo es un avión militar que vuela a baja altura y esparce varios cientos de sensores sobre campo enemigo como en la figura 1.4. En este situación, la forma en que quedarán acomodados los sensores es completamente aleatoria pero una vez establecida no cambiará mucho.

Un posible uso para esta red de sensores es registrar el movimiento del enemigo y reportar la actividad a un punto clave determinado. Una operación típica de esta red sería determinar cuales son los sensores cuya actividad es mayor a cierto valor y reportarlos apropiadamente. Para esto se emitiría una petición a todos los sensores, se recolectarían las respuestas y serían enviadas hacia algún destino para ser procesadas. De esta forma sí un tanque enemigo pasa por esta zona sería detectado por los sensores y no solo por donde está pasando sino también su velocidad, aceleración, dirección y otras cosas que pueden resultar ventajosas en una situación como esa.

Este tipo de operaciones requieren ser implementadas considerando las limitaciones técnicas de los sensores como por ejemplo la comunicación, la energía, la memoria, etc. Debido a las características de los algoritmos geométricos, los diseños de redes de sensores [19, 15] ya incluyen un dispositivo de posicionamiento global del sistema GPS como parte del sensor. Esto les proporciona las coordenadas geométricas con respecto a un marco de referencia estándar y da pauta a la aplicación de algoritmos geométricos. El diseño de algoritmos para redes de sensores tiene un énfasis especial en la memoria limitada y en la localidad de los datos, lo cual una vez logrado puede ser aplicado en redes con menos restricciones para optimizar recursos.

1.2.4. Comunicación en mallas

En la computación distribuida y paralela las tareas o procesos pesados (computacionalmente hablando) se dividen para ser atendidos por separado en varios procesadores o

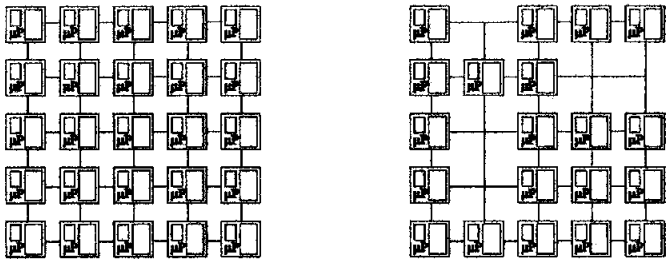

Figura 1.5: Malla de procesadores.

dispositivos de cómputo. Por ejemplo, los sistemas multi-procesador dividen la tarea entre ellos resolviendo tareas más pequeñas que posteriormente se integrarán para obtener la solución final.

Los procesadores requieren *ponerse* de acuerdo para dividir y sincronizar tareas y entregar los resultados; esta comunicación se realiza mediante una red que, idealmente, estaría integrada por enlaces directos entre cada uno de los procesadores, sin embargo esta solución es costosa tanto por su elaboración como por su mantenimiento. Entonces el problema es encontrar la mejor manera de formar la red, problema ampliamente estudiado en el área de cómputo distribuido. La red que se use depende en gran medida de las características que se quieran obtener, por ejemplo las mallas[16] son un esquema de comunicación altamente escalable y sencillo de implementar

La figura 1.5 del lado izquierdo muestra una malla de procesadores donde cada uno tiene a lo más cuatro conexiones de tal manera que para comunicarse con otro procesador lo hará pidiéndole a uno de estos cuatro vecinos que le pase el mensaje a algún vecino y este a su vez a otro hasta llegar a su destino. Además de tener un número constante de conexiones, sin importar el tamaño de la malla, se cuenta con un esquema de comunicación muy sencillo ilustrado por el algoritmo 2.

Algoritmo 2 Comunicación en mallas

1: Mandar el mensaje al vecino cuya columna se acerque más a la columna del destino.
2: Repetir el paso 1 hasta llegar a la columna destino.
3: Mandar el mensaje al vecino cuya fila se acerque más a la fila del destino.
4: Repetir el paso 3 hasta llegar a la fila destino.
5: Terminar.

La posición de los procesadores se conoce de antemano dado que son sistemas que no cambian su topología, por lo que se les asigna fila y columna en un marco de posicionamiento local, relativo e invariante.

Hasta este punto la solución no puede ser más sencilla pero desafortunadamente los sistemas están propensos a fallas y si algún procesador deja de funcionar o está demasiado ocupado el mensaje puede perderse en el camino, ver la figura 1.5 del lado derecho. Estas fallas no son permanentes y a medida que se restablecen la topología regresa a su estado

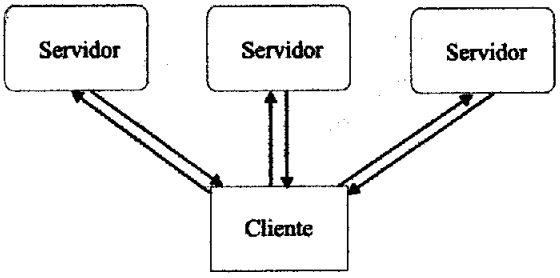

Figura 1.6: Modelo cliente/servidor.

normal, sin embargo estos ciclos son aleatorios y por lo tanto es necesario robustecer el esquema de comunicación haciéndolo tolerante a fallas en enlaces y microprocesadores.

Para diseñar un esquema de fallas en este contexto, es importante recordar la característica de localidad porque estando en un procesador determinado solo se puede estar seguro del estado de los procesadores vecinos, es decir, que solo se cuenta con información local. También se necesita que el esquema sea sencillo para no consumir tiempo y memoria del procesador que están destinados a otras funciones, esta característica lo hará también escalable.

En este caso no es necesario el uso de dispositivos de posicionamiento para convertirla en una red geométrica porque la topología de la red ofrece un marco de referencia de tipo local y relativo que permite la aplicación de algoritmos geométricos sin la necesidad de incluir dispositivos extra. Es decir que la geométrica de la topología de la red y su carácter altamente estático permiten que se puedan aplicar las soluciones planteadas en esta tesis a problemas en el área del cómputo de alto rendimiento.

1.2.5. Modelo de agentes móviles

En la vida real un *agente* es una persona que representa a otra persona y realiza tareas en favor de ella, por ejemplo el agente de viajes revisa los horarios de vuelo y paquetes que más convengan y realiza las reservaciones adecuadas. Un *agente de software* es un programa que realiza tareas en favor de un usuario o tal vez de otro programa. Un *agente de software móvil* es aquel que opera en una red de dispositivos y es capaz de migrar autónomamente de un dispositivo a otro. Las tareas que estos agentes móviles pueden realizar van desde la compra de productos en Internet hasta el cómputo científico distribuido.

Los agentes móviles son un paradigma de computación y comunicación que surge como respuesta a los problemas actuales a los que los paradigmas previos no pueden dar solución de forma satisfactoria. Algunos de estos esquemas previos son la llamada de procedimiento remoto o RPC (Remote Procedure Call) [4] y la evaluación remota o REV [26], que están basados en la arquitectura popular conocida como *cliente servidor* ver figura 1.6.

En RPC los dispositivos mandan peticiones a otros dispositivos dentro de la red para

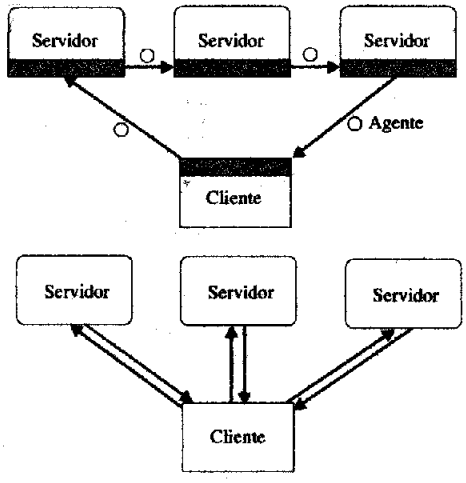

Figura 1.7: Modelo de agente móvil.

que ejecuten un procedimiento de forma remota y posteriormente les envíen de regreso el resultado, mientras que en REV se envía el procedimiento que se desea ejecutar en el otro dispositivo, se ejecuta y se manda la respuesta. En ambos casos cuando se requiere de los servicios de dos dispositivos distintos es necesario establecer comunicación con cada uno de ellos por separado.

El modelo de agentes móviles propone mandar el código, los datos y el contexto para que se dispersen en la red en forma de agente. Este agente se desplaza por la red para realizar su trabajo y después regresa con los resultados, todo esto sin que el usuario intervenga en el proceso de movilidad del agente. En la figura 1.7 se observa la migración autónoma del agente entre los dispositivos servidores lo cual permite que el número de enlaces de comunicación se reduzca con respecto al modelo cliente servidor.

Sin embargo el agente requiere de un *ambiente* en el cual se pueda desplazar, en este caso la figura 1.7 lo muestra de color gris. Emerald [14] es un sistema que ya presenta estas características y Telescript [24] fue el primer sistema diseñado explícitamente para trabajar con agentes móviles siendo pionero para este tipo de entornos o ambientes. Actualmente el lenguaje de programación java [25] está teniendo gran demanda para la implementación de estos ambientes a nivel comercial.

Las ventajas de este tipo de entornos con respecto a los anteriores fueron analizadas en [6] de donde se deriva la siguiente lista:

- Se reduce el tránsito en la red.
- Favorece el asincronismo.
- Actualizaciones dinámicas.
- Concurrencia.

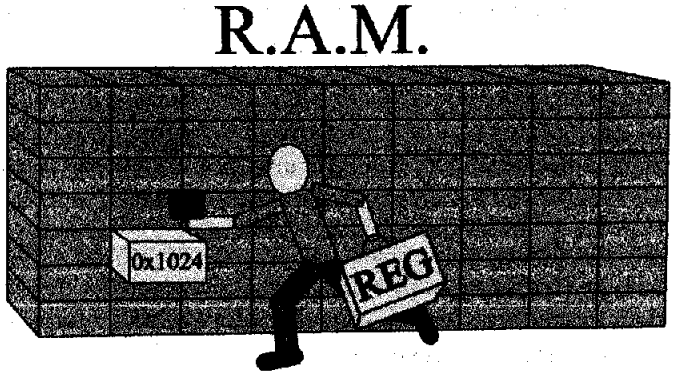

Figura 1.8: Un microprocesador visto como agente móvil, su entorno es la memoria R.A.M..

Una de las desventajas más notables de los agentes móviles es la vulnerabilidad a ataques, por lo que la seguridad es el tópico principal cuando se habla de agentes móviles. Otro problema es la variedad de dispositivos en los que se desplaza el agente algunos de los cuales no tienen una gran capacidad de cómputo por lo que es conviene que las aplicaciones diseñadas para este paradigma utilicen pocos recursos. En general se requiere de un agente *ligero* que optimice los recursos que lleva consigo, que no haga uso excesivo de los recursos del sistema que visita y que sea seguro en el sentido de no violar la integridad de los datos o comprometer al usuario del agente.

Con el bajo costo de los dispositivos de posicionamiento global es posible dotar a las computadoras de su posición y llevar a los agentes hacia un modelo de *agentes móviles geométricos* en el cual sea posible aplicar algoritmos geométricos de memoria limitada para resolver problemas de memoria e incluso de seguridad.

1.2.6. Microprocesador

Este último ejemplo es el más abstracto y tal vez el más interesante por estar relacionado con el principal sistema de cómputo, el microprocesador. A pesar de que existen distintas arquitecturas de procesamiento existen elementos comunes a todas ellas como por ejemplo la *pirámide de memoria.*

La *pirámide de memoria* es una representación gráfica de la cantidad y la rapidez de la memoria disponible para los sistemas de cómputo y obedece a la regla que dice que a medida que se está más cerca del procesador la cantidad de memoria es menor y de acceso más rápido y al estar más lejos del procesador la cantidad de memoria es mayor pero de acceso cada vez más lento. Aunque la tecnología se encargará de estrechar esta brecha este no es el tipo de soluciones que interesan en las ciencias de la computación.

En la pirámide de memoria los registros del microprocesador son los más rápidos y su cantidad generalmente no va más allá del orden de los kilobytes. La cantidad de registros es

constante independientemente del sistema y, a pesar de que es posible contar con módulos de memoria adicionales, los registros son suficientes para realizar todas las tareas.

La memoria principal de un sistema de cómputo es la memoria de acceso aleatorio o R.A.M. (Random Access Memory) y constituye la principal región de trabajo para el microprocesador. Esta memoria se puede expandir muy por encima de la cantidad de memoria de los registros hasta el orden de los megabytes incluso gigabytes pero no deja de ser constante.

Analizando una de las operaciones básicas del procesador como por ejemplo la suma, se observa una serie de pasos que implican el desplazamiento de la información, desde que los datos se encuentran fuera del sistema y son colocados en la memoria, hasta que el procesador los *adquiere*, procesa y entrega el resultado de la suma. Estos pasos se listan a continuación:

1. Ir por el primer dato y ponerlo en el registro A.
2. Ir por el segundo dato y ponerlo en el registro B.
3. Realizar la operación $SUMA$ y poner el resultado en el registro C.
4. Dejar el resultado del registro C en la memoria.

Si al término de la operación el procesador debe restar un valor al resultado anterior, ¡este no será capaz de *recordar* el resultado anterior! y por lo tanto irá de nuevo a la memoria para obtenerlo o lo calculará otra vez.

El procesador constituye un *agente* con memoria constante que se desplaza en un gran espacio de memoria para realizar sus tareas bajo la dirección de los programas de cómputo. Como la memoria está organizada en forma de malla el procesador se encuentra en un sistema de referencia local como en el caso de las mallas de microprocesadores y por lo tanto puede ser conceptualizado como un problema geométrico. La figura 1.8 muestra el concepto gráfico del agente microprocesador en el entorno de la memoria.

En este caso el procesador es capaz de modificar su entorno (la memoria) pero la semejanza con los ejemplos anteriores es indudable y es un claro ejemplo de que con memoria constante e información local se pueden resolver problemas que involucran a todo un sistema.

1.3. Validez del modelo

Los sistemas que se expusieron en los ejemplos anteriores contienen limitaciones serias, pero totalmente reales, que se están presentando con mayor frecuencia hoy en día. Ante esta situación no es posible esperar a que las limitaciones sean cubiertas por desarrollos tecnológicos que permitan usar más memoria o mejores rangos de comunicación, por ejemplo. Lo que se busca son algoritmos que utilicen los recursos e información disponibles de forma óptima para dar solución a los problemas en condiciones limitadas.

El modelo que se propone en la tesis se basa en el uso de información geométrica para cubrir las limitaciones de los sistemas, dando lugar a soluciones de problemas globales con

información local y muy poca memoria. En principio las coordenadas geométricas pueden obtenerse de varias formas, pero destacan las tecnologías de posicionamiento global como por ejemplo el sistema G.P.S.

Actualmente son muchas las disciplinas que se están sumando a la investigación en el área de soluciones geométricas en redes de comunicación. Esta situación se debe a que el modelo resulta práctico por la sencillez y naturalidad de los conceptos así como también por la viabilidad técnica que ofrecen los sistemas de posicionamiento global actuales.

1.4.. Motivación

Cuando se incluye información geométrica en un problema es posible obtener soluciones alternativas que pueden resultar en una mejor solución. También en algunos casos, diferentes problemas se pueden ver sintetizados mediante el uso de información geométrica. Este hecho no resulta tan extraño si se considera que las gráficas han sido un instrumento de abstracción muy poderoso que tiene esta finalidad. Además, el entorno físico de cualquier aplicación se puede modelar mediante espacios geométricos y por ello la geometría contribuye de manera tan natural a la solución de los problemas.

Todos los problemas que tengan como estructura de fondo a una gráfica geométrica serán susceptibles de ser estudiados bajo las condiciones del modelo que aquí se presenta y cuyas características principales son la localidad en la información y las restricciones en los recursos.

Particularmente lo que llama la atención es que las soluciones geométricas resultan mejores comparadas con las tradicionales, sin tomar en cuenta que solo usan información local. Todo esto favorece una solución descentralizada y robusta contra fallas además del uso óptimo de la memoria.

Aparte del interés científico por estos modelos, existe el interés por la aplicación práctica motivado por la tendencia de la tecnología móvil inalámbrica. Muy pronto las tecnologías inalámbricas como *Bluetooth* se integrarán en la vida cotidiana marcando el futuro de las redes. Por otro lado las redes de sensores han llamado tanto la atención de investigadores, académicos y militares, que ya han destinado varios millones de dólares a la investigación de estas redes.

De esta manera sobran las motivaciones para dedicarse al estudio de este tema tan fascinante y apasionante. Su aplicación en las nuevas tecnologías está garantizada y esto pone a México y a la UNAM en competencia real con el resto del mundo, de manera que la aportación de esta tesis a futuros desarrollos es otro motivo importante de la misma.

Capítulo 2

Preliminares

En este capítulo se presentan conceptos, definiciones y modelos matemáticos que servirán de base pasa seguir adecuadamente el desarrollo de los algoritmos y las ideas detrás de ellos, en los capítulos posteriores.

Primeramente se presentan conceptos clásicos de teoría de gráficas así como de gráficas geométricas para entender el entorno donde se desenvuelven los algoritmos, sus propiedades y la variedad de opciones que tienen. Posteriormente se introduce una clasificación de algoritmos basada en la cantidad de memoria que utilizan para resolver un problema y en la forma en que están distribuidos los datos de entrada, esto con el objetivo de ubicar mejor el tipo de soluciones que se están proponiendo. Por último se expone el *ruteo compacto*, el cual se puede decir que es un antecedente directo del *ruteo geométrico* en general. El ruteo compacto es una muestra de como los problemas globales pueden ser resueltos localmente con cantidades de memoria mínimas, siendo este uno de los punto claves de toda la tesis.

2.1. Gráficas geométricas

En 1736 el trabajo del célebre matemático Leonhard Euler [29] resultó en una de las abstracciones más utilizadas en las ciencias exactas : *las gráficas*. Las gráficas son modelos matemáticos que permiten ver los problemas de manera gráfica o visual representando sus componentes y la relación entre ellos mediante círculos y líneas. La figura 2.1 muestra la abstracción de los puentes de Kônigsberg, usados en el famoso problema propuesto por Euler [29] donde los pueblos son modelados mediante círculos negros y los puentes que los unen mediante líneas, los círculos son los *vértices* y las líneas son las *aristas* que en conjunto forman una *gráfica*.

Definición 1 *Una gráfica está compuesta por dos conjuntos, uno de ellos no vacío y el otro de pares no ordenados de elementos del primero. Una gráfica G se denota por $G = (V, E)$, donde V es el conjunto no vacío llamado conjunto de vértices y E es el conjunto de aristas o arcos tal que:*

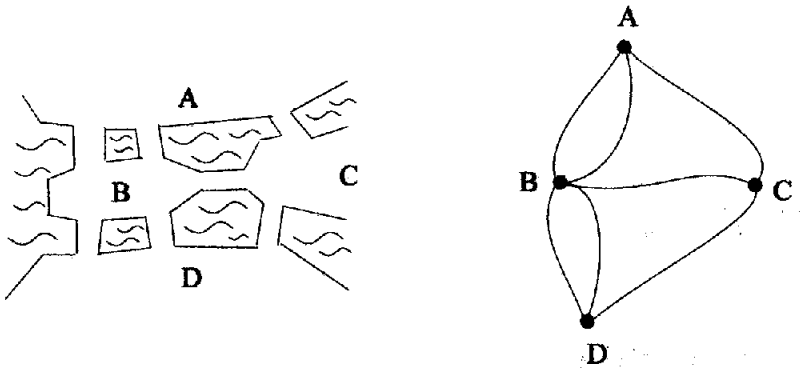

Figura 2.1: Los siete puentes de Kônigsberg y la gráfica que los modela.

- $E \subseteq \{\{u, v\} | u, v \in V\}$.

Las gráficas por sí mismas son un concepto muy útil pero algunos problemas necesitan una realización más práctica de una gráfica, la cual se obtiene mediante una *inmersión en el plano*. La idea es llevar la gráfica a un *dibujo* en el espacio R^2.

Definición 2 *La inmersión de una gráfica $G = (V, E)$ es un mapeo Γ que asigna a cada vértice de una gráfica un punto en R^2, de manera que las aristas se convierten en segmentos de recta entre estos puntos.*

$$\Gamma : V \mapsto R^2$$

Definición 3 *Una gráfica geométrica es una gráfica cuyo conjunto de vértices es $V \subset R^n$ y las aristas son segmentos de recta que tienen como puntos finales a los vértices que ellas relacionan.*

Una inmersión define una gráfica geométrica en R^2 a partir de una gráfica. Las gráficas proporcionan información acerca de la relación (aristas) que existe entre los elementos representados (vértices) pero las gráficas geométricas cuentan además con la información acerca de la posición o ubicación de los elementos dentro del sistema de referencia R^2.

Las gráficas geométricas son modelos matemáticos muy intuitivos que se usan cuando existe la necesidad de trabajar con este tipo de información como por ejemplo en mapas, figuras en tercera dimensión, planos, diagramas, etc. En esta tesis se exponen problemas para los cuales no es necesario usar la información geométrica pero al utilizarla el problema se resuelve de mejor manera.

2.1.1. Gráficas geométricas planas

Dentro de las gráficas geométricas existe una familia que tiene un interés especial para muchas aplicaciones debido a su sencillez relativa con respecto a las gráficas geométricas en

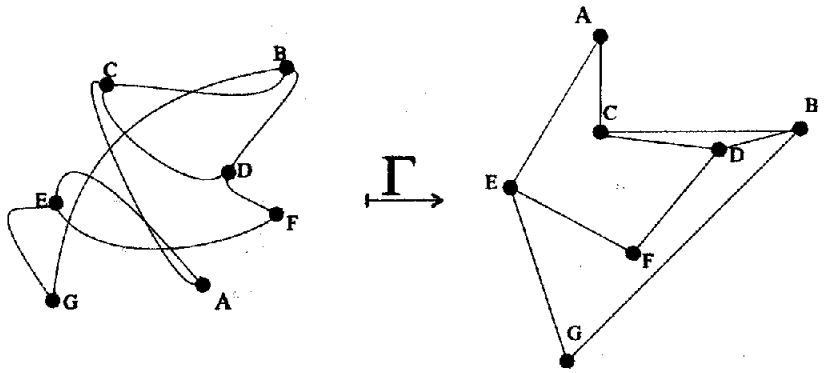

Figura 2.2: Inmersión plana de una gráfica.

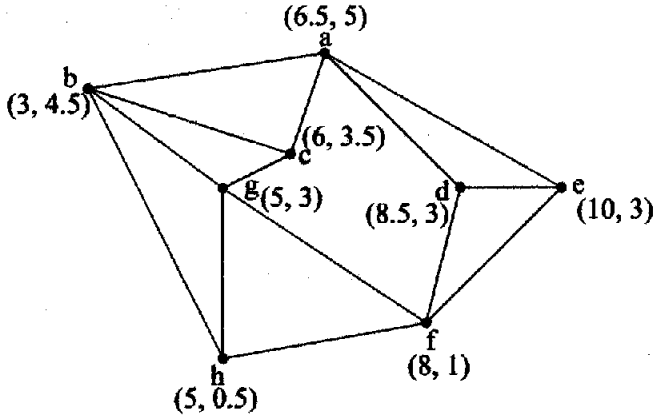

Figura 2.3: Gráfica geométrica plana.

general; estas gráficas se conocen como gráficas geométricas *planas*.

Definición 4 *Una gráfica geométrica plana $G(V, E)$ es aquella gráfica geométrica donde los segmentos de recta que representan las aristas no se intersectan a menos que las aristas tengan vértices en común.*

Cuando una inmersión crea una gráfica geométrica plana se dice que es una *inmersión plana*, ver figura 2.2. Es claro que existen gráficas para las cuales no existe inmersión plana e inmersiones no planas de gráficas para las cuales si existe una inmersión plana.

La idea de *planaridad* tiene que ver con el hecho de que algunas gráficas no se pueden dibujar en un plano sin que las aristas se crucen. Las gráficas *planas* son menos complejas de las gráficas que no lo son y por esta razón la planaridad es una de las propiedades más deseadas para una gráfica cuando se resuelve o modela un problema. Las gráficas planas no

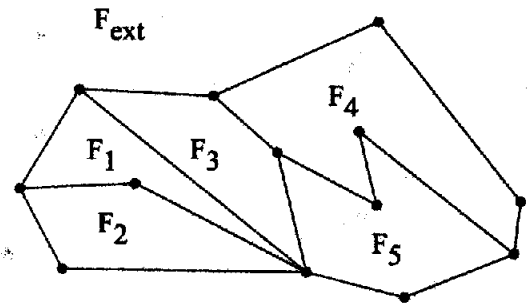

Figura 2.4: Caras inducidas por una gráfica geométrica plana.

solo son sencillas sino que además se apegan a muchos problemas reales, son muy útiles y tienen propiedades bien estudiadas. La figura 2.3 muestra un ejemplo de una gráfica plana $G = (V, E)$ con $V = \{a, b, c, d, e, f, g\}$ y $E = \{\{a, b\}, \{a, c\}, \{a, d\}, \{a, e\}, \{b, c\}, \{b, g\}, \{b, h\}, \{c, g\}, \{d, e\}, \{d, f\}, \{e, f\}, \{f, g\}, \{f, h\}, \{g, h\}\}$.

De aquí en adelante cuando se refiera a una *gráfica* debe entenderse que se trata de una *gráfica geométrica plana* a menos que se mencione otra cosa.

Definición 5 *Una gráfica geométrica plana $G(V, E)$ divide el plano R^2 en un conjunto de regiones conocidas como* caras, *incluyendo la cara externa, las cuales están delimitadas por los vértices y aristas de G.*

A la división del plano en caras por parte de la inmersión de una gráfica se le conoce como *subdivisión* ó *partición*. La figura 2.4 muestra las caras de una subdivisión incluyendo la cara externa.

Las caras de una gráfica son áreas que complementan a los vértices y aristas con respecto al plano y están rodeadas por ellos. Las caras modelan o representan regiones geográficas, superficies, rangos de transmisión, etc. Las gráficas geométricas pueden ser clasificadas de muchas maneras, una de ellas es por la forma de sus caras.

Definición 6 *Una subdivisión convexa es aquella para la cual cada cara es un polígono convexo a excepción de la cara exterior la cual es el complemento de un polígono convexo (que se forma de la unión de las caras interiores).*

Las gráficas convexas tienen diseños aún más sencillos que las gráficas planas, por esta razón también es más fácil trabajar con ellas aunque a medida que se simplifican las cosas obviamente se pierde generalidad.

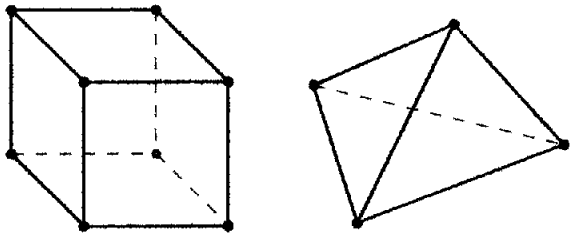

Figura 2.5: Politopos regulares, el cubo y el tetraedro.

2.1.2. Triangulaciones

Definición 7 *Una triangulación T es una subdivisión convexa en la que cada cara interior es un triángulo.*

Una triangulación es una gráfica plana con el máximo número de aristas. Es importante notar que no se define triangulación como aquella gráfica en la que *todas* las caras son triángulos sino aquella en la que solo las caras interiores son triángulos.

Definición 8 *Triangulación regular es aquella gráfica $G = (V, E)$ en la que V y E son proyecciones ortogonales en el plano de los vértices y aristas del recubrimiento convexo inferior de algún* politopo P.

Coxeter [7] define un politopo como la sucesión punto, segmento de línea, polígono, poliedro, etc. En general un politopo es una región en un espacio n dimensional delimitada por un número finito de hiperplanos. Algunas veces también se llama politopo a la unión de todos estos puntos, es decir a los hiperplanos que delimitan y a la región en sí.

Para un politopo en R^3 el *recubrimiento convexo* es el sólido convexo de menor tamaño de contiene todos los puntos del politopo. Un sólido convexo es aquel donde existe un segmento de línea que une a cada par de puntos dentro del sólido de tal manera que el segmento es parte del sólido. La parte *inferior* del recubrimiento convexo son los puntos que se pueden observar desde *abajo* considerando que es un sólido y no se puede ver a través de él. Los politopos regulares se caracterizan por su alto grado de simetría ya que sus lados y ángulos deben ser todos iguales. La figura 2.5 muestra algunos politopos regulares sencillos y familiares.

La *triangulación de Delaunay* es un caso particular de una triangulación regular donde el politopo que proyecta la triangulación es un *paraboloide*, ver figura 2.6. Esta triangulación es una de las más estudiadas por sus propiedades y aplicaciones, ver figura 2.7.

Definición 9 *La triangulación de Delaunay es aquella triangulación en la que para cada triángulo el círculo que lo circunscribe no contiene vértices además de los que forman el triángulo.*

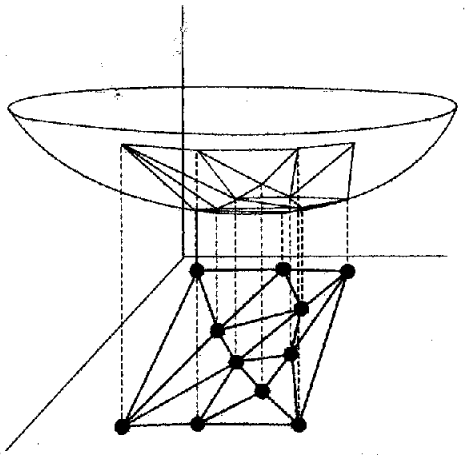

Figura 2.6: Proyección de un paraboloide en el plano.

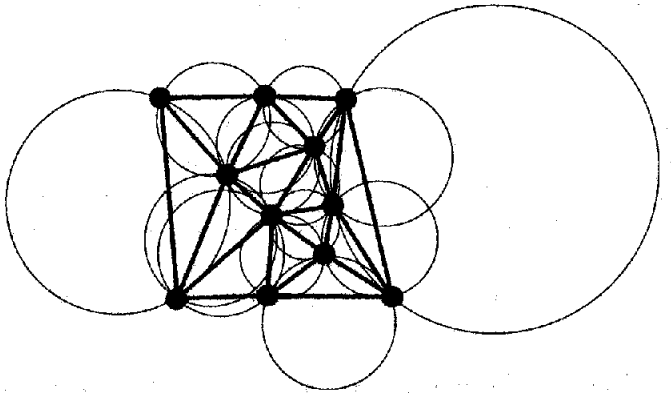

Figura 2.7: Triangulación de Delaunay y los círculos que circunscriben los triángulos.

