VNIVERSIDAD NACIONAL AVTNOMA DE MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"ENCUENTRO DE VARIOS AGENTES UTILIZANDO MEMORIA CONSTANTE"

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN CIENCIAS

(COMPUTACIÓN)

P R E S E N T A:

ARELI ROSAS NAVARRETE

DIRECTOR DE TESIS: DR. JORGE URRUTIA GALICIA

México, D.F. Junio de 2006.

Al Señor

"Porque todas las cosas proceden de él, y existen por él y para él.

¡A él sea la gloria por siempre! Amén. "

Romanos 11:36

Índice general

Índice de figuras

Índice de cuadros

Capítulo 1

Introducción

Considérese la situación siguiente: un par de agentes se encuentra en dos nodos distintos de una red y quieren reunirse. Supongamos que están en los vértices u y v de la red de la figura 1.1.

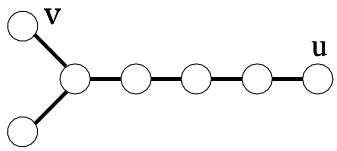

Figura 1.1: Dos agentes en una red

Si se desea utilizar sólo **información local**, es decir información sobre los vecinos inmediatos ¿Será posible desarrollar una estrategia que al ser utilizada por ambos agentes, garantice que en un tiempo finito se encontrarán? Considerando que cada nodo dispone de una lista de vecinos, por medio de ella un agente con memoria constante puede saber el grado del nodo en que se encuentra y es capaz de viajar por la red de la siguiente forma: si ha llegado al nodo por la arista i, saldrá del mismo por la arista $i + 1$ de la lista; la figura 1.2 muestra con flechas el desplazamiento de un agente utilizando este criterio, que es un desplazamiento que utiliza información local y memoria constante.

En las condiciones arriba descritas, el agente está en posibilidad de saber si se encuentra en una hoja (nodo de grado uno) y en el caso de la figura 1.1, podrá detectar al único nodo de grado tres, puesto que un bit de memoria bastará para detectar si ha pasado dos veces por un nodo de grado tres sin haber visitado una

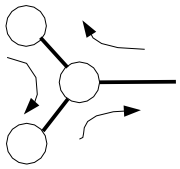

Figura 1.2: Ruta de un agente con memoria constante

hoja antes. Así, los agentes podrán encontrarse usando una estrategia de memoria constante.

En el ejemplo vimos que para responder a la pregunta fue necesario hacer suposiciones sobre los recursos de los agentes (cantidad de memoria), si pueden o no dejar marcas a lo largo de su camino e incluso considerar la forma de la red (topología). La estrategia utilizada se basa en la existencia de un punto identificable de forma única, pero en una red como la de la figura 1.3 no hay tal.

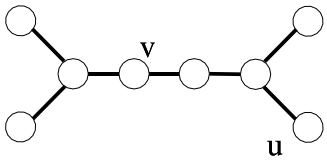

Figura 1.3: Una red simétrica

Una estrategia que consista únicamente en viajar por la red de la forma arriba descrita, reunirá a los agentes del ejemplo cuando se encuentren en una red como la mostrada en la figura 1.3 siempre que la distancia inicial entre ellos sea menor a tres. Cuando comienzan a distancia tres, colocados en una posición como la que muestran los nodos marcados como u y v o una posición isométrica a ésta, cabe la posibilidad de que uno de los agentes comience a viajar siempre detrás del otro y el encuentro no podrá llevarse a cabo.

Para enfrentar esta situación se requiere de un cambio de estrategia; si proporcionamos a cada agente una bandera que pueda colocar en un nodo o desplazar a voluntad, el encuentro se podrá efectuar aplicando el siguiente algoritmo:

1. Desplazarse por la red hasta encontrar un nodo de grado tres, una vez alcanzado este nodo, poner la bandera en él. Si ya hay otra bandera, permanecer en el nodo, si no, ir al paso 2.

2. Continuar desplazándose por la red hasta alcanzar un nodo de grado tres, sin que se haya pasado antes por una hoja, entonces, ir al paso 3.

3. Salir de ese nodo por la arista de entrada y continuar el recorrido hasta encontrar una bandera.

4. Al encontrar la bandera, repetir desde el paso 3 hasta encontrar al otro agente.

El algoritmo anterior efectúa el encuentro debido a que produce una de estas dos situaciones, dependiendo de las posiciones iniciales de los agentes:

- Dos banderas son depositadas en uno de los nodos de grado tres, creando una diferencia entre ellos. Así, los agentes podrán distinguir a uno y elegirlo como punto de encuentro.

- Ambos nodos de grado tres tienen una bandera, entonces, aunque no es posible distinguir entre ellos, el algoritmo mantendrá a los agentes viajando en el segmento sin hojas que los comunica, de este modo, el encuentro es inevitable.

Hasta aquí hemos podido apreciar la dificultad que conlleva la utilización de información local y memoria constante; también hemos visto que para proponer una estrategia efectiva, es importante considerar factores como la topología de la red y la disponibilidad de recursos alternos como las banderas.

Por último, considérese una red como la de la figura 1.4 y una pareja de agentes en las posiciones indicadas por u y v. Al igual que en el ejemplo anterior, ambos agentes *ven* exactamente lo mismo, sin embargo, en esta situación no importa de cuántos recursos se les provea, el encuentro no será posible.

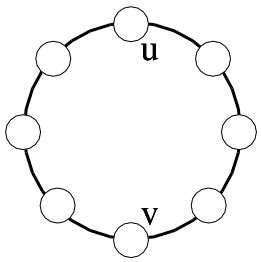

Figura 1.4: Anillo de ocho nodos

La afirmación del párrafo anterior se debe a que al aplicar la misma estrategia en una situación como la arriba mencionada, la estrategia se traducirá en una serie de movimientos simétricos. Sin importar de cuántos recursos (memoria, procesamiento,etc) dispongan los agentes, ambos tendrán exactamente la misma *vista* desde dos lugares distintos y como utilizan la misma estrategia, compartirán el contenido de la memoria y el resultado de todo cálculo que realicen será igual para ambos, como si sólo se tratara de un agente, por lo tanto, llevar a cabo el encuentro es imposible. Ocurre algo semejante con k agentes en un anillo de n nodos si éstos pueden colocarse a una distancia exacta de n/k nodos, ésta es una situación simétrica. Puntualizando, llamaremos **situación simétrica** a aquella en la que el entorno se ve exactamente igual desde lugares diferentes, lo que hace imposible distinguir en cuál de estas ubicaciones se está.

El presente trabajo tiene como objetivo análizar la posibilidad de reunir k agentes en una red de anillo. Ésta clase de problemas se denominan búsquedas de reunión o encuentro. El paradigma de búsqueda de reunión[1] (encuentro) fue planteado en 1960 por Schelling[13] y consiste en reunir, en el menor tiempo posible, un par de individuos que se han separado de forma imprevista, tomando en cuenta las características del medio donde se hallan y los recursos a su disposición.

Éste es un problema de búsqueda óptima y debido a ello fue analizado desde la perspectiva de investigación de operaciones, matemáticas aplicadas y estadística. Tiempo después se convirtió en objeto de interés para teoría de juegos, y ahora la búsqueda de encuentro ha entrado en el dominio de la teoría de la computación al surgir como un reto algorítmico en las redes de computadoras.

Como es natural, los actores del problema han sido denominados de diferente forma, dependiendo de las cualidades que cada autor les atribuye. En este documento serán considerados agentes, pues es un concepto abstracto que permite incluir gran cantidad de posibilidades.

El grado de dificultad que la solución del problema requiere, depende de los siguientes factores:

- Características del lugar donde se efectúa la búsqueda.
- Número de agentes participantes.
- Información que poseen.
- Información que es posible adquirir en el transcurso de la búsqueda.

[1] Del inglés *rendezvous search problem*

- Capacidad de procesamiento de los agentes en cuanto a complejidad de los cálculos que pueden hacer.
- Cantidad de memoria de que dispone cada agente.
- Existencia de medios de coordinar el proceso o comunicarse entre si.
- Si hay un límite de intentos o tiempo para encontrarse.

Para hacer un análisis que resulte útil, es necesario fijar los valores de algunos de estos parámetros. Una importante delimitación al problema viene de definir el lugar de la búsqueda. Para el presente trabajo se ha elegido el **anillo**, que es un entorno discreto compuesto por nodos conectados entre sí formando un ciclo. El objetivo es enunciar un algoritmo que pueda reunir k agentes en un anillo, dado que éstos tienen una cantidad constante de memoria.

En el capítulo 2 encontraremos además de otros trabajos de investigación relacionados, el enunciado del problema, la exposición de conceptos importantes, la justificación de las condiciones de trabajo consideradas y cuatro puntos que permiten definir claramente un problema de encuentro. El orden de presentación de los antecedentes corresponde a distintas fases de *progreso* en la investigación del problema hacia el modelo que presentaremos en la sección 2.4.

En el capítulo 3 se hace un análisis sobre los cambios en la forma de los algoritmos propuestos por Flocchini et al. [7], mientras se va reduciendo la cantidad de memoria utilizada.

El núcleo de resultados de investigación se encuentra en el capítulo 4, donde se estudia un algoritmo de memoria constante; se demuestra que éste consigue realizar el encuentro para $k = 2, 3$ y se realiza un cálculo exacto del tiempo de encuentro para estas dos versiones del problema. Finalmente es propuesta una conjetura con base en los resultados de simulación obtenidos con el programa que se incluye en el apéndice.

Para conseguir el encuentro se han desarrollado diferentes técnicas, la mayoría de las cuales involucran procesos aleatorios, cierto conocimiento del medio en que se efectúa la búsqueda y están pensadas para dos agentes. De ahí la importancia de este estudio, pues aquí se observa cómo las estrategias orientadas hacia varios agentes son más complicadas de analizar y la dificultad del análisis aumenta conforme la estrategia se hace más sencilla y la memoria con que cuentan los agentes disminuye.

Capítulo 2

El problema del encuentro

El enunciado de nuestro problema es el siguiente: "¿Qué deben hacer k agentes que se encuentran separados en un anillo de n nodos para reunirse en el menor tiempo posible dado que todos se mueven a velocidad unitaria?".

Para poder comprender el problema y evaluar de manera uniforme los trabajos de investigación relacionados será necesario introducir algunos conceptos.

En la siguiente sección se hablará sobre las condiciones de simetría en las cuales se trabaja, cómo éstas incrementan la dificultad en la solución del problema y la razón por la que estas condiciones son establecidas. En la sección 2.2 serán definidos los parámetros con los cuales se hará la revisión de antecedentes en la sección 2.3. Estos trabajos fueron seleccionados por haber sentado bases para la investigación que nos concierne. Cabe mencionar que se ha dejado para el próximo capítulo un último trabajo de investigación, pues con él se hará un análisis y comparación diferente debido a las similitudes que guarda con el presnte estudio. Para terminar este capítulo, en la sección 2.4 es presentado el conjunto de suposiciones bajo el cual se trabaja. Ésta es la descripción puntual del problema.

2.1. Simetría

Supongamos que cierta galería de arte alberga una exposición donde las salas están dispuestas de modo secuencial, el recorrido sólo se puede realizar en una dirección sin retroceder, con un único acceso en la primera sala, que es al mismo tiempo, el término del recorrido (ver figura 2.1).

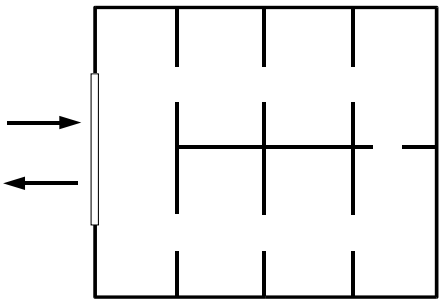

Figura 2.1: Exposición secuencial

Una pareja visitante que se separa en estas circunstancias, podrá reunirse con facilidad si el miembro que se adelantó espera a su compañero en la salida.

La situación que planteamos en el párrafo anterior se puede dar en el contexto de redes de computadoras, cuando un par de agentes móviles quieren reunirse en algún punto de la red. Una red de anillo unidireccional con un nodo *distinguido* resultaría parecida a nuestro esquema de galería. Los nodos del anillo corresponden a las salas y las flechas entre ellos, son como las puertas que comunican una sala con la otra. El nodo distinguido sería equivalente a la sala donde está el acceso.

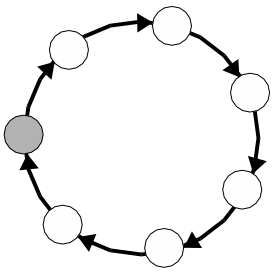

Figura 2.2: Red de anillo unidireccional

Identificamos a un nodo como *distinguido* en la red, ya sea por su función o mediante un identificador, lo mismo que las salas de nuestra galería albergan diferentes obras de arte y pueden tener nombre.

Ahora veamos el impacto de un cambio en el área de búsqueda. Supongamos que un nuevo expositor ha dispuesto de forma simétrica la galería y no estableció restricciones sobre la forma de hacer el recorrido. Nuestra pareja de amantes del arte se separa de nuevo. En estas circunstancias no podrán encontrarse del mismo modo que la vez anterior (ver figura 2.3). Este es un buen momento para tomar en cuenta

otros factores como la posibilidad de comunicarse con el otro o la información que nuestros agentes poseen de antemano.

