TESIS DEFENDIDA POR

Liz Ibeth Gaxiola Olivas

Y APROBADA POR EL SIGUIENTE COMITÉ

Dr. José Alberto Fernández Zepeda

Director del Comité

Dr. Pedro Gilberto López Mariscal

Miembro del Comité

Dr. Carlos Alberto Brizuela Rodríguez

Miembro del Comité

Dra. María del Carmen Maya Sánchez

Miembro del Comité

Dra. Ana Isabel Martínez García

Coordinador del programa de posgrado en Ciencias de la Computación

Dr. David Hilario Covarrubias Rosales

Director de Estudios de Posgrado

30 Noviembre de 2009

CENTRO DE INVESTIGACIÓN CIENTÍFICA Y DE
EDUCACIÓN SUPERIOR DE ENSENADA

PROGRAMA DE POSGRADO EN CIENCIAS
EN CIENCIAS DE LA COMPUTACIÓN

ESTUDIO DE LA TÉCNICA DE ACOPLAMIENTO EN ALGORITMOS AUTO-ESTABILIZANTES

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN CIENCIAS

Presenta:

LIZ IBETH GAXIOLA OLIVAS

Ensenada, Baja California, México, Noviembre 2009

RESUMEN de la tesis de **LIZ IBETH GAXIOLA OLIVAS**, presentada como requisito parcial para la obtención del grado de MAESTRO EN CIENCIAS en CIENCIAS DE LA COMPUTACIÓN. Ensenada, Baja California, Noviembre 2009.

ESTUDIO DE LA TÉCNICA DE ACOPLAMIENTO EN ALGORITMOS AUTO-ESTABILIZANTES

Resumen aprobado por:

Dr. José Alberto Fernández Zepeda

Director de Tesis

La tolerancia a fallas es una característica deseable de los sistemas distribuidos donde los sistemas auto-estabilizantes ofrecen soporte de manera intrínseca. Uno de los retos de los algoritmos auto-estabilizantes es demostrar su convergencia, es decir probar que el algoritmo llega a un comportamiento deseado. En esta tesis se propone realizar un análisis de la convergencia de algoritmos auto-estabilizantes con topologías asimétricas utilizando la técnica de acoplamiento. Debido a que las investigaciones preliminares sobre esta técnica sólo la utilizan en algoritmos con topologías muy restringidas. Dicha técnica comprueba la convergencia de algoritmos auto-estabilizantes combinando dos medidas de convergencia de una cadena de Markov: el tiempo esperado de alcance y el tiempo de absorción$-\varepsilon$. La importancia de esta tesis radica, en ampliar el uso de la técnica de acoplamiento para analizar algoritmos auto-estabilizantes con topologías más generales. De igual manera, la contribución de este trabajo es mostrar que ésta técnica se puede aplicar para demostrar la convergencia de los algoritmos auto-estabilizantes en un conjunto más amplio de confguraciones.

Palabras Clave: Auto-estabilización, convergencia, técnica de acoplamiento

ABSTRACT of the thesis presented by **LIZ IBETH GAXIOLA OLIVAS**, in partial fulfillment of the requirements of the degree of MASTER OF SCIENCE in COMPUTER SCIENCE. Ensenada, Baja California, November 2009.

STUDY OF THE COUPLING TECHNIQUE FOR SELF-STABILIZING ALGORITHMS

To be fault tolerant is a desirable feature of distributed systems and self-stabilizing systems offer intrinsical support. One of the challenges of self-stabilizing algorithms is to prove convergence, i.e. to prove that an algorithm reaches a desired behavior. This thesis proposes an analysis of the convergence of self-stabilizing algorithms with asymmetric topologies by using the coupling technique, since preliminary investigations on this technique show that it is only used on restricted topologies. This technique proves the convergence of self-stabilizing by combining two measures of convergence for a Markov chain: expected hitting time and $\varepsilon-$absorption time. The importance of this thesis is to expand the use of the coupling technique to analyze self-stabilizing algorithms with more general topologies. Similarly, our contribution is to show that this technique can be used to prove convergence in self-stabilizing algorithms on a wider set of configurations.