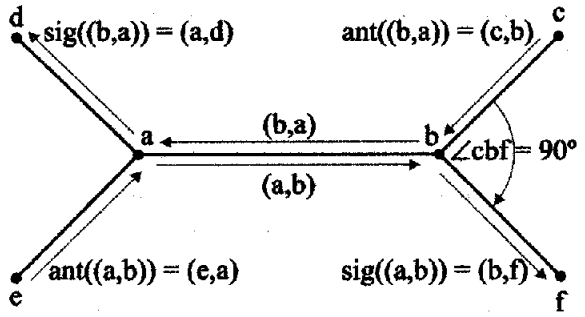

Figura 2.8: Operaciones básicas para gráficas geométricas

Existe una gran variedad de tipos de gráficas, cada una surge del estudio de algún problema en particular y cuando se estudian a fondo es posible encontrar subfamilias. Cada una de ellas tiene sus aplicaciones y propiedades particulares, algunos ejemplos son los árboles, polígonos simples, cubos, cuadrilaterizaciones, etc. En el presente trabajo se presta especial atención a las gráficas planas y dentro de estas a los árboles, pero también se trabaja con triangulaciones y un tipo especial de gráfica no plana que se denomina *gráfica cuasi plana.*

2.2. Operaciones básicas

Las gráficas como elementos matemáticos son susceptibles a operaciones básicas que facilitan la descripción de los algoritmos. Parte fundamental de la tesis se centra en el hecho de que estas operaciones pueden realizarse con memoria constante, información local y sin alterar el estado del vértice o arista. Estas condiciones básicas son las que permiten el desarrollo de operaciones y algoritmos más complejos. La figura 2.8 muestra estas operaciones básicas.

Definición 10 ***L(v)*** *obtiene toda la información local del vértice* v.

$$L : V \mapsto V^*$$

Definición 11 ***xcor(v)*** *obtiene la coordenada en el eje* x *del vértice* v.

$$xcor : V \mapsto \Re$$

Definición 12 ***ycor(v)*** *obtiene la coordenada en el eje* y *del vértice* v.

$$ycor : V \mapsto \Re$$

Definición 13 ***sig(e)*** *obtiene la arista que le sigue a* $e = (u, v)$ *en el sentido contrario de las manecillas del reloj, teniendo como punto de referencia el vértice* v.

$$sig : E \mapsto E$$

Definición 14 ***ant(e)*** *obtiene la arista que le sigue a* $e = (u, v)$ *en el sentido de las manecillas del reloj, teniendo como punto de referencia el vértice* u.

$$ant : E \mapsto E$$

Definición 15 $\angle abc$ *denota el ángulo formado por los vértices* a, b *y* c *con centro en* b *y medido del segmento* $\overline{ba}$ *al segmento* $\overline{bc}$ *en el sentido de las manecillas del reloj.*

Estas son las operaciones más utilizadas como apoyo dentro de los algoritmos expuestos a lo largo de la tesis, sin embargo existen otras operaciones igual de sencillas y algunas otras más complejas que también se pueden utilizar dentro de los algoritmos, por ejemplo la detección de intersecciones y la comparación de valores. En general están permitidas todas aquellas operaciones que puedan realizarse con memoria constante, información local y sin alterar el estado de la gráfica.

2.3. Clasificación de algoritmos

El uso informal de la palabra *algoritmo* se remonta a la edad media cuando era usada para referirse a la ejecución sistemática de operaciones matemáticas mediante el uso de números arábigos. El término *algoritmo* se deriva del trabajo en aritmética del matemático árabe Al-Khwarizmi [3] en Bagdad cerca del año 825 d.C.. De manera coloquial un algoritmo es un procedimiento, receta, lista de pasos o conjunto de reglas u órdenes que permite obtener un resultado o realizar una tarea en particular [30]. En este caso se trata de las tareas que realizan los sistemas de cómputo o modelos semejantes.

Definición 16 *Algoritmo es una secuencia finita de pasos computacionales bien definidos que transforma un conjunto de valores llamado* entrada *en otro conjunto de valores llamado* salida *[45] en un tiempo finito para cualquier entrada.*

2.3.1. Orden

Los algoritmos pueden ser evaluados y clasificados de muchas maneras pero el principal parámetro para determinar que tan bueno es un algoritmo es su rapidez, es decir, que tan rápido obtiene la salida una vez que se le proporciona la entrada. Debido a la diversidad de dispositivos de cómputo la rapidez no puede ser medido en unidades de tiempo porque la tecnología del dispositivo afectaría el resultado.

Para lograr una independencia entre la tecnología y el algoritmo, se calcula la taza de crecimiento del tiempo con respecto al tamaño de la entrada. Para obtener una idea del tiempo que tardará en obtener la salida se obtiene el número de operaciones significativas para cada problema. Cabe mencionar que cada problema tiene una operación significativa propia. Una vez con esta estimación del tiempo en relación con el tamaño de la entrada se

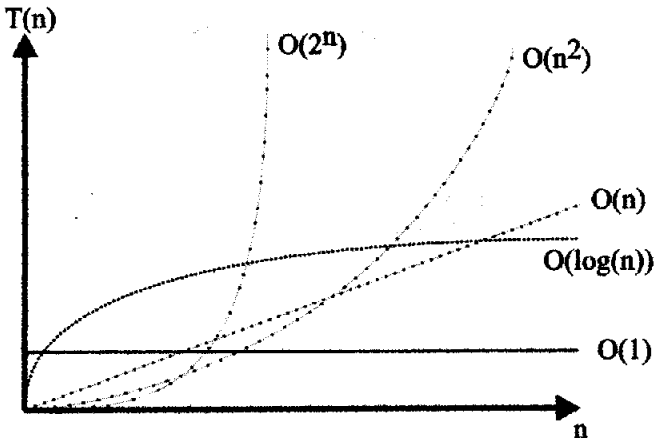

Figura 2.9: Órdenes más comunes para un algoritmo, en orden ascendente de complejidad $O(1)$, $O(\log(n))$,$O(n)$,$O(n^2)$ y $O(2^n)$.

clasifica en un orden de funciones. Este orden se define en base a la taza de crecimiento de la función y se llama coloquialmente *notación O mayúscula*. En este orden las funciones que tengan una taza semejante son agrupadas formando conjuntos de funciones entre los cuales está bien definido cual es el orden más rápido.

El orden de un algoritmo puede ser calculado en base al número de operaciones, a la cantidad de memoria, al número de mensajes emitidos o cualquier otro parámetro que sea crítico para el problema que se pretende resolver. El orden de un algoritmo habla del desempeño del algoritmo en cuanto al uso de recursos y siempre se busca el orden más bajo, el más rápido, el más pequeño. La figura 2.9 muestra algunos de los órdenes más comunes.

2.3.2. Paradigma

En computación al igual que otras áreas un mismo problema se puede resolver de diferentes maneras, por lo tanto los algoritmos se pueden clasifican en base a la forma en que resuelven un problema. A la forma en que resuelven el problema se le conoce como *paradigma*. Un ejemplo es el paradigma *divide y vencerás* en donde el algoritmo divide los datos de entrada y resuelve dos problemas de tamaño menor para después juntar las salidas de cada uno y calcular la salida final.

En base a este criterio existen muchos paradigmas, sin embargo algunos de ellos destacan más que otros debido a que han demostrado ser útiles en más de una ocasión y sirven de base para diseñar nuevos algoritmos. Estos son los paradigmas conocidos y estudiados dentro de las ciencias de la computación. A medida que el desarrollo de algoritmos avanza surgen nuevos paradigmas, algunos se mezclan con los anteriores, otros quedan obsoletos y existen muchos que aunque ya son utilizados ampliamente todavía no se les reconoce como tales.

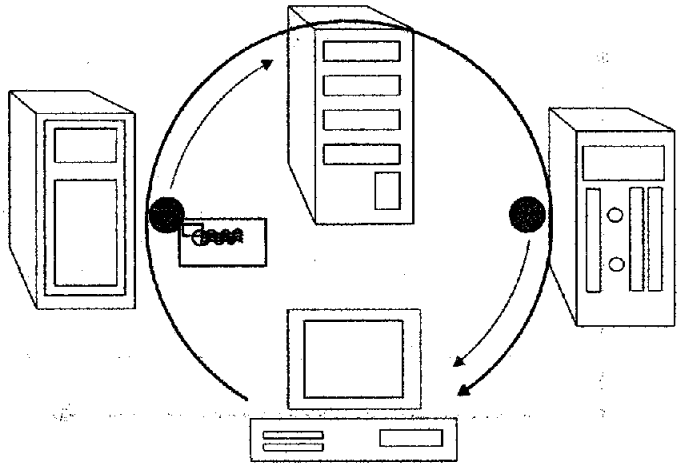

Figura 2.10: Agente de correo que se desplaza por los servidores entregar el correo.

2.3.2.1. Agentes móviles

Un paradigma particularmente interesante es el de los *agentes móviles*, recientemente utilizado para resolver problemas donde está involucrada la tecnología inalámbrica. Un agente móvil es un programa que migra autónomamente de un dispositivo de cómputo a otro dentro de una red, es decir que el agente decide cuando y a donde ir. El agente móvil puede suspender su ejecución en cualquier momento, transportarse a otro dispositivo y continuar su ejecución allí.

Por ejemplo, un agente móvil de correo electrónico puede tomar un correo y moverse a un ruteador dentro de la red, preguntar por el destino y moverse hacia él para después entregar el mensaje y terminar. El agente móvil es capaz de realizar tareas complejas para lograr su objetivo como pueden ser el ruteo o la búsqueda, la figura 2.10 muestra un ejemplo.

Durante mucho tiempo el paradigma cliente-servidor predominó como la principal solución a los problemas en red, sin embargo los agentes móviles tienen muchas ventajas con respecto a este paradigma, por ejemplo :

1. **Eficiencia.** Los agentes consumen menos recursos de la red debido a que mueven el cómputo hacia los datos y no los datos hacia el cómputo.

2. **Tolerancia.** Los agentes no necesitan una conexión contínua entre dos dispositivos.

3. **Conveniencia.** El paradigma oculta los canales de comunicación pero no la ubicación del cálculo.

4. **A la medida.** Los agentes permiten a los clientes y servidores tener más funciones mediante la programación entre ellos.

Existen otros paradigmas que tienen algunas de estas características, como por ejemplo las llamadas a procedimientos remotos, pero estas solo se aplican a un tipo de problema en particular mientras que un sistema de agentes móviles es un marco de trabajo sencillo y homogéneo en el que se pueden implementar diferentes tipos de aplicaciones en forma distribuida, fácil y eficiente.

2.3.3. Distribución de datos y cálculos

Las redes de computadoras, la comunicación inalámbrica y otros factores tecnológicos han originado una nueva manera de clasificar a los algoritmos basandose en la manera en que están distribuidos tanto los datos como el propio cálculo. Y es que todos estos factores han permitido distribuir los datos de entrada y de salida entre más de un sistema de cómputo así como también la ejecución de varios algoritmos al mismo tiempo en distintos puntos y la posibilidad de usar recursos computacionales compartidos. En base a esto es posible clasificarlos de la siguiente manera:

- **Algoritmos Centralizados.** Todos los datos de entrada se conocen *a priori* y están disponibles dentro de la misma unidad de procesamiento en todo momento.

- **Algoritmos Distribuidos.** Todos los datos de entrada se conocen *a priori* y se distribuyen en distintas unidades de procesamiento para hacer más rápido su procesamiento al hacer más pequeñas las tareas. Se requiere de comunicación entre las unidades para dar un resultado final.

- **Algoritmos Locales.** No se conocen *todos* los datos *a priori* solo algunos de ellos, los datos locales. Los demás datos se van obteniendo a medida que avanza la ejecución del algoritmo.

Los primeros algoritmos que se desarrolláron fueron centralizados debido a la naturaleza de los primeros sistemas de cómputo. Los algoritmos distribuidos nacen de la necesidad de aumentar el poder de cómputo compartiendo recursos mediante redes de computadoras. Por último los algoritmos locales tienen su origen en versiones *On-line* de problemas clásicos, es decir, aquellos en donde los datos de entrada se proporcionan poco a poco y no se sabe nada acerca de los datos que faltan por llegar. A medida que llegan los datos se procesan y se obtiene un resultado parcial que será completado cuando llegue el último dato, precisamente por estas razones se les conoce como algoritmos *on-line* o *en línea.*

La comunicación móvil y las redes especializadas son un campo de aplicación activo muy interesante para los algoritmos que solo usan información local. Debido a que los dispositivos portátiles no disponen de muchos recursos y que los cambios en la red son tantos, es conveniente y necesario que la información utilizada en cada punto sea puramente local.

2.4. El modelo de memoria

En la mayoría de los problemas la cantidad de memoria no es un factor importante porque actualmente las capacidades de almacenamiento han superado las necesidades en casi todas la áreas. Sin embargo, como ya se ha visto, el tipo de problemas que se resuelven aquí requieren que la cantidad de memoria sea mínima al igual que el tiempo de procesamiento.

Los algoritmos diseñados están pensados para ser ejecutados por agentes móviles bajo un modelo de memoria restringido que destaca las características y ventajas del mismo.

1. Información local $L(u)$. En cualquier momento de la ejecución el agente solo conoce una parte de la información, solo la información que es local al punto donde se encuentra. Para algún estado o punto u la información local se denota como $L(u)$.

2. Memoria constante $\gamma()$. La función $\gamma()$ transforma y determina la cantidad de bloques de memoria a utilizar, la cual solo puede ser una cantidad constante con respecto al tamaño de la entrada.

3. Sin marcas. El modelo no contempla marcar o dejar marcas en cada estado o punto en el que se encuentre, es decir los estados o ubicaciones son inalterables.

Los algoritmos para los agentes se pueden clasificar en base a la cantidad de memoria que utilizan.

2.4.1. Agentes locales sin memoria

Se dice que un agente es local y sin memoria (memoryless o stateless) cuando el siguiente paso del agente depende solo de la información local $L(u)$ disponible. Matemáticamente se puede modelar como una función:

$$\delta\colon [L(u)] \mapsto v$$

Donde u es el estado o punto actual y v será el nuevo estado o lugar al que el agente llegará. Es muy importante observar que este tipo de algoritmos tienen la misma salida para un mismo estado o punto, es decir que no importa cuantas veces pasen por un estado siempre llegarán al mismo resultado. En general esto ocasiona que este tipo de algoritmos sean propensos a quedar atrapados en ciclos.

2.4.2. Agentes locales con memoria

Un agente que recuerda lo ocurrido en un punto anterior es capaz de tomar mejores decisiones y esto solo es posible si cuenta con cierta cantidad de memoria adicional. Un

agente local con memoria es aquel que para determinar su siguiente estado o movimiento utiliza, además de la información local $L(u)$, una cantidad de K bloques de memoria.

Esta cantidad de memoria se da en términos del tamaño de entrada por lo que se puede hablar de algoritmos con memoria de orden cuadrático $O(n^2)$ ó de algoritmos con memoria de orden constante $O(1)$ por ejemplo. En la práctica los bloques de memoria son valores compuestos de $O(\log(n))$ bits, por lo que técnicamente no son bloques de tamaño constante. Esto es comprensible y sucede en cualquier aplicación computacional, pues si se considera un algoritmo que cuente el número de vértices de una gráfica se necesita como mínimo una cantidad de memoria igual a $\Omega(\log n)$ para n vértices, porque de otra manera el resultado sería incorrecto. A pesar de tener que usar bloques de tamaño no constante, esto no representa una limitación para el modelo pues es una asunto natural con el que trabajan todas las implementaciones, pero es importante que se tome en cuenta.

Una función que describe el comportamiento de un agente con K bloques de memoria es la siguiente:

$$\delta\colon [L(u), M^k] \mapsto v$$

Donde M representa un bloque de memoria y las otras variables el mismo significado que en la definición anterior. Además de determinar el estado siguiente en base a la memoria y la información local es necesario contar con otra función γ que determine la forma en que se manejan los bloques de memoria.

$$\gamma\colon [L(u), M^k] \mapsto M^k$$

En cada paso del agente, esta función se encarga de mantener actualizados los valores de la memoria y es muy probable que en ella se coloquen los valores de entrada al iniciar el algoritmo del agente y que al terminar su ejecución aquí esté la salida del mismo.

2.4.3. Agentes locales con memoria $O(1)$

La principal preocupación y objetivo de esta tesis es mantener la cantidad de memoria del agente en un valor constante mínimo con respecto a la entrada, al mismo tiempo que el agente debe ser capaz de realizar sus tareas. El factor de crecimiento del número de bloques de memoria con respecto al tamaño de la entrada debe ser constante a medida que aumenta el tamaño de la entrada. Esto permite al modelo ser altamente escalable y es una de sus principales ventajas en comparación con modelos anteriores. A este tipo de agentes o algoritmos se les conocer como algoritmos con memoria constante ó algoritmos con memoria $O(1)$ haciendo referencia a su comportamiento asintótico.

2.5. Propuesta geométrica

Usar gráficas geométricas para problemas donde basta usar gráficas sin información geométrica es una idea fundamental para esta tesis. Todo parece indicar que esta idea surgió mediante el estudio del ruteo, de hecho se podría decir que el *ruteo compacto* [32] es el antecedente inmediato del ruteo geométrico. Por esta razón vale la pena revisar los resultados básicos del ruteo compacto para entender las ventajas e importancia del uso de la información geométrica.

El ruteo compacto se originó en el área conocida como cómputo de alto rendimiento donde las redes de comunicación de microprocesadores son fundamentales para lograr altas velocidades en la ejecución de programas. Estas redes tienen problemas de comunicación muy particulares debido a que cuentan con memoria constante dentro de los microprocesadores y además la mayoría de los recursos computacionales disponibles son usados para la ejecución de programas de tal manera que la comunicación no debe afectar el desempeño de estos.

Una solución es conectar directamente todos los microprocesadores entre sí, pero esto no es viable debido a que los canales de comunicación son muy costosos y sí aumenta el número de microprocesadores sería casi imposible organizar las conexiones.

Debido a esto es necesario buscar otras formas de interconección de preferencia formas que al combinarlas con esquemas de comunicación resulten en mejores soluciones. Algunos aspectos deseables para estas formas de conectar los elementos de la red son:

- **Escalabilidad.** Que al agregar otro elemento el desempeño no se vea afectado.
- **Competitividad.** Que la ruta más corta para cualquier pareja de vértices en la red sea competitiva.
- **Tolerancia a fallas.** Que la red siga funcionando a pesar de que uno o varios microprocesadores o enlaces de comunicación dejen de funcionar.

El área del cómputo en paralelo tiene bien estudiadas las propiedades de diversas topologías y los esquemas de ruteo que son eficientes para cada una de ellas. Algunas de las topologías más populares están inspiradas en conceptos geométricos como por ejemplo anillos, mallas, árboles, cubos y pirámides. Esta relación entre geometría y desempeño en las redes permite darse cuenta de su importancia.

Aunque estas topologías son muy eficientes, su aplicación está dirigida a redes estáticas que no cambian su configuración y por lo tanto es casi imposible aplicarlas en redes dinámicas como Internet ó redes móviles inalámbricas. En respuesta directa a esto los esfuerzos se concentraron en buscar algoritmos de ruteo que funcionaran para topologías arbitrarias y se dejó un poco de lado la importancia de los conceptos geométricos en relación con la eficiencia del ruteo.

Cuadro 2.1: Tabla compacta del vértice b de la gráfica 2.11

Por la Arista	Se llega a los vértices
(b,a)	$\{a,e,d\}$
(b,c)	$\{c\}$
(b,g)	$\{g,f\}$
(b,h)	$\{h\}$

2.5.1. Tablas compactas

Los algoritmos tradicionales de ruteo están basados en el uso de tablas que contienen información acerca de las rutas hacia todos los demás vértices de la gráfica. Aquí lo más importante es *llenar* las tablas con la información adecuada de forma rápida, actualizarlas cuando existan cambios en la red y poder consultarlas de forma ágil. El principal problema en todos estos puntos es que las tablas generalmente son muy grandes.

El ruteo compacto ataca esta problemática reduciendo el tamaño de las tablas al mínimo y para lograrlo establece un *orden matemático* entre los vértices de la gráfica mediante el cual se establece una relación entre las aristas de salida y los destinos *alcanzables* mediante esa arista.

2.5.1.1. El principio

El ruteo compacto está basado en un principio muy simple: *al estar ubicados en cualquier vértice de la gráfica, la ruta hacia cualquier otro vértice tiene que pasar por alguna de sus g aristas incidentes, siendo g el grado del vértice.* Aunque esto resulta obvio tiene implicaciones muy importantes para la tesis porque en escencia se traduce en el hecho de que con tan solo g renglones, cada uno integrado por la relación entre aristas de salida y destinos, es posible resolver el problema del ruteo. Estas tablas de g renglones son una especie de *tablas compactas.*

Sí se trata de construir la tabla compacta de manera directa, por ejemplo mediante un algoritmo centralizado, se formaría una tabla de g renglones pero la información en los renglones sería tan grande como la gráfica misma resultando de muy poca utilidad. Ahora lo que se intentará hacer es que cada renglón pueda ser reducido o compactado mediante la agrupación de la información en cada uno de los g renglones. Esta agrupación se puede realizar mediante una operación sencilla de tal manera que se facilite la consulta de la tabla y el ruteo en sí. De esta manera a cada arista le correspondería una operación que recibe como parámetro el vértice destino e que indica si es ó no la arista por la cual se llega a este destino.

El siguiente paso sería buscar que operaciones pueden relacionar a los vértices del mismo renglón, y sobre todo que estas operaciones sean coherentes en todos los vértices. Sin

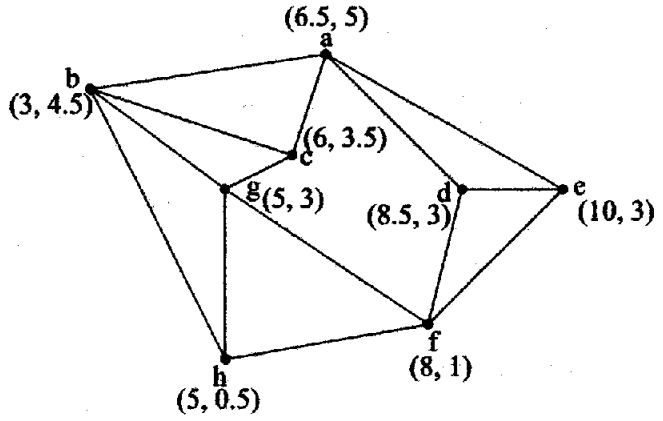

Figura 2.11: Gráfica.

embargo los identificadores de cada vértice carecen de *relación* en cada renglón de la tabla, como lo muestra el cuadro 2.1 para el ejemplo del vértice b en la gráfica de la figura 2.11. Cabe mencionar que este ejemplo fue elaborado manualmente en base a aproximaciones, pero se pueden utilizar otras técnicas para llenar la tabla.

Para relacionar a los vértices dentro del mismo renglón, el ruteo compacto asigna un identificador extra a cada vértice. Estos identificadores tienen un *orden* entre sí y permiten utilizar una función para agruparlos. Así pues son los identificadores asignados los que permiten determinar la operación dentro las tablas compactas de ruteo de g entradas.

Las operaciones y los identificadores a usar no son fáciles de calcular porque dependen de la topología de la gráfica y en muchos casos no es posible determinar funciones que entreguen rutas cortas como se esperaría en un principio. Algunos ejemplos de ruteo compacto se muestran en [42].

2.5.2. Ruteo por intervalos

Un esquema concreto de ruteo compacto es el *ruteo por intervalos*[32] en el cual los vértices se identifican mediante números enteros y las aristas tienen asociado un *intervalo* el cual sirve para agrupar los vértices del mismo renglón en la tabla de ruteo compacta.

Definición 17 *Sean los enteros a, b, n con $a, b < n$. Un intervalo $[a, b]$ con respecto a n es el conjunto de enteros consecutivos entre a y b. Se considera que n y 0 son consecutivos. Formalmente:*

$$[a,b] = \begin{cases} \{i | a \leq i \leq b\} & si \quad a \leq b \\ \{i | b \leq i \leq n\} \cup \{i | 1 \leq i \leq a\} & en \quad otro \quad caso. \end{cases}$$

Este tipo de intervalos son *cíclicos* es decir que el intervalo $[5, 2]$ para $n = 7$ incluye a los números $\{5, 6, 7, 0, 1, 2\}$, como si la cuenta desde el 7 diera vuelta de regreso al 0.

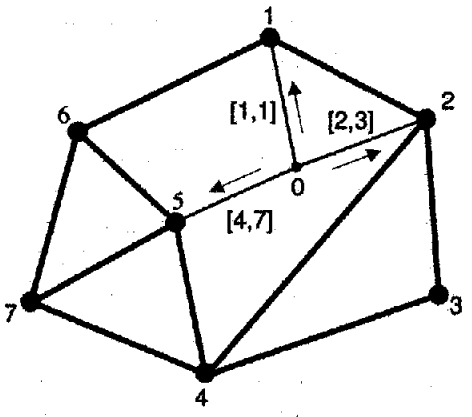

Figura 2.12: Gráfica donde se aplica ruteo por intervalos.

Todos los intervalos son respecto al número de vértices de la gráfica, es decir n. Cada arista de salida tiene asociado un intervalo de tal manera que para todo vértice la unión de los intervalos de todas sus aristas de salida de como resultado el conjunto total de vértices. La idea principal es que para llegar al vértice destino, estando en cualquier otro vértice, solo es necesario desplazarse por la arista cuyo intervalo contenga al vértice destino.

Se necesita de un algoritmo que determine las asociaciones entre vértices e identificadores y aristas e intervalos [32], pudiendo haber más de una forma de hacer estas asignaciones. Una vez realizada la asignación el ruteo es muy sencillo y solo se necesita revisar los intervalos de las aristas de salida y avanzar por la arista que contenga al identificador del destino. La tabla de ruteo compacta de cada vértice está formada por el conjunto de intervalos de sus aristas de salida. El algoritmo 3 muestra los pasos a seguir para realizar el ruteo una vez que se ha construido la tabla de ruteo compacta mediate algún otro algoritmo.

Algoritmo 3 Ruteo compacto

1: Revisar el intervalo de cada una de las aristas de salida.
2: Avanzar por la arista de salida cuyo intervalo contenga al identificador del vértice destino.
3: Repetir desde el paso 1 hasta llegar al destino.

En la figura 2.12 se observa un ejemplo donde el vértice con identificador 0 tienen tres aristas de salida cada una ellas con sus respectivos intervalos $[1,1]$, $[2,3]$ y $[4,7]$. De esta manera para mandar un mensaje desde el vértice 0 al vértice 3 utilizaríamos la arista cuyo intervalo es $[2,3]$ porque $2 \leq 3 \leq 3$.

En [23] se demuestra que toda gráfica tiene una asociación que permite realizar ruteo por intervalos, pero en [37] se demuestra que existen gráficas para las cuales el ruteo por intervalos no proporciona rutas óptimas. El ruteo compacto es una idea que en principio resulta prometedora pero la asignación de intervalos y etiquetas requiere del conocimiento total de la gráfica, topología y número total de vértices, requisitos difíciles de cumplir incluso para redes tradicionales y particularmente improbables en sistemas dinámicos. Por esta razón

los resultados se aplican a redes donde los elementos se pueden considerar altamente estáticos como por ejemplo las redes de microprocesadores.

2.5.3. Etiquetas geométricas e información local

Lo que llama la atención del ruteo compacto es que no utiliza tablas que hagan referencia a toda la gráfica, sino tablas con una cantidad de información mucho más *ligera*. Se puede observar que esto depende en gran medida del tipo de etiquetas sobre los vértices de la red porque son los que permiten simplificar las relaciones entre rutas y destinos.

Se denomina *propuesta geométrica* al uso de etiquetas con las coordenadas (x, y) de cada vértice como parte de su identidad, lo cual lleva a las redes hacia una inmersión en el plano. La idea de usar estas coordenadas es que se puedan crear *tablas geométricas compactas* en cada vértice y que estas estén compuestas solamente por las coordenadas de los vértices adyacentes o vecinos, es decir solo información local.

Lo interesante de usar coordenadas es que existen más relaciones entre vértices y aristas las cuales van más allá de la adyacencia y se extienden a parámetros como la distancia, dirección y posición dando origen a relaciones más complejas como líneas, círculos, cierres convexos, árboles generadores de peso mínimo, etc. Esta variedad de relaciones ofrece una amplia gama de posibilidades para todos los algoritmos que trabajan en gráficas geométricas incluyendo al ruteo.

El uso de información local es una necesidad real en la mayoría de los sistemas actuales, por ejemplo en las redes inalámbricas. En este tipo de entornos no es posible conocer el estado de la red en forma global y mucho menos al instante. Una visión más objetiva es considerar que la información que se dispone en todo momento es escencialmente local.

Así pues la propuesta geométrica encaja bien en sistema que están cambiando constantemente pues la información geométrica permite realizar operaciones locales para resolver problemas globales, como se verá en el siguiente capítulo.

2.6. Conclusión

Las gráficas geométricas tienen una aplicación práctica muy importante debido a que tienen mayor información sino que son altamente intuitivas para el ser humano. Hasta hace poco las gráficas geométricas eran utilizadas cuando el problema así lo ameritaba, es decir solo cuando era necesario. En los problemas que aquí se presentan se usa la geometría para compensar la falta de memoria de los agentes y la localidad de la información, de tal manera que se pueda dar solución a problemas que de otra manera no sería posible resolver dadas las restricciones. Estas soluciones están siendo aplicadas en problemas como el ruteo y la comunicación inalámbrica.

Gracias a los dispositivos de localización o mediante topologías como las mallas es posible introducir la geometría en problemas tradicionales. Una vez que la geometría es parte

del problema, los agentes móviles pueden realizar sus tareas de forma eficiente usando una cantidad de memoria constante. La intención de esta investigación no solo es encontrar algoritmos que funcionen bajo este esquema sino que además se puedan ejecutar bajo condiciones restringidas en cuanto a información y a la capacidad de modificar su entorno.

Con estas sorprendentes cualidades los algoritmos de memoria constante en entornos geométricos limitados ya han llamado la atención de muchos investigadores en diferentes áreas. Sin embargo el diseño de este tipo de algoritmos representa un reto significativo porque se sale de los parámetros estándar. En el diseño de estos algoritmos se busca minimizar no solo el tiempo de procesamiento sino también la memoria y la comunicación.

En este contexto se desarrollan los siguientes capítulos exponiendo algoritmos para agentes móviles que trabajan en gráficas geométricas en dos dimensiones, utilizando memoria constante, sin alterar la gráfica o el estado de sus elementos y disponiendo de información local en cada punto donde se encuentra el agente.

Capítulo 3

Ruteo en gráficas geométricas

Actualmente las redes de comunicación juegan un papel muy importante en el desarrollo de la humanidad, por ejemplo las redes de computadoras, redes celulares e incluso las redes de microprocesadores que hacen posible cosas como la simulación de fenómenos naturales, reacciones químicas o el cómputo de nóminas y transacciones bancarias.