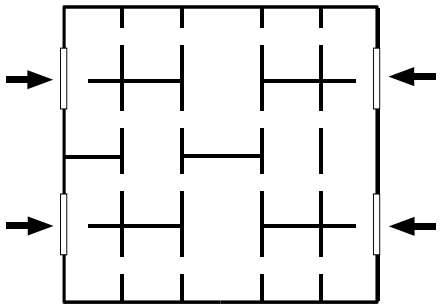

Figura 2.3: La exposición simétrica

Si cuentan con teléfonos portátiles, lo mejor sería utilizarlos para ponerse de acuerdo y elegir una sala para reunirse. De no ser así, podemos considerar la posibilidad de que ambos sepan que están interesados en la misma pieza de la exhibición, entonces podrían dirigirse hacia la sala donde ésta se encuentra y esperar ahí por su compañero. Aquí es importante considerar que existen factores que pueden retrasar el momento del encuentro, pues no hemos hablado de cómo harán para llegar a la sala destino. Puede ser que tengan un mapa donde se indique la distribución de las piezas en la exposición, que puedan preguntar a alguien o que parte de la visión del artista sea desorientar a los asistentes al no proporcionar información ni mapas y así causarles la sensación de estar perdidos.

La figura 2.3 ilustra un caso de navegación particularmente difícil, mismo que enfrentaremos en el anillo. Un nodo identificable en esta topología de red, correspondería a la pieza única de interés común. La existencia de control central o alguna forma de comunicación entre los agentes sería equivalente a que los visitantes contaran con teléfonos móviles. Sin embago, es importante desarrollar algoritmos que no hagan uso de estos recursos por tres razones:

- No siempre se cuenta con un nodo distinguido.
- No es recomendable usar los nodos distinguidos, pues en caso de fallo o ataque no será posible llevar a cabo el algoritmo.
- Por cuestiones de balanceo de carga en una red, es importante que los algoritmos consideren a todos los nodos iguales y que no se dependa de un control central.

Para nuestro ejemplo el no hacer uso de un control central, de algún medio de comunicación entre agentes o características de los nodos, equivaldría a tener una galería vacía con todas las paredes pintadas del mismo color. Con una distribución simétrica como la que vemos en la figura 2.3, fácilmente perderíamos el sentido de orientación sin poder distinguir las salas de sus homólogas en el extremo contrario y por lo mismo no podríamos elegir de forma unívoca un punto de encuentro.

Un **entorno simétrico** es un ambiente dispuesto de tal forma que desde puntos distintos se tiene la misma vista, por ejemplo una galería vacía de paredes blancas cuyas salas estén distribuidas de la forma que muestra la figura 2.3.

Como consecuencia de las nuevas restricciones de información, habrá varias dificultades de navegación. En un entorno simétrico como el anillo además de conocer la disposición de las salas, es necesario orientarse a fin de llevar control sobre dónde estamos, hacia dónde queremos ir y como llegar ahí. Un mapa equivaldría a tener información global en una red. En general, es mejor no hacer uso de esta información, pues así damos margen a que se modifique la cantidad e identidad de los nodos que conforman la red durante la ejecución del algoritmo sin afectarlo. En estas circunstancias, es posible que se desee poner marcas en las paredes para distinguir un lugar en donde ya se ha estado, sin embargo, esto no es recomendable debido al volumen de tráfico en las redes, pues si todos hicieran lo mismo agotarían los recursos de memoria del nodo almacenando cada uno sus marcas en ellos. Las marcas también equivalen a un medio de comunicación entre los agentes.

Con todas estas restricciones es necesario producir la asimetría por medios artificiales. Una forma de hacerlo es coordinando las actividades de los agentes e instruirlos para que actúen de diferente forma. Sin embargo la naturaleza autónoma de los agentes nos lleva a buscar que éstos sean capaces de realizar sus tareas, sin depender de las instrucciones de un control central o líder, utilizando sólo los recursos propios. De aquí la importancia de disponer de algoritmos que produzcan asimetrías o encuentren asimetrías de forma independiente para cada agente.

El anillo sin etiquetar y considerando iguales a todos los nodos, se convierte en un entorno homogéneo es decir, el agente ve exactamente lo mismo desde cualquier nodo, ante esta situación muchos trabajos de investigación apostaron por producir asimetría diferenciando a los participantes, o valiéndose de algún fenómeno aleatorio que sirviera de guía a las acciones de éstos. Dado que no siempre es posible hacer lo primero y los procesos aleatorios implican un grado de incertidumbre, se ha optado por buscar otros medios como las marcas en el nodo de inicio.

2.2. Otros conceptos importantes

Todas las acciones en un algoritmo de encuentro tienen como objetivo el aproximar a los agentes hasta una distancia menor o igual al radio de detección definido. Esto puede conseguirse al dejar fijo a un agente mientras el resto se va aproximando a él. También pueden mantenerse todos en movimiento y disminuir las distancias entre ellos paulatinamente hasta reunirlos.

En teoría de juegos se llama **estrategia** a un criterio que permite al jugador decidir qué acción realizar. Normalmente, tomará la decisión con base en la situación en la que se encuentra. Una estrategia es llamada **pura (simple o uniforme)** cuando el criterio de decisión es determinista. La estrategia se llama **mixta** si la decisión se basa en una distribución probabilística entre todas la acciones posibles. Un algoritmo puede llamarse mixto o uniforme, bajo los mismos criterios. Cuando todos los agentes son iguales, es necesario que todos ejecuten el mismo algoritmo, no hay modo de hacer que ciertos agentes ejecuten otro diferente. Esta es la simetría de agentes que Alpern[1] llamó estrategia de encuentro **simétrica** .

El aplicar de forma síncrona una estrategia simétrica pura en un entorno simétrico, significa correr el riesgo de que todos los agentes mantengan constante la distancia entre ellos sin llegar nunca a encontrarse. Esto se vería como un bloque de soldados en un desfile militar, todos moviéndose pero conservando siempre la formación original. Es ésta la simetría que queremos evitar.

Ya se mencionó que el anillo es un entorno discreto; a fin de facilitar el análisis del problema en éste, se hacen los siguientes supuestos:

- El tiempo está dividido en segmentos iguales (tiempo discreto).
- Los agentes viajan a la misma velocidad.
- Los agentes tardan el mismo tiempo siempre en viajar entre cualquier pareja de nodos conectados entre sí.
- El traslado entre nodos y las operaciones que realicen en ellos toman una unidad de tiempo.

Los últimos tres puntos describen lo que en adelante llamaremos **sincronía**.

A continuación se enlistan cuatro parámetros que pueden utilizarse para describir un problema de búsqueda de reunión específico. Con base en ellos serán considerados los resultados de investigación que analizaremos en la sección 2.3 y nos auxiliaremos de esta caracterización para contrastarlos también.

Entorno de la búsqueda. Lejos de existir una solución general, los trabajos de investigación se circunscriben a ciertos ambientes como la línea, gráficas con ciertas características, el círculo, etc. A la *forma* del terreno de búsqueda y otras características peculiares del mismo, le llamaremos entorno de búsqueda.

Características de los agentes. Se refiere a las *cualidades* de éstos en cuanto a recursos de memoria, información, capacidad de cálculo, velocidad, etc. Éstas pueden ser diferentes o iguales para todos.

Características de la estrategia. Ésta puede ser pura, mixta, simétrica o antisimétrica.

Espacio discreto o continuo. Esto se refiere a la forma de considerar el entorno (como un espacio continuo o formado por secciones intercomunicadas) y el tiempo (continuo o dividido en segmentos de la misma longitud).

2.3. Antecedentes

Un trabajo pionero sobre la búsqueda de reunión es el de Schelling[13], quien habla de cooperación entre los participantes y apuesta por la existencia de lo que llama **puntos focales**, también conocidos como puntos de Schelling. Éstos son puntos en el terreno de la búsqueda con características especiales que los distinguen entre los demás de forma única, de modo que con una elección independiente los agentes pueden elegir el mismo. A este enfoque se le llama **juego de un sólo tiro** porque consiste en la elección del punto y el resultado puede ser el encuentro o no. En nuestro paradigma de búsqueda, ésta no termina hasta efectuado el encuentro, además en un anillo anónimo no podemos hacer uso de este tipo de estrategias.

El trabajo de Anderson y Weber[3] nos resulta muy interesante porque le dan un nuevo giro al problema proponiendo las siguientes condiciones:

- El entorno de búsqueda son n lugares idénticos, entre los cuales los agentes pueden viajar de forma instantánea.

- Los agentes comienzan la ejecución del algoritmo al mismo tiempo, son idénticos y deben usar la misma estrategia.
- El tiempo se encuentra dividido en segmentos tales que cada agente puede visitar sólo un lugar en cada segmento.
- La estrategia se lleva a cabo por etapas que duran cierto número de segmentos de tiempo. La duración de las etapas debe ajustarse; los autores recomiendan etapas de $n-1$ unidades de tiempo para obtener el mejor resultado. En cada etapa el agente decide de acuerdo a una probabilidad dada, entre quedarse estático o moverse hacia un lugar no visitado.

Este trabajo difiere del nuestro en que los autores (Anderson y Weber) para evitar la simetría, proponen una estrategia mixta para dos agentes.

De entre los resultados presentados por los autores, resultan de interés para este trabajo de investigación los dos siguientes:

- Si uno de los agentes queda estático mientras el otro busca entre los lugares en orden aleatorio y sin repetición, el tiempo esperado para el encuentro es al menos $\frac{(n+1)}{2}$.
- Cualquier algoritmo de encuentro para dos agentes, basado en mensajes, tendrá como mínimo el tiempo esperado para que un agente encuentre un mensaje. De esta forma los autores establecen que el menor tiempo posible de encuentro de cualquier algoritmo basado en mensajes no puede ser menor que el mínimo tiempo necesario para que un agente encuentre un mensaje dejado por otro, este valor se denota como S. El valor de S se calcula como la suma de la probabilidad de no haber visitado ningún lugar en común desde el primer paso del algoritmo hasta el paso $n/2$, considerando que en cada paso del algoritmo un agente elige al azar y sin repetición visitar un lugar. El valor de S se puede escribir como la siguiente fórmula, donde n es el número de lugares donde se realiza la búsqueda y k es el número de paso del algoritmo.

$$S = \sum_{k=1}^{n/2} \left\{ \prod_{j=1}^{k} \left(1 - \frac{k}{n-j+1}\right) \right\}$$

De las variaciones en las condiciones del problema que los autores proponen, nosotros hemos trabajado con las dos siguientes:

Estructura en los lugares de búsqueda. Ellos plantean la necesidad de determinar si proponer una estructura para los lugares de búsqueda afectaría la resolución del problema; ¿cambiarían en algo las circunstancias si éstos estuvieran conectados entre sí formando un ciclo de arcos dirigidos?. En el caso de tres lugares, esta configuración no reduce el tiempo esperado de reunión, sin embargo, los autores dejan abierta esta pregunta para un mayor número de lugares.

Sincronización. Considerar una sincronía de movimientos entre ambos jugadores facilita el análisis, aunque no la resolución del problema. Debemos tener cuidado con este tipo de suposiciones al momento de la aplicación de las estrategias, pues la sincronía no es una circunstancia natural en las redes de computadoras. Los autores mencionan que el excluir la sincronía de las condiciones iniciales, requerirá de mayor complejidad en las estrategias, además, la estrategia óptima dependerá del conocimiento que los agentes posean sobre la diferencia en los tiempos de inicio de la ejecución del algoritmo.

Responder a preguntas como las arriba mencionadas no es fácil y requiere de mucho trabajo de investigación. Pero, como señala Alpern[2], aún cuando los problemas que dieron origen a este campo de investigación sigan sin respuesta, se han generado problemas subsidiarios con soluciones y teoría nueva, como los trabajos que revisaremos en los párrafos siguientes.

En 1990 Yu y Yung[15] plantean el problema directamente para k agentes en una red. La red es representada por una gráfica con n nodos homogéneos. Para evitar la simetría usan algoritmos basados en caminatas aleatorias. Éstos no resultan muy interesantes debido a que pueden ser muy lentos. También proponen un algoritmo determinístico *no uniforme*, que termina siendo una forma sofisticada de tener dos tipos de jugadores, ya que producen asimetría al asignar una cadena binaria a cada agente con la garantía de que no habrá k cadenas iguales. La asignación de las cadenas se realiza suponiendo que existe un conjunto U de agentes dividido en subconjuntos de cardinalidad $k-1$, a cada uno de éstos se le asigna una cadena binaria diferente. La cadena se utiliza como *entrada* del algoritmo para decidir las acciones del agente, de forma que en algún momento al menos un agente estará actuando de forma diferente al resto y con ello se garantiza el encuentro. Además, cada uno de los agentes tiene un identificador único.

Los autores hacen el análisis considerando un posible desfasamiento δ entre el momento en que cada agente comienza la ejecución del algoritmo con respecto a otro. Llaman Δ al mayor de todos estos desfasamientos.