Keywords: Self-stabilization, convergence, coupling

A mis padres por su amor y comprensión...

Agradecimientos

Agradezco a mi asesor, Dr. José Alberto Fernández Zepeda, por sus sabios consejos, dedicación y paciencia.

A mi cómite de tesis, Dr. Pedro Gilberto López Mariscal, Dr. Carlos Alberto Brizuela Rodríguez y Dra. María del Carmen Maya Sánchez por su tiempo, dedicación y consejos durante el desarrollo de esta tesis.

Al departamento de ciencias de la computación, por contribuir en mi formación académica y brindarme la oportunidad de seguir creciendo como profesional y como persona.

A mi familia, por sus palabras de aliento y estar conmigo en todo momento.

A Roberto por su amor, apoyo y estar conmigo.

A la generación 2007 por todo el tiempo, risas y tristezas compartidas a lo largo de la maestría.

A Pablo Aguilar por sus sabios consejos, gran apoyo y ayuda invaluable en todo este tiempo.

Al Centro de Investigación Científica y de Educación Superior de Ensenada.

Al Consejo Nacional de Ciencia y Tecnología por su apoyo económico.

Contenido

Página

Contenido (continuación)

Lista de Figuras

Lista de Tablas

Capítulo I

Introducción

En la actualidad, gracias al incremento en el uso de redes de computadoras mucho del procesamiento se hace de forma remota, así que es necesario garantizar de alguna manera que este procesamiento se haga de forma adecuada. Los sistemas de procesamiento remoto deben ser muy confiables, ya que si un componente del sistema falla, otro componente debe ser capaz de reemplazarlo. Se dice que un sistema falla cuando no cumple con el objetivo para el cual fue creado.

Como es necesario tener sistemas de procesamiento remoto confiables, se deben utilizar mecanismos para lograr la confiabilidad e incorporar características que les permitan ser tolerantes a fallas. Un sistema es tolerante a fallas si continúa proporcionando los servicios especificados aún en presencia de fallas de hardware o errores de software. El objetivo del diseño y construcción de sistemas tolerantes a fallas es garantizar que el sistema continúe funcionando de manera correcta, incluso en la presencia de fallas.

Hay varios mecanismos que apoyan a los sistemas tolerantes a fallas como lo son: la redundancia de información, redundancia de recursos, redundancia de tiempo, la detección temprana de fallas, aislamiento de fallas severas y en particular en esta tesis se profundiza en la auto-estabilización.

La auto-estabilización aparece como un mecanismo que ayuda a la tolerancia a

fallas transitorias (mal funcionamiento temporal). Se dice que un sistema es auto-estabilizante, si el sistema al detectar alguna falla, realiza acciones adecuadas para corregir esa falla y logra la recuperación del sistema sin ayuda externa. El presente trabajo de tesis se enfoca al estudio de cómo demostrar que un sistema es auto-estabilizante. Más específicamente, se profundiza en cómo demostrar la propiedad de convergencia usando la técnica de acoplamiento.

Este capítulo introduce el concepto de auto-estabilización. En la Sección I.1 se presenta la definición de auto-estabilización, así como conceptos básicos asociados. En la Sección I.2 se presentan algunas definiciones sobre la convergencia de los algoritmos auto-estabilizantes. La Sección I.3 presenta el objetivo de este trabajo de investigación. La Sección I.4 explica la panorámica de la tesis, es decir, las actividades de investigación que se realizaron para elaborar esta tesis. En la Sección I.5 se muestra la importancia de este trabajo de investigación. Finalmente, la Sección I.6 muestra un panorama general de cómo está organizada la tesis.