Desde su creación estas redes han sido estáticas, es decir que los dispositivos que pertenecen a ella no cambian la forma en que se conectan entre sí. Parte de esta naturaleza estática radica en la necesidad de usar cables o alambres para poder comunicarse, la otra parte se debe a que no son dispositivos portátiles pues para poder serlo requerirían grandes cantidades de energía y un tamaño apropiado.

Recientemente tecnologías como la miniaturización han hecho posible la existencia de computadoras portátiles y comunicaciones inalámbricas abriendo paso a redes más dinámicas y desafiantes para las ciencias de la computación. Estas redes se han venido aplicando en el monitoreo del clima con redes de sensores, comunicaciones personales con redes de área personal y otras más.

Aunque ya existen soluciones de ruteo para redes inalámbricas es necesario analizar cuidadosamente todos los aspectos del problema y estudiar a fondo otras alternativas para tratar de cumplir con las espectativas que se tienen de las redes inalámbricas.

Una de estas alternativas se centra en el uso de información geométrica y agentes de memoria limitada. Parte fundamental de esta propuesta es integrar tecnologías de posicionamiento (global o local) de tal manera que los elementos en la red conozcan sus coordenadas (x, y) dentro de un espacio geométrico y sea posible modelar estos problemas con gráficas geométricas. Los agentes usan estas coordenadas para navegar dentro de la red y realizar diversas funciones a pesar de las limitaciones propias y del entorno.

Actualmente los algoritmos para agentes de memoria limitada están siendo usados en la etapa experimental de problemas de ruteo sobre redes de microprocesadores, redes inalámbricas, redes ad hoc y redes de sensores. Estas aplicaciones se caracterizan por que cuentan con pocos recursos y porque existe la posibilidad real de incluir dispositivos de localización que las proyecten como gráficas geométricas. Es precisamente el ruteo el primer

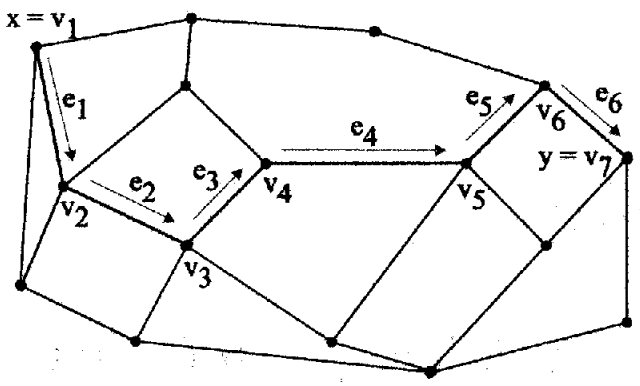

Figura 3.1: Ruta $x \rightsquigarrow y$.

problema en abordarse dentro de la tesis y también el que más resultados tiene actualmente.

3.1. Antecedentes

El ruteo, también conocido como *path finding* o descubrimiento de camino, se refiere al proceso de encontrar una ruta o camino entre dos puntos que generalmente son el punto donde se origina la necesidad de encontrar la ruta y el punto al que se desea llegar. El ruteo es inherente al desplazamiento pues cualquier movimiento implica el seguimiento de una ruta. El ruteo es utilizado por repartidores de pizza, aves que migran en invierno y los *ruteadores* en redes de computadoras como Internet para enviar datos por todo el mundo.

En los términos que conciernen a las ciencias de la computación el *ruteo* es el problema en el cual dada una gráfica se desea encontrar una secuencia de vértices y aristas llamada *ruta* de tal manera que recorriendo esta secuencia sea posible *ir* de un vértice **fuente** a un vértice **destino**.

Definición 18 *Una* ruta *es una sucesión finita y alternada de vértices y aristas de una gráfica $G(V, E)$ que inicia en un vértice origen x y termina en un vértice destino y. La ruta del vértice x al vértice y se denota por $x \rightsquigarrow y$.*

$$x \rightsquigarrow y = \{v_1 e_1 v_2 e_2 \dots e_n v_{n+1}\}$$

Con $x = v_1$, $y = v_{n+1}$, $v_i \in V$ y $e_i \in E$. Los vértices y aristas en la ruta se pueden repetir.

Definición 19 *La* longitud *de una ruta $x \rightsquigarrow y$ es el número de aristas en ella y se denota como $|x \rightsquigarrow y| = n$.*

Definición 20 *La distancia euclidiana d_e entre un vértice u con coordenadas (x_1, y_1) y otro vértice v con coordenadas (x_2, y_2) se define como:*

$$d_e(u, v) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

Definición 21 *La distancia euclidiana de una ruta se define como la suma de las distancias euclidianas entre los vértices de las n aristas de la ruta. Es decir que para la ruta* $x \rightsquigarrow y = \{v_1 e_1 v_2 e_2 \dots e_n v_{n+1}\}$, *la distancia euclidiana es:*

$$d_e(x \rightsquigarrow y) = \sum_{i=1}^{n} d_e(v_i, v_{i+1})$$

3.1.1. Tipos de rutas

La figura 3.1 muestra una ruta posible para el par de vértices x y y pero existen otras rutas y generalmente existe una que es mejor que las demás en base a algún criterio. Por ejemplo si se desea llegar al destino lo más rápido posible la mejor ruta será aquella cuyo recorrido total se realize en el menor tiempo y en condiciones normales esto sucede con la ruta de menor longitud. Esta es la llamada *ruta más corta* y es un problema bastante estudiado cuyas soluciones se utilizan en la elaboración de tablas de ruteo. Dos de los esquemas más populares para evaluar las rutas cortas son la longitud de la ruta o distancia de enlace y la distancia euclidiana. En [2] se presenta un análisis completo acerca de las ocasiones y condiciones en las cuales es más conveniente usar cada una de estas dos métricas.

Cuando se dispone de toda la información referente a la gráfica, como se supone tradicionalmente, existen muchos algoritmos centralizados y distribuidos que resuelven el problema de rutas cortas para todos los vértices por ejemplo el popular algoritmo de Bellman Ford[45]. Sin embargo no se conoce alguna solución para resolver el problema de la ruta más corta en gráficas geométricas utilizando información constante, información local y sin alterar la gráfica, entre otras limitaciones. Una conjetura personal es que no es suficiente una cantidad constante de memoria para resolver este problema. No obstante existen algoritmos que obtienen *rutas competitivas*, es decir rutas k veces más largas que la distancia más corta entre el origen y el destino. Estas rutas se conocen como rutas k *competitivas*, donde el factor k es una constante. La ventaja de este tipo de rutas es que aseguran un tiempo de recorrido acotado por el factor k. En condiciones de tráfico normal las rutas cortas o competitivas requieren menos energía para establecer una comunicación, en general cualquier factor de consumo asociado a la distancia entre dos vértices se ve minimizado al usar rutas cortas o competitivas, por esto es que son tan útiles.

Pero ¿Existe alguna ventaja al usar rutas cortas o rutas competitivas como una solución global?. Si así fuera todos los vértices utilizarían rutas competitivas o cortas para comunicarse entre sí y en la mayoría de las gráficas esto provocaría que algunos vértices y aristas sean visitados más que otros y por consiguiente que se presenten problemas de comunicación conocidos como *cuellos de botella* ó congestionamientos, que refiere al hecho de que la cantidad de mensajes supera las capacidades técnicas para manejarlos. Debido a esto también existen propuestas para distribuir el tráfico ya sea determinando más de una ruta para dos vértices de tal manera que el tráfico no se cargue en algún punto ó mediante topologías cuyas rutas cortas no provoquen este tipo de comportamiento negativo.

Por esta razón al usar cualquier tipo de rutas se debe prever el tipo de comportamiento

que tendrán al ser usadas en conjunto por todos los elementos de la red. Los algoritmos que aquí se muestran encuentran rutas en general y competitivas en algunos casos. Encontrar rutas cortas en una gráfica geométrica con un agente de memoria constante es un problema que permanece abierto, incluso en gráficas geométricas planas.

3.2. El modelo

Los algoritmos que se presentan en este capítulo están concebidos bajo el mismo modelo que se describió en el capítulo anterior. Recordando estas condiciones para el problema de ruteo se tiene que:

- En cada momento el agente conoce su posición (x, y) dentro de la gráfica así como la posición del origen y del destino de la ruta que busca.
- El agente dispone de una cantidad constante de memoria para almacenar datos, direcciones, etc. Este espacio de memoria es parte del agente y viaja con él. Esto implica que el agente no puede conocer la forma global de la gráfica ni llevar un registro de la ruta recorrida.
- En cada vértice de la gráfica existe información acerca de la posición de los vértices vecinos. La información constante del agente y esta información local son los únicos elementos disponibles para determinar el siguiente vértice en la ruta.
- No está permitido alterar la información dentro de los vértices. Esto quiere decir que la única manera que el agente tiene para saber si ya visitó un vértice o no, es si la dirección del vértice está almacenada en su espacio de memoria constante. El vértice no puede almacenar datos más que el código del agente y sus datos constantes, todo estos solo durante su visita al vértice, porque una vez que el agente ha terminado su trabajo y se desplaza a otro vértice, se pierde todo registro de que el agente estuvo allí.

Se buscan algoritmos que permitan el ruteo en redes geométricas de tal manera que al mensaje solo se le agrege una cantidad de memoria constante. Esta información incluye coordenadas geométricas, datos del algoritmo y al propio algoritmo. De esta forma el conjunto de información que viaja junto con el mensaje es precisamente el agente móvil de memoria limitada que ruteará para entregar el mensaje al destino.

3.3. Ruteo por caras

Originalmente planteado en [12] como Compass Routing II también se conoce como *"Face Routing"* y tiene una variante que mejora su desempeño conocida como *"Face Routing 2"* [35]. Se le denomina ruteo por caras debido a que el agente recorre la secuencia de caras que une al origen con el destino.

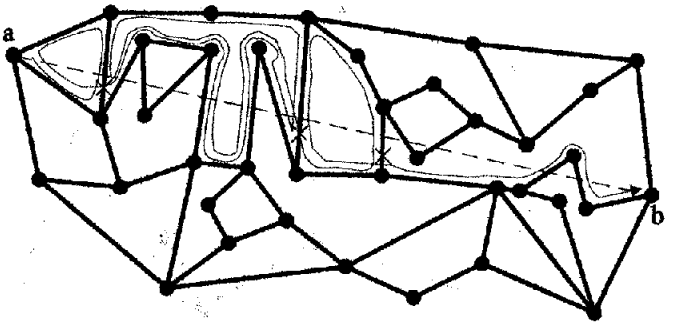

Figura 3.2: Ruta seguida por el ruteo por caras, las cruces muestran los puntos donde se cambia de cara.

Algoritmo 4 Ruteo por caras

1: Usar el segmento de recta que une los vértices origen y destino.
2: Identificar la cara adyacente que intersecta este segmento.
3: Recorrer la cara usando la regla de la mano derecha y recordar el punto de intersección más lejano con la línea.
4: Volver a recorrer la cara hasta ese punto.
5: Cambiar de cara en la arista que contiene ese punto.
6: Repetir desde el paso 3 hasta llegar al destino.

La idea principal es recorrer la cara *actual* dentro de la ruta y encontrar el punto más cercano al destino sobre la línea de referencia. Una vez localizado este punto, se regresa a la arista que lo contiene y cambia de cara. Este proceso se repite hasta llegar a la cara que contiene el vértice destino. El algoritmo 4 muestra el procedimiento.

Este algoritmo hace uso del segmento de recta que une al origen con el destino y toma en cuenta las intersecciones de este segmento con las aristas de las caras que se recorren. La figura 3.2 muestra un ejemplo con cruces en los lugares donde se hace el cambio de cara. Este algoritmo es capaz de encontrar una ruta para dos vértices dentro de una gráfica plana en un tiempo de $O(n)$ [35] y solo necesita memoria constante.

Este algoritmo garantiza encontrar una ruta siempre y cuando la parte de la gráfica involucrada en el ruteo permanezca estática durante el tiempo que toma encontrar la ruta [35]. Esta condición es necesaria porque a pesar de que es posible modificar este algoritmo para trabajar en condiciones más dinámicas hasta el momento no existe un modelo adecuado bajo el cual se puedan realizar estas adaptaciones.

A pesar de esto el algoritmo se pueden extender fácilmente para trabajar como un algoritmo de difusión sin la necesidad de enviar más de una copia del mensaje, así como también como un algoritmo de difusión geográfica o geométrica donde el mensaje se entrega solo en una zona determinada.

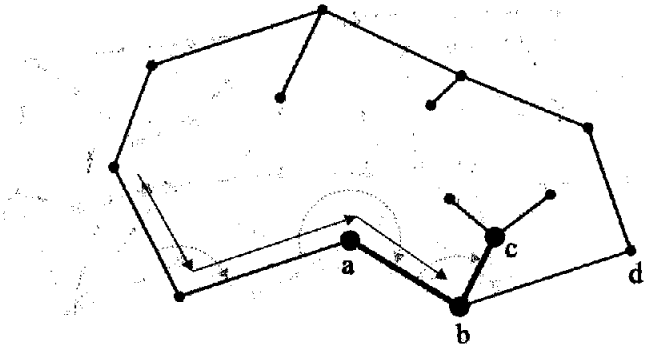

Figura 3.3: Regla de la mano derecha, cuando el agente llega al vértice b, puede recordar por donde llegó, vértice a y determinar cual de los vértices vecinos restantes c ó d es el que le sigue simulando que lleva la mano derecha pegada a la arista (a, b).

3.4. Ruteo por la regla de la mano derecha

Algoritmo 5 Ruteo por la regla de la mano derecha

1: Usar la línea que une al vértice origen con el destino.
2: Recorrer hacia la derecha la cara que intersecta esta línea.
3: Ignorar las aristas que intersecten la línea.

Cuando el algoritmo de ruteo por caras se aplica sobre gráficas convexas su funcionamiento resulta idéntico al ruteo basado en la *regla de la mano derecha* [38]. La regla de la mano derecha es una técnica usada en la exploración de laberintos que dice que si al recorrer un laberinto se mantiene la mano derecha pegada a la pared eventualmente se encontrará la salida. Usando esta sencilla regla es posible recorrer todos los vértices y aristas de cualquier cara [20] usando información local y memoria constante. Por esta razón la regla de la mano derecha es una operación básica para las aplicaciones geométricas en general.

La figura 3.3 muestra un recorrido usando esta regla. Análogo a este concepto está la regla de la mano izquierda que simplemente realiza un recorrido en sentido contrario.

Lo interesante del recorrido es que se visitan todos los vértices de la cara utilizando memoria constante e información local. Sí el origen y el destino están en la misma cara, el recorrido encontrará una ruta entre ellos e incluso dos ruta si se usa *la regla de la mano izquierda.*

El algoritmo de ruteo basado en la regla de la mano derecha funciona haciendo que los dos vértices, origen y destino, se encuentren en la misma cara. Como es de esperarse esto no sucede en todos los casos, es decir cada vértice está en su propia cara por lo tanto hace falta un procedimiento que parezca que están en la misma cara. Esta cara que en realidad no existe, pero que sirve para el ruteo, se conoce como *cara virtual* y se forma mediante la eliminación de aristas. En particular se eliminan (ó no se toman en cuenta) las aristas que

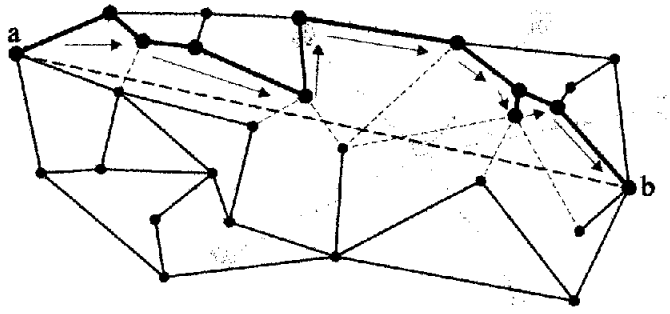

Figura 3.4: Ruta seguida por el ruteo de la mano derecha.

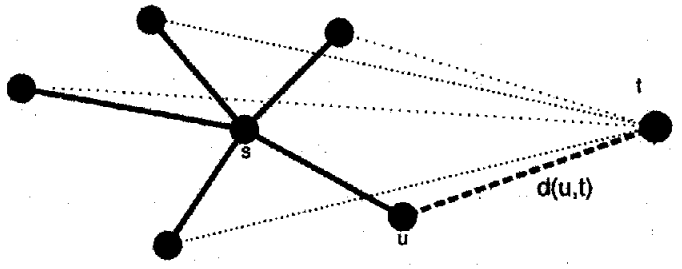

Figura 3.5: La distancia $d(u,t)$ es la más corta para todos los vecinos.

intersectan al segmento de recta que une el destino con el origen. La *cara virtual* que se forma contiene a los vértices origen y destino, y al recorrerla con la regla de la mano derecha se encuentra la ruta deseada. El algoritmo 5 muestra el procedimiento.

El primer paso es identificar la cara que contiene al segmento de recta origen destino, después recorrer esta cara mediante la regla de la mano derecha y al detectar una intersección con el segmento de recta origen destino no tomarla en cuenta para el recorrido y continuar con la siguiente arista. La figura 3.4 muestra un ejemplo de esta técnica.

Este algoritmo solo recorre las caras de la gráfica hasta que encuentra una intersección por lo tanto resulta ser muy sencillo, sin embargo solo funciona para gráficas con caras convexas y su complejidad es la misma que el ruteo por caras, es decir $O(n)$.

3.5. Algoritmo voraz

Algoritmo 6 Ruteo voraz

1: Identificar al vértice vecino que este más cercano al destino.
2: Mandar el mensaje por ese vértice.

El paradigma de programación voraz dice que para resolver un problema siempre se

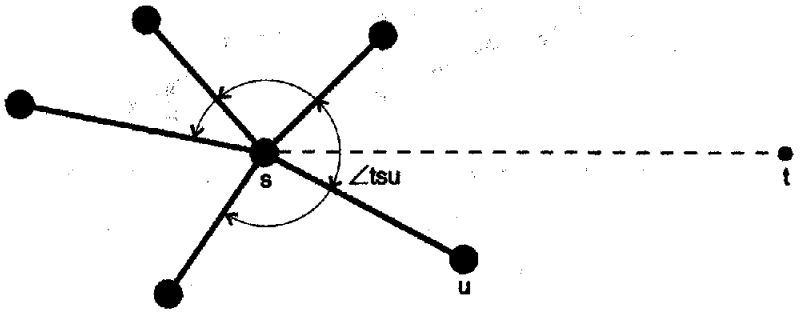

Figura 3.6: El ángulo $\angle tsu$ es el más pequeño para todos los vecinos.

debe buscar la solución parcial que más se acerque al objetivo hasta ese momento, de tal manera que los avances en cada paso dan como resultado la solución esperada. De igual manera la estrategia de ruteo voraz se desplazará al vértice que más lo acerque al destino, tomando en cuenta que solo conoce las coordenadas de los vértices adyacentes y del destino.

El algoritmo 6 es una de las estrategias de ruteo geométrico más sencilla, un agente de memoria limitada que siga este ruteo se desplazará vértice a vértice y en cada uno de estos avances acortará la distancia que lo separa del destino.

La figura 3.5 muestra la elección que el agente tomaría si estuviera en el vértice s y quisiera ir al vértice t. En este caso usamos la distancia euclidiana pero es posible usar alguna otra. De una manera formal y recordando el modelo planteado en la sección 2.4 del capítulo anterior las funciones que modelan este agente son:

$$\begin{aligned} \delta_{voraz}(L(a),t) &= u|min\{d(u,t)\}\forall u \in L(a) \\ \gamma_{voraz}(L(a),t) &= t \end{aligned}$$

Solo se utiliza un bloque de memoria extra que es el del destino y la complejidad es de $O(n)$, sin embargo solo funciona para triangulaciones de Delaunay y en casos particulares para otras gráficas geométricas.

3.6. Ruteo por brújula

Algoritmo 7 Ruteo por brújula

1: Usar la línea que une al vértice destino con el actual.
2: Encontrar la arista incidente con el ángulo menor con respecto a esta línea.
3: Mandar el mensaje por esa arista.

El algoritmo de ruteo por brújula refleja lo que haría un turista perdido en el centro de la Ciudad de México con intención de visitar la Torre Latinoamericana, simplemente se

orientaría observando la Torre y compararía esta dirección con las direcciones de las calles que pueda tomar. El algoritmo 7 muestra los pasos a seguir dentro de una gráfica, que es lo que haría un agente de memoria limitada.

La figura 3.6 muestra como, al estar en el vértice s, se compara la orientación mediante el ángulo que forman las aristas incidentes a s y el segmento de línea imaginario que une al vértice actual con el destino. De todas las orientaciones se toma la más cercana al segmento de línea que lleva al destino, es decir que el agente se comporta como si llevará consigo una brújula con *el norte* ubicado en el destino. Las funciones según el modelo serían:

$$\begin{aligned} \delta_{brujula}(L(a),t) &= u|min\{\angle uat\}\forall u \in L(a) \\ \gamma_{brujula}(L(a),t) &= t \end{aligned}$$

Desafortunadamente el ruteo por brújula solo funciona para triangulaciones *regulares* las cuales son una generalización de la triangulación de Delaunay, en otro tipo de gráficas el algoritmo puede quedar atrapado en un ciclo alrededor del destino.

3.7. Ruteo brújula-voraz

Algoritmo 8 Ruteo brújula-voraz

1: Usar la línea que une al vértice destino con el actual.
2: *En el sentido de las manecillas del reloj*, encontrar la arista incidente con el ángulo menor con respecto a esta línea.
3: *En el sentido contrario de las manecillas del reloj*, encontrar la arista incidente con el ángulo menor con respecto a esta línea.
4: De estas dos aristas tomar la que más se acerque al destino.

El ruteo por brújula y el ruteo voraz son estrategias claras y muy sencillas que funcionan en la mayoría de los casos, de hecho el tipo de gráficas para las cuales no logran rutear están bien definidas y afortunadamente tienen características diferentes entre si. De esta manera es posible usar una combinación de estos dos algoritmos para obtener uno que funciona para en cualquier triangulación.

La idea es tener dos vértices candidatos y decidirse por el más cercano al destino, ver algoritmo 8. Los candidatos son los dos vértices vecinos con el menor ángulo con respecto a la línea que va del vértice actual al destino uno en el sentido de las manecillas del reloj y otro al contrario, como se puede ver en la figura 3.7. Si hay empates para decidir al más cercano se toma cualquiera de los dos de una manera determinista. El modelo queda de la siguiente manera:

$$\delta_{brujula-voraz}(L(a),t) = s|min\{d(s,t)\}s \in \{u,v\},$$

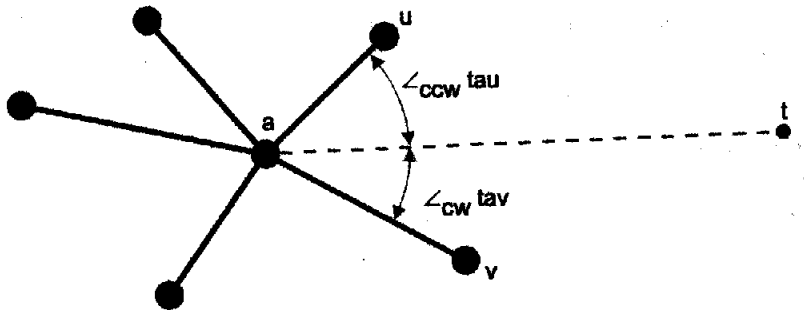

Figura 3.7: Ángulos en el sentido de las manecillas del reloj $\angle_{cw}$ y en el sentido inverso $\angle_{ccw}$.

$$u = min\{\angle_{ccw} tau\}, v = min\{\angle_{cw} tav\} \forall v, u \in L(a)$$

$$\gamma_{brujura-voraz}(L(a), t) = t$$

Aprovechando la sencillez de estos últimos algoritmos se usó el modelo matemático del capítulo anterior para modelarlos, de tal manera que es posible apreciar sus características desde otro punto de vista. Desafortunadamente a medida que los algoritmos avanzan en complejidad, generalidad y capacidades se hace más difícil describir su modelo matemático y lo peor de todo es que se hace mucho más difícil de entender. Por esta razón de aquí en adelante no se exponen las funciones para los demás algoritmos aunque si se describe la idea detrás del algoritmo y sus fundamentos matemáticos.

3.8. Algoritmos sin memoria

Cuando se sabe cual es la cantidad mínima de memoria necesaria para resolver un problema se puede decir que un algoritmo es *sin memoria* si solamente usa esta cantidad mínima.

En [38] se demuestra que este tipo de algoritmos no pueden encontrar una ruta para dos vértices dados ni siquiera cuando se trata de gráficas geométricas planas convexas. Para probarlo utilizan un conjunto finito de gráficas para las cuales en algún punto el agente tomará una decisión que deberá repetir más adelante y que lo llevará a un ciclo. Como no es capaz de recordar que ya pasó por allí y salir del ciclo no hay manera de llegar al destino usando solo información local y los datos del vértice destino y el vértice origen.

A pesar de esto existen algoritmos sin memoria que pueden rutear en gráficas planas particulares como por ejemplo en triangulaciones. Los algoritmos de ruteo voraz, ruteo por brújula y ruteo brújula-voraz son un ejemplo de algoritmos *memoryless* porque solo utilizan la información local a cada vértice además de las coordenadas del vértice origen y el destino.

Cabe mencionar que Cucka [36] evaluó experimentalmente el desempeño de estos algoritmos y lo que obtuvo fue que cuando se considera la distancia euclidiana el algoritmo

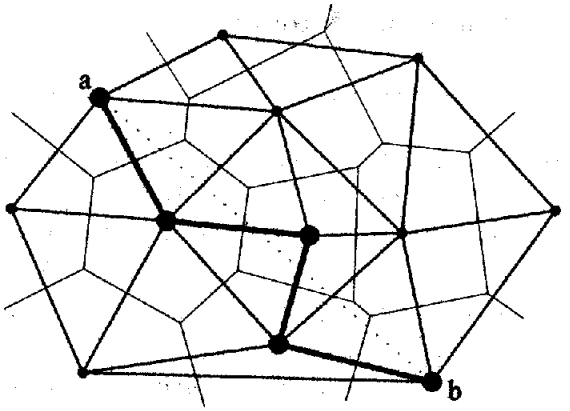

Figura 3.8: Triangulación de Delaunay y Diagrama de Voronoi de un conjunto de puntos y la ruta que encuentra el ruteo Voronoi.

de ruteo voraz se comporta mejor que el de ruteo por brújula y si se considera la longitud de la ruta (número de aristas en ella) el ruteo por brújula es mejor.

Estos algoritmos son sencillos pero solo funcionan para una clase de gráficas, sin embargo son adecuados para sistemas mucho muy restringidos donde se pueda ejercer cierto grado de control sobre la topología de tal manera que se pueda llevar la gráfica a una de las clases donde este tipo de ruteo funciona adecuadamente. Con esto se pretende ganar sencillez en la implementación, sin embargo existen aplicaciones donde no se puede ejercer este tipo de control sobre la gráfica.

3.9. Ruteo competitivo

Los algoritmos anteriores demuestran que es posible encontrar una ruta para cualesquiera dos vértices de una gráfica plana bajo condiciones de memoria e información limitada pero no aseguran nada acerca del tipo de ruta que se encontrará. Por lo tanto se despierta el interés natural por saber si es posible tener algoritmos capaces de encontrar rutas con alguna característica especial.

El siguiente algoritmo es una extensión más del ruteo por caras donde las caras sobre las que se realiza el ruteo son las regiones de Voronoi. Este algoritmo solo trabaja en triangulaciones de Delaunay porque aquí el cálculo de las regiones de Voronoi solo require memoria constante e información local. El algoritmo muestra una manera de obtener rutas competitivas bajo condiciones limitadas. Las rutas competitivas tienen una longitud mayor en un factor constante k con respecto a la distancia entre los dos vértices.

3.9.1. Ruteo Voronoi en paralelo

El diagrama de Voronoi es una construcción que muestra las regiones en donde los puntos son más cercanos a los vértices de la gráfica y resulta ser el dual de la triangulación de Delaunay. La figura 3.8 muestra un ejemplo de estas estructuras clásicas que han sido estudiadas y utilizadas durante muchos años en distintas áreas. En el diagrama de Voronoi cada vértice de la gráfica tiene asociada una región delimitada por los segmentos de recta perpendiculares a las arista de la triangulación de Delaunay y centradas justo a la mitad.

El algoritmo podría considerarse como *ruteo por regiones* porque se recorren las regiones de Voronoi en el orden en que intersectan al segmento de recta que va del origen al destino. Como a cada región le corresponde un vértice, el orden de las regiones nos ofrece un orden de los vértices asociados y es esta la ruta que se conoce como *ruta Voronoi.*

Debido a que la gráfica es una triangulación de Delaunay, el agente solo tiene que usar información local y memoria constante para determinar la ruta Voronoi, el algoritmo 9 muestra los pasos a seguir.

Algoritmo 9 Ruteo Voronoi

1: Usar el segmento de recta que une al vértice origen y destino.
2: Recorrer la *región Voronoi* actual hasta encontrar la *arista Voronoi* cuya intersección con el segmento esté más cerca del destino.
3: Ir al vértice vecino que forma esta *arista Voronoi.*
4: Repetir desde el paso 2 hasta llegar al destino.

Por si mismo el algoritmo 9 es un ejemplo de ruteo con memoria constante e información local, pero la ruta puede no ser competitiva. Lo interesante es que sí se toman los vértices y aristas de todos los triángulos que intersectan el segmento de recta que une a dos vértices así como los vértices y aristas de la ruta Voronoi, se obtiene una subgráfica denotada como G_{dfs}[8]. Esta subgráfica contienen una ruta con una longitud de $(1+\sqrt{5})\frac{\pi}{2}$ veces la distancia entre dichos vértices, es decir una ruta $(1+\sqrt{5})\frac{\pi}{2}$ competitiva.

En [38] se demuestra que si solo se toman los vértices y aristas de los triángulos que intersectan la línea origen destino se forma una subgráfica que también contiene la ruta competitiva de la subgráfica G_{dfs}, de esta manera para encontrar la ruta $(1+\sqrt{5})\frac{\pi}{2}$ competitiva no es necesario recorrer la ruta Voronoi.

En [8] se demuestra que la ruta competitiva contiene a los vértices de la ruta Voronoi que se encuentran en la parte superior del segmento de recta origen-destino en el mismo orden que son encontrados por el algoritmo de ruteo Voronoi, ver figura 3.9.

Ahora solo se necesita buscar la ruta más corta entre los vértices superiores de la ruta Voronoi y para esto existen dos opciones: que al estar en un vértice de la ruta se continue el recorrido por por *arriba* hacia el siguiente vértice superior o por *abajo* usando un *atajo* por la parte inferior.