A continuación hacemos una pequeña reseña de dos resultados que nos intere-

san. El primero es un algoritmo determinístico $O(\Delta + n \log \frac{|U|}{k-1})$ para gráficas generales. El algoritmo utiliza un bit de la cadena binaria del agente como entrada. Dependiendo del valor de este bit el agente queda estático o recorre la gráfica. Los agentes estáticos tomarán nota del nodo inicial del agente dinámico de menor identificador; para en el próximo paso dirigirse hacia ese nodo, mientras que los agentes que hicieron el recorrido de la gráfica esperan en sus nodos iniciales. Posteriormente, los agentes reunidos en el nodo detendrán a los agentes restantes mientras éstos atraviesan nuevamente la gráfica. Esta serie de acciones se repetirá, consumiendo bit a bit la cadena de entrada (un bit por iteración), hasta llegar a la primera posición donde las cadenas de entrada son diferentes. Este algoritmo funciona porque al menos una de las cadenas de entrada de los agentes es diferente a las demás. Con ello se elimina la simetría de acciones entre los agentes.

En segundo lugar está un algoritmo aleatorio de $O(\frac{n}{k} \log n + \frac{n}{2})$ que reune k agentes en un anillo de n nodos con una probabilidad de $1 - O(\frac{1}{n^c})$ de efectuar el encuentro (para una c lo suficientemente grande). El algoritmo consiste en iniciar la caminata con una probabilidad de 0.25 en un sentido u otro, o quedarse quieto con una probabilidad de 0.5. La decisión se toma en cada paso hasta encontrar a otra unidad. Cuando dos unidades se encuentran, atraviesan el ciclo en direcciones opuestas haciendo que todas las unidades estáticas a su paso las sigan. Si encuentran otra unidad haciendo la travesía, seguirán a la unidad de identificador menor.

El trabajo de investigación de Anderson y Weber, así como el de Yu y Yung, son pioneros en tratar con la simetría bajo circunstancias parecidas a las que nosotros planteamos, sin embargo la forma en que ellos logran la asimetría es através de fenómenos aleatorios que rigen la realización de dos secuencias distintas de acciones, información de entrada dispar, identificadores únicos de los agentes o alguna otra forma de crear diferencias entre ellos. Los algoritmos presentados en este trabajo no hacen uso de ninguno de estos recursos.

En 1995 Steve Alpern[1] publica una manera de definir formalmente el problema de búsqueda de encuentro en términos suficientemente abstractos, de modo que puede utilizarse para expresar las diferentes versiones que existen. Su especificación formal consiste en definir tres aspectos que caracterizan el problema:

1. Lugar donde ocurre la búsqueda. Éste se considerará un espacio métrico compacto (X, ρ), donde ρ es la función de distancia asociada al espacio X.
2. Radio de detección δ. Es la distancia máxima a la cual es posible que un agente detecte la presencia de otro.
3. Un grupo de isometrías G que representa la información que poseen los agentes

sobre el lugar de la búsqueda. Esta información les permite ubicar su posición en un mapa común a ambos.

Con base en lo anterior, un problema de encuentro se define como un par (X, G) al cual le asociaremos un valor R **(valor de reunión)**, que es el menor tiempo esperado de encuentro que puede ser alcanzado por las estrategias permitidas. El autor hace aún otra distinción importante: cuando los agentes utilizan la misma estrategia, se llamará **valor de reunión simétrico** R^s y **valor de reunión asimétrico** R^a en caso contrario.

El autor plantea el problema en un espacio continuo y considera que la búsqueda se llevará a cabo sobre una trayectoria elegida por cada agente. En algunos casos, cuando ambos agentes eligen la misma clase de trayectorias, es posible que conserven constante la distancia entre ellos y el valor R^s sea infinito. Para evitar esta situación el autor propone la utilización de estrategias mixtas, que producen asimetría usando procesos aleatorios independientes. A pesar de que esto separa su trabajo del análisis que aquí realizamos, es importante presentar los límites para R encontrados por el autor y otros datos interesantes.

Para una red de m arcos de longitud uno, con diámetro D y radio de detección $\delta = 0$, se cumple lo siguiente:

$$\begin{array}{lrcl} 1. & V(X,G) & \leq & 2mD \leq 2m^2 \\ 2. & R^s(X,G) & \leq & 4m \\ 3. & R^a(X,G) & \leq & m \end{array}$$

El autor llama V al valor del juego de búsqueda, donde uno de los agentes evade. De las inecuaciones arriba mostradas podemos observar que, aunque V es mayor que R, no es significativamente más grande. Alpern concluye también que no es necesario moverse más rápido para encontrar a un evasor que a un compañero de reunión.

Para la gráfica cíclica, el autor utiliza el algoritmo que estudiamos con Anderson y Weber[3] sin embargo, éste viene del trabajo de Ruckle[12] que se basa en estrategias Markovianas simétricas para un juego de suma-cero, donde evasor y buscador tienen diferentes probabilidades de moverse. El análisis que nos presenta Alpern es más sencillo, pues sólo estudia la forma de minimizar el tiempo de encuentro y asigna igual probabilidad de movimiento a ambos agentes. Después presenta unos resultados de simulación comparativos entre el juego de búsqueda y el encuentro. En las gráficas de la simulación podemos observar que a mayor número de nodos la

probabilidad óptima de movimiento es mayor, aunque ésta es menor para el algoritmo de encuentro que para el de búsqueda.

Steve Alpern[1] propone varios temas de investigación futura. En este trabajo hemos tomado los puntos siguientes:

- Estudio del encuentro con k agentes, dado que la generalidad de los trabajos entonces existentes (aún ahora la mayoría) estaban pensados para dos agentes.
- El autor consideró que los agentes pueden ver toda la región de búsqueda y propone realizar un análisis para el caso en que los agentes tienen una percepción local del entorno. Para nosotros, ésto significa información local en cada nodo, de modo que no se puede calcular el número de nodos entre dos puntos elegidos sin hacer el recorrido.

Al analizar el problema para varios agentes, una cuestion importante es determinar la mejor estrategia para seguir en el momento de encontrarse. Es necesario considerar si conviene que los agentes permanezcan juntos desde ese momento en adelante, ésta es la estrategia de **fusión al encuentro** o utilicen una estrategia de **reenuentro**, que consiste en separarse después de acordar una cita futura en cierto nodo conocido para ambos agentes. Ésta es la cuestión que Thomas y Pikounis estudian en su artículo publicado en 2001[14]. Ellos analizan el problema en el contexto de una gráfica completa de n vértices (red de n nodos), sin restricciones para buscar entre ellos. En su modelo, los encuentros sólo se efectuarán en los nodos y un agente viaja entre nodos en una unidad de tiempo. Aunque los nodos se consideran homogéneos, una vez que un agente ha visitado uno, lo puede distinguir de los demás, regresar a él e incluso comunicarle a otro agente cómo llegar ahí. No se permiten acuerdos previos al comienzo de la ejecución del algoritmo y se usan estrategias simétricas. Los autores se restringen a un grupo de estrategias que llaman *estacionarias sin memoria*, las denotaremos con $\Pi^{\mathrm{sym}}_{\mathrm{nomem}}$. Este conjunto está constituido por estrategias mixtas que consisten en decidir con cierta probabilidad p si quedarse en el nodo o moverse a otro elegido al azar.

El objeto de estudio es el valor de S_k, que se define como sigue:

$$S_k = \inf_{\pi \in \prod^{\mathrm{sym}}_{\mathrm{nomem}}} T_k(\pi)$$

Donde T_k es el número esperado de movimientos hasta que se efectúa la reunión, usando la estrategia $\pi \in \prod^{\mathrm{sym}}_{\mathrm{nomem}}$. S_k será entonces el número mínimo espera-

do de movimientos, considerando que inicialmente los agentes se ubican en nodos diferentes elegidos al azar.

El hecho de utilizar estrategias sin memoria implica que un agente no recuerda nada sobre a quién ha encontrado, ni dónde y mucho menos cómo regresar a ese punto; estas condiciones nos restringen a la *fusión al encuentro* donde una vez reunidos, los agentes no se separan. Los autores prueban que la estrategia óptima de fusión en una gráfica completa de n nodos, tendrá probabilidad de quedarse fija en un nodo $p = \frac{1}{n}$. También hacen la definición formal de *estrategia óptima estacionaria sin memoria* para el problema de reunir k agentes en una gráfica completa de n vértices y muestran que en tal estrategia los agentes usan la fusión al encuentro y que $S_k < S_{k+1}$ para $k = 1, \ldots, n-1$.

De entre los resultados de simulación que estos mismos autores exponen, cabe destacar lo siguiente:

- Cuando los agentes utilizan la estrategia de reencuentro la probabilidad óptima que minimiza el tiempo esperado de reunión es la misma que la requerida al usar la estrategia de fusión. Esta probabilidad es $p = \frac{1}{n}$.
- El mejor tiempo esperado para las estrategias de reencuentro es el $20\,\%$ del tiempo esperado usando estrategias de fusión.
- En las estrategias de reencuentro, al aumentar el número de jugadores, disminuye el tiempo del encuentro de todos, sin embargo en la medida en que k se convierte en una proporcion importante de n, el convenir citas se hace difícil y la estrategia de reencuentro se comporta pobremente.

Hasta este punto todos los trabajos presentados utilizan estrategias mixtas, debido a que sólo de esta forma se evita la situación fatídica en la cual los agentes permanecen a la misma distancia durante todo el tiempo que dura la búsqueda.

Motivados por encontrar una forma de producir condiciones de asimetría que se pudieran aprovechar usando estrategias simétricas no mixtas, Baston y Gal[4] en el año 2001 publican un análisis sobre la utilización de marcas de inicio, éstas constituyen una forma de información adicional obtenida durante la búsqueda, también son el caso más simple de intercambio de mensajes entre los agentes. A ésta nueva forma del problema le llaman encuentro con marcas iniciales[1] y el mínimo valor esperado de reunión se simboliza con M. Los autores hacen el análisis del problema para dos agentes en tiempo continuo sobre la línea y el círculo orientado.

[1] *markstart rendezvous*

También lo hacen para una gráfica completa de n nodos con tiempo discreto. Resulta interesante la distinción que hacen para la forma de las estrategias: M^a para estrategias asimétricas, M^s para estrategias simétricas mixtas y M^{ps} para simétricas puras. Éstas últimas se refieren a tener un conjunto particular de instrucciones a seguir sin considerar ninguna distribución de probabilidad. Hasta ahora no habíamos visto un artículo que tratara el encuentro simétrico usando estrategias puras, pues llevarlo a cabo en esas circunstancias raramente es posible y en general no hay garantía de lograrlo a menos que cada agente conozca su ubicación precisa. Aún así, a decir de los autores, cuando se utilizan marcas de inicio es más frecuente encontrar valores finitos para M, mismos que suelen ser distintos para cada tipo de estrategia (asimétrica, pura simétrica o simétrica).

Ahora transcribimos algunas de las estrategias de los autores que resultan relevantes para nuestro problema por ejemplo, la estrategia propuesta por Baston y Gal[4] aplicada en el círculo fue utilizada por Kranakis et al.[10] para un anillo. También incluimos algunos comentarios respecto al análisis realizado en gráficas completas.

Para dos agentes en una circunferencia orientada cuando ellos conocen la distancia que los separa (d), la estrategia pura simétrica consiste en avanzar ésta distancia en el sentido de las manecillas del reloj; el agente que encontró una marca continuará en esa dirección, mientras que el otro regresará. Esto da un valor para $M^{ps} = \frac{3d}{2}$.

En la gráfica completa una estrategia pura equivale a una regla que para cada instante de tiempo le dice al agente qué nodo ocupar, esta elección dependerá de si el vértice inicial del otro jugador ha sido identificado o no. Se considera que el encuentro se efectúa sólo cuando dos agentes ocupan la misma posición. Los autores afirman que en K_n el valor de M^{ps} es infinito para $n = 2$ y para $n \geq 4$. Para $n = 3$, $M^{ps} = \frac{7}{6}$ usando la siguiente estrategia: los agentes viajan a otro nodo para determinar cuál es el único que no tiene marca inicial. El encuentro se lleva a cabo en este nodo *diferente*. La estrategia para M^s es visitar otros nodos regresando periodicamente al nodo inicial hasta encontrar la otra marca, desde entonces el agente visitará alternativamente ambos nodos con una probabilidad de moverse $p = \frac{1}{2}$.

Cabe hacer hincapié en que ninguno de los autores arriba mencionados trabajó con un modelo exactamente como el nuestro, sin ambargo, obtuvieron resultados que nos conciernen. Los cuadros 2.1 y 2.2 presentan un resumen de estos.

AUTOR(ES)	Entorno	Agentes	Espacio
Anderson y Weber	n lugares homogéneos	dos, idénticos	tiempo en segmentos (discreto)
Yu y Yung	gráfica de nodos homogéneos	k agentes con identificador	algoritmo por etapas
Alpern	red, círculo, ciclo y línea	dos agentes	continuo excepto en la red
Thomas y Pikounis	Gráfica completa de n nodos (K_n)	k agentes	discreto
Baston y Gal	círculo, línea y K_n	dos agentes	discreto para K_n

Cuadro 2.1: Versiones del problema

Anderson y Weber fueron los primeros en estudiar un ambiente discreto. Utilizando estrategias mixtas obtienen que el tiempo esperado para el encuentro cuando un agente queda estático mientras el otro busca es de al menos $\frac{(n+1)}{2}$, donde n es el número de lugares donde se realiza la búsqueda.

Yu y Yung son los primeros en considerar varios agentes en una red de computadoras modelada por una gráfica; este es un acercamiento al problema desde el área de teoría de la computación. Los autores buscan aplicar una estrategia simétrica pura, esto es, un algoritmo determinístico para todos los agentes y como resultado proponen un algoritmo de orden logarítmico.

Steve Alpern analiza una red de m arcos unitarios y compara los valores esperados de encuentro para la búsqueda de reunión y el juego de búsqueda. Para el primero ambos agentes cooperan mientras que para el segundo uno de los agentes realiza actividades de evasión. El autor muestra que el valor de encuentro no es mucho menor que el valor esperado de la búsqueda.

Thomas y Pikounis estudian y comparan dos estrategias aplicables para el caso de encuentro de varios agentes. Utilizan estrategias simétricas en el análisis del problema en una gráfica completa. En caso de no disponer de memoria, es necesario fusionar los agentes cuando se encuentran.

Baston y Gal exploran una gráfica completa en tiempo discreto, utilizando marcas en los nodos de inicio para dos agentes. Los autores son los primeros en hacer un estudio sobre estrategias puras simétricas.

AUTOR(ES)	Estrategia	Resultado de interés
Anderson y Weber	mixta, simétrica	Tiempo esperado para que un agente encuentre un mensaje dejado por otro.
Yu y Yung	pura simétrica controlada por valores de entrada	Algoritmo logarítmico para gráficas generales.
Alpern	mixta para el ciclo	El valor de reunion es ligeramente menor que el del juego de búsqueda.
Thomas y Pikounis	simétrica mixta	El valor de encuentro es una función creciente en el número de agentes para cualquier estrategia óptima.
Baston y Gal	pura simétrica	Estrategia para un círculo orientado cuando los agentes conocen la distancia que los separa y estrategia para K_3.

Cuadro 2.2: Resultados relevantes

2.4. Modelo

El entorno de la búsqueda que se ha elegido para este trabajo es una red de anillo de n nodos idénticos.

El tiempo se encuentra dividido en unidades de igual tamaño y se considera que a cualquier agente le tomará una unidad de tiempo viajar de un nodo al nodo vecino en el anillo.

Los agentes se encuentran distribuidos en distintos nodos de la red, todos son iguales y disponen de una cantidad constante de memoria. Es indispensable que conozcan el número de ellos al comenzar el algoritmo. No conocen su distribución inicial en el anillo y cada uno posee una bandera que coloca en el nodo en que inicia la ejecución del algoritmo de reunión; ésta inicia de forma paralela en todos los agentes. El algoritmo de reunión aplicado es el mismo y cada paso del algoritmo toma una unidad de tiempo.

Un **encuentro** se lleva a cabo cuando en un mismo instante de tiempo dos agentes se hallan en el mismo nodo.

Capítulo 3

Los efectos de la reducción de memoria

Todos los algoritmos de computadora tienen en común dos recursos: el tiempo que tardan en terminar su ejecución y el espacio de almacenamiento que usarán para sus datos de entrada y las operaciones necesarias. Midiendo estos dos recursos podemos realizar comparaciones entre algoritmos.

Los algoritmos que resuelven el problema del encuentro se ven afectados por la cantidad de memoria de los agentes y el tipo de operaciones que éstos pueden realizar. Estos dos factores afectan la complejidad en términos de tiempo del algoritmo.

Un agente con **memoria constante** cuenta con una cantidad de bits en memoria que es fija e independiente del tamaño del anillo o el número de agentes que ejecutan el algoritmo.

En este capítulo trataremos principalmente con agentes de **memoria logarítmica**, es decir agentes que poseen tantos bits en la memoria como el logaritmo del número de nodos en la red.

3.1. Relación entre factores

La cantidad de memoria de que disponen los agentes pone un límite sobre el grado de dificultad y el tipo de operaciones que pueden realizar; también restringe la

cantidad de información sobre el entorno que puede manejarse dentro del algoritmo para tomar decisiones o realizar cálculos.

Hemos visto que no es mucha la información útil que puede adquirirse del entorno en las condiciones que aquí se proponen, sin embargo, el utilizar marcas de inicio permite establecer diferencias en el entorno, por ejemplo, habrá banderas más alejadas de otras e incluso podría determinarse la imposibilidad de llevar a cabo el encuentro, como lo establecen Flocchini et al.[7] para un esquema de banderas fijas, donde el encuentro es posible si y sólo si la secuencia de distancias entre las banderas es aperiódica y se requiere un mínimo de $\Omega(\log\log n + k)$ memoria para cada agente. Este límite se establece con el argumento de que ésta es la cantidad mínima necesaria para que no sea posible que el estado de la memoria de un agente sea el mismo en dos nodos diferentes de una red de anillo.

En este punto ya se han descartado las estrategias basadas en diferencias entre los agentes o entre los nodos de la red, y se ha decidido utilizar banderas como recurso auxiliar, sin embargo, la cantidad de memoria disponible restringirá la forma en que éstas son utilizadas. Por ejemplo, con banderas y memoria logarítmica es posible saber si la posición inicial de los agentes es simétrica; con memoria constante, no es posible determinar tal cosa. Así, queda expuesto cómo la cantidad de memoria disponible para los agentes, afecta directamente la capacidad de adquirir información del medio.

La cantidad de memoria limita la información que es posible utilizar para hacer inferencias que permitan diferenciar un nodo de los demás o a cierto agente de los otros. Por ejemplo, con marcas de inicio es posible calcular la distancia entre los agentes y usar esto para producir asimetría o establecer si su posición inicial es simétrica; pero para poder hacer uso de toda esa información es necesario almacenarla en memoria.

El tipo de operaciones que se pueden realizar está relacionado a su vez con la cantidad de datos que es posible almacenar en memoria, si se puede guardar el recorrido o los estados anteriores del agente, se pueden hacer cálculos y proyecciones sobre la situación pasada o el estado de memoria anterior, estas operaciones suelen agilizar el desenvolvimiento del algoritmo. También dependen de la cantidad de memoria los datos con los que se pueda operar, por ejemplo, una lista de números primos, recordar una lista de valores y compararlos para tomar decisiones, calcular el punto medio de una distancia o ruta, saber si hay un número par o non de nodos en un intervalo y calcular un órden lexicográfico.

Si no se imponen restricciones sobre la cantidad de memoria, será más sencillo diseñar un algoritmo veloz de encuentro, sin embargo, esta posición no será viable

para implementar, sobre todo porque se trata de flujo de información en una red.

Un agente con memoria constante no puede recordar rutas o citas, así que su algoritmo no debe incluir cálculos o decisiones que se basen en la historia del agente.

Kranakis et al.[10] afirman que hay una relación entre la cantidad de memoria disponible y el tiempo que tarda el algoritmo en llevar a cabo el encuentro, de modo que la primera puede ser reducida a expensas del segundo.

Siguiendo esta misma idea, Flocchini et al.[7] presentan tres algoritmos que efectúan el encuentro de k agentes en un anillo síncrono de n nodos. El cuadro 3.1 fue tomado del artículo [7] y muestra la cantidad de memoria por agente que cada algoritmo requiere y el tiempo que tardará en completar la tarea.

Algoritmo	Memoria	Tiempo
1	$O(k \log n)$	$O(n)$
2	$O(\log n)$	$O(kn)$
3	$O(k \log \log n)$	$O(\frac{n \log n}{\log \log n})$

Cuadro 3.1: Tiempo esperado de encuentro y cantidad de memoria [7]

Para estos tres algoritmos, los autores consideran que los agentes conocen el número de ellos (k) y el anillo no es orientado. El algoritmo 1 consiste en dar una vuelta al anillo y almacenar las distancias que separan a los agentes entre sí, con esta información, cada agente podrá identificar al mismo nodo haciendo algunos cálculos de manera independiente de los demás. Para el segundo algoritmo la cantidad de memoria fue reducida, de modo que los agentes ya no pueden almacenar la secuencia de distancias que los separan. Con esta cantidad de memoria, completarán el algoritmo en rondas que consisten en dar una vuelta al anillo y decidir entre permanecer activo o no. Los agentes activos en la ronda i almacenarán la distancia que los separa de la i-ésima bandera y la compararán contra las demás distancias entre banderas, con base en esta diferencia los agentes que tengan más cerca a sus vecinos decidirán quedar inactivos hasta que al final sólo resta el agente que tiene más lejos a sus vecinos que los demás y será él quien concrete la reunión. Por último, en el algoritmo 3 cada agente podrá tener k contadores con los que sabrá las distancias entre banderas $\mod p$ donde p es un número primo, de esta forma, podrán los agentes distinguir cierto nodo y encontrarse en él.

Analizando las mejoras progresivas hechas a los algoritmos en el transcurso del tiempo, vemos que mientras más información pueda tener en memoria el agente, necesitará dar menos recorridos por el anillo para poder hacer alguna distinción entre los nodos o producir una asimetría entre los agentes que permita elegir un

líder y éste lleve a cabo el encuentro.

Un algoritmo de memoria constante estará restringido a operaciones que no requieran conocimiento más allá del estado actual del agente y del nodo en que éste se encuentra. De este modo el agente decidirá cómo actuar únicamente con base en el estado actual del nodo y el propio. Esto restringe las operaciones posibles a aquellas que llamaremos **operaciones básicas**: avanzar, dejar una bandera, tomar una bandera y cambiar de dirección. Un algoritmo constituido por operaciones básicas es más difícil de diseñar y requiere de ideas más complicadas para realizar la demostración correspondiente, pues ya no resulta evidente el cómo o por qué se consigue el resultado.

A partir de este punto llamaremos **bandera inicial** a aquella que el agente coloca en el primer paso del algoritmo y **bandera extremo** a la primera bandera que encuentra en la dirección en que inicialmente avanza. El conjunto de nodos y arcos comprendido entre la bandera inicial y la bandera extremo será llamado **segmento**. La bandera extremo de un segmento, será al mismo tiempo la bandera inicial para otro segmento.

3.2. Algoritmo de espera con memoria logarítmica

Algoritmo 3.1 ***Algoritmo de espera con banderas fijas***

1. *Elegir una dirección*
2. *Avanzar contando el número de nodos hasta encontrar otra bandera u otro agente.*
3. *Sea r el número de nodos hasta la bandera; si no se encontró a otro agente, esperar r^2 tiempos.*
4. *Al encontrar a otro agente fusionarse y verificar si ya estan todos juntos, de no ser así, eliminar la primera bandera encontrada y caminar hasta la próxima bandera.*
5. *Invertir la dirección*
6. *Repetir desde el paso 2.*

Este algoritmo utiliza únicamente un contador. La forma de sus operaciones está restringida al estado actual del agente y el nodo en que se encuentra, esto es

operaciones básicas, que son el tipo de operaciones que se pueden realizar utilizando memoria constante. El motivo de ello es que este algoritmo será utilizado para perfilar el comportamiento del algoritmo de memoria constante que se presenta en el capítulo siguiente.

Para determinar el tiempo esperado de encuentro, debemos considerar que el algoritmo tardará más tiempo en reunir a los agentes cuando todos comienzan a avanzar en la misma dirección, así que haremos el cálculo del tiempo esperado de encuentro para esta situación. También es importante destacar que cuando todas las distancias entre banderas son iguales, la situación es simétrica y el encuentro no se puede llevar a cabo.

Sean dos agentes A_1 y A_2 a distancia inicial de r y s. Considérese que ambos agentes comienzan viajando en el mismo sentido, llamemos b_1 y b_2 a las banderas iniciales respectivas y $s = r + c$, con $c > 0$. El tiempo esperado de encuentro cuando $c \leq 2r^2 + r$ es $O(r^2)$ y se obtiene haciendo un análisis como el mostrado en el cuadro 3.2. Podemos calcular de esta forma el tiempo esperado de encuentro para cada caso posible, sin embargo la idea que se quiere resaltar por medio de los cuadros y las figuras es que el encuentro ocurrirá en un tiempo menor o igual a $2r^2 + 2r$ cuando $c \leq 2r^2 + r$. El valor anterior se obtiene observando que en este caso el agente A_1 recorrerá su segmento, esperará r^2 tiempos y regresará a su bandera inicial, si para este momento A_2 aún no ha llegado, A_1 comenzará otra espera de r^2 durante la cual es seguro que el agente A_2 llegará y le encontrará.

Tiempo	A_1		A_2		Tiempo
	posición	dist. a recorrer	posición	dist. a recorrer	transcurrido
0	b_1	r	b_2	s	r
r	b_2	esperar r^2	$b_1 - c$	c	c
s	b_2	esperar $r^2 - c$	b_1	esperar s^2	$r^2 - c$
$r^2 + r$	b_2	r	b_1	esperar $c^2 + 2rc + c$	r
$r^2 + 2r$	b_1		b_1		Encuentro

Cuadro 3.2: Tiempo esperado de encuentro para dos agentes ($c \leq r^2$)

Explicando el contenido del cuadro, tenemos que en el tiempo cero, los agentes comienzan la ejecución del algoritmo en sus banderas, en el tiempo r uno de los agentes, digamos A_1, alcanzará b_2 su bandera extremo, entonces esperará en ese nodo r^2 tiempos, mientras tanto en el tiempo $r + c$, A_2 llegará a b_1 y comenzará su espera de $s^2 = r^2 + 2rc + c^2$ tiempos. En el tiempo $r^2 + r$ terminará la espera de A_1, mientras que al agente A_2 le restan $c^2 + 2rc + c$ tiempos de espera en b_1. De lo anterior es claro que en el tiempo $r^2 + 2r$, cuando A_1 regrese a b_1 encontrará al agente A_2.