El algoritmo 10 busca cual de las dos opciones es la mejor avanzando un paso por la primera ruta y dos por la segunda, luego cuatro por la primera y ocho por la segunda,

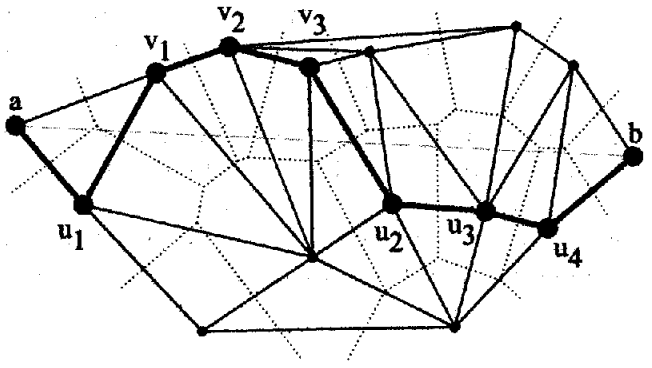

Figura 3.9: Los vértices v_i están en la ruta competitiva, en ese orden.

así sucesivamente hasta llegar al destino. Este proceso se conoce como búsqueda en *paralelo* de donde se toma el nombre del algoritmo. En a los más nueve *vueltas* sobre las dos rutas exploradas se llega al destino, por lo tanto el algoritmo es $9(1+\sqrt{5})\frac{\pi}{2}$ competitivo para una triangulación de Delaunay.

Algoritmo 10 Ruteo Paralelo-Voronoi

1: Usar el segmento de recta que va del origen al destino.
2: Tomar la distancia entre el origen y el segundo vértice de la primer ruta.
3: Tomar la distancia entre el origen y el segundo vértice de la segunda ruta.
4: Usar la distancia más pequeña como referencia.
5: Avanzar por la primera ruta hasta pasar la distancia de referencia.
6: Regresar al origen.
7: Duplicar la distancia de referencia.
8: Avanzar por la segunda ruta hasta pasar la distancia de referencia.
9: Duplicar la distancia de referencia.
10: Terminar al llegar al destino.
11: Repetir desde el paso 5.

3.9.2. Ventajas de la competitividad

En este punto debe ser claro que la competitividad le cuesta muchas *vueltas* a los algoritmos debido básicamente a la falta de memoria y a las limitaciones impuestas. Así pues es comprensible hablar de un intercambio de desempeño entre la competitividad de la ruta y el tiempo que tarda en encontrarla.

Encontrar una ruta competitiva ocasiona un mayor consumo de energía debido al intenso tránsito del agente, contrario a lo que pasa si solo se desea encontrar una ruta cualquiera. Entontes ¿Cuál es el propósito de buscar rutas competitivas? la respuesta puede venir en muchos sentidos; por ejemplo, las rutas competitivas una vez descubiertas pueden

almacenarse y así se compensa el alto costo de calcular la ruta con el número de veces que la ruta será utilizada. Por otro lado, el conocer los límites reales de este modelo de comunicación sirve de base para realizar mejoras o variaciones.

La decisión de buscar rutas competitivas depende de las necesidades y restricciones que se tengan. Algunos factores a considerar son el costo de las transmisiones, la memoria del algoritmo, lo rápido que se necesite establecer una comunicación, etc. Actualmente los algoritmos que encuentran rutas competitivas en gráficas, que no sean triangulaciones específicas, permanece como un problema abierto bajo las condiciones de localidad y memoria constante.

3.10. Ruteo en redes inalámbricas

Un ejemplo concreto de la aplicación del ruteo geométrico se encuentra en las redes inalámbricas que utilizan tecnología de comunicación de tipo radial como las ondas de radio. Los dispositivos de estas redes son capaces de establecer comunicación entre ellos solo si se encuentran dentro del volumen de cobertura esférica de cada uno. Estas redes generalmente se modelan bajo las consideraciones siguientes:

- **Dispositivos homogêneos.** Todos los dispositivos tienen la misma capacidad de cómputo y comunicación.
- **Rango de comunicación fijo.** Los dispositivos se pueden comunicar entre si solo sí la distancia entre ellos es menor que el radio de comunicación R_c, el cual no puede cambiar.
- **Posicionamiento.** Los dispositivos conocen su ubicación en algún sistema de referencia ya sea global o local.
- **Localidad.** Los dispositivos conocen las ubicaciones de los dispositivos con los que se pueden comunicar.
- **Bidireccional.** La comunicación entre dispositivos es bidireccional.
- **Sin atenuación.** No se toma en cuenta la atenuación correspondiente debido a la distancia o a las interferencias de objetos sólidos.

Para poder comunicarse con todos los participantes de la red en estas condiciones, un dispositivo primero necesita *ubicar* al destino, es decir averiguar su posición. Si está dentro del rango R_c se comunicará directamente, sí no es así mandará el mensaje a un dispositivo vecino y este a su vez repetirá el proceso hasta que el mensaje llegue a su destino. Esta aproximación se conoce como multi saltos o *multi hop* y es usada de manera regular en sistemas de ruteo y comunicación.

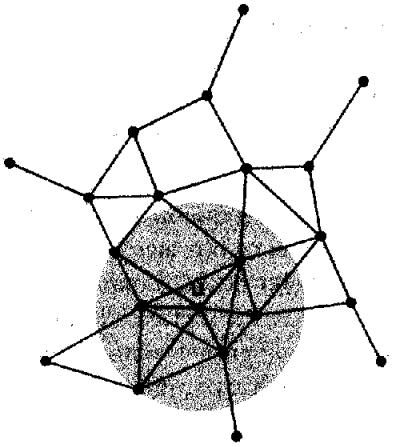

Figura 3.10: Gráfica de disco unitario, el vértice u solo se conecta con los vértices a distancia menor a 1.

3.10.1. Gráfica de disco unitario

El modelo matemático que más se usa para trabajar con redes inalámbricas es la *gráfica de disco unitario*, suponiendo que con la escala apropiada el rango de transmisión es igual a la unidad, es decir que $R_c = 1$. La figura 3.10 muestra un ejemplo de gráfica de disco unitario.

Definición 22 *La gráfica de disco unitario para un conjunto de vértices V, es aquella que contiene a los vértices V y a las aristas que unen a dos vértices distintos de V que se encuentran a una distancia menor o igual a la unidad.*

Esta gráfica suele confundirse con la gráfica de distancia unitaria la cual solo agrega las aristas cuyos vértices están a una distancia exactamente igual a la unidad. De la misma manera existen definiciones de esta gráfica donde se supone que los enlaces son estrictamente menores que la unidad con el objetivo de facilitar algunos cálculos y demostraciones. La gráfica de disco unitario generalmente no es plana y puede que tampoco sea conexa sin embargo en todos los trabajos, incluida esta tesis, se hace la suposición de que siempre es conexa.

La gráfica de disco unitario es un modelo válido para muchos casos prácticos bajo la suposición de un rango de comunicación fijo, pero cuando las aplicaciones manejan rangos de comunicación variable las gráficas se convierten fácilmente en gráficas generales en cuyo caso el modelo puede no ser tan exacto. Por esta razón es conveniente seguir buscando solución al ruteo en gráficas generales bajo las limitaciones impuestas y abarcar un rango de aplicaciones mayor.

3.10.2. Subgráficas planas

Los algoritmos anteriores solo trabajan con gráficas planas y hasta el momento no existen algoritmos de ruteo que trabajen sobre gráficas en general, aunque existen avances en esta dirección [11].

Ante la necesidad de soluciones que involucran el uso de gráficas no planas, como la gráfica de disco unitario, la solución es obtener una subgráfica plana a partir de la original, es decir *planarizar* la gráfica. Dicho de otra manera, se intenta usar una subgráfica de la gráfica original que contenga todos los vértices de esta y que además sea conexa y plana. Los algoritmos para obtener una subgráfica plana conexa de una gráfica son muy variados pero es difícil determinar cuales son los parámetros adecuados para evaluar la calidad de las subgráficas obtenidas.

La idea detrás de usar subgráficas planas es que un agente de memoria limitada pueda usarlas para desplazarse y realizar operaciones sobre los vértices de una gráfica no plana. Este esquema constituye la principal solución al problema del ruteo en gráficas geométricas porque permite aplicar los algoritmos de ruteo para gráficas planas en gráficas que no lo son.

Sin embargo se debe tomar en cuenta que existe un intercambio de desempeño cuando se usan subgráficas planas principalmente porque al reducir el número de conexiones o aristas se pierden enlaces directos y por lo tanto se incrementa el tráfico en otros enlaces.

Durante la planarización de una gráfica se determina cuales aristas serán utilizadas y cuales no, de esto dependerán las características de la subgráfica. Las características más deseadas para una subgráfica son las siguientes:

1. Conexa.
2. Competitiva con respecto a las rutas cortas.
3. Pocas aristas.
4. Fácil de rutear.

Algunas gráficas que cumplen con estas propiedades son las estructuras clásicas de la geometría y teoría de gráficas como por ejemplo el *árbol generador de peso mínimo*, que se forma con las aristas que conectan a todos los vértices de tal manera que la longitud total de las aristas usadas sea mínima con respecto a cualquier otra elección. Este árbol es por definición la estructura más ligera en cuanto al número de aristas pero puede dejar a algunos vértices demasiado alejados, es decir con rutas que no son competitivas.

3.10.3. Estructuras recubridoras

Las *estructuras recubridoras* son aquellas subgráficas que contienen todos los vértices de la gráfica original y son conexas, es decir que existe una ruta para cada par de vértices

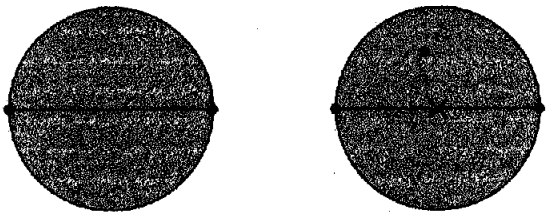

Figura 3.11: Prueba de Gabriel: si hay algún vértice dentro del círculo sombreado, la arista se elimina.

dentro de la estructura recubridora. Una estructura recubridora tiene un parámetro inherente a ella conocido como *factor de recubrimiento*, el cual permite evaluar la calidad de sus rutas y determinar que estructura recubridora es mejor.

Definición 23 *Para una gráfica $G(V, E)$, sea la distancia entre dos vértices $u, v \in V$ la longitud total de la ruta más corta entre ellos denotada por $d_G(u, v)$. Una gráfica* t-recubridora *denotada por $S(V, E')$ con $E' \subseteq E$, es aquella donde se cumple que para cada par $u, v \in V$ $d_S(u, v) \leq t \cdot d_G(u, v)$. Al valor de t se le conoce como* factor de angostura.

3.10.4. Gráfica de Gabriel

La gráfica de Gabriel fue introducida por K. Gabriel en [28]. La gráfica de Gabriel de una gráfica geométrica G se denota como $GG(G)$ y se construye aplicando *la prueba de Gabriel* a todas la aristas para determinar cuales serán eliminadas y cuales no. Esta prueba revisa el círculo centrado a la mitad de la arista y diámetro igual a la longitud de la arista. Sí este círculo contiene algún vértice de la gráfica la arista se elimina, sí no la arista se conserva, ver la figura 3.11.

Esta es una de las pruebas más importantes para la eliminación de aristas y planarización de gráficas. La gráfica de Gabriel es usada como estructura implícita para el algoritmo de ruteo por caras [35]. La importancia de esta gráfica se debe a que sí la gráfica original es conexa, su gráfica de Gabriel también lo será y además contendrá al árbol generador de peso mínimo.

3.10.5. Gráfica de vecindad relativa

La gráfica de vecindad relativa se usa para la difusión de mensajes en gráficas por el algoritmo usado en [31]. Esta gráfica se obtiene al aplicar la *prueba de la vecindad relativa* a un conjunto de vértices y determina cuales son los vecinos más cercanos. Esta gráfica se denota por $RNG(G)$ y es un concepto propuesto por Toussaint [18]. La prueba consiste en conservar todas las aristas de la gráfica cuya intersección de los círculos centrados en los vértices de la arista y de radio igual a la longitud de la arista, no contengan ningún otro vértice de la gráfica, la figura 3.12 ilustra la prueba.

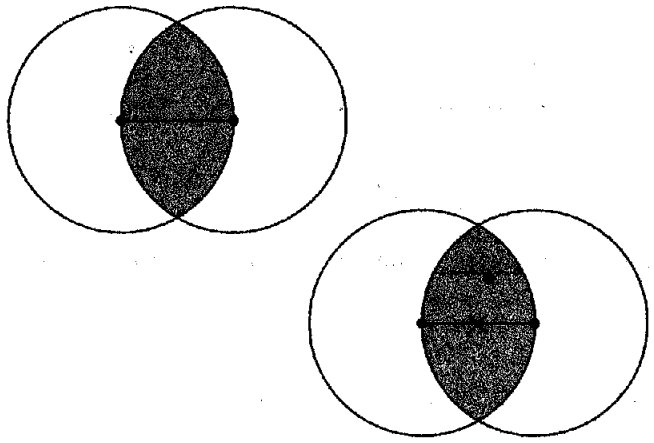

Figura 3.12: Prueba de vecindad relativa: sí hay algún vértice dentro de la luna sombreada la arista se elimina.

A diferencia de la gráfica de Gabriel el área que se revisa durante esta prueba es más grande y por lo tanto no contiene aristas que son conservadas por la gráfica de Gabriel, dando como resultado que la gráfica de vecindad relativa sea una subgráfica de la gráfica de Gabriel. Esta gráfica también contiene al árbol generador de peso mínimo.

3.10.6. Estructuras de pozo

Debido a la naturaleza minimalista de los problemas que aquí se manejan es deseable que la mayoría de los parámetros de la gráfica se mantengan acotados por una constante, tal es el caso del grado de los vértices. En el modelo de la gráfica de disco unitario los dispositivos pueden conectarse con todos los dispositivos dentro del rango de comunicación, por lo tanto el número de enlaces posibles es proporcional al número total de dispositivos. De esta manera la cantidad de recursos asociados a cada enlace también pueden llegar a ser lineales por cada vértice, es decir cuadráticos en total.

Las *estructuras de pozo* son una familia de estructuras geométricas que tienen grado acotado en sus vértices, aunque desafortunadamente también pueden resultar ser gráficas no planas. Las estructuras de pozo fueron diseñadas durante el trabajo con redes inalámbricas pensando en que un número de aristas acotado se traduce en un número de canales de comunicación fijo lo cual mejora el diseño de los dispositivos haciéndolos más económicos y portables.

La gráfica Yao es un ejemplo de estructura de pozo que se define en base a un entero k mayor o igual a 6 de la siguiente manera:

1. Para cada vértice dividir el espacio en k conos de ángulos iguales y centrados en el vértice.

2. Por cada cono se agrega una arista que va de la punta del cono al vértice más cercano dentro del cono, si existe.

La gráfica Yao se denota como $Yao_k(G)$ y es muy parecida a la gráfica θ que en lugar de tomar el vértice más cercano toma el vértice que tenga la proyección más cercana al eje del cono.

La estructura de pozo propuesta en [46] usa una técnica de [44] para obtener una estructura de pozo plana con grado acotado de forma local. Esta estructura combina las ventajas de la estructura de grado acotado y la planaridad.

1. Para cada vértice se calculan sus aristas Yao.
2. Los vértices adyacentes a estas aristas ya no se toman en cuenta.
3. Por cada vértice adyacente se calculan sus aristas Yao tomando en cuenta solo los vértices en su *cono*. Se repite el proceso hasta que ya no hay vértices que unir con aristas.

Existen muchas otras estructuras planas recubridoras, algunas de ellas se forman mediante la aplicación de diferentes pruebas, por ejemplo aplicar la estructura Yao a la gráfica de Gabriel o la gráfica de Gabriel a la de Yao, etc. Cada prueba le proporciona a la gráfica final una característica, por ejemplo la gráfica Yao tiene grado acotado pero no es plana y la gráfica de Gabriel si es plana, de esta manera la gráfica Yao-Gabriel será plana y con grado acotado.

3.10.7. Prueba Morelia

Las estructuras anteriores se construyen usando solo información local a cada vértice para determinar la eliminación sistemática de aristas que ocasionan cruces o son potencialmente propensas a crearlos. Por lo tanto es posible que existan aristas que son eliminadas de forma innecesaria provocando que algunas rutas contengan más aristas ó saltos de los necesarios. Además de que las rutas se alargan, algunos vértices serán más transitados que otros causando un mayor uso de recursos en ellos.

La prueba Morelia fue introducida en [33] y determina si es o no necesaria la eliminación de una arista verificando que no ocasione cruces. Esta prueba tiene tres objetivos:

1. **Mantener las aristas largas.** Estos enlaces evitan tener rutas con muchos saltos.
2. **Obtener una gráfica plana.** Para poder aplicar otros algoritmos.
3. **Usar información local.** Naturalmente debido a las restricciones.

Mantener aristas largas es un punto controvertido porque estas aristas requieren mayores recursos de comunicación y energía. La intención de utilizarlas es sacrificar un poco los recursos en favor de evitar que exista más tráfico sobre la red debido al efecto *multi salto*.

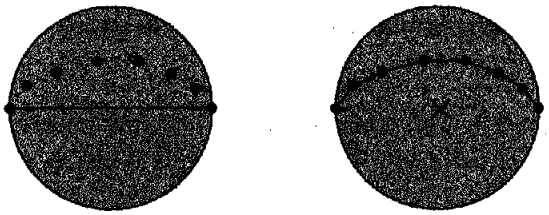

Figura 3.13: Efecto multi salto: al eliminar la arista mediante la prueba de Gabriel la ruta que conecta a los vértices de la arista tiene muchos saltos o vértices intermedios.

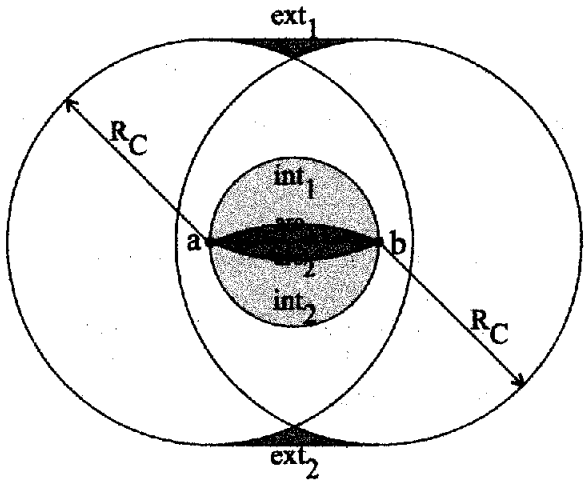

Figura 3.14: Las áreas sombreadas son las que se verifican en la prueba Morelia.

3.10.7.1. Efecto multi salto

El efecto multi salto ocurre cuando al eliminar una arista que no ocasiona cruces se obtiene una ruta con más vértices intermedios entre los vértices de la arista que se eliminó. La gráfica de Gabriel y la de la vecindad relativa tienen este problema como lo muestra la figura 3.13 donde se aplica la prueba de Gabriel.

3.10.7.2. Áreas de prueba

La prueba Morelia utiliza seis áreas definidas por: un círculo centrado en la mitad de la arista y de diámetro igual a la mitad de la longitud de la arista, dos arcos de radio R_c que van de un extremo de la arista a otro y dos círculos de radio R_c centrados en cada vértice de la arista, la figura 3.14 muestra estas regiones.

Los arcos definen las regiones arc_1 y arc_2, el resto del círculo forma las regiones int_1 e int_2 y por último la tangente horizontal de los círculos forman las regiones ext_1 y ext_2. El

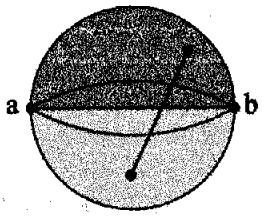

Figura 3.15: **Cruce central :** Se buscan cruces dentro del círculo de diámetro (a, b).

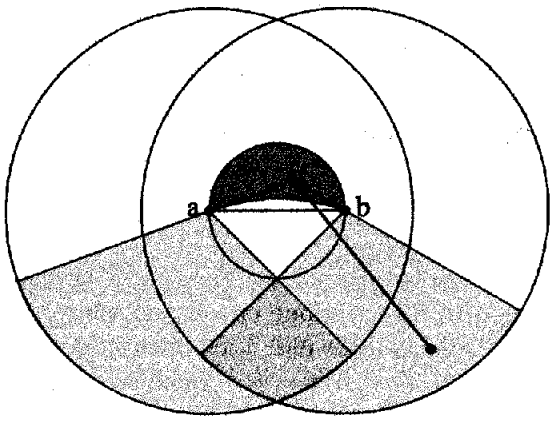

Figura 3.16: **Cruce alto :** Se buscan cruces de $L(a) \cup L(b)$ a la parte alta del medio círculo de diámetro (a, b).

subíndice 1 es para las regiones por encima de la arista y el 2 para las regiones por debajo de esta.

3.10.7.3. La prueba

La prueba Morelia consiste en la aplicación de las siguientes tres reglas a una arista (a, b):

1. **Cruce central.** Si existe al menos un vértice en el medio círculo por encima del segmento (a, b) y al menos un vértice en el medio círculo de la parte baja, la arista (a, b) se elimina. Ver figura 3.15.

2. **Cruce alto.** Si la regla anterior no eliminó la arista (a, b) entonces uno de los medios círculos está vacío, sin pérdida de generalidad se supone que fue el de abajo. Si existe al menos un vértice en la zona int_1, se verifica que no existan vértices en $L(a) \cup L(b)$ que ocasionen cruces, si los hay la arista ab se elimina. Ver figura 3.16.

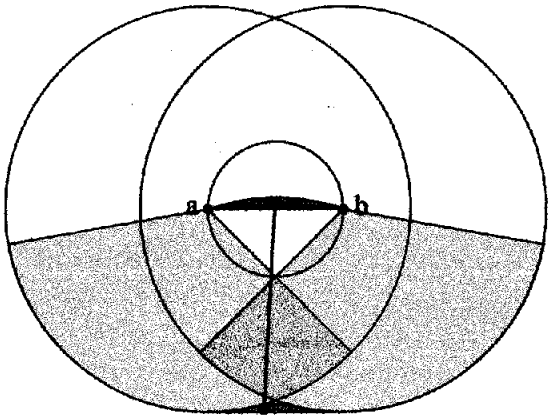

Figura 3.17: **Cruce bajo :** Se buscan cruces de $L(a) \cup L(b)$ a la parte baja del medio círculo de diámetro (a, b).

3. **Cruce bajo.** Si aún no se elimina la arista (a, b) verificar que si existe al menos un vértice en la zona arc_1 que ocasione cruces con vértices en $L(a) \cup L(b)$, si los hay la arista ab se elimina. Además se le pide a los vértices en arc_1 que busquen en su vecindad vértices en la zona ext_2, si existen cruces también se elimina (a, b). Ver figura 3.17.

Teorema 3.10.1 *Al aplicar la prueba Morelia a todas las aristas de una gráfica geométrica conexa G, la subgráfica G' resultante es conexa y plana.*

Prueba 3.10.1 Conexidad. *La prueba de Gabriel elimina una arista si existe algún punto dentro del círculo central de la prueba Morelia, por lo tanto la primera, segunda y tercera prueba conservan todas y cada una de las aristas que se conservan durante la aplicación de la prueba de Gabriel, por lo tanto la gráfica G' contiene a la gráfica de Gabriel y como esta última es conexa G' también lo es.*

Planaridad. *Supóngase la existencia de un cruce en G' determinado por las aristas (a, b) y (c, d). La primera regla de la prueba Morelia impide que algún vértice de la arista (c, d) esté dentro del círculo de diámetro (a, b) centrado en la mitad del segmento (a, b), por lo tanto la arista (c, d) tiene una longitud mayor que (a, b).*

Pero la primera regla también impide que algún vértice de la arista (a, b) esté dentro del círculo de diámetro (c, d) centrado en la mitad del segmento (a, b), por lo tanto la arista (a, b) tiene una longitud mayor que (c, d), lo cual es una contradicción y tal cruce no puede existir debido a la aplicación de la primera regla de la prueba Morelia. Luego entonces la gráfica G' es plana. □

Las operaciones para detectar el cruce en la primer y segunda regla se realizan localmente de forma sencilla a un costo computacional de $O(1)$ con memoria constante. Sin embargo la regla tres requiere mandar mensajes a los vértices en la zona arc lo que representa

una sobre carga a pesar de esto la prueba se realiza con un número constante de saltos y por lo tanto con información local.

Sin embargo esta última regla tiene una probabilidad muy baja de aplicarse. El tamaño máximo de la zona *arc* se obtiene cuando la arista (a, b) tiene longitud igual a R_c, esta área es $\frac{\pi}{6}{R_c}^2 - \frac{\sqrt{3}}{4}{R_c}^2 = \frac{2\pi-3\sqrt{3}}{12}{R_c}^2$. Por otro lado el medio círculo de (a, b) tiene un área de $\frac{\pi}{2}(\frac{R_c}{2})^2 = \frac{\pi}{8}{R_c}^2$, esto indica que el área *arc* es $\frac{\frac{2\pi-3\sqrt{3}}{12}{R_c}^2}{\frac{\pi}{8}{R_c}^2} = \frac{4\pi-6\sqrt{3}}{3\pi} < 0{,}24$ más chica que el medio círculo, por lo tanto la probabilidad de que un vértice aparezca allí es muy baja.

Así pues, la aplicación de la prueba Morelia a las aristas de una gráfica permite a un agente móvil ver una subgráfica plana conexa que no tiene el efecto multi salto. El único inconveniente es que se paga un costo adicional en la comunicación muy bajo (tercera regla).

3.11. Conclusión

Los resultados anteriores [12, 35, 38, 20, 33, 11, 31, 46] han demostrado que el ruteo es posible en términos reales y prácticos en ambientes limitados con recursos limitados gracias a la ayuda de la información geométrica. También es claro que a medida que estas restricciones llegan al extremo se pierde generalidad en las soluciones, es decir que solo funcionan para gráficas específicas como por ejemplo las triangulaciones y los algoritmos *sin memoria.*

La idea en la solución de estos problemas es asignar la menor cantidad de memoria posible a los algoritmos sin que el desempeño se vea afectado, en el caso del ruteo sobre gráficas geométricas planas esto se logra con una cantidad constante de memoria $O(1)$ usando el *ruteo por caras.* Este algoritmo sirve de base para otros algoritmos ya sea que se le modifique o que se use como paso intermedio. Sin embargo, no existe un algoritmo que encuentre la ruta más corta para cualesquiera dos vértices en una gráfica plana bajo las condiciones planteadas.

A pesar de su importancia el ruteo solo es una muestra de lo que se puede lograr con una aproximación geométrica y, aunque es un tema que tiene muchos problemas abiertos en este momento, sirve de base para otras aplicaciones como se podrá ver en el siguiente capítulo.

De esta manera las contribuciones expuestas contextualizan los resultados propios pues son las primeras piedras en la construcción de un área que promete tener mucho futuro por delante y en la cual está desarrollada esta tesis.

Capítulo 4

Resultados

Ante la problemática en las redes inalámbricas *ad hoc*, donde no existe una infraestructura de comunicación fija, la aproximación geométrica ha demostrado aportar soluciones interesantes, útiles y prácticas. Los algoritmos presentados hasta el momento [11, 12, 23, 32, 33, 34, 35, 38, 40, 46, 47, 15, 19, 21] fueron ideados para trabajar con información geométrica bajo el modelo de agentes móviles de memoria limitada. El modelo de agente por sí solo tiene ventajas como, por ejemplo, un menor número de enlaces en el proceso de comunicación y un entorno de programación estándar[15, 19]. La aproximación geométrica agrega escalabilidad, eficiencia, sencillez y flexibilidad gracias al uso de memoria constante y que se resuelven problemas globales son solamente información local.

Estos resultados motivaron la investigación de soluciones a problemas en gráficas geométricas que no hubieran sido resueltos bajo las mismas condiciones, es decir, información geométrica disponible, memoria constante, información local, no alterar los vértices y agentes móviles.

Al final se han obtenido una serie de algoritmos capaces de resolver los siguientes problemas: el barrido de línea, la conexidad por vértices y por aristas, 6-coloración, el cálculo del árbol generador de peso mínimo, la distancia entre cualesquiera dos vértices de un árbol, la distancia máxima de un vértice en un árbol y el centro de un árbol. Todos estos problemas utilizan un total de $O(n)$ unidades de memoria y sus complejidades temporales varian desde $O(l \cdot n)$, donde l es la longitud de la ruta que se busca, hasta $O(n^8)$ para la verificación de la 6-conexidad. Lo realmente interesante de estos problemas es que se pueden resolver, aunque es obvio que hace falta trabajar para reducir la complejidad temporal en los casos donde esto sea posible.

Los problemas resueltos se presentan de forma real en las redes de comunicación en general y particularmente en las redes *ad hoc*. El mantenimiento de la red, el monitoreo de actividades ó los servicios de seguridad son un ejemplo de los problemas que resuelven los algoritmos, pero se pueden encontrar más incluso en otras áreas.

A continuación se resumen las condiciones bajo las cuales se desarrolláron los algoritmos:

- **Memoria constante.** La cantidad de memoria disponible para el agente que se desplaza por la gráfica es constante con respecto al tamaño de la entrada (número de vértices), es decir que en términos generales solo puede recordar unos cuantos datos, como coordenadas, contadores, etc.
- **Información geométrica local.** Los vértices son capaces de obtener su posición (x, y) y la de los vecinos dentro de un rango de comunicación limitado y constante.
- **No alterar la gráfica.** El vértice solo puede almacenar al agente junto con sus datos (todo en conjunto tiene un tamaño constante) y proporcionarle sus coordenadas y las de los vecinos. El agente no puede almacenar variables dentro del vértice para después revisarlas cuando vuelva a pasar.

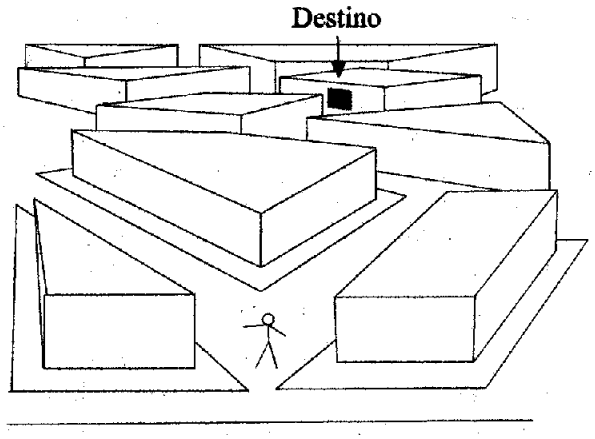

Figura 4.1: El turista no es capaz de ver su destino.

4.1. Recorrido de gráficas geométricas

Imaginando de nuevo al turista perdido en la ciudad surge la pregunta ¿Qué pasa si no puede ver desde lejos el lugar a donde va, como en el caso de la figura 4.1? Tal vez dispone de una fotografía y solo sea capaz de identificar el lugar si pasa por enfrente de él. En este caso tampoco puede preguntar u obtener información más allá de sus propios sentidos, así como tampoco puede dejar marcas por la ciudad o llevar un mapa y su memoria sigue siendo muy mala, ¿será posible que encuentre el lugar?.

En general el problema al que se hace referencia es al de la localización, el cual se refiere al hecho de que para mandar un mensaje el origen necesita conocer de antemano la dirección del destino. Las soluciones van desde consultar un servidor de nombres/direcciones hasta mandar el mensaje a todos los participantes (incluido el destino por supuesto).

Los agentes dentro de las gráficas geométricas se enfrentan a este mismo problema cuando pretenden realizar tareas como el ruteo, aunque en un principio se hace la suposición de que el usuario del agente conoce la ubicación (x, y) del destino.

La solución propuesta es que el agente *busque* el nodo destino mediante un recorrido de todos y cada uno de los vértice de la red. Esta misma búsqueda puede ser aprovechada por otros algoritmos, por ejemplo aquellos que necesitan aplicar alguna operación sobre todos los vértices. La solución concuerda con la situación real de la redes inalámbricas pues las posiciones de los vértices cambian constantemente y no es posible implementar una solución centralizada como el servidor de nombres.

El algoritmo presentado no solo recorre los vértices sino también todas las aristas y todas las caras de la gráfica de tal manera que en principio de pueden aplicar operaciones o búsquedas sobre todas las partes de la gráfica.

Definición 24 *Un recorrido de la gráfica $G = (V, E)$ reporta una sola vez todos y cada uno*

de los elementos de la gráfica, es decir vértices, aristas y caras.

En la literatura se encuentran varios algoritmos de recorrido, entre los que destacan *búsqueda primero en profundidad* y *búsqueda primero en amplitud*[45], pero las restricciones de memoria y la naturaleza dinámica del problema hacen sencillamente imposible aplicar directamente estas técnicas. No obstante, es posible realizar recorridos bajo el modelo de memoria limitada y dicho sea de paso este es uno de los resultados más importantes en el área porque la complejidad del algoritmo puede ser reducida a un tiempo de $O(n \log n)$, considerado como óptimo dadas las restricciones [34].

En general, los recorridos de gráficas pueden ser usados para resolver el problema del ruteo, pues es obvio que el recorrido da como resultado una ruta que pasa por todos los vértices. A pesar de que el sentido común indica que es una solución poco práctica, en la realidad no existen soluciones para el ruteo en gráficas planas en general pero si para el recorrido por lo que resulta atractivo usarlas mientras se descubren otras alternativas.

4.1.1. Arista de entrada

El método fue propuesto en [35] y está fundamentado en el concepto de *arista de entrada*, la cual es una referencia que se aprovecha para recorrer sistemáticamente la gráfica. La arista de entrada se define a partir de una cara y es única, por lo tanto la gráfica tiene tantas aristas de entrada como caras, incluida la cara exterior.

Definición 25 *Dada una cara, su arista de entrada es aquella que:*

- *Contiene el vértice más a la derecha, es decir el que tiene la coordenada x más grande. En caso de empate, se toma el vértice más bajo, es decir el que tiene la coordenada y más chica.*
- *Es la arista siguiente a este vértice recorriendo la cara en el sentido de las manecillas del reloj.*

La figura 4.2 muestra un ejemplo que ilustra la definición.

Lema 4.1.1 *La arista de entrada para una cara es única y se puede encontrar en un tiempo $O(n)$, donde n es el número de aristas de la cara.*

Prueba 4.1.1 *Supóngase que existen dos aristas de entrada distintas (u, v) y (z, w) con los vértices en el orden de las manecillas del reloj. Esto quiere decir que tanto u como z tienen la misma distancia al eje x (primera línea) y por lo tanto fue necesario aplicar la segunda regla y resultó que tanto u como z eran los más cercanos al eje y. Como la gráfica es plana u tiene que ser igual a z. Recorriendo la gráfica en el sentido de las manecillas del reloj solo es posible encontrar uno y solo un vértice siguiente a $u = z$, por lo tanto $v = w$ y las dos*

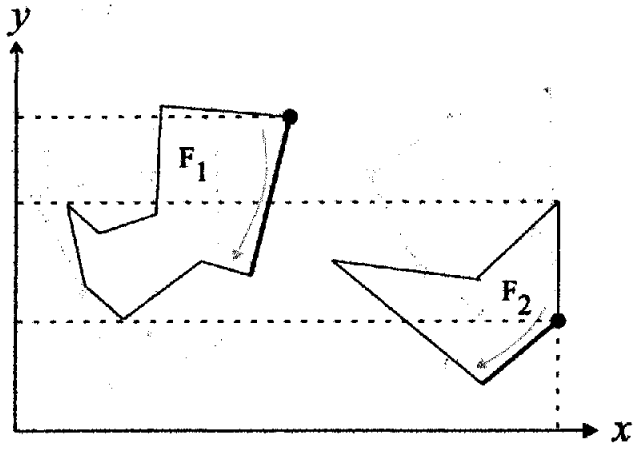

Figura 4.2: Aristas de entrada para F_1 y F_2 y su relación con los ejes x y y.

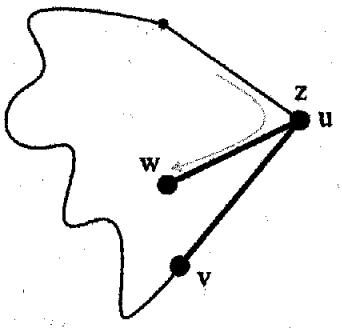

Figura 4.3: Por la primera condición de la arista de entrada $u = z$. Siguiendo el orden de las manecillas del reloj solo puede existir una arista de entrada.

aristas (u, v) *y* (z, w) *son iguales, es decir que solo existe una arista de entrada, ver figura 4.3.*

Usando la regla de la mano derecha se recorre la cara hasta recorrer todos los n *vértices, en una vuelta se calcula el vértice más a la derecha y más abajo. En una segunda vuelta por los* n *vértices se localiza la arista de entrada, por lo tanto en un tiempo total de* $O(n)$ *y con memoria constante se ubica la arista de entrada pues solo es necesario recordar el valor de las coordenadas* (x, y) *de un solo vértice.*

Lema 4.1.2 *Sea* E_e *el conjunto de las aristas de entrada de una gráfica* G*. Entonces* $G - E_e$ *es un árbol.*

Prueba 4.1.2 *Supóngase que la gráfica* $G - E_e$ *no es un árbol y que por lo tanto contiene*

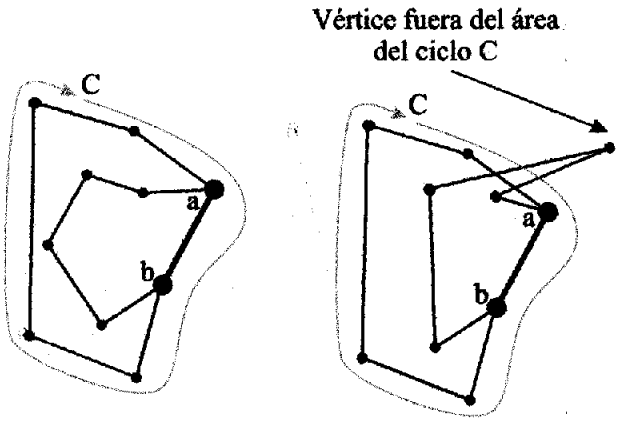

Figura 4.4: La arista de entrada del ciclo C es (a, b), la cual es arista de entrada de su propia cara. A la derecha se muestra que de no ser así la gráfica no sería plana.

un ciclo C compuesto por aristas que no son de entrada. Como la gráfica es plana el ciclo C encierra a todos los vértices de las caras dentro del ciclo.

Es posible identificar la arista de entrada del ciclo C, denotada por E_c, aplicando el criterio de la arista de entrada. Esta arista contiene al vértice más a la derecha de todo el ciclo C y de todos los vértices dentro del ciclo. La arista E_c pertenece a una cara de G y se supone que no es arista de entrada, pero si no lo es entonces debería de existir otra arista de la misma cara que estuviera más a la derecha, pero esto no es posible porque el ciclo C limita las caras de las aristas que contiene, por lo tanto la arista E_c tiene que ser de entrada, no pertenece a $G - E_e$ y esta gráfica no tiene ciclos. Ver figura 4.4.

Con este resultado lo único que necesita un agente para recorrer el árbol $G - E_e$ es determinar si una arista es o no de entrada. El árbol $G - E_e$ no tiene un vértice especial como raíz por lo que el agente puede iniciar su recorrido en cualquier vértice. La figura 4.5 muestra la gráfica con las aristas de entrada sombreadas.

4.1.2. El recorrido

El algoritmo 11 muestra como un agente puede recorrer el árbol que queda al no tomar en cuenta a las aristas de entrada de una gráfica plana, es decir $G - E_e$. Al mismo tiempo que se recorre la gráfica se pueden realizar búsquedas o reportar los elementos de la gráfica, de tal manera que se puede aplicar en muchos otros problemas.

La parte importante del algoritmo está centrada en la identificación de las aristas de entrada de cada cara. El algoritmo no necesita de una pila o una cola para realizar el recorrido en profundidad. Esto es posible porque al entrar en una cara no es necesario recordar por donde llegó, simplemente continúa recorriendo la gráfica y eventualmente atravesará en sentido inverso la arista de entrada.

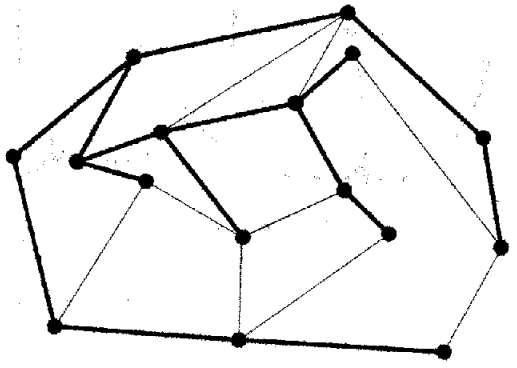

Figura 4.5: Gráfica menos aristas de entrada, $G - E_e$.

Algoritmo 11 Recorrido de gráficas geométricas

1: Iniciar en cualquier vértice.
2: De todas las arista incidentes, no tomar en cuenta las aristas de entrada y desplazarse por cualquiera de las restantes.
3: Recordar la arista que se acaba de visitar.
4: De todas las arista incidentes, no tomar en cuenta las aristas de entrada y desplazarse por la arista siguiente a la que se acaba de visitar, en el sentido de las manecillas del reloj.
5: Terminar cuando se visite el primer vértice por i-ésima vez, donde i es el grado del vértice.

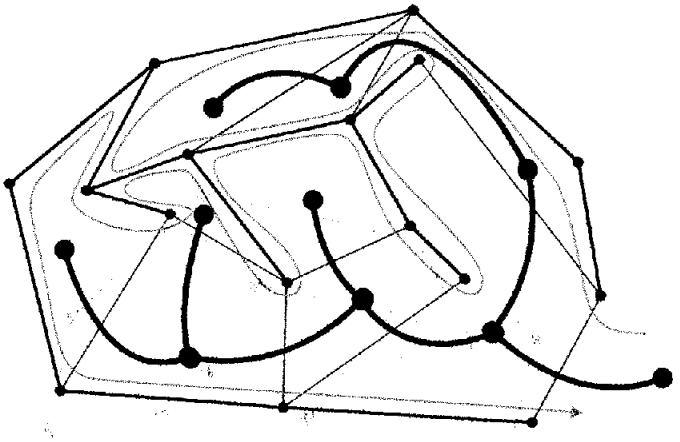

Figura 4.6: Árbol de caras. Las aristas de entrada determinan la forma de árbol.

De esta forma la arista de entrada funciona como una especie de *marca virtual* en la cara y le permite al agente ubicarse y decidir el camino por donde continuará su búsqueda o recorrido.

En principio es necesario recorrer toda la cara para encontrar la arista de entrada. Este procedimiento da como resultado global un total de $O(n^2)$ verificaciones, sin embargo existe una modificación que ofrece mejor desempeño. Al observar que las caras forman ciclos o anillos, se pueden aplicar algoritmos de elección de líder como el que se encuentra en [10], de donde el tiempo para encontrar a la arista de entrada se puede reducir hasta $O(n \log n)$.

Otra estructura importante que se forma a partir del concepto de arista de entrada es el llamado *árbol de caras*[38]. Este árbol tiene un vértice por cada cara de la gráfica incluida la cara exterior. El vértice asociado a la cara exterior es la raíz del árbol. Por cada arista de entrada el árbol de caras tiene una arista entre los vértices de las caras que contienen a la arista de entrada, es decir que si dos caras contienen una arista de entrada, los vértices que representan estas caras estarán unidos en el árbol de caras.

Cuando se recorre la gráfica $G - E_c$ con el algoritmo 11 el árbol de caras de G es visitado como en la búsqueda en profundidad. La figura 4.6 muestra un ejemplo donde se puede observar el recorrido de un agente que sigue este método y el árbol de caras de la gráfica.

4.1.3. Recorrido de gráficas no planas

El recorrido mediante eliminación de aristas de entrada es eficaz para gráficas geométricas planas, pero ¿será posible recorrer gráficas que no sean planas? la respuesta es sí, pero solo en cierto tipo de gráficas conocidas como *gráficas cuasi planas*[11]. Este tipo de gráficas tienen permitidos cruces entre aristas pero deben de cumplir ciertas condiciones. La figura 4.7 muestra un ejemplo de gráfica cuasi-plana.

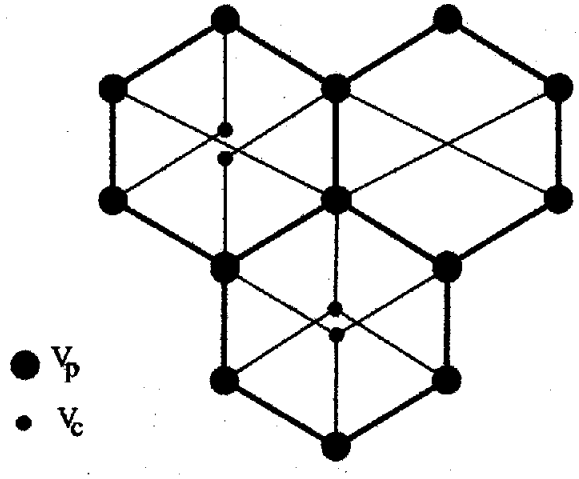

Figura 4.7: Gráfica cuasi-plana.

Definición 26 *Sea $G = (V, E)$ una gráfica geométrica en el plano R^2 tal que:*

- $V = V_p \cup V_c$,
- V_p *induce una gráfica conexa plana* P,
- *la cara exterior de* P *no contiene vértices de* V_c *ó aristas de* $G - P$,
- *las aristas de* P *no son cruzadas por otras aristas de* G.

Existe un tipo especial de gráficas cuasi planas que cumplen con una sencilla regla, la regla del vecino izquierdo.

Definición 27 *Una gráfica cuasi-plana $G = (V_p \cup V_c, E)$ cumple la regla del vecino izquierdo sí cada vértice $v \in V_c$ tiene un vértice vecino u tal que $xcor(u) < xcor(v)$.*

El algoritmo de recorrido de gráficas solo funciona sobre gráficas cuasi planas que cumplan con la *regla del vecino izquierdo*. Una gráfica cuasi plana tiene en P a una subgráfica plana subyacente y son las caras de P las que sirven de apoyo para el recorrido. Las gráficas cuasi planas utilizan el concepto de cara de una manera diferente pues las *cuasi caras* son ciclos definidos por operaciones básicas como lo son las operaciones geométricas $sig(e)$ y $pre(e)$.

Definición 28 *Se le llama cuasi-cara al ciclo formado por las aristas $e_i e_{i+1} \ldots e_n$, tales que:*

$$e_{i+1} = sig(e_i)$$

para cualquier arista $e_i \in E$, de una gráfica $G(V, E)$.

Esta definición de cuasi-caras es compatible con la noción de cara en gráficas planas y también contienen una arista de entrada. Para poder definir la arista de entrada de cualquier cuasi cara es necesario definir dos tipos de vértices: vértice derecho y vértice izquierdo. Identificando la arista de entrada ya es posible recorrer la gráfica con el algoritmo de recorrido visto anteriormente.

Definición 29 *Para una arista $e = (u, v)$ su vértice* derecho *se define como el vértice u tal que $(xcor(u), ycor(u)) \leq (xcor(v), ycor(v))$ y el vértice izquierdo v se define como aquel que no es el derecho.*

Esta definición es equivalente a decir que el vértice derecho es el que está más cerca del eje x y en caso de empate el que está más cerca del eje y.

Los parámetros que definen a la arista de entrada son:

- Coordenada x del vértice izquierdo de la arista.
- Coordenada y del vértice izquierdo de la arista.
- Ángulo de la arista con respecto al rayo vertical que pasa por el vértice izquierdo.
- Coordenada x del vértice derecho de la arista.
- Coordenada y del vértice derecho de la arista.

La prueba sobre el orden total que inducen estos parámetros se encuentra en [11]. Una vez identificadas las aristas de entrada el algoritmo de recorrido es el mismo y el procedimiento para verificar a la arista de entrada también, lo único que cambia son los parámetros para definir a la arista de entrada.

4.2. Enumeración efectiva de vértices y aristas

En la mayoría de los problemas en redes y en general en las comunicaciones es necesario conocer la identidad de los vértices que participarán en la solución, por ejemplo en el ruteo se debe conocer de antemano el destino. Esta suposición puede no ser tan sencilla de cumplir debido a que solo se puede estar seguro de los datos inmediatos o locales. De la misma manera, para calcular las rutas hacia todos los demás vértices, es necesario saber o conocer quienes y cuantos son los demás vértices. El problema surge porque en el caso práctico no se conocen las posiciones de todos los vértices, solo las locales.

Una solución es asignar un nombre o identificador a cada vértice y arista de la gráfica de forma única. Este proceso se conoce como *enumeración efectiva* de vértices y aristas. La forma centralizada de hacerlo es mediante el uso de bases o puntos que contienen las identidades de cada participante en la red por ejemplo una central telefónica. Sin embargo en este caso la solución centralizada no es viable.

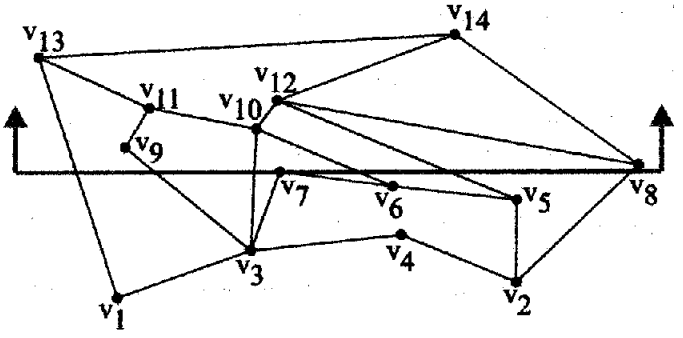

Figura 4.8: Barrido de línea y la enumeración que proporciona a los vértices.

Definición 30 *La enumeración efectiva de vértices y aristas es un mapeo que a cada vértice y arista de una gráfica $G(V, E)$ les asigna un número natural, de tal manera que se puedan enumerar en las secuencias $v_1, v_2, \ldots, v_{|V|}$ para vértices y $e_1, e_2, \ldots, e_{|E|}$ para aristas.*

Si un agente fuera capaz de conocer o calcular el número que le corresponde a cada vértice podría fácilmente realizar operaciones sobre cada uno de los vértices en un orden dado y sin necesidad de tener información global.

4.2.1. El barrido de línea

El recorrido de una gráfica mediante el uso del concepto de arista de entrada establece una enumeración efectiva de los vértices debido al orden en que se visita el árbol de caras de la gráfica.

Para el algoritmo que aquí se expone el orden de los vértices está dado por un concepto más familiar y que tiene que ver con las coordenadas de cada vértice. De esta manera se pretende obtener un algoritmo de enumeración efectiva basado en un criterio ordenado más natural. Específicamente esta basado en la técnica conocida como *barrido de línea*[45]. Esta técnica se usa ampliamente en geometría computacional para resolver una cantidad importante de problemas y consiste en tomar una línea recta (generalmente horizontal) y desplazarla poco a poco por el plano de manera ordenada y en una sola dirección, deteniéndose para realizar alguna operación sólo cuando es necesario, que en este caso es la enumeración efectiva, ver figura 4.8.

La línea de barrido cubre todos los vértices y es colocada por debajo del vértice más bajo de la gráfica de tal manera que al ir subiéndola poco a poco se intersecte con cada uno de los vértices en el orden de sus coordenadas y. De esta forma el primero en tocar la línea será el vértice más bajo y se le asignará el número uno, obviamente el vértice más alto será el último en tocar la línea y por lo tanto tendrá el número $n = |V|$. Si existen dos o más vértices con la misma coordenada y, la enumeración se realizará de izquierda a derecha sobre

Algoritmo 12 Enumeración efectiva de vértices

1: Iniciar en cualquier vértice.
2: Usar ruteo por caras para llegar a la cara externa.
3: Recorrer toda la cara externa hasta llegar al vértice más bajo, es decir el vértice $n = 0$.
4: Recorrer todas la caras que intersectan la línea horizontal (línea de barrido) que pasa por el vértice n (el actual).
5: Encontrar en este recorrido al vértice con la coordenada y siguiente, este será el vértice $n + 1$.
6: Enumerar de derecha a izquierda a las aristas incidentes que estén por encima de la línea horizontal.
7: Repetir desde el paso 4 hasta enumerar todos los vértices y aristas.

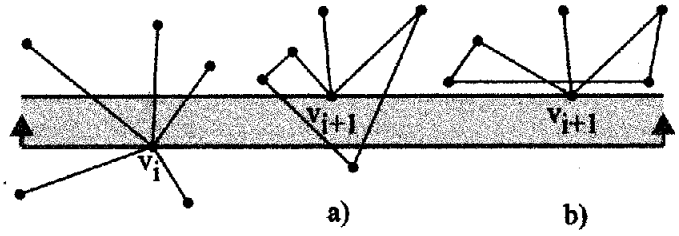

Figura 4.9: La cara inferior que contiene a v_{i+1} intersecta a la línea de barrido que pasa por v_i (a), si esto no fuera cierto el vértice o arista de la cara tendría que pasar por encima de v_{i+1} porque la zona sombreada esta vacía, lo cual es imposible porque son gráficas planas.

la línea. El algoritmo 12 realiza este procedimiento usando solo información local y memoria constante.

Teorema 4.2.1 *El algoritmo de enumeración efectiva realiza una enumeración efectiva de la gráfica G en un tiempo de $O(n^3)$ usando una cantidad de memoria constante, es decir $O(1)$.*

Prueba 4.2.1 *Se hace la suposición de que el siguiente vértice en la enumeración v_{i+i} se encuentra en alguna de las caras que intersectan con la línea horizontal (la línea de barrido) que pasa por la coordenada y del vértice anterior, es decir por $y(v_i)$.*

Suponiendo que el vértice v_{i+1} no está en estas caras, todos los vértices de las caras que pasan por debajo de v_{i+1} tendrán coordenadas y mayor que v_{i+1}. Pero esto no puede ser porque debe de existir una arista o vértice que pase por abajo de v_{i+1} para poder cerrar la cara y como no puede haber vértices más abajo que v_{i+1} ó más arriba que v_i la arista o vértice de esta cara está por debajo de la línea horizontal que pasa por v_i y por lo tanto intersecta a una de las caras que contienen a v_{i+1}, ver figura 4.9.

Para encontrar el vértice siguiente v_{i+1} se avanza sobre la línea de barrido horizontal de izquierda a derecha recorriendo todas las caras incluyendo la exterior, este procedimiento

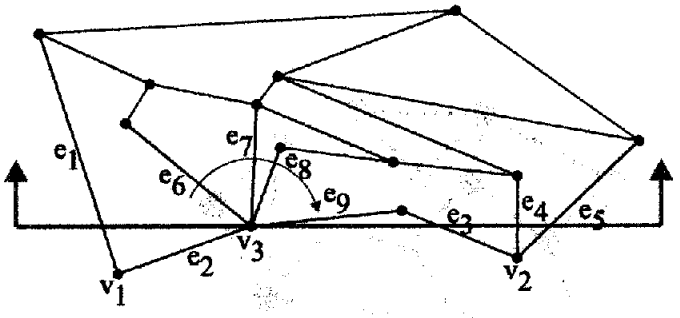

Figura 4.10: En cada vértice se enumeran las aristas en el orden descrito.

tarda $O(n^2)$ en el peor de los casos. Para alcanzar la cara exterior se lleva un tiempo de $O(n)$ y para subir de v_i a v_{i+1} se lleva $O(n^2)$, como son n vértices, el tiempo total es de $O(n^3)$, con memoria $O(1)$.

La enumeración de las aristas se hace cada vez que se enumera un vértice siguiendo el sentido de las manecillas del reloj para todas las aristas que están por encima de la línea de barrido, ver figura 4.10.

Este procedimiento permite visitar todos y cada uno de los vértices y aristas de la gráfica, de tal manera que se pueden aplicar procedimientos como el mantenimiento, el monitoreo, la búsqueda y otros. Al mismo tiempo es una aplicación directa del concepto de barrido de línea que es base para otros procedimientos en geometría computacional como por ejemplo la detección de cruces.

La enumeración efectiva ofrece la ventaja de poder identificar a los vértices mediante un orden natural y hace posible operaciones que involucren a uno o más vértices sin necesidad de tener conocimiento previo de sus posiciones.

4.3. Conexidad

Una de las características más deseadas para una red de comunicación es que sea tolerante a las desconexiones, es decir que siga funcionando a pesar de que algunos nodos o aristas dejen de funcionar o participar. Por ejemplo, en la red de la figura 4.11, la eliminación del vértice v no afecta la comunicación porque siguen existiendo rutas alternativas para comunicar los puntos cuyas rutas pasaban por v. Sin embargo, en la figura 4.12 los vértices v, w y z son indispensables para mantener la comunicación total y no pueden ser eliminados. Al igual que los conjuntos de vértices, existen conjuntos de aristas que resultan indispensables.

Pero ¿de qué sirve saber cuales son estos elementos importantes, si de todos modos van a fallar?. La importancia radica en el enfoque preventivo que se le pueda dar a la información, es decir tratar de no tener actividades que propicien el fallo de las aristas y

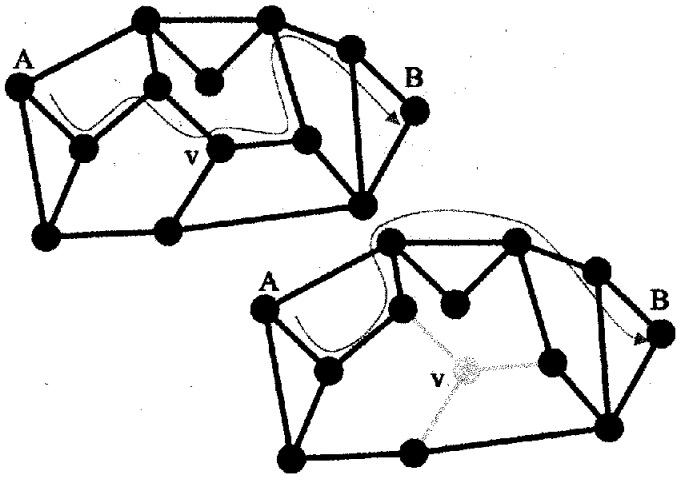

Figura 4.11: En la gráfica original, el vértice v es parte de la ruta, pero al ser eliminado otros vértices toman su lugar y la gráfica sigue siendo conexa.

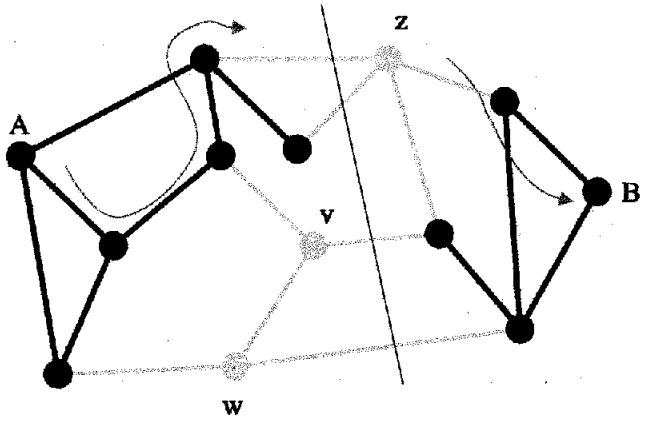

Figura 4.12: Si se eliminan los vértices v, w y z la gráfica se parte en dos y la comunicación entre las dos partes no es posible.

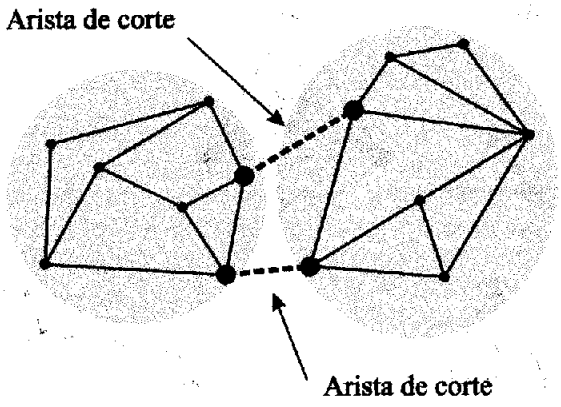

Figura 4.13: Gráfica 2 conexa por aristas, las aristas de corte están en líneas punteadas.

vértices implicados. Por ejemplo, no saturarlos para que no pierdan rápidamente su energía o procurar establecer enlaces alternativos en caso de poder controlar la topología de la red.

Además de la importancia de conocer cuales son los vértices y aristas que al retirarlos dividen la gráfica, existe un parámetro muy útil para determinar la vulnerabilidad de la misma ante este tipo de situaciones, la *conexidad.*

Definición 31 *Si para cada par de vértices $a, b \in V$ de una gráfica $G(V, E)$, existe una ruta $a \rightsquigarrow b$ la gráfica es conexa.*

En base a esto existen tres formas de perder la conexidad de la gráfica, retirando aristas, vértices o ambos. Por lo tanto es muy importante conocer que tan propensa es la gráfica a quedar desconectada o *disconexa* y cuales son los puntos críticos que requieren mayor atención para que esto no suceda. La conexidad de una red o gráfica permite evaluar su vulnerabilidad a las fallas y detectar donde se encuentran, ya sea para reforzarlas o para asignar un grado de confianza menor a esta parte de la red.