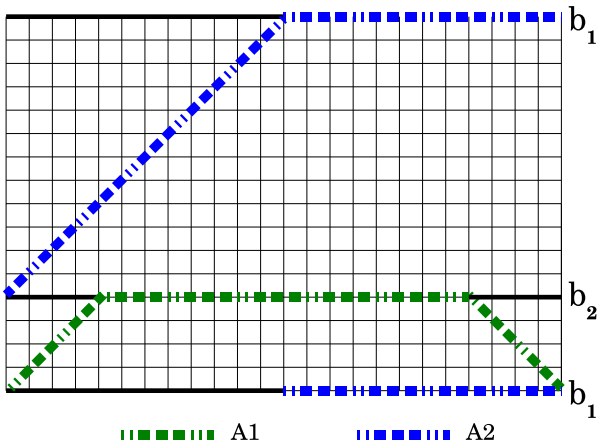

Figura 3.1: Dos agentes ejecutando el algoritmo 3.1 para $c \leq 2r^2 + r$

La figura 3.1 nos muestra un ejemplo donde $c \leq 2r^2 + r$. El tiempo avanza sobre el eje horizontal y los nodos del anillo se encuentran representados en el eje vertical, por lo que la última línea horizontal que podemos ver en la figura, representa al mismo nodo que la primera y ambas tienen la etiqueta b_1. Observemos que el encuentro ocurrirá en un máximo de $2r^2 + 2r$ unidades de tiempo debido a que A_1 encontrará en su bandera inicial al agente A_2 mientras éste espera.

Para el caso en que $c > 2r^2 + r$ el encuentro ocurrirá en un tiempo máximo de $s + r^2 + 2r - 1$. En este caso A_1 regresa, espera y abandona su bandera inicial (b_1) una o más veces antes de que el agente A_2 pueda llegar a ella. Esto se puede observar con la ayuda del cuadro 3.3 y la figura 3.2.

Tiempo	A_1		A_2		Tiempo a transcurrir
	posición	dist. a recorrer	posición	dist. a recorrer	
0	b_1	r	b_2	s	r
r	b_2	esperar r^2	$b_1 - (2r^2 + r + 1)$	c	r^2
$r^2 + r$	b_2	r	$b_1 - (r^2 + r + 1)$	$c - r^2$	r
$r^2 + 2r$	b_1	esperar r^2	$b_1 - (r^2 + 1)$	$c - r^2 - r$	r^2
$2r^2 + 2r$	b_1	r	$b_1 - 1$	1	1
$2r^2 + 2r + 1$	$b_2 - 1$	$r - 1$	b_1	esperar s^2	$r - 1$
$2r^2 + 3r$	b_2	esperar r^2	b_1	esperar $2r^2 + r$	r^2
$3r^2 + 3r$	b_2	r	b_1	$r^2 + r$	r
$3r^2 + 4r$	b_1		b_1	Encuentro	

Cuadro 3.3: Tiempo esperado de encuentro para dos agentes (con $c = 2r^2 + r + 1$)

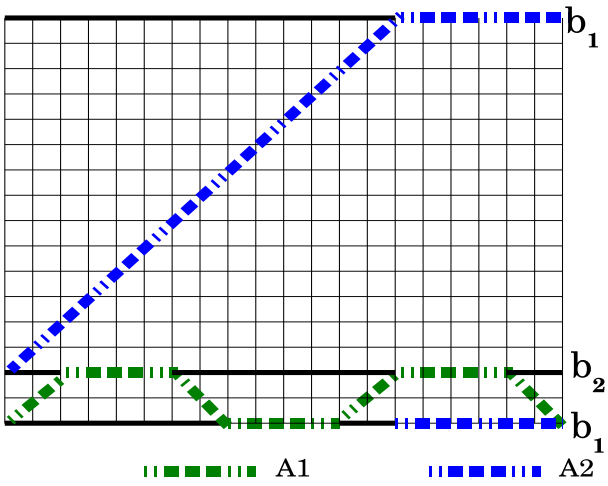

Figura 3.2: Dos agentes ejecutando el algoritmo 3.1 para $c > 2r^2 + r$

La razón de que $s + r^2 + 2r - 1$ sea un límite máximo es que en el peor de los casos en el momento en que A_2 alcanza b_1, A_1 acaba de abandonar este nodo y por lo tanto tardará $r^2 + 2r - 1$ unidades de tiempo en volver. Dado que la espera de A_2 en b_1 será de al menos $(2r^2 + 2r + 1)^2$ unidades de tiempo, podemos ver que es seguro que el encuentro ocurra en cuanto A_1 regrese a su bandera inicial.

Para tres agentes sólo existe una combinación donde las tres distancias son diferentes. En todos los demás casos es posible realizar una reducción hacia el caso de dos agentes que ya estudiamos, pues sabemos que los agentes en segmentos del mismo tamaño están en situación simétrica y no se encontrarán, por lo que podemos ignorarlos y analizar su comportamiento con respecto al otro agente, de ahí que el análisis se hace primero sobre la pareja formada por el agente cuya bandera extremo es la bandera inicial del agente en el segmento de menor tamaño y este último agente. El tiempo esperado de encuentro será la suma del tiempo esperado de encuentro de la primera pareja (que se comportará según lo pronosticado en el análisis, la figura 3.3 ilustra esta afirmación) más el tiempo esperado de encuentro de esta pareja con el agente restante, lo que mantiene el orden de complejidad arriba calculado.

Podemos hacer algunas observaciones generales sobre el comportamiento de este algoritmo. A continuación las exponemos.

- Cuando el tamaño de todos los segmentos es igual, no es posible llevar a cabo el encuentro.

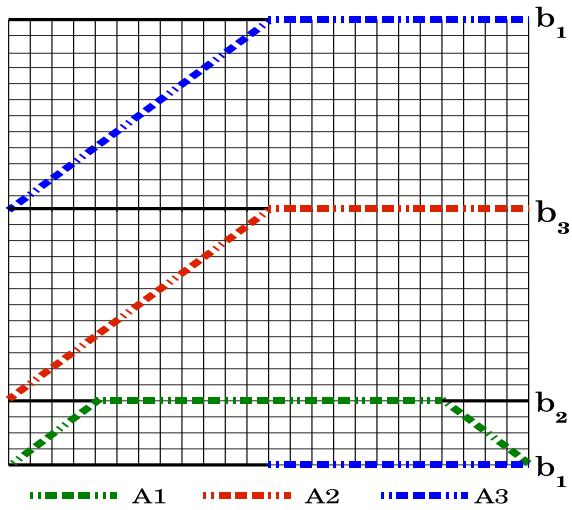

Figura 3.3: Tres agentes ejecutando el algoritmo 3.1 (dos distancias diferentes)

- Para hacer un análisis del comportamiento del algoritmo para k agentes, las combinaciones posibles se pueden reducir a casos anteriores excepto cuando las distancias entre banderas son todas diferentes; esta situación requiere de análisis particular.
- El análisis para $k > 3$ se complica mucho debido a que es necesario considerar las combinaciones posibles de vecinos del segmento más chico y estas combinaciones aumentan conforme k crece.
- Es importante hacer notar que el algoritmo elimina una bandera después de una fusión de agentes, por lo tanto conseguirá efectuar el encuentro de k agentes siempre que después de eliminar una bandera no resulte una situación simétrica.
- Para tres agentes el orden de complejidad enunciado para dos agentes se mantiene. Recordemos que éste es $O(r^2)$ para $c \leq 2r^2 + r$ y $O(s + r^2)$ cuando $c > 2r^2 + r$.
- Haciendo un análisis aproximado, el tiempo esperado de encuentro es como máximo la suma del tiempo esperado de encuentro para cada pareja de agentes y habrá cuando más $k - 1$ parejas, por lo que se conjetura un tiempo esperado de encuentro de $O((k-1)r^2)$ para $c \leq 2r^2 + r$ y $O((k-1)(s + r^2))$ cuando $c > 2r^2 + r$.

Capítulo 4

Encuentro de agentes con memoria constante

Un agente con memoria constante se ve limitado en sus opciones de acción, en cuanto a reunirse con otros se refiere, al ser incapaz de realizar las siguentes actividades:

- Recordar una ruta.
- Recordar si se ha encontrado a otro agente.
- Recordar y distinguir los lugares en los que ya ha estado.
- Concertar una cita y llevar una agenda.
- Contar distancias entre nodos en la red, banderas u otros elementos cuyo tamaño esté en función del número de nodos y agentes de que se trate.

Debido a lo anterior, un agente de esta clase no será capaz de determinar si es posible llevar a cabo el encuentro o no, de modo que cualquier algoritmo de memoria constante considerará que el encuentro es posible como precondición. He aquí otra diferencia importante con el trabajo de Flocchini et al. revisado en el capítulo anterior, pues los autores consideraban que un algoritmo resolvía el problema del encuentro si los agentes se detenían cuando el encuentro no era posible o se reunían en caso contrario.

A continuación presentamos el algoritmo de memoria constante que Kranakis et al. utilizan[10] para reunir dos agentes que no conocen el número de nodos del anillo donde se encuentran, ni la distancia entre ellos al comenzar la ejecución del algoritmo.

Algoritmo 4.1 ***Algoritmo de banderas desplazables para dos agentes con memoria constante***

1. *Poner bandera.*
2. *Elegir una dirección y comenzar a caminar.*
3. *Al encontrar otra bandera invertir la dirección.*
4. *Mover la bandera un nodo en la nueva dirección.*
5. *Continuar caminando en la nueva dirección.*
6. *Repetir desde el paso 3 hasta que se efectúe el encuentro.*

Este algoritmo mantendrá a los agentes viajando entre su bandera inicial y la bandera extremo a la par que intentan reunirlas desplazándolas un nodo más cerca la una de la otra cada vez que llegan a ellas. Los agentes repetirán estas acciones hasta que ambos se encuentren en la bandera que comparten u ocurra un empalme de banderas, después de lo cual ambos agentes estarán viajando en el mismo segmento y forzosamente se encontrarán.

Un **empalme de banderas o empalme** ocurrirá cuando uno de los agentes consiga reunir su bandera extremo y su bandera inicial en un mismo nodo; a partir de este momento las banderas se considerarán como una sola.

En la siguiente sección analizaremos el comportamiento de la generalización de este algoritmo hacia k agentes, posteriormente presentamos un autómata finito determinístico que utilizamos para modelar el comportamiento del algoritmo e inferir algunas propiedades. En la sección 4.3 se encuentra la demostración del funcionamiento del algoritmo para $k = 2, 3$ agentes y por último, en la sección 4.4 presentamos un cálculo exacto del tiempo esperado de encuentro de la primera pareja de agentes bajo diversas circunstancias.

4.1. Reuniendo k agentes con memoria constante

Algoritmo 4.2 ***Algoritmo de banderas desplazables para k agentes con memoria constante***

1. *Plantar bandera en nodo, elegir una dirección.*
2. *Avanzar. Al encontrar una bandera, tomarla e invertir la dirección.*
3. *Avanzar un nodo en la nueva dirección y plantar la bandera.*
4. *Repetir desde el paso 2 hasta que se efectúe un encuentro o un empalme de banderas.*
5. *Al efectuarse un encuentro, fusionar agentes en uno solo y verificar si ya están juntos todos los agentes. De ser así, el algoritmo termina. En caso contrario, elegir una dirección e ir al paso 2.*
6. *En un empalme de banderas el agente se quedará estático en el nodo donde éstas fueron reunidas.*

Veamos el funcionamiento del algoritmo gráficamente en un anillo de once nodos con tres agentes a distancias tres y cuatro, suponiendo que todos comenzaron a caminar en la misma dirección (en contra de las manecillas del reloj), la secuencia muestra los tiempos $t = 0$ y $t = 2$ hasta $t = 5$ de la ejecución del algoritmo.

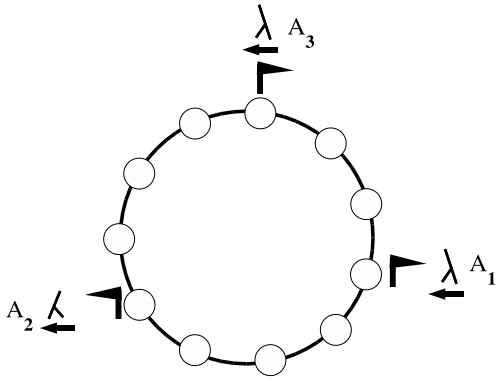

Figura 4.1: Tiempo $t = 0$

La flecha debajo del agente indica la dirección en que éste se dirige, será hacia la izquierda si el agente avanza en dirección contraria a las manecillas del reloj y hacia la derecha en caso contrario. La figura 4.1 muestra que los agentes avanzarán todos en contra de las manecillas del reloj.

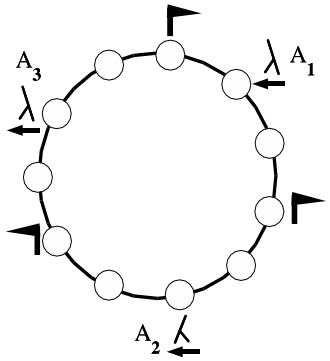

Figura 4.2: Tiempo $t = 2$

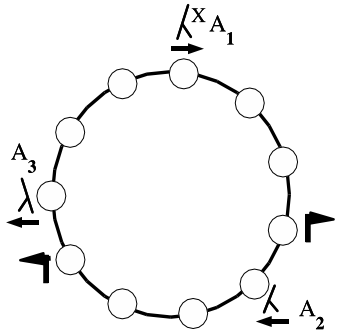

Figura 4.3: Tiempo $t = 3$

En la figura 4.3 podemos ver que el agente A_1 ha alcanzado una bandera, el símbolo **x** indica que la ha tomado y ha modificado su dirección a la que ahora indica la flecha.