4.3.1. Conexidad por aristas

Definición 32 *Una gráfica $G(V, E)$ es k conexa en aristas si existen k aristas tales que $G - \{e_1, e_2, \ldots, e_k\}$ no es conexa, y el número k es el mínimo posible. Las aristas que al retirarlas hacen que la gráfica sea desconexa, se conocen como aristas de corte.*

La figura 4.14 muestra ejemplos de diferentes gráficas conexas, desde una 1 conexa hasta una 5 conexa.

Para resolver el caso más sencillo, es decir determinar si una gráfica es o no 1-conexa y en dado caso que lo sea determinar la arista de corte, el agente puede visitar las aristas

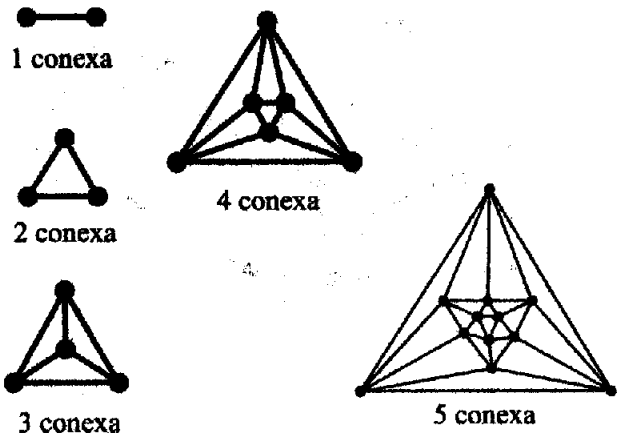

Figura 4.14: Ejemplos de gráficas con diferentes grados de conexidad.

una por una usando el algoritmo de recorrido y un contador de aristas, para después aplicar una prueba a la arista y averiguar si es de corte o no.

Teorema 4.3.1 *La arista de corte e_c de una gráfica plana G que es 1-conexa, solo pertenece a una cara.*

Prueba 4.3.1 *Supóngase que la arista de corte $e_c = \{u, v\}$ pertenece a dos caras C_1 y C_2, es que decir que existen las rutas $u \rightsquigarrow v \in C_1$ y $u \rightsquigarrow v \in C_2$, tal que $e_c \notin u \rightsquigarrow v$. Por lo tanto al eliminar la arista e_c la gráfica no queda disconexa, pues las rutas que pasaban por e_c pueden tomar rutas alternativas ya sea por C_1 ó C_2 que ahora son la misma cara, por lo tanto la arista e_c no es de corte. Ver figura 4.15*

Con esta información el agente solo tiene que recorrer la cara de la arista sospechosa y verificar si la ha encontrado dos veces durante el recorrido, si es así la arista es de corte. El algoritmo 13 realiza este procedimiento y al mismo tiempo puede determinar, después de revisar todas las aristas, sí una gráfica es o no 1 conexa en aristas.

Algoritmo 13 Gráfica 1 conexa en aristas

1: Usar la enumeración efectiva de las aristas.
2: Para cada arista recorrer su cara en un sentido.
3: Si durante el trayecto se recorre la arista en sentido inverso (segunda vez que se recorre) esta es una arista de corte y la gráfica es 1 conexa.
4: Sí al terminar de revisar todas las aristas ninguna es de corte la gráfica no es 1 conexa.

Este algoritmo recorre la cara en $O(n)$ y lo debe de hacer por cada arista que prueba, por lo tanto el algoritmo tarda $O(n^2)$ en determinar sí la gráfica es o no 1 conexa, con un total de $O(1)$ bloques de memoria.

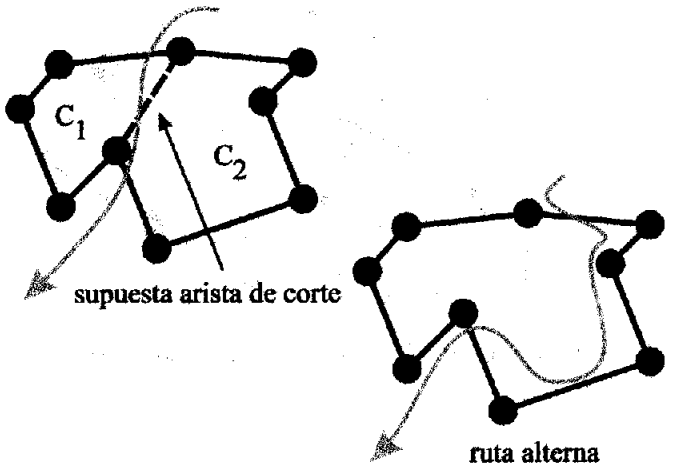

Figura 4.15: Al eliminar la supuesta arista de corte siguen existiendo rutas alternativas para las componentes que supuestamente separó.

4.3.1.1. 2 conexidad

Para saber si una gráfica es 2 conexa se procede de manera similar, es decir, se toman dos aristas candidatas para ser de corte usando la enumeración efectiva y después se aplica una prueba para saber si realmente lo son o no.

Algoritmo 14 Gráfica 2 conexa en aristas

1: Usar la enumeración efectiva de las aristas.
2: Para cada par de aristas a, b, no tomar en cuenta a a.
3: Verificar si la gráfica resultante es 1 conexa con el algoritmo 13, y la arista de corte es b.
4: Ahora no tomar en cuenta a b.
5: Verificar si la gráfica resultante es 1 conexa con el algoritmo 13, y la arista de corte es a.
6: Sí las dos pruebas son afirmativas, las aristas a y b son de corte y la gráfica es 2 conexa.
7: Sí al terminar de revisar todas las parejas de aristas ninguna es de corte la gráfica no es 2 conexa.

Teorema 4.3.2 *Sea la gráfica $G = (V, E)$ una gráfica 2 conexa, con $e_1, e_2 \in E$ sus aristas de corte. Entonces $G_1 = G - e_1$ es una gráfica 1 conexa con e_2 como arista de corte; y $G_2 = G - e_2$ es una gráfica 1 conexa con e_1 como arista de corte.*

Prueba 4.3.2 *La gráfica $G - \{e_1, e_2\}$ no es conexa por definición, Y $G - e_1$ tiene que ser conexa pues de otra forma se tendría una contradicción. $G - e_1$ no puede ser k conexa para $k > 1$ porque entonces la gráfica original sería $k + 1$ conexa, otra contradicción. De esta manera la gráfica $G - e_1$ tiene que ser 1 conexa y su arista de corte es e_2. Lo mismo aplica para $G - e_2$, cuya arista de corte es e_1.*

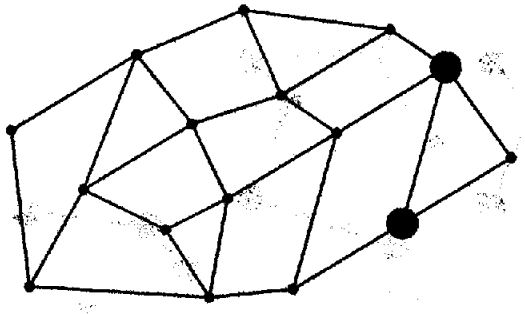

Figura 4.16: Gráfica 2 conexa en vértices, al eliminar los vértices sobre saltados la gráfica pierde conexidad.

Ahora el algoritmo solo tiene que proponer dos aristas de corte candidatas, ignorar una de ellas y verificar que la gráfica resultante es 1 conexa con la arista de corte igual a la otra candidata. En caso que si lo sea la gráfica es 2 conexa y ambas aristas candidatas son sus aristas de corte. El proceso se repite para cada pareja de aristas, hasta determinar si la gráfica es o no 2 conexa, ver algoritmo 14.

Para elegir un par de aristas solo se requieren dos bloques de memoria, y son un total de $O(n^2)$ posibles combinaciones. Para cada combinación se requiere hacer dos recorridos por las caras de cada una, ambos recorridos se realizan en $O(n)$, por lo tanto el algoritmo en total tarda $O(n^3)$ para determinar la 2 conexidad de una gráfica plana usando $O(1)$ bloques de memoria.

4.3.1.2. k conexidad

La idea general para verificar si una gráfica es k conexa por aristas es proponer un conjunto de k aristas de corte, ignorar $k-1$ de estas aristas y verificar si la gráfica resultante es 1 conexa, con la arista de corte igual a aquella arista que no se eliminó del conjunto de k aristas. Si las k pruebas son exitosas, la gráfica es k conexa.

Como el agente solo funciona en gráficas geométricas planas y no existe gráfica plana que sea 7 conexa[45], entonces el algoritmo propuesto puede ejecutarse con memoria constante y proporciona la conexidad por aristas de una gráfica geométrica plana. El tiempo que tarda por recorrer la cara es de $O(n)$ y para verificar la k conexidad se tienen $O(n^k)$ combinaciones, por lo tanto se obtiene un algoritmo de $O(n^{k+1})$ con memoria $O(1)$ para determinar la k conexidad de una gráfica.

4.3.2. Conexidad por vértices

Definición 33 *Una gráfica $G(V,E)$ es k conexa en vértices si existen k vértices tales que $G-\{v_1,v_2,\ldots,v_k\}$ es no es conexa y k es mínimo para cualquier otra elección. A estos*

vértices se les conoce como vértices de corte.

De manera muy similar a los algoritmos de conexidad por aristas, la k conexidad por vértices se determina proponiendo un conjunto de k vértices candidatos para determinar sistemáticamente si son o no de corte y una prueba de recorrido lineal para verificar cada uno de ellos.

Al igual que en la conexidad por aristas primero se determina si la gráfica es 1 conexa por vértices y este algoritmo servirá de base para los demás. El algoritmo 15 muestra el procedimiento y la figura 4.16 muestra un ejemplo de una gráfica 2 conexa en vértices.

Algoritmo 15 Gráfica 1 conexa vértices

1: Usar la enumeración efectiva de las aristas.
2: Contar el número de vértices.
3: Por cada vértice contar el número de vértices, ignorando a las aristas que contengan dicho vértice.
4: Si el número de vértices en ambos recorridos difiere en más de una unidad el vértice es un vértice de corte y la gráfica es 1 conexa.
5: Si al terminar de revisar todos las vértices ninguno es de corte la gráfica es 1 conexa.

Para contar el número de vértices se toma un tiempo igual al recorrido de la gráfica, usando el mejor algoritmo disponible se tarda $O(n \log n)$, por lo tanto para calcular la 1 conexidad por vértices se aplica este conteo a los n vértices, dando un total de $O(n^2 logn)$. En este caso el concepto para determinar la k conexidad por vértices es contar el número de vértices que quedan en la gráfica después de ignorar al vértice que se está probando. Como resultado, si la gráfica es k conexa, la subgráfica contendrá menos de $n-k$ vértices siendo n el número de vértices de la gráfica original y los k vértices serán sus vértices de corte.

Para la k conexidad por vértices se tienen $O(n^k)$ combinaciones que deben verificarse, y por cada combinación se debe realizar un conteo que toma $O(n \log n)$, entonces, el algoritmo tarda $O(n^{k+1} \log n)$ para obtener la k conexidad de una gráfica plana, con una memoria de $O(1)$.

4.4. 6-Coloración usando marcas

Durante el siglo pasado, las comunicaciones han registrado grandes avances en poco tiempo, desde el telégrafo hasta las redes celulares, de la radio a la televisión de alta definición, pasando por dispositivos como los satélites artificiales y la fibra óptica. Uno de los principales elementos en estos esquemas de comunicación son los *canales de comunicación*, que en términos generales determinan cuantos enlaces de comunicación se pueden tener a la vez. De manera general, dos dispositivos establecen comunicación al usar la misma frecuencia, un ejemplo común son los canales de televisión y las estaciones de radio, en donde cada canal tiene establecida una frecuencia única y para poder escucharlo se requiere *sintonizar* la frecuencia.

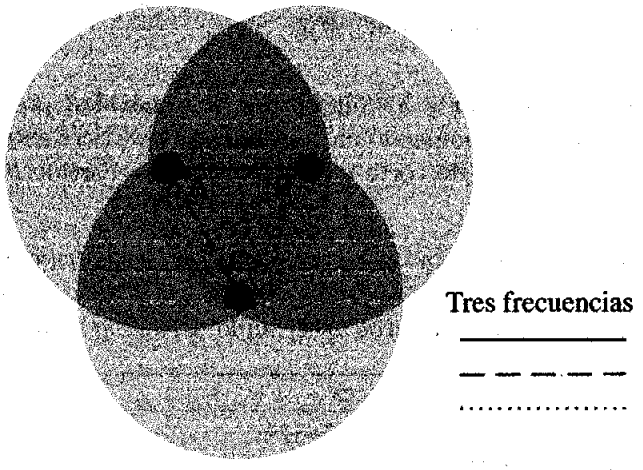

Figura 4.17: Los dispositivos A, B y C necesitan usar tres frecuencias diferentes porque de otra manera un mensaje exclusivo de A para B podría ser recibido por C.

Los dispositivos portátiles con capacidad de comunicación inalámbrica también funcionan bajo el mismo principio, de tal manera que si se tienen tres dispositivos A, B y C y se desea establecer la comunicación entre ellos, primero deben estar dentro del alcance de sus transmisores/receptores, después asignar una frecuencia a cada enlace y por último comunicarse mediante esa frecuencia.

Como se puede observar en la figura 4.17, cada enlace usa su propia frecuencia, porque de no ser así los mensajes enviados, por ejemplo, de A hacia B podrían ser leidos por C. Por lo tanto se requiere de tres frecuencias distintas para enlazar tres vértices y en la figura 4.18 se observa que para cuatro vértices también se requieren tres frecuencias.

El problema de asignación de frecuencias se refiere precisamente a determinar la frecuencia que le corresponde a cada enlace y el número de frecuencias a utilizar. Cuando la red se modela con una gráfica, el problema de la asignación de frecuencias se modela mediante el problema de coloración de aristas. La coloración de vértices es un problema similar en donde los vértices no tienen que tener el mismo color si son adyacentes.

Definición 34 *La coloración de una gráfica es la asignación de colores a sus vértices. Una coloración propia es cuanto los colores de los vértices adyacentes tienen colores distintos. La k coloración es una coloración con k colores distintos.*

En este caso siempre se habla de coloraciones propias. Existe además un resultado fundamental conocido como el teorema de los cuatro colores[45] que dice que una gráfica plana se puede colorear con tan solo cuatro colores. Aunque este es un valor óptimo, resulta difícil de conseguir debido a las limitaciones de memoria, por esta razón el algoritmo propuesto utiliza seis colores y tan solo dos marcas distintas. Este algoritmo es la única excepción

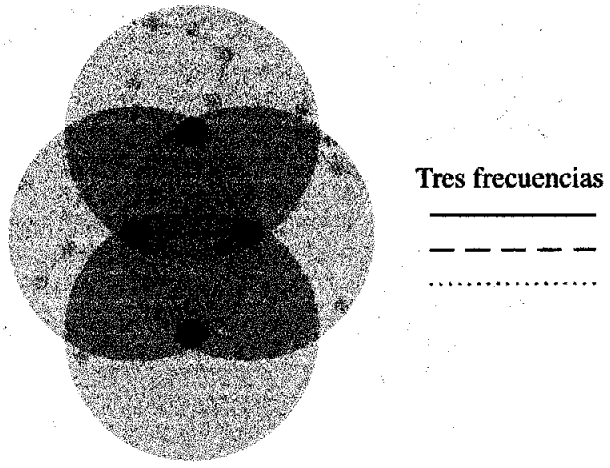

Figura 4.18: Los dispositivos A, B, C y D necesitan usar solo tres frecuencias.

a la regla de no dejar marcas en la gráfica debido a su naturaleza, pues la asignación de frecuencias implica un cambio de estado en los dispositivos.

El algoritmo trabaja sobre subconjuntos de vértices y los va coloreado con un algoritmo voraz. Para diferenciar los vértices se usan dos marcas t y t', la primera indica que ya han sido coloreados y la segunda que se están coloreando. Es decir que durante la ejecución del algoritmo se tienen tres tipos de vértices, los coloreados (t), los que se están coloreando(t') y los que aún no se procesan (no tienen marca).

Para determinar los conjuntos de vértices a colorear se toman subconjunto de vértices V_i, donde todos estos subconjuntos contienen vértices de grado menor o igual a 5, con respecto a su propio conjunto. Se puede observar que al igual que en otros algoritmos de coloración el parámetro básico es el grado de los vértices. En este caso la agrupación se hace de la siguiente manera:

1. $V_1 = \{v|g(v) \leq 5, v \in V\}$.
2. $V_{i+1} = \{v|g(v) \leq 5, v \in G - \bigcup_{j=1}^{i} V_j\}$.

El conjunto base V_1 contiene a los vértices con grados menores, los vértices de grado 6 estarán en V_2 y así sucesivamente hasta que en algún momento de la agrupación, se tendrá un conjunto V_k a partir del cual los demás estarán vacíos, es decir que se tendrán k conjuntos V_i no vacíos, y $V_{k+1} = \emptyset$. Cada uno de estos conjuntos contendrá vértices asociados por su grado.

Cada una de estas gráficas se puede colorear con 6 colores de forma local y con respecto a la misma agrupación, pues el grado máximo es 5 por definición. El algoritmo de coloración local es el que sigue:

1. Recorrer los vértices en algún orden determinado.

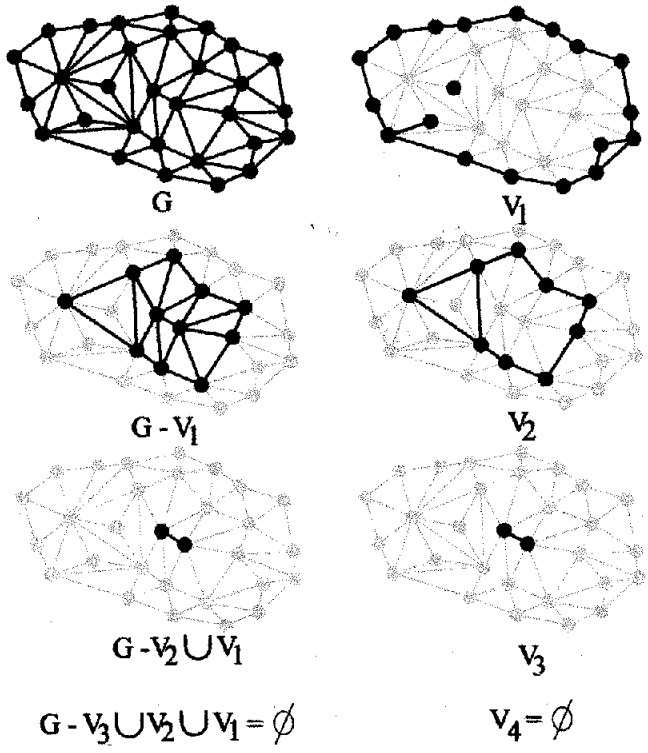

Figura 4.19: Conjuntos V_i de la gráfica G.

2. Asignar a cada vértice un color (en forma ordenada) no asignado a sus vecinos.

Ahora solo es necesario tomar en cuenta las aristas entre V_i y V_{i+1}, para lo cual se utilizan las marcas. En todo momento la marca t' indica el conjunto V_i y la marca t indica los conjuntos $\bigcup_{j=1}^{i-1} V_j$ que ya han sido coloreados. La figura 4.19 muestra un ejemplo.

El algoritmo 16 realiza una 6-coloración en una gráfica plana usando marcas de manera sistemática para distinguir subconjuntos de vértices y colorearlos vorazmente.

Algoritmo 16 6 Coloración

1: Iniciar con $i = 1$.
2: Recorrer la gráfica y marcar con t' a los vértices de V_i $(i = 1)$.
3: Suponer que ya están marcados con t todos los vértices de V_j para $j < i$.
4: Colorear el conjunto V_i vorazmente considerando solo las marcas t y t'.
5: Terminar si ya están todos marcados con t ó t'.
6: Si no, cambiar las marcas t' por t.
7: Repetir desde del segundo paso.

El algoritmo realiza un recorrido para colorear $(O(n \log n))$ cada V_i. Si en el peor de

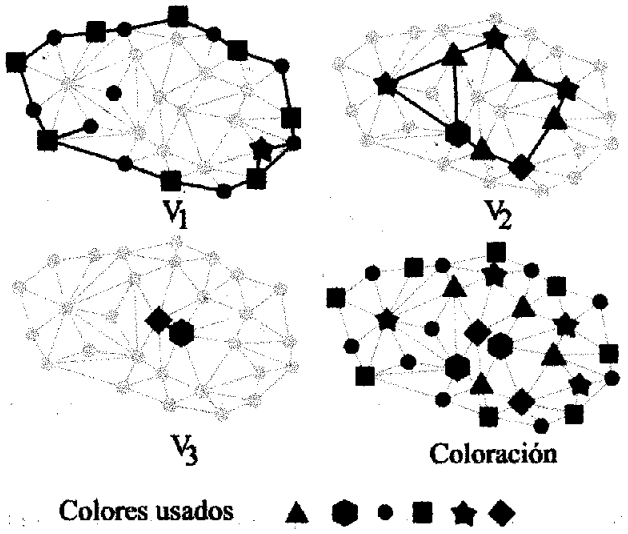

Figura 4.20: Ejemplo de 6-coloración, para cada uno de los conjuntos V_i y el resultado final.

los casos se tienen n niveles de V_i se tiene un algoritmo de tiempo $O(n^2 \log n)$. El algoritmo será presentado en [27]. La figura 4.20 muestra la gráfica del ejemplo ya coloreada.

4.5. Árbol generador de peso mínimo

La motivación para extraer árboles a partir de las gráficas originales es que son más simples en su topología y fáciles de recorrer. Además incluyen todos los vértices de la gráfica original y se pueden usar para aplicar operaciones sobre ellos.

Uno de los árboles más importantes de una gráfica es el *árbol generador de peso mínimo* que, al igual que la ruta más corta, posee propiedades prácticas que implican eficiencia en el uso de recursos asociados a los enlaces y comunicación sobre la gráfica, la figura 4.21 muestra un ejemplo.

Definición 35 *Para un árbol (gráfica sin ciclos) $T(V, E)$, su peso está dado por:*

$$Peso(T) = \sum_{i=1}^{n} d_e(e_i) | e_i \in E$$

Definición 36 *Dada una gráfica G el árbol generador de peso mínimo contiene todos los vértices de la gráfica G y las aristas cuya suma de pesos total sea mínima de tal manera que la gráfica resultante sea conexa.*

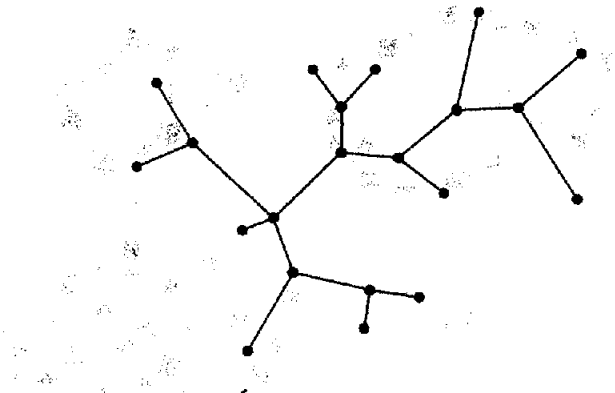

Figura 4.21: Árbol generador de peso mínimo.

La función de peso más utilizada es normalmente la distancia euclidiana porque se le asocian factores de desempeño críticos como la potencia de transmisión y porque en general las gráficas que proporcionan menores distancias euclidianas son más deseables.

Debido a que el agente no puede almacenar mucha información, la forma de calcular el árbol generador de peso mínimo es a medida que se recorre. Entonces se requiere de un método para determinar cuales aristas pertenecen al árbol generador de peso mínimo y que pueda avanzar por ellas utilizando, por ejemplo, la regla de la mano derecha.

El siguiente lema proporciona la información necesaria para establecer las condiciones bajo las cuales una arista pertenece o no al árbol generador de peso mínimo. El algoritmo 17 realiza un recorrido por las aristas del árbol de peso mínimo, y para determinar cuales son realiza un recorrido sobre las aristas de longitud menor a la que está probando.

Lema 4.5.1 *Sea $G(V,E)$ una gráfica con todas sus aristas de longitud distinta y $G_i = G - \{e|l(e) > l(e_i)\}$ para alguna $e_i \in E$, con $l(e)$ como la longitud de la arista e. Entonces e_i pertenece al árbol generador de peso mínimo solo sí G_i no contiene ciclos que contengan a la arista e_i.*

Prueba 4.5.1 *Supóngase que G_i tiene un ciclo que contiene a e_i. Como todas las aristas tienen una longitud menor a la de e_i, si se elimina e_i del ciclo se obtiene una gráfica con un peso menor, por lo tanto e_i no pertenece al árbol generador de peso mínimo, ver figura 4.22.*

Teorema 4.5.1 *El algoritmo 17 recorre el árbol generador de peso mínimo determinando si una arista es o no parte del árbol en un tiempo $O(n^2)$ con respecto al número de aristas.*

Prueba 4.5.2 *Si la arista está en un ciclo obviamente no puede pertenecer a un árbol pues por definición un árbol es una gráfica acíclica, por otro lado como todas las aristas son de peso menor a ella si lo estuviera, al eliminarla se conseguiría una gráfica de peso menor.*

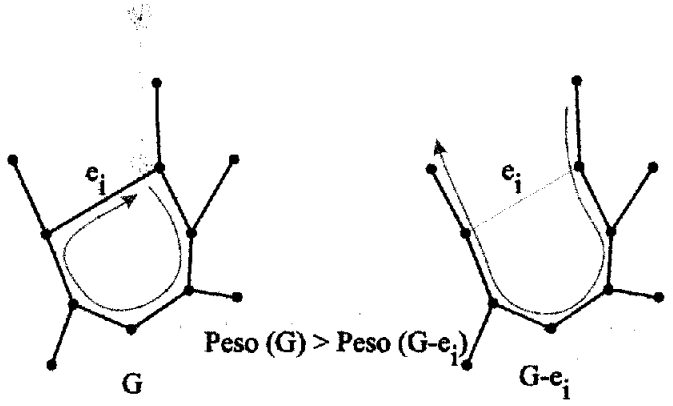

Figura 4.22: Si las aristas de longitud menor que e_i forman un ciclo es posible reducir el peso, por lo tanto e_i no está en el árbol generador de peso mínimo.

Algoritmo 17 Recorrido del árbol de recubrimiento mínimo

1: Iniciar en cualquier vértice.
2: Recorrer una de sus caras en el orden de las manecillas del reloj.
3: Tomar la arista a recorrer como candidata.
4: Usar la longitud de la arista candidata.
5: No tomar en cuenta a las aristas de longitud menor.
6: Recorrer las dos caras a las que pertenece la arista usando la regla de la mano derecha.
7: Si se forma un ciclo la arista no está en el árbol.
8: Si no, la arista sí está en el árbol.
9: Tomar la siguiente arista en sentido de las manecillas del reloj como candidata.
10: Repetir desde el paso 4 y terminar al darle la vuelta al árbol.

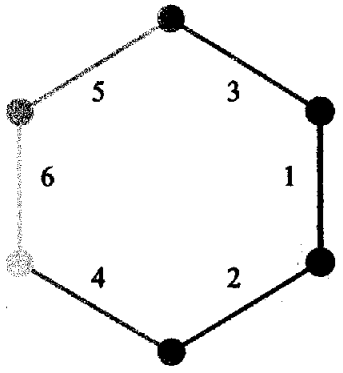

Figura 4.23: Los números indican el orden de las aristas según el criterio del vértice más a la derecha y a la izquierda.

Si la arista no está en un ciclo y todas la aristas que faltan por considerar son de peso mayor no existe arista que conecte con un menor peso a las componentes que une la arista que se prueba, por lo tanto está en el árbol generador de peso mínimo.

Para verificar si una arista está en un ciclo se recorre la cara con la regla de la mano derecha y posteriormente se verifica si durante el recorrido se visita la arista en sentido inverso. Si se visita en sentido inverso no está en un ciclo. La sencillez de este procedimiento se debe en gran medida a que se basa en resultados anteriores permitiendo avances significativos.

Este algoritmo necesita que las longitudes de las aristas sean distintas, pero esta suposición no altera la generalidad del método y se puede implementar fácilmente. Este requisito se debe a que si existen aristas iguales sería un problema determinar que arista eliminar porque cualquier selección realizada altera las demás decisiones y es necesario llevar un registro, lo cual es imposible porque la memoria es constante. Por esta razón es necesario que las aristas tengan longitudes distintas.

En caso de tener un ciclo con aristas de la misma longitud es necesario establecer un orden entre ellas para diferenciarlas. El siguiente criterio es el mismo que se usa para encontrar la arista de entrada de una cara y permite establecer un orden entre las aristas según los siguientes valores:

- Vértice más a la derecha.
- Si hay empates, el vértice que esté más abajo.

La figura 4.23 muestra un ejemplo en donde los números indican el orden para considerarlas mayores o menores y posteriormente eliminar la que no cumpla las condiciones de la prueba. En la figura 4.23 la arista 1 se considera de longitud mayor que todas las demás y como está en un ciclo de aristas de longitud menor no es parte del árbol generador de peso mínimo.

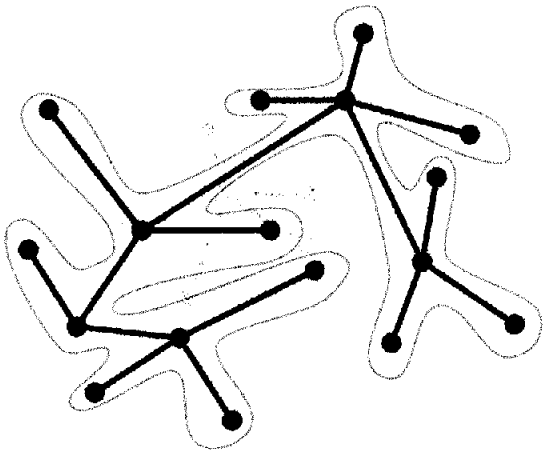

Figura 4.24: La cara externa del árbol contiene a todos los vértices del árbol.

4.6. Ruteo y distancia en árboles

Los árboles son estructuras sencillas con características importantes como por ejemplo que se pueden realizar recorridos sistemáticos de una manera óptima[45]. Ya se han expuesto algoritmos que permiten a una agente *ver* la gráfica como si esta fuera un árbol, por ejemplo el árbol de recubrimiento mínimo o incluso el árbol que se forma al retirar las aristas de entrada de las caras. De esta forma, es posible aplicar algoritmos sobre los árboles obtenidos y en este caso se expone el ruteo en árboles.

Definición 37 *Un árbol es una gráfica G donde no existen los ciclos.*

Una de las implicaciones de esta definición es que las rutas entre vértices son únicas y que todas la aristas se encuentran en la misma cara, la cara exterior. La figura 4.24 muestra como al seguir la orilla del árbol se vuelve al mismo punto y esta cara contiene a todas las aristas.

Entonces el ruteo por caras es naturalmente un algoritmo con memoria constante e información local que es capaz de rutear mensajes en árboles en un tiempo $O(n)$. Sin embargo el algoritmo no es capaz de determinar la ruta única entre los dos vértices.

Pero ¿para qué usar la ruta si se pueden mandar mensajes? Definitivamente mandar mensajes es una tarea que resulta sencilla en árboles, pero la ruta única contiene información que resulta valiosa cuando las configuración del sistema se vuelve estable o simplemente cuando se va utilizar más de una vez. Es decir que a medida que pasa el tiempo, el usar la ruta única tiene un mejor desempeño comparado con usar el ruteo por caras.

Como la ruta es única el agente solo necesita verificar cual es la arista de salida que lo lleva al destino deseado. Desde el punto de vista de un agente *parado* en un vértice y con

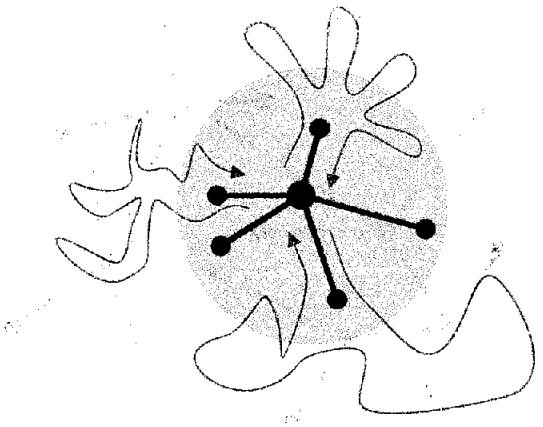

Figura 4.25: Desde el vértice central solo se ven los vecinos, él agente solo sabe que al salir por una arista y recorrer el árbol, regresará a la arista de partida.

información local, el árbol se verá como en la figura 4.25. Lo único que el agente sabe es que al salir por alguna arista tendrá que regresar por la misma arista, y que si recorre todas las aristas de salida de esta manera recorrerá el árbol en su totalidad. Por lo tanto el agente al dar un recorrido completo, sabrá cual es la arista que contiene el vértice destino. y podrá avanzar por ella. Estando en el nuevo vértice, el agente puede recordar por donde llegó para no recorrer esa parte de la gráfica. Así pues en el peor de los casos el agente tardará $O(n^2)$ en calcular la ruta y la distancia de la misma entre dos vértices de un árbol. El algoritmo 18 muestra este procedimiento.

Algoritmo 18 Distancia entre dos vértices

1: Recorrer el árbol usando la regla de la mano derecha hasta llegar al primer vértice.
2: Por cada una de las aristas incidentes a este vértice, recorrer el árbol con la regla de la mano derecha, iniciando y terminando en esa misma arista.
3: Se avanza por la arista que reporte al segundo vértice, incrementando la distancia apropiadamente.
4: Repetir desde el paso 2 hasta llegar al destino.

En cada *vuelta* se descubre una arista de la ruta única que une a los dos vértices. La figura 4.26 muestra las tres *vueltas* que el agente da usando la regla de la mano derecha, en cada una de ellas solo puede estar seguro de que la arista por la que llega, después de *ver* el vértice b, está en la ruta y por lo tanto avanza sobre esa arista.

Este algoritmo es sensible a la salida, en este caso es la longitud de la ruta, de tal manera que si el número de aristas en la ruta es l, la complejidad será de $O(l \cdot n)$ y en el peor caso es $O(n^2)$.

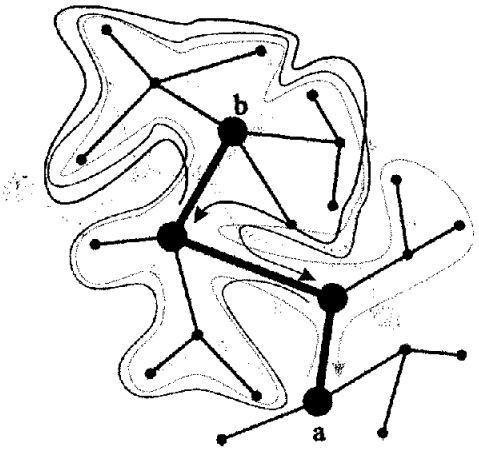

Figura 4.26: Al hacer el recorrido el agente solo puede estar seguro de que la arista por la que salió está en la ruta de a hacia b.

4.7. Distancia máxima en árboles

Otro tipo de información valiosa es la *distancia máxima* y el vértice con el cual se obtiene. Un vértice que conoce cual es su distancia máxima puede determinar las distancias de recorrido mínimas para sus mensajes, es decir que sí el vértice a está ofreciendo un servicio y quisiera pedirle a otro vértice que lo apoye ofreciendo el mismo servicio, el vértice ideal sería el que estuviera más lejos para que cada uno cubra por lo menos la mitad de los vértices dentro de la ruta, como lo muestra la figura 4.27.

Sí se intenta encontrar el vértice más lejano aplicando repetidas veces el algoritmo de distancia sobre árboles, la complejidad se puede elevar hasta $O(n^3)$. Pensando en esto se plantea otra aproximación en base a algunas observaciones. La primera es que el vértice más lejano con respecto a otro vértice siempre estará en una hoja por lo tanto se pueden descartar todos los vértices internos. En general lo que el agente tiene que averiguar es sí al caminar sobre la cara está aumentando la distancia o la está disminuyendo.

Para poder encontrar el algoritmo se supuso que está información se conocía de tal forma que se observó que las aristas en donde la distancia aumenta no han sido recorridas y las aristas donde la distancia disminuye ya han sido recorridas por lo menos una vez. De esta manera solo es necesario un recorrido por vértice, dando una complejidad de $O(n^2)$.

El recorrido se realiza de la siguiente forma, el agente avanza por la primera arista y la distancia aumenta, después avanza por la cara exterior hacia una arista candidata y se regresa al inicio. Si durante este recorrido de regreso visita la arista candidata dos veces, la distancia se decrementa, si no la distancia aumenta, el agente regresa a la arista candidata y avanza a la nueva arista candidata hasta llegar al inicio. Durante todo este proceso el agente guarda un registro de la distancia actual y un registro de la distancia máxima junto el vértice donde se logra esto. El algoritmo 19 calcula la distancia hacia todos los vértices del árbol, sin

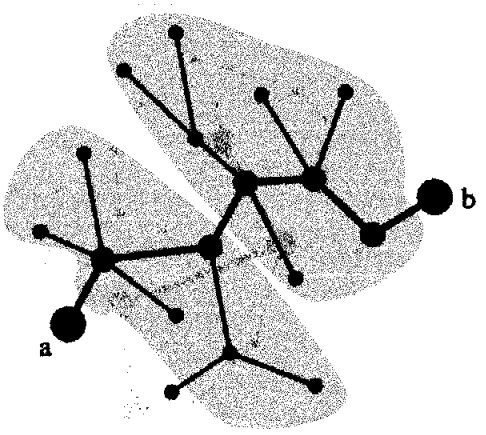

Figura 4.27: Los vértices a y b al estar a una distancia máxima, cubren entre los dos más vértices con menos mensajes.

embargo debido a la memoria limitada solo puede reportar un número constante de vértices, en este caso el vértice de la distancia máxima.

Algoritmo 19 Distancia a todos los vértices de un árbol

1: Recorrer el árbol por la cara exterior hasta llegar al vértice de inicio.
2: Avanzar por la arista candidata, es decir la arista que sigue en el recorrido de la cara exterior.
3: Regresar en sentido inverso hasta el vértice de inicio.
4: Si en el recorrido inverso se encuentra dos veces a la arista candidata, decrementar la distancia.
5: Si solo la encuentra una vez, aumentar la distancia.
6: Comparar con la distancia máxima y actualizar debidamente.
7: Regresar a la arista candidata.
8: Repetir desde el paso 2 hasta llegar al vértice original.

La figura 4.28 muestra un ejemplo donde se puede ver que las aristas son probadas dos veces por el algoritmo, de manera que la primera revisión aumentará la distancia y la segunda la disminuirá. A pesar de esto el algoritmo no es capaz de saber si es la segunda o primera vez que visita la arista porque no tiene la memoria suficiente para almacenar n aristas.

El algoritmo 19 usa memoria $O(1)$ y un tiempo $O(n^2)$ para determinar la distancia máxima y el vértice que se encuentra a esa distancia.

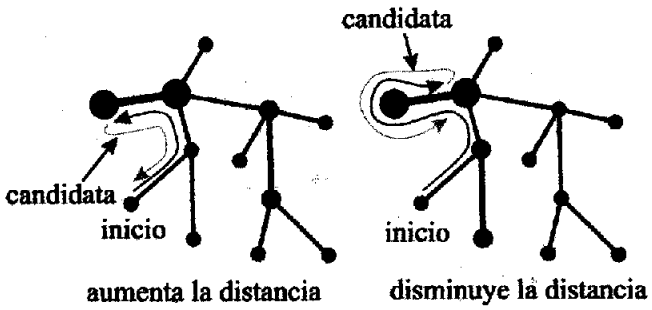

Figura 4.28: La distancia al vértice inicio aumenta o disminuye dependiendo de las veces que la arista candidata se encuentre en la ruta de regreso.

4.8. Centro de un árbol

En las situaciones más comunes siempre existen lugares mejores que otros, por ejemplo los miradores son sitios altos desde donde se puede obtener una vista panorámica ó el centro de la ciudad que es ideal para llegar rápido a cualquier otro lugar. Estos lugares estratégicos resultan ser de gran interés para ciertas aplicaciones, por ejemplo un observatorio se ubica mejor en lugares altos y sin mucha luz, pero la estación de bomberos deberá estar en un lugar central a la zona que esté cubriendo. En la gráficas y redes sucede lo mismo, se tienen aristas que son poco transitadas, vértices con un alto grado, caras que abarcan gran parte de la gráfica, etc.

Uno de estos lugares estratégicos utilizado para ubicación de servicios es el *centro de la gráfica*, un lugar desde donde la distancia a todos los vértices es mínima. El centro de una gráfica es de gran utilidad en problemas de cobertura o para detectar puntos críticos de una red.El centro del árbol junto con las distancias entre vértices permiten evaluar parámetros de la red concernientes a la distribución de servicios y disponibilidad de recursos. Esta información es muy valiosa si se desea usar los recursos de manera óptima. Por esta razón se decidió diseñar un algoritmo que encontrase el centro de una gráfica, para lo cual se trabajó con el problema más particular de encontrar el centro de un árbol pues, como ya se ha mencionado, existen métodos para extraer árboles a partir de gráficas más generales.

Definición 38 *El centro de un árbol es el punto desde donde la distancia a todos los vértices del árbol es mínima. Este punto puede estar en un vértice o a la mitad de una arista.*

El centro del árbol se encuentra a la mitad de la ruta que une a los dos vértices más alejados, los cuales serán vértices hoja. De esta forma para encontrar el centro del árbol solo se necesita encontrar estos dos vértices.

La idea del algoritmo es encontrar dos parejas de vértices alejados al máximo cada uno. Esto no garantiza encontrar el centro del árbol, hace falta decir que estas dos aristas necesitan tener un vértice en común de tal manera que la ruta que conecta a el vértice en

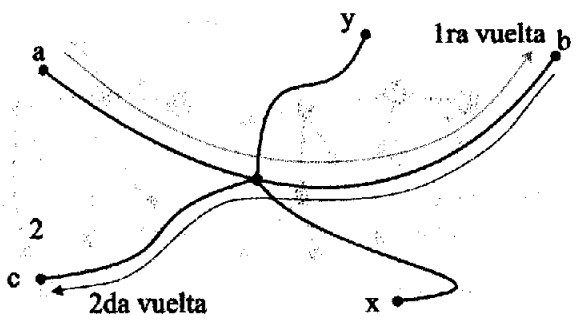

Figura 4.29: Centro de un árbol. En la segunda vuelta se encuentra la ruta más larga. Si la ruta de x a y fuera más larga, sería descubierta en esta segunda etapa.

común y el segundo extremo contenga el centro de la gráfica. Es decir que en solo dos pasos es posible encontrar el centro del árbol. El algoritmo 20 muestra el procedimiento.

Algoritmo 20 Centro de un árbol

1: Recorrer el árbol usando la regla de la manor derecha, hasta llegar a una hoja.
2: Calcular la distancia a todos las hojas desde allí.
3: Desplazarse a la hoja más lejana.
4: Calcular la distancia a todos las hojas desde esta segunda hoja.
5: Rutear hacia la hoja más lejana.
6: En la mitad de esta ruta está el centro del árbol.

Teorema 4.8.1 *El algoritmo 20 calcula el centro de un árbol en un tiempo de $O(n^2)$ usando memoria constante e información local.*

Prueba 4.8.1 *Iniciando en cualquier vértice hoja a, el algoritmo 19 encuentra el vértice más lejano b (otra hoja). Aplicando el mismo procedimiento al vértice b se obtiene el vértice hoja c cuya distancia a b es máxima. Estos dos pasos se consiguen en tiempo $O(n^2)$ usando memoria constante e información local.*

Supóngase que existe una pareja de vértices x y y que están más alejados que c y d, es decir $d(x,y) > d(c,d)$. Esto quiere decir que al calcular la distancia máxima para a, en el paso uno, se obtuvo que $d(a,x) \leq d(a,b)$, $d(a,y) \leq d(a,b)$ y $d(a,c) \leq d(a,b)$. Del paso dos se tiene que $d(b,x) \leq d(b,c)$, $d(b,y) \leq d(b,c)$ y $d(b,a) \leq d(b,c)$. Por lo tanto $d(x,y) \leq d(b,c)$, la suposición no es válida y la ruta que conecta a b y c contiene el centro del árbol justo a la mitad, ya sea una arista o un vértice.

Algoritmo	Complejidad	Bloques de memoria
k Conexidad vértices (k constante)	$O(n^{k+1} \log n)$	$O(1)$
k Conexidad aristas (k constante)	$O(n^{k+1})$	$O(1)$
Barrido de línea	$O(n^3)$	$O(1)$
Coloración	$O(n^2 \log n)$	$O(1)$[1]
Árbol generador de peso mínimo	$O(n^2)$	$O(1)$
Distancia Máxima en árboles	$O(n^2)$	$O(1)$
Centro de un árbol	$O(n^2)$	$O(1)$
Distancia en árboles (l número de aristas entre los vértices)	$O(n \cdot l)$	$O(1)$

Tabla 4.9. Algoritmos propuestos.

4.9. Conclusiones

En este capítulo se atacaron problemas geométricos usando memoria constante, información local y tratando de no alterar la gráfica, por lo menos en los casos donde esto no es necesario. Las ventajas de este método son evidentes y van más allá de lo que el ruteo geométrico había ofrecido hasta ahora. También se puede ver que el ruteo es base para otros problemas, de la misma manera que los problemas que se resuelven aquí pretenden ser base para problemas más complejos. Otro aspecto importante a concluir es el hecho de que los problemas se van resolviendo de lo particular a lo general, utilizando para ello métodos que permiten pasar de una instancia general a otra en particular dentro del mismo modelo. En el capítulo anterior se presentaron métodos para obtener gráficas planas y en el presente capítulo se presentaron métodos para obtener árboles a partir de gráficas planas. Todo esto proporciona un conjunto de herramientas para trabajar sobre gráficas geométricas y aplicar soluciones a diferentes niveles. Aunque es cierto que al tener menos aristas la red se congestiona más, los métodos de exploración para los vértices son más rápidos y en la mayoría de las aplicaciones el interés se centra en los vértices y no en las aristas.

A pesar de que los problemas ya cuentan con soluciones prácticas en la literatura del área [45], hay que recordar que no existen algoritmos que usen memoria constante e información local, las cuales son las principales limitaciones de los sistemas tratados. Es decir, que las soluciones encontradas satisfacen las condiciones que se presentan en los sistemas como redes inalámbricas *ad hoc*, de una manera que las soluciones actuales no han demostrado satisfacer, sobre todos en el manejo de recursos.

Los algoritmos también mantienen las características de los primeros resultados del ruteo geométrico, que son: la no alteración del estado de la gráfica y por lo tanto la posibilidad de ejecutarse en paralelo; y la resolución de problemas globales de forma local, lo que es una muestra clara de la alta escalabilidad de los resultados obtenidos.

Por esta razón los algoritmos no pueden ser comparados con las versiones no geométricas ó centralizadas, pues en todos los casos se trata de métodos nuevos. La tabla 4.9 muestra un resumen de los métodos expuestos y su complejidad.

Capítulo 5

Conclusiones

Cuando los sistemas de comunicación tienen un alta movilidad y su topología cambia rápidamente, la información acerca de ellos solo es confiable si se trata de información local. De la misma manera la movilidad del sistema significa, en la mayoría de los casos, que los recursos disponibles son pocos, por ejemplo que la energía para mantenerlo operando suelen ser baterías recargables, que los circuitos de memoria no pueden ser más de cierto número, etc. De una manera muy práctica la tesis aporta soluciones para estas dos características de los sistemas de comunicación móviles.

La investigación de la tesis tiene su origen en el trabajo presentado en [12] donde se expone el notable algoritmo conocido como *ruteo por brújula*. En él se establecen las condiciones básicas para dar una solución geométrica a los problemas de un sistema de comunicación móvil, en particular el ruteo en redes *ad hoc*. Las condiciones incluyen: conocer la posición (coordenadas) de los vecinos y la propia, usar memoria constante y no dejar marcas. Entre otras cosas esto permite la ejecución paralela de los algoritmos, un uso óptimo de la memoria y un alto grado de escalabilidad, además de que se obtienen soluciones localmente a problemas globales. Debido a esto el modelo fue ampliamente aceptado y continuaron obteniéndose resultados prometedores entre los que destaca el recorrido de gráficas planas en un tiempo $O(n \log n)$[35].

Esta tesis se suma a los resultados en el área con algoritmos para solucionar problemas como la coloración, la conexidad, el centro de un árbol, la distancia máxima en un árbol, entre otros más. Los algoritmos se aplican a cualquier problema modelado mediante gráficas geométricas donde se necesite usar memoria constante y tener alta escalabilidad.

Hasta el momento la principal aplicación de los algoritmos es sobre redes *ad hoc*, las cuales no cuentan con una infraestructura de comunicación fija como lo pudieran ser las antenas de los servicios celulares o los cables de las líneas telefónicas. Los resultados e implicaciones de la *aproximación geométrica* son tan eficientes que sería muy interesante aplicarla en sistemas de redes tradicionales ó incluso en otras áreas. El hecho de modelar los algoritmos como agentes de software (entidades que se mueven autónomamente sobre las gráficas para realizar su tarea) significa tener un tráfico menor y mayor tolerancia a fallas, comparado con otras alternativas.

La seguridad y el cómputo distribuido son dos de las posibles líneas de investigación a futuro que pueden continuar con el espíritu de la tesis, sin embargo quedan abiertos muchos otros problemas como soluciones en gráficas no planas, dimensiones superiores, además de posiblemente reducir la complejidad de los algoritmos aquí presentados.

5.1. Agentes de memoria limitada

Los agentes de memoria limitada son programas que hacen uso de una cantidad de memoria constante para realizar sus tareas, es decir una cantidad $O(1)$ con respecto a una entrada de tamaño n. Es claro que existe una independencia entre la cantidad de memoria y el tamaño de problema dejando de lado las implicaciones técnicas de la implementación.

Una de las ventajas más sobresalientes de los agentes es que son altamente escalables debido a que la cantidad de memoria no depende del tamaño de la entrada, es decir que esta puede seguir creciendo y el desempeño del algoritmo no se ve afectado.

Por esta razón los agentes son herramientas útiles en cualquier campo de aplicación sin embargo su uso es poco común debido a que no en todos los problemas se requiere de agentes o se cuenta con memoria limitada, además de que el diseño de este tipo de algoritmos es más complicado.

Estos agentes son móviles y están concebidos para trabajar en entornos dinámicos reales como redes de computadoras y redes móviles. Las ventajas de este modelo se ven reflejadas en un menor número de enlaces y en una red con menor tráfico.

5.2. Entornos geométricos

La geometría computacional se dió a conocer por sus aplicaciones en animación y simulación pero el verdadero potencial de la geometría como herramienta en las ciencias de la computación está aún por descubrirse según se puede apreciar por los resultados de esta tesis.

Se habla de *geometrizar* los procesos cuando se incluye o se trabaja con información geométrica con el fin de obtener ventajas y realizar tareas que de otra forma y bajo condiciones especiales (como memoria constante) serían imposibles de realizar.

Actualmente la *geometrización* de los problemas es posible gracias a tecnologías de posicionamiento global vía satélite como la que se usa en el GPS (Global Positioning System) ó el sistema Galileo. Estas tecnologías están basadas en la triangulación de señales satelitales de alta cobertura y además ofrecen servicios como sincronización y coordenadas estándares.

5.2.1. Sistemas de posicionamiento global

La historia del hombre ha mostrado que siempre ha existido interés por ubicarse en todo momento y a diversas escalas dentro de su mundo. Para lograrlo ha utilizado puntos cardinales, brújulas, mapas, divisiones política y geográficas, etc, etc. Dentro de estas técnicas para ubicarse se destaca la observación del firmamento que desde el tránsito del sol hasta el movimiento de las estrellas ha permitido la navegación y la posibilidad de llegar un registro del tiempo, entre otras cosas.

Los sistemas de posicionamiento y de navegación fueron mejorando junto con la tecnología hasta llegar a nuestros días donde se utilizan sistemas de posicionamiento satelitales, que curiosamente usan constelaciones de satélites artificiales para triangular la posición de los objetos sobre la superficie de la tierra.

5.2.1.1. El principio de operación

La triangulación de la posición de un punto es el proceso básico que utilizan los sistemas de posicionamiento global y está basado en dos puntos clave:

1. Conocer la posición (coordenadas) de diferentes puntos de referencia.
2. Medir la distancia hacia esos puntos de referencia.

Este método es válido para localizar puntos en cualquier dimensión, siempre y cuando el número de puntos de referencia utilizados en cada triangulación sea mayor a la dimensión del espacio, es decir que para triangular la posición de un punto en tres dimensiones es necesario tener al menos cuatro puntos de referencia.

Algoritmo 21 Triangulación de la posición de un punto (para n-dimensiones).

1: Tomar m puntos de referencia, donde $m > n$.
2: Realizar la medición de la distancia entre el punto a calcular y cada uno de los m puntos de referencia.
3: Usar cada punto de referencia como el centro de una esfera de dimensión n de radio igual a la distancia medida al punto a triangular.
4: Calcular la intersección de las esferas.
5: Las coordenadas de la intersección son las coordenadas del punto que se busca triangular.

El algoritmo 21 describe el proceso de triangulación para n dimensiones pero basta con usar solo tres, es decir con la intersección de cuatro esferas. La idea detrás del algoritmo es que al tomar la primera distancia se sabe que el punto está sobre la superficie de una esfera, al tomar la segunda distancia, se intersectan las esferas y el punto solo puede estar sobre el círculo que se forma de la intersección. Con la tercera distancia solo quedan dos posibles puntos y con la cuarta distancia ya no hay duda de cual de los dos es el punto que se busca. La figura 5.1 ilustra estos pasos hasta la tercera esfera pues en la práctica solo se toman tres medidas y la cuarta es tomada como el radio de la tierra.

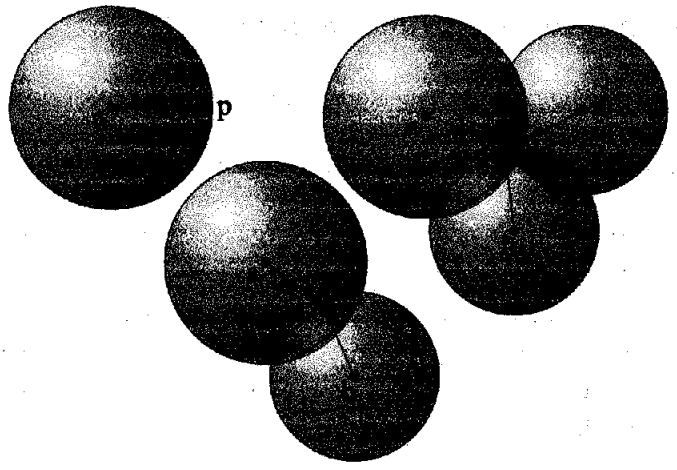

Figura 5.1: Triangulación de un punto p. Se conocen las posiciones de los puntos a, b y c además es posible medir la distancia de p hacia ellos.

Los puntos de referencia conocidos generalmente son estaciones emisoras de radio o satélites en órbita alrededor de la tierra. Las distancias son calculadas midiendo el tiempo que tardan las ondas de radio en llegar de un punto a otro. Debido a que es posible incluir información en estas señales algunos sistemas incluyen servicios como la sincronización de relojes.

5.2.1.2. Ejemplos

En la actualidad los diferentes sistemas de posicionamiento global utilizan satélites artificiales que emiten constantemente una señal que los identifica de manera única. Conjuntamente existen estaciones terrestres que ayudan a determinar las coordenadas correctas de los satélites, porque aunque están en una órbita regular sufren desviaciones.

Con los satélites emitiendo señales, un dispositivo electrónico solo tiene que conocer sus posiciones de forma precisa y calcular las distancias a ellos mediante el tiempo que tarda en recibir las secuencias de los satélites. Sin embargo no todo es tan sencillo pues, por razones de seguridad, la señal está codificada y se introducen errores que solo pueden ser compensados de forma directa por quienes los conocen.

- **GPS (Global Positioning System)** fue creado por el Departamento de Defensa de los Estados Unidos de Norte América con fines militares y consta de 24 satélites a una altura aproximada de 17,600 kilómetros. Cada satélite lleva 4 relojes atómicos de alta precisión que permiten la sincronización de los dispositivos que reciben la señal.
- **GLONASS** es administrado por las Fuerzas Espaciales Rusas del Gobierno de la Federación Rusa con propósitos civiles y militares. Solo cuenta con siete satélites activos

y cuatro de reserva de los 79 que han sido lanzados. A pesar de que el sistema tiene serios problemas económicos, mala calidad de la señal y que se espera que deje de operar pronto, las señales de los satélites son usadas por otros sistemas de posicionamiento global como apoyo para mejorar la precisión y la tolerancia a fallas.

- **Galileo** es administrado por la Unión Europea y la Agencia Europea del Espacio y tendrá plena capacidad operativa para el 2008, contará con 30 satélites en tres órbitas a 23,616 kilómetros sobre la superficie de la tierra. Será independiente de GPS y GLONASS pero podrá usar sus señales.

5.2.1.3. Precisión

La precisión de los sistemas de posicionamiento es un elemento crítico para la aplicación de las soluciones geométricas. Si las coordenadas no son certeras los algoritmos podrían presentar resultados erróneos o quedar ciclados. El efecto de las variaciones en la precisión de las coordenadas no debe menospreciarse. Esta precisión depende básicamente de los siguientes puntos:

- **Puntos de referencia.** Conocer con exactitud las posiciones de los puntos de referencia.
- **Medición.** Calcular las distancias de manera precisa.
- **Cálculo.** Precisión computacional en los cálculos de la intersección de las esferas.

En base a esto es posible determinar algunas medidas para mejorar la precisión del sistema:

- **Planear la medición**. Determinar la hora en que se *ven* más satélites, usar aquellos en buenas condiciones o aquellos de los que se conosca con mayor exactitud su posición.
- **Número de observaciones**. Determinar si basta con una sola observación (**Posicionamiento absoluto**) o varias (**Posicionamiento diferencial**) para realizar comprobaciones entre ellas.
- **Cálculo**. Realizar las operaciones con la mayor precisión disponible.
- **Otros**. Verificar que sistema de referencia se esta usando, determinar el error posible, etc.

Al tomar medidas en varios puntos se forman una o más *líneas base*, las cuales permiten verificar los datos mediante relaciones geométricas. La precisión final es variable y depende del equipo y método de trabajo seleccionado.

5.2.1.4. Ventajas y desventajas

El posicionamiento global tiene muchas ventajas, principalmente porque los satélites requieren poco mantenimiento, tienen un alcance de varios miles de kilómetros y su señal no sufre mucha distorsión por el clima.

Además los dispositivos no requieren dirigir antenas para recibir las señales que son totalmente gratuitas. Se obtienen resultados rápidamente y como son interpretados por uno mismo tienen comprobación.

Las principales desventajas se deben a las propiedades de las señales del satélite, las cuales no llegan a zonas subterráneas y reciben interferencia por los objetos en zonas urbanas o con muchos árboles, por ejemplo.

5.2.2. Sistemas y marcos de referencia

Un *sistema de referencia* es un conjunto de reglas que describen de manera única la superficie de algún objeto y la posición de otros objetos sobre ella. Los *marcos de referencia* están constituidos por puntos materializados en el espacio y ubicados con gran exactitud y precisión según algún sistema de referencia.

Los sistemas y marcos de referencia de la superficie de la tierra son muy útiles para aplicaciones como la navegación, control de tráfico, monitoreo, mapas, etc, etc. Se tienen dos tipos básicos de sistemas:

1. **Con Referencia Celeste**. Se usan las estrellas y otros cuerpos celestes como puntos de referencia por lo tanto son teoricamente fijos.

2. **Con Referencia Terrestre**. Se usa al propio planeta como referencia por lo tanto giran y se trasladan junto con él.

En general los elementos más utilizados para definir sistemas de referencia son: la dirección del eje de rotación terrestre, el círculo del Ecuador, la constelación de Aries, el centro de masa de la tierra y el meridiano de Greenwich. El sistema de referencia terrestre más conocido supone a la tierra como una esfera perfecta y la divide usando líneas imaginarias llamadas meridianos y paralelos. Los meridianos son círculos máximos que pasan por los polos de rotación y los paralelos son círculos menores paralelos al Ecuador que van estrechándose hasta convertirse en un punto en los polos. El Ecuador es el origen de los paralelos y el meridiano que pasa por el observatorio real de Greenwich en Londres es el origen de los meridianos.

El principal sistema de referencia es el ITRS (International Terrestrial Reference System) y el principal marco de referencia el ITRF (International Terrestrial Reference Frame). Para Europa aplica el ETRS (European Terrestrial Reference System) y el ETRF (European Terrestrial Reference Frame), todos ellos compatibles entre sí. Todas las coordenadas tienen

asociado el año en que fueron anunciadas, es decir que están fijas en el tiempo. Esto es muy útil tomando en cuenta la superficie de la Tierra está en constante movimiento.