En la figura 4.4 veremos que los otros dos agentes alcanzaron su bandera, la han tomado y modificaron su dirección. A_1 ha llevado la bandera a un nodo adyacente y en este momento las distancias entre las banderas aparecen modificadas, ahora hay dos, cuatro y cinco arcos entre ellas.

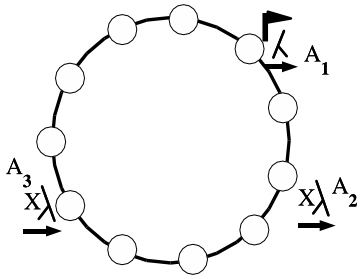

Figura 4.4: Tiempo $t = 4$

Sin embargo para el tiempo $t = 5$ (figura 4.5), las distancias han sufrido una compensación ocasionada porque A_2 aleja la bandera de la que movió A_1, con éste movimiento, la distancia vuelve a ser tres. Como A_3 movió la bandera de forma paralela con A_2, la distancia entre ellas sigue siendo cuatro y por último, cuando A_3 movió la bandera, convirtió la distancia restante en cuatro; devolviéndolas de esta forma a sus distancias originales.

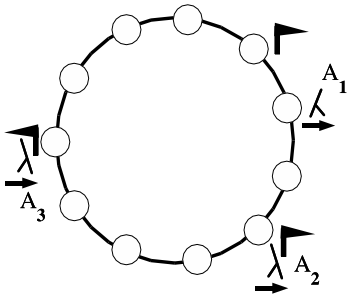

Figura 4.5: Tiempo $t = 5$

Llamemos a cada una de estas figuras **configuración**. Podemos ver la ejecución del algoritmo como una secuencia de configuraciones relacionadas por el orden en que éstas se van presentando. Una configuración nos dice la posición de las banderas, la posición y dirección de los agentes e incluso cuando un agente ha tomado un bandera o va a dejarla en un nodo.

Por la información que proporciona una configuración es posible asociarla en el tiempo con otras dos: una anterior a ella y otra siguiente. Dado que se trata de la ejecución de un algoritmo, si la ejecución se detiene la gráfica de la sucesión formada por las configuraciones tendrá la forma de una trayectoria y en caso de que la ejecución del algoritmo no termine, tendremos un ciclo. Esto resulta muy importante como indicador de que dadas ciertas condiciones iniciales el algoritmo llevará a cabo el encuentro.

Llamaremos **configuración simétrica** a aquella configuración donde las banderas se encuentren equidistantes y los agentes que viajan hacia ellas se encuentren a la misma distancia de su respectiva bandera a alcanzar. En esta configuración, cada bandera es alcanzada y movida en direcciones opuestas por los agentes que la comparten exactamente el mismo número de veces en un intervalo de tiempo t. Esto quiere decir que la distancia de las banderas entre sí, como la de los agentes, se mantiene constante, hecho que impide efectuar un encuentro. Si esto no ocurre, entonces en algún momento habrá un empalme de banderas y se encontrarán dos agentes.

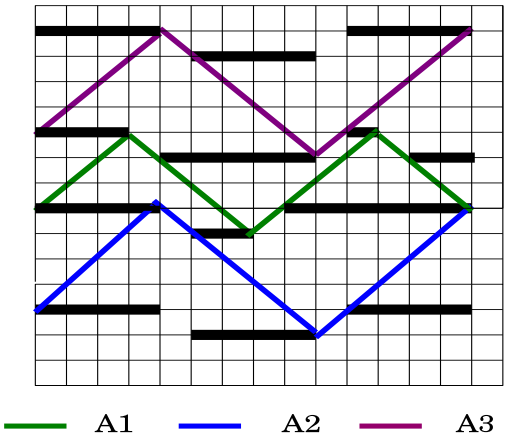

Figura 4.6: Configuraciones extendidas

La figura 4.6 nos muestra la ejecución del algoritmo para el ejemplo propuesto desde el tiempo cero hasta que ocurre el primer encuentro. Las banderas están representadas con una línea negra. Los nodos se encuentran en el eje vertical y el tiempo en el eje horizontal. Este es un ejemplo donde el encuentro se da en el nodo donde hay una bandera compartida por dos agentes antes de que un empalme de banderas ocurra.

Hemos obsevado que el algoritmo lleva a cabo el encuentro de dos formas:

1. Antes de realizar un empalme.
2. En un nodo con una bandera empalmada. El empalme puede comenzar a ocurrir desde que comienza el algoritmo si al menos un segmento cuyo tamaño es r tiene a su lado un segmento de tamaño igual o mayor que $2r$, si esto no ocurre, habrá posibilidad de un empalme de banderas cuando el resultado de $\frac{r}{s-r}$ sea un número entero, considerando que r es el tamaño de un segmento y s el tamaño de cualquiera de sus segmentos vecinos. Se explicará la obtención de estos valores en la sección 4.4.

También hemos observado que el desarrollo del algoritmo depende de las acciones que realizan los agentes en los nodos y un conjunto finito de posibles estados de un nodo. La interrelación de estos dos factores puede modelarse por medio de un autómata finito determinista que presentamos en la siguiente sección.

4.2. Modelando el algoritmo con un autómata finito determinista (AFD)

Las acciones que los agentes realizan son las siguientes:

1. Avanzar un nodo en la dirección actual
2. Tomar bandera e invertir la dirección
3. Dejar bandera

La realización de estas acciones dependerá del estado del nodo en que el agente se encuentra en un instante de tiempo dado. Los estados en que puede encontrarse un nodo se enlistan a continuación.

1. Vacío
2. Con bandera
3. Con bandera y agente

4. Con agente
5. Con dos agentes
6. Con dos agentes y bandera

En una unidad de tiempo un agente llega a un nodo nuevo, inspecciona el estado del mismo (si hay bandera u otros agentes en él) y realiza las acciones necesarias hasta que se encuentra listo para dirigirse hacia otro nodo. Este comportamiento puede modelarse con un AFD que llamaremos N cuyo conjunto de estados son los estados del nodo arriba descritos y su alfabeto de entrada sean las acciones de los agentes. La descripción formal de N no será necesaria para nuestra explicación y por ello no la incluimos, en cambio, en el cuadro 4.1 se hace un resumen de los aspectos que nos interesan acerca de la correspondencia entre el AFD N y el algoritmo.

Estado	Estado del nodo	Entrada	Acción del agente
q_0	Nodo vacío		
q_1	Nodo con bandera	**a**	Dejar bandera
q_2	Nodo con agente	**b**	Tomar bandera
q_3	Nodo con agente y bandera	**c**	Agente en tránsito
q_4	Dos agentes en el nodo	**d**	Procedimiento
q_5	Dos agentes en un nodo con bandera		de encuentro

Cuadro 4.1: Correspondencia autómata - algoritmo

De manera adicional se añadirán unas marcas especiales en los estados donde se encuentre un agente para indicar la dirección de éste y cuando el agente tenga una bandera usaremos un símbolo especial para indicar que el agente encontró la bandera, la tomó y modificó su dirección a la que indica la flecha; si el agente va a dejar la bandera en el nodo, se prescindirá del símbolo (ver cuadro 4.2).

Símbolo	Descripción
$\overleftarrow{q_2}$, $\overrightarrow{q_2}$	El agente lleva la dirección indicada por la flecha
$\overleftarrow{q_3^{\times}}$, $\overrightarrow{q_3^{\times}}$	El agente tomó la bandera del nodo y ahora se dirige en la dirección de la flecha
$\overleftarrow{q_3}$, $\overrightarrow{q_3}$	El agente va a dejar la bandera en el nodo y seguirá avanzando en la dirección que indca la flecha

Cuadro 4.2: Detalle de notación especial

R	$=$	$\{Q, \Sigma, \delta, q_i, F\}$
donde:		
Q	$=$	{ Tupla formada por el estado actual de los n nodos del anillo en un tiempo t_i }
Σ	$=$	{ Tupla de k símbolos de acción del agente }
δ	:	$Q \times \Sigma \rightarrow Q$
q_i	$=$	{ Un estado con $n-k$ elementos q_o y k elementos $\overleftarrow{q_3}$ o $\overrightarrow{q_3}$ }
F	$=$	{ Cualquier estado con x_1 elementos "q_1", x_2 elementos "q_2", x_3 elementos "q_3" y finalmente x_4 elementos "q_0" Donde: $x_2 + x_3 = 1 \quad 1 \leq x_1 \leq k$ $x_1 + x_3 \leq k \quad x_4 = n - (x_1 + x_2 + x_3)$ }

Cuadro 4.3: Descripción formal del autómata R

El autómata N sólo describe un nodo de un anillo donde se está ejecutando el algoritmo 4.2. Para describir la ejecución síncrona y en paralelo del algoritmo utilizaremos un AFD llamado R que se encuentra formado por n autómatas de nodo (como N) que están funcionando en paralelo de forma síncrona. Los estados de este autómata serán equivalentes a una configuración de nuestro algoritmo. El cuadro 4.3 contiene la descripción formal de R. Cabe aclarar que la función de transición δ no se encuentra descrita en detalle debido a su tamaño.

Los estados del autómata R son la conjunción del estado de todos AFD de nodo que lo conforman en un mismo instante del tiempo. Dado que los autómatas de nodo están trabajando de forma sícrona, un estado de R nos dirá en qué nodos están las banderas, dónde están los agentes e incluso la dirección que llevan, es decir, la configuración del algoritmo para un instante de tiempo específico. A partir de un estado del autómata podemos obtener un único estado que tuvo lugar en la unidad de tiempo anterior y otro para la unidad de tiempo siguiente.

Con base en lo anterior podemos concluir que la secuencia de estados en el tiempo producidas por δ sólo puede tener dos formas: una trayectoria o un ciclo.

4.3. Encuentro de dos y tres agentes utilizando memoria constante

En esta sección demostraremos que el algoritmo 4.2 lleva a cabo el encuentro de dos y tres agentes con memoria constante.

Debemos recordar que el caso de análisis es aquel en el que todos los agentes comienzan a avanzar en una misma dirección, en otro caso el encuentro ocurrirá en un tiempo menor.

Teorema 4.1 *En un anillo de n nodos homogéneos, dos agentes con memoria constante se pueden reunir ejecutando en paralelo de forma síncrona el algoritmo 4.2 sí y sólo sí no parten de una configuración simétrica.*

DEMOSTRACIÓN. Supongamos que los agentes se encuentran y el algoritmo comienza con una configuración simétrica. Una **configuración simétrica** para dos agentes ocurre cuando las las banderas están a distancia $n/2$, ambos agentes viajan en la misma dirección y cada uno se encuentra a una distancia d de la bandera que quiere alcanzar. Esto significa que los agentes se encuentran también a una distancia de $n/2$. Un algoritmo formado sólo por operaciones básicas, ejecutado en paralelo y de forma síncrona para ambos agentes hará que éstos conserven la distancia entre ellos. Podría producirse un cambio en esta situación si se condiciona la ejecución de ciertas operaciones al resultado de la evaluación de alguna circunstancia del medio; de modo que si uno de los agentes obtiene un resultado diferente al de su compañero en la evaluación de la condición, se romperá la simetría de acciones y el encuentro será posible.

El paso 2 del algoritmo 4.2 contiene esta condición: *si se ha encontrado una bandera*, de ella depende que un agente cambie su dirección o no, si un agente encuentra una bandera y el otro no, el primero cambiará su dirección, mientras el otro continuará con la que llevaba. En una configuración simétrica ambos agentes encontrarán una bandera al mismo tiempo, como consecuencia obtendrán la misma evaluación de la condición y mantendrán la simetría de acciones, por lo que el encuentro no ocurre. He aquí la contradicción.

Ahora supongamos que los agentes parten de una configuración asimétrica pero el encuentro no se lleva a cabo, por lo tanto los agentes se encuentran en una situación en la que un empalme de banderas no es posible, esto ocurrirá si cada agente puede alcanzar y mover las banderas en una dirección opuesta a la de su compañero el mismo número de veces que éste. Como el algoritmo se ejecuta de forma síncrona, entonces ambos agentes recorren la misma distancia entre banderas; para el caso de dos agentes esto sucederá únicamente si las banderas se encuentran a distancia $n/2$ y ambos agentes a la misma distancia de la bandera que moverán; ésta es una situación simétrica, lo cual contradice la suposición inicial.

Teorema 4.2 *En un anillo de n nodos homogéneos, tres agentes con memoria constante se pueden reunir ejecutando en paralelo de forma síncrona el algoritmo 4.2 sí y sólo sí no parten de una configuración simétrica.*

DEMOSTRACIÓN. Supongamos que el encuentro se lleva a cabo pero partieron de una configuración simétrica, lo anterior significaría que los agentes se encontraban en posiciones simétricas al igual que las banderas, por lo cual los agentes alcanzaban y movían sus respectivas banderas al mismo tiempo, conservando la distancia inicial entre ellos en todo momento, lo que contradice la afirmación de que el encuentro se lleva a cabo.