El GPS utiliza el sistema de referencia WGS84 (World Geodesic System 1984) que también es utilizado para determinar las órbitas de los satélites. Los sistemas Galileo y GLONASS usan el sistema de referencia PZ 90 (Parametry Zemli-1990, Parámetros de la Tierra 1990) ó PE 90 (Parameters Earth-1990). Todos ellos son consistentes con el ITRF.

Es posible establecer una relación geométrica entre dos o más sistemas de referencia si se conoce la posición de un objeto en común dentro de los sistemas. De esta manera un dispositivo puede tomar como referencia satélites de distintos sistemas. De esta forma la precisión también depende de la transformación entre sistemas.

5.2.3. Futuro

Los sistemas de posicionamiento satelital son una herramienta muy útil que formará parte de la vida cotidiana en muy poco tiempo. Su potencial ya ha sido analizado por la industria, la milicia y el sector académico, entre sus aplicaciones más importante se encuentran los mapas electrónicos de alta precisión y sistemas de rastreo y monitoreo. Otras aplicaciones pueden ser el control de tráfico ó para problemas de seguridad.

Un ejemplo alentador se encuentra en el combate contra la piratería. Actualmente se usan códigos de seguridad para evitar la reproducción de materiales como el video fuera de ciertas zonas comerciales. Este método ya ha sido burlado, pero si se usara posicionamiento global un reproductor podría determinar en que zona se encuentra y decidir si reproduce o no el material, e incluso si el propio dispositivo funciona o no. En general se puede observar un futuro prometedor para las aplicaciones y esto ha ocasionado que los precios de este tipo de sistemas estén bajando día a día.

De esta manera existen más posibilidades de aplicar las soluciones propuestas en esta tesis como por ejemplo *geometrizar* problemas y darles una solución bajo el modelo planteado.

5.3. Comunicación inalámbrica

El descubrimiento de las ondas de radio y su aplicación en la comunicación inalámbrica ha tenido un impacto fundamental en la vida moderna principalmente porque favorece un ritmo de vida mucho más dinámico y ágil. Junto con las ondas de radio, la miniaturización y las nanotecnología han permitido ahorrar espacio, energía, materia prima y otros recursos importantes. Todas ellas se han convertido en piezas clave para el concepto de *movilidad.*

Tradicionalmente el concepto de *conexión* esta asociado a los cables y a un sentido bidireccional de la comunicación. Por ejemplo, no se dice que el aparato receptor de televisión está *conectado* a la estación emisora. Sin embargo este concepto está cambiando debido a la comunicación bidireccional inalámbrica portátil como la que usan los teléfonos celulares.

Las ondas de radio son una alternativa para la comunicación mediante cables pero su uso en dispositivos portátiles estaba limitado por la potencia que se requiere para generar las señales desde el dispositivo. Una solución es transmitir señales de baja potencia y usar baterías de alto rendimiento.

5.3.1. La salud

Una de las principales preocupaciones sobre el uso de ondas de radio es el efecto pueden llegar a tener en la salud de las personas. La exposición a las ondas de radio puede ocasionar trastornos en las células del cuerpo humano, sin embargo las condiciones para que esto suceda aún están en investigación. Debido a esta situación algunas revistas científicas dedicadas a las tecnologías inalámbricas incluyen artículos médicos relacionados a este tema. Es muy importante considerar que en un futuro próximo las ondas de radio estarán más concentradas de lo que están ahora y que los efectos que estas puedan tener en la salud tienen que ser investigados apropiadamente.

5.3.2. Alternativas

Aunque las ondas de radio resultan ser la opción número uno para comunicación inalámbrica, existen alternativas como campos magnéticos o la luz infrarroja. La emisión de luz infrarroja se puede realizar entre dos dispositivos teniendo como medio físico el aire, contrario a la fibra óptica por ejemplo donde se requiere de la fibra para transportar la luz. La desventaja es que la luz necesita una línea de visión directa, es decir sin obstáculos. La ventaja es que la potencia que se requiere es mucho menor por tratarse de enlaces dirigidos y no radiales.

La luz infrarroja está siendo usada para entornos más controlados y a distancias más cortas, por lo tanto existe la posibilidad de que coexistan estas dos tecnologías para ofrecer entornos flexibles.

5.4. Ruteo inalámbrico

En redes de computadoras el *ruteo* se define como el acto de mover información a través de la red de un dispositivo origen a un dispositivo destino.

En 1984 la ISO (International Standards Organization) en asociación con el comité consultivo para telegrafía y telefonía internacional mejor conocido como CCITT (Comite Consultatif Internationale de Telegraphique et Telephonique) produjeron un modelo de referencia para la interconexión de *sistemas abiertos* llamado OSIRM u *Open Systems Interconnection Reference Model.* El OSIRM es un estándar que promueve el desarrollo de protocolos que permitan la interconexión de sistemas que están dispuestos a la comunicación y al intercambio de información, es decir *sistemas abiertos.*

El OSIRM divide la tarea de la comunicación entre sistemas en siete capas y cada una proporciona un servicio a la capa inmediata superior y lo hace usando los servicios de la capa inmediata inferior. Como el modelo fue concebido para sistemas en general, las redes inalámbricas también son susceptibles a ser modeladas con el OSIRM. Sin embargo sus condiciones especiales pueden requerir de otros modelos más flexibles y sencillos, debido a la necesidad de optimizar recursos y que solo se puede confiar en la información local.

5.4.1. Ruteo IP

Actualmente el protocolo IP es el estándar de *facto* para redes de computadoras y por eso cuando aparecieron las primeras redes inalámbricas fue el primer protocolo que se adaptó a ellas. Actualmente esta adaptación sigue en proceso y las investigaciones suguieren que es necesario un protocolo especial y no una adaptación.

Junto con el protocolo IP se utilizan tablas para mantener las direcciones o identificadores de los participantes de la red. Esta técnica es usada por los algoritmos de ruteo basados en tablas. Las tablas generalmente contienen la asociación entre destinos y direcciones a donde mandar la información. Las tablas pueden ser creadas con varios criterios, pero en todos los casos esta información es usada por los *ruteadores* para determinar el tráfico en la red. Los ruteadores son un componente básico en este tipo de estructuras, su principal tarea es determinar las rutas mediante el control y mantenimiento de las tablas de ruteo. Otra tarea puede ser el filtrado de paquetes.

5.4.1.1. Ruteo por saltos

El ruteo por saltos es un esquema general de comunicación cuyo principio básico es avanzar poco a poco hasta llegar al destino. En el ruteo por tablas, estos avances son realizados de ruteador a ruteador verificando las tablas de ruteo de la siguiente manera:

1. Conocer la dirección del destino.
2. Localizar al ruteador que le corresponde.
3. Enviar mensaje al ruteador.
4. El ruteador verifica su tabla de ruteo.
5. Se envían los datos según la tabla, ya sea al destino o a otro ruteador.
6. Repetir desde el paso 4 hasta llegar al destino.

Es obvio que todo el sistema depende de las propiedades de las tablas, y que en principio deben contener información suficiente para comunicarse con toda la red. Por esta razón los principales problemas del ruteo por tablas son la creación y el mantenimiento de las mismas.

Cuadro 5.1: Tabla compacta del vértice b de la gráfica

Característica	Descripción	Por tablas	Geométrico
Óptimo:	Seleccionar la mejor ruta.	alta	baja
Simplicidad:	Operaciones sencillas.	pobre	alta
Eficiencia:	Aprovechar al máximo los recursos.	baja	alta
Robustez:	Ser tolerante a las fallas.	media	alta
Convergencia:	Encontrar las rutas rápidamente.	baja	alta
Flexibilidad:	Funcionar bajo condiciones diversas.	media	alta

La sexta versión de IP (IPv6) pretende resolver los problemas de la versión 4 que es más difundida. IPv6 contempla el uso de información geométrica por lo que es posible, en principio, utilizar los algoritmos geométricos diseñados y expuestos en esta tesis. Desafortunadamente en la industria las nuevas soluciones tienden a mantener compatibilidad con versiones anteriores para que se tenga una mayor aceptación aunque esta no sea la mejor solución.

5.4.2. Ruteo geométrico

Ahora que se conocen a grandes rasgos lo que es el ruteo por tablas y lo que es el ruteo geométrico, se puede realizar una comparación entre estos dos esquemas. La tabla 5.1 resumen los puntos más importantes.

Algunas otras características del ruteo geométrico son:

- **Dinamismo**. Mantienen la información local actualizada de forma muy rápida, sin la necesidad de un administrador.
- **Distribución del tráfico**. Al usar más de una ruta para un par de vértices se distribuye el tráfico. Esta característica se deriva del hecho de no almacenar información en los vértices y a su alta movilidad.
- **Igualdad**. Todos los participantes son iguales y actúan de la misma manera, cualquiera puede realizar la función de ruteo.

Entre otras cosas el ruteo geométrico es una solución distribuida donde la tarea del ruteador recae sobre cada uno de los participantes contrario al esquema de tablas tradicional donde los ruteadores se encargan precisamente de esta tarea.

La tabla 5.1 muestra que en la mayoría de los rubros los algoritmos geométricos superan a los basados en tablas sin embargo no alcanzan la optimalidad en el tipo de rutas que producen. Sin embargo, tomando en cuenta que existen problemas de tráfico cuando se

usan rutas cortas, se puede considerar una ventaja el hecho de que no puedan encontrarlas, por lo menos bajo el criterio común de rutas cortas.

En un balance general, el ruteo geométrico tiene más oportunidad de aplicarse en las nuevas tecnologías de comunicación y posiblemente cuando alcance un punto adecuado en su desarrollo podrá aplicarse a las tecnologías tradicionales que aún sigan en uso.

5.4.3. Futuro para el ruteo geométrico

Existen dos formas de dar solución al ruteo, la que adapta o extiende el protocolo IP basado en tablas de ruteo y la que usa enfoques distintos como por ejemplo el ruteo geométrico.

La aproximación geométrica supone que cada participante de la red conoce en todo momento su posición relativa en base a algún sistema de referencia, generalmente mediante dispositivos de localización, convirtiendo a las redes en gráficas geométricas.

El uso de información local es una necesidad para las redes inalámbricas y otros sistemas móviles pero puede resultar en un ahorro considerable de recursos si se adopta en sistemas convencionales.

Tomando en cuenta que existen algoritmos de ruteo que aseguran la entrega de los paquetes bajo en condiciones de movilidad, el futuro del ruteo geométrico es prometedor y más aún cuando existen otros problemas, como la coloración o el recorrido, que se ven favorecidos con soluciones geométricas derivadas del estudio del primer problema.

Actualmente se busca aplicar el protocolo IP en redes inalámbricas pero esta tendencia puede cambiar considerando que al usar gráficas geométricas se mejora el desempeño del ruteo. El ruteo geométrico, como se ha planteado en esta tesis, no contempla capas de protocolos de comunicación y como consecuencia no tiene pérdida de recursos asociada a la transición entre estas capas. Por lo tanto un protocolo geométrico basado en los algoritmos que aquí se exponen tiene mejor desempeño que los enfoques tradicionales en este sentido.

Las ventajas del ruteo geométrico son aprovechadas ampliamente por las redes de sensores y *ad hoc* y solo hacen falta detalles para que lleguen a un mercado comercial más amplio.

También existe la posibilidad de que la solución geométrica y los métodos tradicionales coexistan. Por ejemplo sí a las direcciones IP actuales se les agrega un campo que indíque su posición se consigue una red geométrica y la posibilidad de aplicar algoritmos tanto geométricos como por tablas IP.

5.5. Diseño de redes geométricas

Las redes actuales son susceptibles a formar una red geométrica permitiendo que se aplíquen las soluciones que aquí se plantean. Cuando no es posible obtener las coordenadas

reales con respecto a un sistema de referencia global, se pueden asignar coordenadas fictícias para simular una gráfica geométrica. Esto quiere decir que no es necesario utilizar dispositivos de localización.

El problema es encontrar la forma de asignar las coordenadas fictícias a los participantes de la red, es decir encontrar una buena inmersión en el plano sobre la cual se puedan aplicar los algoritmos vistos en la tesis y en algunos casos hasta poder hacer más que solo asignar coordenadas.

5.5.1. Inmersiones con rutas cortas

La idea de conocer las coordenadas de los participantes de la red es conocer las relaciones entre ellos, principalmente la distancia y dirección, para poder aplicar algoritmos en base a ellas. Sin embargo existen otras posibilidades, por ejemplo asignar coordenadas que no representan en realidad su posición sino la forma de rutear desde el vértice.

La idea expuesta en [38] muestra que es posible codificar una tabla de ruteo dentro de las coordenadas de cada vértice, usando cálculos y almacenamiento de alta precisión. El método básicamente consta de los siguientes pasos:

1. Llevar la red a una inmersión en forma de malla.
2. Calcular la tabla de ruteo de cada vértice mediante algún algoritmo conocido.
3. Codificar cada tabla dentro de un número menor a ϵ.
4. Desplazar cada vértice en la cantidad que codifica su tabla.

El desplazamiento debe ser menor a un número ϵ el cual indica la cantidad mínima para desplazarse sin que la gráfica pierda su forma. Con un límite lo suficientemente grande en la cantidad de decimales usados para representar un número, es posible codificar cualquier cantidad de información en un número real. Entonces el algoritmo calcula las tablas, las codifica en un número real menor que ϵ y desplaza los vértice en esta cantidad. De esta manera los vértices contienen la información sobre el ruteo en sus propias coordenadas.

Para aplicar semejante esquema es necesario que los dispositivos sean capaces de realizar operaciones aritméticas de alta precisión, además de que la información acerca de la gráfica tiene que ser conocida en su totalidad antes de usar el método. En el caso de dispositivos móviles esto es casi imposible actualmente. Lo que se pretende mostrar aquí es el hecho de que existen métodos que llevan redes a proyecciones en gráficas geométricas donde se pueden aplicar los resultados generados en esta tesis e incluso se pueden complementar.

5.5.2. Triangulaciones regulares

En [38] se expone un método que crea gráficas geométricas donde es posible aplicar ruteo por brújula, el cual solo funciona en triangulaciones regulares. El método construye un

politopo en base a la topología de la gráfica y después lo proyecta en el plano para obtener una triangulación regular.

En este método también es necesario conocer la topología de la red para calcular la inmersión, asignar las coordenadas fictícias a cada nodo y generar la gráfica geométrica. Una vez con la información geométrica es posible usar algoritmos geométricos sobre la red.

5.6. Seguridad

Una de las ventajas de usar memoria constante es la seguridad que implica llevar muy poca información. Por ejemplo, algunos algoritmos de ruteo no necesitan recordar donde se originó el ruteo por lo tanto si un agente es capturado antes de llegar al destino no hay manera de conocer quien envió el mensaje. Los vértices no almacenan datos sobre los agentes que pasan por allí, por lo tanto tampoco es posible rastrear al agente por los nodos que ha visitado.

Otro ejemplo es el recorrido de una gráfica mediante el concepto de arista de entrada. La arista de entrada se puede definir en base a otro sistema de referencia diferente al tradicional, el plano cartesiano, pudiendo ser cualquier otro par de líneas ortogonales. De tal manera que el recorrido de la gráfica y el orden en que se reportan los elementos de ella depende de esta elección en particular. Si un agente es capturado a la mitad del recorrido no será posible determinar donde se inició el recorrido.

Las implicaciones de seguridad que tiene un modelo con poca información son alentadoras, sobre todo al considerar que se pueden combinar con esquemas de criptografía de llave pública. Por ejemplo, si se necesita mandar la identidad del vértice origen a través del agente, se puede encriptar con la llave pública del destino de tal manera que sí es capturado no se podrá conocer al vértice origen a menos que conozcan la clave privada del destino.

Este tipo de combinaciones permiten abrir paso a la adopción de soluciones geométricas en aplicaciones que requieren altos niveles de seguridad. Por lo pronto esta es una de las direcciones a seguir en la investigación de este importante modelo.

5.7. Casos prácticos

Actualmente existen proyectos que incluyen el desarrollo de redes de sensores y las aplicaciones para manejarlos. En estos proyectos se puede observar claramente el tipo de condiciones reales bajo las cuales se diseñaron los algoritmos de esta tesis. Todos los proyectos contemplan la posibilidad de usar localizadores que permiten ver las redes como gráficas geométricas.

Llama la atención que los dispositivos (sensores) de la red cuentan con sistemas operativos propios, los cuales son altamente eficientes y pequeños. A pesar de las restricciones, estos sistemas no dejan de ofrecer servicios como: planificación de procesos, comunicación en

red y administración de memoria.

Estos desarrollos no solo apoyan el diseño de algoritmos para agentes de memoria limitada en gráficas geométricas, sino que al mismo tiempo ofrecen diseños originales de sistemas operativos, protocolos de comunicación, hardware y tecnologías de conservación de energía entre otros, por lo que resultan en su conjunto sistemas mucho muy interesantes.

5.7.1. GNOMES

GNOMES [15] es un proyecto que ofrece sensores de prueba como una herramienta para explorar los alcances de la tecnología de sensores inalámbricos y su diseño. La principal característica de GNOMES es que usa sensores heterogéneos es decir que la red está compuesta por sensores de luz, de calor, de movimiento o cualquier otra variable, la idea es que la información se complemente entre ellos. GNOMES usa tecnologías de alimentación de energía alternativas como la energía solar, el viento y el calor. Está diseñado modularmente, lo que facilita el uso de diferentes tipos de sensores, módulos de comunicación inalámbrica, módulos de posicionamiento global y de energía. Algunas de sus especificaciones son:

- Microcontrolador de 16 bits a 7.3728 MHz con 60 Kilobytes de memoria para código y 2 Kilobytes para datos, convertidores y puertos de comunicación.
- Memoria externa de 32 Kilobytes.
- Soporta múltiples fuentes de alimentación, por ejemplo energía solar, del viento, calor, etc.

Su sistema operativo ofrece concurrencia de procesos y sus controladores están programados especialmente para cada modulo incluyendo una implementación parcial de la pila de comunicación de Bluetooth.

5.7.2. MANTIS

El proyecto MANTIS[19] incluye el desarrolló de software y hardware para una red de sensores inalámbrica y está diseñado para que los programadores aprendan a trabajar con ella rápidamente. Al mismo tiempo ofrece flexibilidad en las aplicaciones facilitando la investigación de algoritmos, esquemas y protocolos referentes a redes de sensores.

Cuenta con un sistema operativo multihilos con capacidad de sincronización de entrada salida y pila para el protocolo de comunicación de red, todo con menos de 500 bytes de memoria. MANTIS ofrece la posibilidad de realizar actividades de forma remota que son de vital importancia para el mantenimiento y actualización de cualquier sistema, por ejemplo reprogramar los sensores, realizar depuraciones y simulaciones con dispositivos reales y virtuales.

Los paquetes de datos que se pueden enviar y recibir en esta red se sensores van desde los 12 hasta los 64 bytes en 30 canales de comunicación. Los sensores MANTIS tienen la capacidad de reconocer sus posiciones mediante el chip Trimble Lassen SQ que cuenta con capacidades GPS.

5.8. Para finalizar

La presente tesis expone y justifica las bases de una propuesta, los resultados más significativos en el área y resultados originales, así como las ventajas y desventajas frente a otras alternativas.

Es importante resolver los problemas computacionales haciendo un uso óptimo de los recursos no solo del tiempo de procesamiento sino también de la memoria. En este caso al agregar información geométrica es posible resolver problemas con recursos muy limitados. Los agentes de software llevan menos información consigo usando información local de manera que hay menos tráfico en la red y se gana escalabilidad.

Los algoritmos presentados en esta tesis no solo cuidan los recursos de memoria, procesamiento y comunicación sino que además no alteran su entorno. La ventaja en general es que se obtienen ambientes de comunicación y de cómputo menos saturados, más intuitivos y eficientes.

Dejando de lado el hecho de que para representar un número n hacen falta $O(\log n)$ bits, todos los algoritmos presentados utilizan una cantidad de memoria constante $O(1)$ con respecto al tamaño de la entrada es decir que son altamente escalables y son considerados como algoritmos que no usan *memoria extra*.

Estos algoritmos son diseñados bajo el modelo de agentes móviles geométricos con memoria limitada en entornos restringidos y son capaces de realizar tareas como búsquedas, exploración, cálculo de árboles generadores, ruteo, coloración, etc.

La idea en el desarrollo de estos algoritmos ha sido restringir la cantidad de memoria al mínimo exigido por cada problema y observar las capacidades de los algoritmos para después ofrecer un poco de memoria y ver hasta donde pueden llegar.

Se trabajó en el problema inverso de la complejidad computacional, es decir: ¿Cual es el problema más complejo que se puede resolver con una cantidad fija de memoria en cierto tiempo?.

Una vez que se resuelven los problemas con restricciones temporales, es conveniente dedicar igual cantidad de esfuerzo para resolverlos con restricciones espaciales y por último encontrar el punto medio entre el obvio intercambio de desempeño que existe entre restricción espacial y temporal.

Los agentes presentados son capaces de realizar sus tareas porque su entorno es geométrico resultando un complemento perfecto. Prueba de ello son los fascinantes resultados presentados, porque un agente con memoria limitada sin información geométrica difícilmente

podría hacer las tareas que se realizan con información geométrica.

Los mecanismos de posicionamiento tienen una tendencia en su costo a la baja, por lo que la solución geométrica es viable desde hace algunos años y formará parte de la vida cotidiana. Es solo cuestión de tiempo para que las aplicaciones se *geometrícen* y al mismo tiempo se empiecen a realizar aplicaciones específicamente geométricas.

Por estas razones en estos algoritmos se encuentran los estándares del mañana para tecnologías de comunicación, cómputo, entretenimiento e incluso seguridad.

Bibliografía

[1] A. Borodin and R. El-Yaniv, Online Computation end Competitive Analysis. Cambridge University Press, 1998.

[2] A. Maheshwari, J.R. Sack, H jidjev. Link Distance problems. In J.R. Sack and J. Urrutia, editors, handbook of Computational Geometry, pages 519-558. Elsevier Science, 2000.

[3] Al-khwarizimi. Kitab al jabr w'al-maqabala. c. A.D. 825. Mathematical Foundation of Programming. FS. Beckman Addison-Wesley 2ed. 1981.

[4] B. H. Tay and A. L. Ananda. A Survey of Remote Procedure Calls. Operating Systems Review, 24(3):68-79, July 1990.

[5] Carlos de Morais Cordeiro and Dharma P. Agrawal. Mobile Ad Hoc Networking. OBR Research Center for Distribuited and Mobile Computing 2002.

[6] Colin G. Harrison, David M. Chess and Aaron Kershenbaum. Mobile Agents: Are they a good idea? Technical report, IBM Research Division, T.J. Watson Research Center, March 1995. http://www.research.ibm.com/massdist/mobag.ps

[7] Coxeter, H. S. M. Introduction to Geometry, 2nd ed. New York. Wiley, 1969.

[8] D. Dobkin, S. J. Friedman, and K. J. Supowit. Delaunay graphs as good as complete graphs. Discrete and Computational Geometry, 5:399-407,1990.

[9] D. Estrin et al. New Century Challenges : Scalable Coordination in Sensor Networks. ACM Mobicom, 1999.

[10] D. S. Hirschberg and J.B. Sinclair. Decentralized extrema-finding in circular configurations of processes. Comunications of the ACM, 23(11):627-628, 1980.

[11] E. Chavez, S. Dobrev, E. Kranakis, J. Opatrny, L. Stacho, J. Urrutia, Traversal of a Quasi-Planar Subdivision Without Using Mark Bits. Journal of Interconnection Networks, Vol. 5, No. 4, pp. 395 - 408, 2004.

[12] E. Kranakis, H. Singh, and J. Urrutia. Compass routing on geometric networks. In Proc. Of 11th Canadian Conference on Computational Geometry, 51-54, 1999.

[13] E. Royer and C. Toh, A review of current routing protocols for ad-hoc mobile wireless networks. IEEE Personal Comunications, Apr 1999.

[14] Eric Jul, Henry Levy, Norman Hutchinson and Andrew Black. Fine-Grained Mobility in the Emerald System. ACM Transactions on Computer Systems, 6(1):109-133, February 1988.

[15] Erik Welsh, Walt Fish, J. Patrick Frantz. GNOMES: A Testbed for Low-Power Heterogeneous Wireless Sensor Networks. IEEE International Symposium on Circuits and Systems (ISCAS), Bangkok, Thailand, 2003.

[16] F. T. Leighton. Introuction to Parallel Algorithm and Architectures: Arrays, Trees and Hypercubes. Morgan Kaufman. 1992.

[17] Fabian Kuhn, RogerWattenhofer, Yan Zhang, Aaron Zollinger. Geometric Ad-Hoc Routing: Of Theory and Practice. Department of Computer Science, ETH Zurich, Zurich Switzerland 2003.

[18] Godfried T. Toussaint. The relative neighborhood graph of a finite planar set. Pattern Recognition, vol. 12 no 4, pp.216-268, 1969.

[19] H. Abrach, S. Bhatti, J. Carlson, H. Dai, J. Rose, A. Sheth, B. Shucker, J. Deng, R. Han. Mantis: System support for MultimodAl NeTworks of In-situ Sensors. WSNA 2003, September 19, 2003, San Diego California, USA.

[20] J. A. Bondy and U. S. R. Murty. Graph Theory with Applications. Elsevier/North-Holland, 1976.

[21] J. M. Khan. New Century Challenges : Mobile Networking for Smart Dust. ACM Mobicom, 1999.

[22] J. M. Kleinberg. On-line algorithms for robot navigation and server problems. Master thesis, MIT, Cambridge, MA, 1994.

[23] J. van Leeuwen and R. B. Tan, Routing With Compact Routing Tables, Tech. Rep. RUU-CS-83-16, Dept. of Computer Science, Ultrecht University (1983). Also as : computer Networks with Compact Routing Tables, in G. Rozenberg and A. Salomaa (Eds.) The book of L, Springer-Verlag, Berlin (1986) pp. 298-307.

[24] James E. White. Mobile Agents. Technical report, General Magic, Inc., October 1995.

[25] James Gosling, Bill Joy y Guy Steele. The Java Language Specification. Addison-Wesley, August 1996.

[26] James W. Stamos y David K. Gifford. Remote Evaluation. ACM Transactions on Programming Languages and Systems, 12(4): 537-565, October 1990.

[27] Jurek Czyczowicz, Evangelos Kranakis, Nicola Santoro, Jorge Urrutia. Traversal of Geometric Planar Networks using a Mobile Agent with Constant Memory. en preparación.

[28] K. Gabriel and R. Sokal. A new statistical approach to geographic variation analysis. Systematic Zoology, 18:259-278, 1969.

[29] Leonhard Euler. The Seven Bridges of Königsberg. World of Mathematics Vol I p 573. Simon and Schuster, New Work 1956.

[30] Machtey Michael. Introduction to the general theory of algorithms. 2ed 1978. Elsevier North Holland, New York.

[31] Mahtab Seddigh, J. Solano Gonzales, and I. Stojmenovic, RNG and internal mode based broadcasting algorithms for wireless one-to-one networks", ACM Mobile Computing and Comunications Review, vol. 5, no. 2, pp. 37-44, 2002.

[32] N. Santoro and R Khatib. Routing Without Routing Tables. Tech. Rep. SCS-TR-6, School of Computer Science, Carlenton University (1982). También como : Labelling and Implicit Routing in Networks, Computer Journal 28 (1), (1985), pp.5-8.

[33] P. Boone, E. Chavez, L. Gleitzky, E. Kranakis, J.Opatrny, G. Salazar, J. Urrutia, Morelia Test: Improving the Efficiency of the Gabriel Test and Face Routing in Ad-hoc Networks, Proceedings of SIROCCO 2004, LNCS 3104, pp. 23-34, 2004.

[34] P. Bose and P. Morin. An Improved algorithm for subdivision traversal without extra storage. Internat. J. Comput. Geom. Appl. 12(4):297-308, 2002. Annual International Symposium on Algorithms and Computation (Taipei 2000).

[35] P. Bose, P. Morin, I. Stojmenovic and J. Urrutia, Routing with guaranteed delivery in ad hoc wireless networks, 3rd int. Workshop on Discrete Algorithms and methods for mobile computing and communications, Seattle, August 20, 1999.

[36] P. Cucka, N. S. Netanyahu, and A. Rosenfeld. Learning in navigation : Goal finding in graphs. International Journal of Pattern Recognition and Artificial Intelligence, 10(5): 429-446, 1996.

[37] P. Ružička, On Efficiency of Interval Routing Algorithms, in M.P. Chytil, L. Janiga, V. Koubek (Eds.) Mathematical Foundations of Computer Science 1988, Springer-Verlag LNCS 324 (1988) pp. 492-500.

[38] Patric Ryan Morin. Online Routing in Geometric Graphs. Ph.D. Thesis, Ottawa, Ontario, 2001.

[39] Petros Zerfos, Gary Zhong, Jerry Cheng, Haiyun Luo, Songwu Lu and Jeffrey Jia-ru Li. DIRAC: A Software-based Wireless Router System. ACM MOBICOM (International Conference on Mobile Computing and Networking) 2003 pp. 230-244, San Diego, California, September 2003.

[40] Prosenjit Bose and Pat Morin. An Improved Algorithm for Subdivision Traversal without Extra Storage. International Symposium on Algorithms and Computation, pp 444-455, 2000.

[41] R. Min, M. Bhardwaj, S. Cho, A. Sinha, E. Shih, A. Wang, and A. P. Chandrakasan. An Architecture for a Power-Aware Distributed Microsensor Node. IEEE Workshop on Signal Processing Systems (SiPS '00) pp. 581-590, October 2000.

[42] Richard B. Tan and Jan van Leeuwen. Compact routing methods: A survey. In Proceedings of Colloquium on Structural Information and Communication Complexity (SICC'94), SCS, Carleton University, Ottawa, pages 99–109, 1995.

[43] Ronald L. Grahamm Donald E. Knuth, and Oren Patashnik, Concrete Mathematics. Addison Wesley, 2nd edition, 1994.

[44] Sunil Ayra, Gautam Das, David Mount Jeffrey Salowe, and Michael Smid. Euclidian spanners: short, thin and lanky. Proc 27th ACM STOC, 1995, pp. 489-498.

[45] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. Introduction to algorithms. The MIT Press, 2000.

[46] Xiang Yang Li. Ad Hoc Wireless networking X. Cheng, X. huang and D. Z. Du Editors. 2003 Kluver Academic Plublishers.

[47] Xiang Yang Li, G. Calinescu, and Peng Yun Wan. Distribuited construction of planar spanner and routing for ad hoc wireless networks. 21st Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2002, vol 3.

[48] Xiang Yang Li, Ivan Stojmenovic, and Yu Wang. Partial delaunay triangulation and degree limited localized bluetooth scatternet formation. in AdHocNow 2002.