Ahora supongamos que partiendo de una configuración asimétrica el encuentro no se efectúa. Aquí hay dos casos:

CASO 1. Dos segmentos son de un mismo tamaño y el tercero es diferente. En esta situación, la bandera común a los segmentos de igual tamaño será movida igual número de veces en direcciones opuestas, por lo que es posible ignorar este movimiento y considerar éste como un caso de dos agentes en situación asimétrica. Está demostrado que el encuentro sí se lleva a cabo en estas circunstancias.

CASO 2. Los tres segmentos son de longitudes distintas, digamos r, s y t. En cada segmento se encuentra un agente, digamos A_r, A_s y A_t respectivamente. Si el encuentro no se lleva a cabo, entonces A_r puede mover las banderas de su segmento tantas veces como los agentes A_s y A_t. Entonces, si el agente A_r mueve ambas banderas en p pasos, en este mismo intervalo de tiempo el agente A_s habrá movido la bandera que comparte con A_r una vez y el agente A_t habrá movido la bandera que comparte con A_r una vez.

Dado que el encuentro no se lleva a cabo, esta situación se mantiene siempre, es decir, cada vez que el agente A_r da p pasos, puede mover las dos banderas de su segmento; los agentes A_s y A_t también podrán mover cada uno la bandera que comparte con A_r una vez es decir, en cada ciclo de p pasos los agentes A_s y A_t mueven una vez la bandera que comparten con A_r.

El algoritmo 4.2 hace que un agente viaje entre las dos banderas de su segmento invirtiendo la dirección que lleva cada vez que éste alcanza una de las banderas por lo cual, un agente no puede mover dos veces consecutivas una misma bandera; esto forzosamente significa que en algún momento durante los p pasos A_s y A_t recorren su segmento y mueven la otra bandera. Como la ejecución del algoritmo es síncrona esto último sólo será posible si $s = t = r$,

lo que constituye una contradicción al supuesto inicial de que las longitudes de los tres segmentos son distintas.

Con base en la demostración de los teoremas 4.1 y 4.2, la aparente inclinación del algoritmo hacia un comportamiento inductivo y las observaciones realizadas en la sección 4.2, conjeturamos que para k agentes con memoria constante, el algoritmo 4.2 llevará a cabo el encuentro siempre que éste sea posible.

4.4. Cálculo del tiempo esperado de encuentro en un escenario de peor caso

Recordando que hay dos formas en que ocurre un encuentro de agentes (sección 4.1), en esta sección se calculará el mayor tiempo esperado de encuentro para el caso en que los agentes se reunen antes de que ocurra un empalme de banderas. Debido a que es un cálculo para el peor caso, se considerará que los gentes comienzan a avanzar en la misma direción.

Calcularemos el tiempo esperado de encuentro para tres agentes (A_1, A_2 y A_3), cuando éstos tienen una distribución tal que A_1 se encuentra a r arcos de A_2. Entre A_2 y A_3 hay s arcos, al igual que entre A_3 y A_1. Si r y s cumplen las especificaciones que se indican a continuación, el algoritmo presenta un comportaminto que hace posible realizar este cálculo.

- $s < 2r$. Esta condición se impone debido a que en otro caso, el agente que viaja en el segmento de tamaño r comenzará a hacer logros para conseguir el empalme de banderas acortando su segmento, de esta forma ya no se presentaría el comportamiento que aquí exponemos.
- $r < s$
- Definimos $b = s - r$, $m = r \mod b$ y $c = \frac{(r-m)}{b}$

El cuadro 4.4 nos muestra el tiempo esperado de encuentro de la primera pareja de agentes en las circunstancias arriba descritas. Posteriormente explicaremos cómo se realizó este cálculo.

Los agentes en un anillo sólo pueden avanzar en dos direcciones: en el sentido de las manecillas del reloj (0) o en sentido contrario (1); aún cuando no se encuentren al tanto de cuando viajan en un sentido u otro.

Dirección del agente			Tiempo esperado de encuentro
A_1	A_2	A_3	
0	0	0	$c(s+1)^*$
0	0	1	$s/2$ (se encuentran A_2 y A_3)
0	1	0	$r/2$ (se encuentran A_1 y A_2)
0	1	1	$r/2$ (se encuentran A_1 y A_2)
1	0	0	$s/2$ (se encuentran A_1 y A_3)
1	0	1	$s/2$ (se encuentran A_2 y A_3)
1	1	0	$s/2$ (se encuentran A_1 y A_3)
1	1	1	$c(s+1)^*$

*si r es múltiplo de b

Cuadro 4.4: Tiempo esperado de encuentro de la primera pareja con $k = 3$

Analicemos el comportamiento de tres agentes en posición simétrica. Digamos que todas las banderas están a distancia s, entonces, al tiempo s los agentes se encontrarán en sus banderas extremo, en el tiempo $s + 1$ todos habrán movido ésta bandera un nodo en dirección a su bandera inicial, por lo tanto estarán a distancia s de ésta, así, al tiempo $2s + 1$ habrán completado una vuelta y todos los agentes se encuentran en su bandera inicial, exactamente en la misma situación que al principio. De este modo, podemos ver que los agentes completarán la i-ésima vuelta en el tiempo $i(2s + 1)$. Ahora veamos cuánto cambia este comportamiento si reducimos uno de los intervalos.

Digamos que es reducido el segmento que recorre el agente A_1 en b arcos, de modo que ahora mide r arcos, mientras que los otros dos segmentos son de s arcos. A_1 alcanzará y moverá su bandera extremo antes que los otros dos, así, al tiempo $r + 1$, le faltarán $r - 1$ arcos por recorrer para completar su vuelta mientras que el segmento de donde la bandera recién movida es la bandera inicial, tendrá $s + 1$ arcos, sin embargo, al tiempo $s + 1 = r + b + 1$ los otros dos agentes habrán movido sus banderas extremo causando el siguiente efecto: el segmento que medía $s + 1$ arcos ha quedado de s arcos porque la bandera inicial de ese segmento no se movió, en cambio, ambas banderas del segmento que medía s arcos han sido movidas, pues la bandera inicial de ese segmento es la bandera extremo del segmento que medía $s+1$, entonces, el segmento quedará de s arcos, por último, como la bandera inicial del A_1 es la bandera extremo del segmento que mide s, ésta será alejada un nodo de A_1 y el segmento de A_1 volverá a medir r. El cuadro 4.5 muestra como varían en el tiempo las longitudes de los segmentos y la distancia de los agentes a la bandera, suponiendo que todos comienzan a avanzar en el sentido de las manecillas del reloj.

t	tamaño de segmento			distancia a bandera		
	A_1	A_2	A_3	A_1	A_2	A_3
0	r	s	s	r	s	s
r+1	$r-1$	$s+1$	s	$r-1$	$b-1$	$b-1$
r+b+1	r	s	s	$r-b$	s	s
2r+1	r	s	s	r	$2b$	$2b$
2r+2	$r-1$	s	$s+1$	$r-1$	$2b-1$	$2b-1$
2r+2b+2	r	s	s	$r-2b$	s	s
3r+2	r	s	s	r	$3b$	$3b$
3r+3	$r-1$	$s+1$	s	$r-1$	$3b-1$	$3b-1$
3s+3	r	s	s	$r-3b$	s	s

Cuadro 4.5: Variación en el tiempo de los segmentos y de la distancia de los agentes hacia sus banderas

Considerando que los agentes avanzan a la misma velocidad y se mueven en paralelo, por cada vuelta de A_2 y A_3, A_1 podrá completar una vuelta y avanzar b arcos, de este modo, al cabo de c vueltas de A_2 y A_3, A_1 habrá completado c vueltas, y además si $m = 0$, alcanzará la bandera extremo, que a su vez es la bandera inicial de alguno de los otros agentes, por lo que se encontrará con A_2 o A_3. En el párrafo siguiente calcularemos el tiempo necesario para que esto suceda.

En el cuadro 4.5 podemos observar que para recorrer un segmento de longitud r, bastan r pasos, sin embargo, si se requiere observar el impacto del movimento de la bandera, hay que esperar un paso más. En el párrafo anterior se determina el tiempo transcurrido para el encuentro con base en vueltas de los agentes en los segmentos de tamaño s, ahora, el avance del algoritmo por vuelta, como se explica en el párrafo anterior, se verá reflejado no en el tiempo s, sino en el tiempo $s + 1$, ya que se han movido las banderas; de este modo se tiene que el tiempo esperado de encuentro es $c(s + 1)$ cuando $s = cb$, es decir, cada $(s + 1)$ tiempos, los segmentos volverán a sus longitudes iniciales y el agente en el segmento de tamaño r avanzará b arcos hacia su bandera extremo, de esta forma, al cabo de c vueltas la alcanzará. Si $s = cb + m$, entonces en el tiempo $c(s + 1)$ comezará el empalme de banderas y el encuentro ocurrirá una vez hayan sido reunidas las banderas del segmento r.

4.4.1. Análisis para k agentes

Podemos aplicar los valores del análisis anterior a un par de casos de k agentes donde las distancias entre ellos tienen la particularidad de que se pueden analizar como un caso de tres agentes con dos distancias diferentes.

El primero de ellos es el de k agentes en un anillo de n nodos distribuidos de la siguiente manera:

- Uno de los agentes se encuentra a r arcos de su vecino más próximo.
- Los restantes $k-1$ agentes se encuentran a distancia s.

Recordemos que en una configuración simétrica la distancia entre banderas y la de los agentes hacia las banderas permanece constante. En esta configuración $k-3$ de los agentes tendrán segmentos vecinos de la misma longitud que el propio segmento, por lo que se comportarán como si estruvieran en una configuración simétrica, conservando la distancia entre ellos y hacia las banderas respectivas. De esta forma podemos centrar nuestra atención en los tres agentes restantes. El agente en el segmento de r arcos tiene por vecinos dos agentes, cada uno en un segmento de s arcos. Éstos a su vez tienen un vecino en un segmento de r arcos y otro vecino en un segmento de s arcos; éstas son las mismas circunstancias que las del caso analizado en la sección anterior, por lo que tendremos el mismo tiempo esperado de primer encuentro.

El segundo caso es una reducción hacia un caso con tres distancias diferentes r, s y t. En este caso, k-5 agentes se encuentran en una situación simétrica con respecto a sus vecinos, de esta forma podemos enforcar nuestra atención en los tres agentes que se encuentran en segmentos de longitud distinta. Para este caso sabemos que el agente en el segmento de menor tamaño conseguirá un empalme de banderas, a lo cual seguirá el primer encuentro.

Capítulo 5

Conclusiones

En este trabajo se ha desarrollado el tema de reunir varios agentes en un anillo bajo las siguientes restricciones:

- Ausencia de medios de orientación global.
- Memoria limitada (logarítmica y constante).
- Información local (visión limitada).

Estas restricciones nos llevan a considerar las dificultades que un entorno simétrico representa, pues la memoria constante del agente le impedirá *notar* la falta de progreso en la consecución de su objetivo por encontrarse en esta clase de entorno.

La razón del establecimiento de estas restricciones es que existen problemas de aplicación que lo requieren, por ejemplo la navegación en un terreno desconocido con visión limitada[11], redes ad hoc o redes inalámbricas de sensores[5]; además, el paradigma mismo de agentes móviles[8, 6], requiere que el agente se mueva con datos y el procesamiento que realice se encuentre dentro de la capacidad del nodo que lo hospeda.

En este trabajo fueron presentados dos algoritmos para reunir k agentes, con las respectivas demostraciones para $k = 2, 3$ agentes y otros casos especiales. Ambos poseen sencillez en su estructura, sin embargo, el análisis de los mismos es muy complicado. Lo anterior nos lleva a la siguiente reflexión: el diseño de algoritmos eficientes se complica en la medida que los recursos disminuyen. Aquí se presenta un

fenómeno curioso, mientras el algoritmo en sí mismo adquiere simplicidad, analizar su comportamiento y evaluar su rendimiento, se vuelven procesos más complicados.

En los trabajos relacionados encontramos un proceso de transformación y mejora de los algoritmos, durante el cual el problema del encuentro fue resuelto utilizando técnicas cada vez más sutiles con las cuales se satisfacían restricciones más difíciles. De esta manera, se llega hasta el trabajo de Flocchini et al.[7], que es el que más se acerca al presente desarrollo, sin embargo en él no se toca el tema de memoria constante, pues ello requiere cambiar del esquema de banderas fíjas a banderas móviles.

En cuanto a la demostración del algoritmo de memoria constante, si se considera el desarrollo del mismo como la secuencia de estados que produce la función δ de un autómata que lo modela (ver sección 4.2), llegamos a la conclusión de que la secuencia resultante sólo puede tener dos formas: una trayectoria o un ciclo.

Nuestra conjetura es que el único caso en que la ejecución del algoritmo da origen a una secuencia cíclica de estados es cuando la configuración inicial es simétrica. La demostración (nada trivial) de este hecho probaría que el algoritmo 4.2 puede llevar a cabo el encuentro siempre que éste sea posible. Aún hay bastante por hacer en este aspecto pues no se puede asegurar que no exista otro algoritmo de memoria constante o incluso un paradigma diferente con un mejor comportamiento o más sencillo de analizar.

Durante el desarrollo de este trabajo observamos que las banderas son un buen recurso ante las situaciones simétricas. Resultaría muy interesante investigar el encuentro en árboles utilizando banderas y memoria constante, tema que a primera vista parece más sencillo que el aquí tratado, sin embargo no existe mucha literatura al respecto, lo que sugiere que es un problema poco explorado. Un análisis superficial nos daría una idea del grado de dificultad del tema y a su vez explicaría la cantidad de literatura que existe al respecto. Las razones anteriores nos llevan a proponerlo como un buen candidato de investigación a futuro.

Apéndice A

Simulador para el algoritmo 4.2

A continuación describiremos el programa con que se simuló el algoritmo de memoria constante 4.2. El fin de elaborar este programa fue agilizar el análisis del comportamiento de este algoritmo. El programa fue elaborado en lenguaje *JAVA* y consta de tres clases principales: `Nodo`, `Agente` y `Algoritmo`. Como interfaz para utilizarlas se crea la clase `Texto`. Cabe mencionar que esta simulación podría hacerse utilizando hilos y ello requeriría forzozamente establecer sincronía entre los hilos.

Las clases principales tienen una parte de código correspondiente al algoritmo y otra que es necesaria para conseguir la simulación, pues recordemos que estamos simulando la ejecución síncrona y en paralelo de un algoritmo sobre un sistema de un procesador.

Comenzaremos con la clase `Agente`, que modela las caracteristicas necesarias en un agente para simular el algoritmo 4.2 y que básicamente son las cosas que el agente sabe o puede hacer durante la ejecución del algoritmo: tomar una bandera, transportar una bandera, dejar la bandera en un nodo, tener una dirección y avanzar. Además de éstas se añadieron otras características que no son requeridas por el algoritmo para llevar a cabo el encuentro. Éstas últimas son necesarias para la simulación y se enlistan a continuación:

- Etiqueta o identificador.
- Ubicación en el anillo (existe una etiquetación global de los nodos).
- Para simular la fusión, el agente seguirá a aquel con el que se haya fusionado. Esto se indica por medio de una variable, que se encontrará en -1 mientras el agente no se haya fusionado.

- Para simular que el agente nota la presencia de otro agente en el mismo nodo, se utiliza una función que registra la presencia del agente en el nodo. Esta misma función registrará la salida del agente cuando éste se dirija hacia otro nodo. De esta forma será posible saber qué agente o agentes ocupan un nodo en especifico.

- Con el propósito de analizar el comportamiento del algoritmo se añadió al agente la capacidad de contar las veces que recorre un segmento.

Para realizar todas estas funciones, el agente poseerá los siguientes atributos:

```
boolean marcador; //indica si agente lleva bandera
int posicion;     //número de nodo en el que se encuentra
int direccion;    // +1 ó -1 en o contra sentido manecillas
int imita;        // índice agente con que está fusionado
(-1 antes)
int registro;    //posición de registro en nodo
int recorridos;   //número de veces que recorrió segmento
String nombre;   //identificador del agente
```

Mismos que se podrán modificar mediante los siguientes métodos:

```
public void setPosicion(int pos)
public void setDireccion(int dir)
public void setNombre(String nom)
public void setImita(int val)
public void setRegistro(int reg)
public void setRecorridos()
public int getPosicion()
public String getNombre()
public boolean getBandera()
public int getDireccion()
public int getImita()
public int getRegistro()
public int getRecorridos()
```

Las acciones del agente relacionadas al algoritmo están representadas por los siguientes métodos:

```
public void avanzar()
public void voltear()
public void tomaBandera()
public void dejaBandera()
```

La clase `Nodo` simula un nodo del anillo, a efectos del algoritmo, el nodo debe ser capaz de almacenar una bandera y hospedar k agentes. Ahora, para la simulación, se añadirán como parte de la funcionalidad del nodo las siguientes capacidades:

- Saber cuántos agentes están en él.
- Conocer la etiqueta de los agentes que se encuentran en él.

Los atributos de la clase son:

```
boolean bandera;     //indicador de bandera en el nodo
int nagentes;       // número de agentes en el nodo
int[] huesped = new int[2] ;   //id. de agente en el nodo
```

Los métodos que simulan las capacidades del nodo necesarias para el algoritmo son:

```
public void ponBandera()
public void quitaBandera()
public boolean hayBandera()
public void regAgente(int var) //agentes en nodo
public int getNagentes()
```

Los siguientes métodos permitirán detectar un encuentro y llevar a cabo el proceso de fusión simulado, que consiste en determinar cuál de los dos agentes seguirá ejecutando el algoritmo y cuál se limitará a copiar la posición del otro. Es muy importante que el agente que siga ejecutando el algoritmo sea el de identificador menor para que el otro agente simule que se ha fusionado con él (copiando su posición).

```
public int regHuesped(int ag)     //id. agente en nodo
public void saleHuesped(int num) //Borra id. agente
public int getHuesped(int g)  //Recupera id. ordenado
public int getAgente()       //Recupera id. sólo un agente
public void ordenaHuespedes()    //Ordena los id's.
```

La última clase principal (`Algoritmo`) se servirá de los medios provistos por las otras dos clases para efectuar la simulación. Esta clase es una especie de observador externo que hace que sucedan las cosas, por ejemplo, si hay un encuentro lleva a cabo el procedimiento de fusión, es capaz de presentar el estado de la red y controla el final de la ejecución del algoritmo, pues es el que *sabe* cuando ya se han reunido todos los agentes, e incluso una vez aproximado el orden de complejidad del algoritmo, se le puede añadir una condición de terminación si el tiempo transcurrido supera la estimación, sin perder de vista que se trata de una simulación.

Al comenzar la ejecución esta clase considera que el nodo donde se encuentra cada agente ya está marcado con una bandera y los agentes comienzan con el peor de los escenarios: todos avanzan en la misma dirección. Se ha impuesto que sea con las manecillas del reloj, esto no afecta, igualmente puede ser en dirección contraria.

La parte donde se lleva el control de la ejecución es el método `paso`. La ejecución se realiza de forma secuencial y para simular el paralelismo las actividades se llevan a cabo en el siguiente orden:

1. Avanzan todos los agentes

2. Para cada agente revisa el estado del nodo donde se encuentra y realiza una de estas acciones:

   a) Si hay más agentes en el nodo los fusiona, esto es, el agente de identificador mayor dejará de ejecutar el algoritmo y copiará la posición del otro agente, de forma que sólo este último continuará la ejecución en representación de los dos.

   b) Si el agente trae una bandera, la deja en el nodo al que llegó.

   c) Si encuentra una bandera, la toma y cambia su dirección por la opuesta.

Inmediatamente después, el algoritmo podrá desplegar el estado de la red.

La red se simula mediante un arreglo de nodos. El control de los agentes se hace por medio de un arreglo de agentes. Los atributos de la clase se enlistan a continuación.

```
componentes.Nodo[] red = new componentes.Nodo[100];
//arreglo de nodos para simular red de anillo
componentes.Agente[] agentes = new componentes.Agente[10];
//agentes
private int Tiempo = 0; //contador global de pasos dados
private int N= 0; //número de nodos de la red
private int K= 0; //número de agentes en la red
private int restantes = -1; //cuenta de número de agentes que
//son movidos para identificar la terminación (sólo queda uno)
private boolean reunion = false;
```

A continuación detallamos los métodos con los que se realizan las acciones *mover agente* y *fusión al encuentro* del método `paso`.

```
public void moverAgente(int a)
public void encuentro(int i)
```

El método `moverAgente`, recibe como parámetro el identificador del agente que se esta moviendo, registra su salida del nodo en el que está y pone sus datos en las variables de registro de último agente visitante. Después hace avanzar al agente y lo registra en el nodo al que llegó. El método `encuentro` recibe el número de nodo donde hay dos agentes, de esta forma determina cuáles son y le indica a uno de ellos que copie la posición del otro desde este momento en adelante (el agente *"líder"* será el de identificador más pequeño).

Los siguientes métodos no tienen relación con el funcionamiento del algoritmo, fueron añadidos para observar el comportamiento del mismo en el tiempo.

El método `foto` representa numéricamente la posición y dirección de cada uno de los agentes en el anillo, así como las posiciones de las banderas.

El método `dibuja`, presenta el anillo abierto puesto tantas veces como se quiera sobre una línea infinita. Los nodos vacíos se representan con "$-$", aquellos que tienen bandera con "$>$", las posiciones de los agentes con un "$*$" y si éstos tienen bandera se marcará con "&".

Por último, la clase `Texto` es la interfaz que nos permite llevar a cabo la simulación. Por medio de ella definimos el problema de encuentro a simular: número de nodos en la red, número de agentes, posición inicial de cada agente y número de repeticiones para el despliegue del anillo abierto sobre la línea. Ésta clase únicamente valida los parámetros recibidos, crea una instancia de la clase `Algoritmo` con

ellos y ejecuta su método paso hasta que el mismo objeto indica que ha concluido la ejecución.

```
public void foto() {
    int i;
    int d=0;
    char dir = ' ';

    for (i=0;i<K;i++) {
       System.out.print(agentes[i].getNombre());
       System.out.print("nodo ( ");
       d=agentes[i].getDireccion();
       switch(d) {
        case -1: dir = '-';
          break;
        case 0: dir = ' ';
          break;
        case 1: dir = '+';
          break;
       }
       System.out.print(  + dir + );
       System.out.print(agentes[i].getPosicion());
       System.out.print(") ");
       System.out.print(agentes[i].getBandera());
       System.out.print("Vueltas: ");
       System.out.println(agentes[i].getRecorridos());
    }
    System.out.println("BANDERAS");
    for (i=0; i<N;i++) {
       if (red[i].hayBandera()) {
          System.out.print(i);
          System.out.print();
       }
    }
    System.out.println();
    System.out.println("================");
}
```

```
public void dibuja(int iter)
{
    int ix_n, pos;
    char[] linea = new char[N];
    String despliegue = ;

    for (ix_n=0;ix_n<N;ix_n++) {
    //Revisa nodos y si el nodo tiene una bandera,
    //la marca en el arreglo linea
      if (red[ix_n].hayBandera())
      {
        linea[ix_n] = '>';
      }
      else
      {
        linea[ix_n] = '-'; // '-' indicador de "vacio"
      }
    } //fin for paso1

    //Marca agentes cuidando de no borrar banderas
    for (ix_n=0;ix_n<K;ix_n++) {
      pos =agentes[ix_n].getPosicion();
      if (linea[pos]== '-')
      { //No hay bandera
        if (agentes[ix_n].getBandera()) {
        //el agente tiene bandera
          linea[pos]='&';
        }
        else
        { //Marca sólo el agente
          linea[pos]='*';
        }
      }
      else
      {//Hay bandera
        linea[pos]= '&'
      }
    } //fin for paso2
    for(ix_n=0;ix_n<N;ix_n++)
    {
      despliegue = despliegue + "|"+ linea[ix_n]x;
    }
    for(ix_n=1;ix_n<iter;ix_n++)
    {
      System.out.print(despliegue);
    }
    despliegue = despliegue + "|";
    System.out.print(despliegue);
    System.out.println(Tiempo);
}
```

Índice alfabético

Bibliografía

[1] Steve Alpern. The rendezvous search problem. *SIAM Journal of control and optimization*, 33(3):673–683, 1995.

[2] Steve Alpern. Rendezvous search: a personal perspective. *Operations Research INFORMS*, 50(5):772–795, septiembre-octubre 2002.

[3] Edward Anderson and Richard Weber. The rendezvous problem on discret locations. *Journal of applied probability*, 28:839–851, 1990.

[4] Vic Baston and Shmuel Gal. Rendezvous search when marks are left at the starting points. *Naval Research Logistics*, 48:722–731, 2001.

[5] Carlos de Morais Cordeiro and Dharma Agrawal. *Mobile Ad Hoc Networking*. 2002.

[6] Rafael Norman Saucedo Delgado. Algoritmos para gráficas geométricas con agentes de memoria limitada. Master's thesis, Posgrado en Ciencia e Ingeniería de la Computación (UNAM), 2005.

[7] Paola Flocchini, Evangelos Kranakis, Danny Krizanc, Nicola Santoro, and Cindy Sawchuk. Multiple mobile agent rendezvous in a ring. In *proceedings of LATIN 2004*, volume 2976 of *Lecture Notes in Computer Science*, pages 599–608. M. Farach-Colton, 2004.

[8] Colin G. Harrison and Aaron Kershenbaum David M. Chess. Mobile agents: Are they a good idea? Technical report, IBM Research division, 1995.

[9] John Hopcroft, Rajeev Motwani, and Jeffrey Ullman. *Introducción a la teoría de autómatas, lenguajes y computación*. Addison Wesley, 2001.

[10] Evangelos Kranakis, Danny Krizanc, Nicola Santoro, and Cindy Sawchuk. Mobile agent rendezvous in a ring. *International Conference on Distributed Computing Systems*, pages 592–599, 2003.

[11] N. S. V. Rao, S. Hareti, W. Shi, and S. S. Iyengar. Robot navigation in unknown terrains: Introductory survey of non-heuristic algorithms. Technical Report ORNL/TM-12410, Oak Ridge National Laboratory, July 1993.

[12] W. H. Ruckle. Pursuit on a cyclic graph. *International Journal of Game Theory*, 10:91–99, 1983.

[13] Thomas Crombie Schelling. *The Strategy of Conflict*. Harvard University Press, 1960.

[14] Laurent Thomas and Moti Pikounis. Many-player rendezvous search: Stick together or split and meet? *Naval Research Logistics*, 48:710–721, 2001.

[15] Xiangdong Yu and Moti Yung. Agent rendezvous: A dynamic symmetry-breaking problem. In *proceedings of ICALP '96*, volume 1099 of *Lecture Notes in Computer Science*, pages 610–621. Springer, 1996.