I.1 Auto-estabilización

En 1974 Edsger W. Dijkstra introdujo el concepto de auto-estabilización como un paradigma para el diseño de algoritmos distribuidos tolerantes a fallas. Dijkstra define auto-estabilización como: a pesar del estado inicial de un sistema, éste garantiza que sin intervención externa se encontrará en un estado legítimo después de un número finito de pasos (Dijkstra, 1974).

Dijkstra además observó que es díficil diseñar un sistema auto-estabilizante y esta

dificultad radica en que los nodos del sistema sólo contienen información local (es decir, información de sí mismo y de sus vecinos) y de alguna manera se tiene que lograr la estabilización solamente con esta información. Antes de definir de manera formal auto-estabilización se describen algunos conceptos preliminares.

La topología de un sistema es el grafo dirigido formado por los componentes del sistema, donde los nodos representan a los nodos y las aristas representan las líneas de comunicación. Cada componente del sistema contiene un estado local (conjunto de variables locales de cada nodo). El estado global del sistema se define como la unión de los estados locales de los componentes del sistema.

Schneider (1993) define formalmente auto-estabilización como:

Para un sistema S y un predicado P (donde P es una especificación para una ejecución correcta del sistema S), el sistema S es auto-estabilizante con respecto a P si satisface dos propiedades:

1. Convergencia: Iniciando de un estado global arbitrario, se garantiza que S alcanza un estado global que satisfaga P en un número finito de transiciones de estado.

2. Cerradura: P es cerrado bajo la ejecución de S; esto es, una vez que P es verdadero en S, no se puede convertir en falso.

En pocas palabras, la propiedad de convergencia dice que el sistema, a partir de cualquier estado inicial, llegará a un estado legal o legítimo en un número finito de pasos. Un estado legal se define como un estado libre de errores, en caso contrario, el estado se llama ilegal o ilegítimo. La propiedad de cerradura dice que una vez que se ha alcanzado el estado legal, el sistema no se cambiará a un estado ilegal en la ausencia

de fallas.

Un sistema auto-estabilizante tiene dos propiedades útiles e interesantes:

1. No se necesita inicializar. Por la propiedad de convergencia, el sistema en algún momento alcanza un estado legal, así que no es necesario que el sistema inicie en un estado legítimo predeterminado.

2. Puede recuperarse de errores transitorios. Después de haber sufrido una falla, el sistema puede iniciar una recuperación para corregir la falla.

El uso de sistemas auto-estabilizantes se ha ido ampliando por la gran ventaja que ofrecen, ya que la habilidad para recuperarse de errores sin intervención externa es muy deseable en los sistemas distribuidos.

Se han diseñado y utilizado una gran cantidad de algoritmos auto-estabilizantes para muchas aplicaciones como lo son: protocolos de comunicación (Dolev *et al.*, 1995; Dolev y Kat, 2004; Cramer y Fuhrmann, 2005; Chen *et al.*, 2005; Kersch *et al.*, 2008), para sistemas operativos (Dolev y Yagel, 2005b,a), para sincronizadores (Boulinier *et al.*, 2008, 2004; Xu *et al.*, 2003), por mencionar algunos ejemplos.

I.2 Comprobación de Convergencia

Un punto fundamental asociado a los algoritmos auto-estabilizantes es el demostrar su propiedad de convergencia. Debido a esta propiedad, el diseñador requiere ser muy cuidadoso al crear un algoritmo auto-estabilizante. Se tienen que tomar en cuenta dos factores al analizar los algoritmos auto-estabilizantes.

- Demostrar la convergencia, es decir, comprobar que el algoritmo se estabiliza con un número finito de pasos.

- Calcular el tiempo de convergencia. Como no es necesario inicializar el algoritmo en algún estado, la tarea de calcular el tiempo en que el algoritmo se estabiliza requiere de técnicas sofisticadas.

Es por esto, que se han desarrollado técnicas y estrategias para probar la convergencia de estos algoritmos, con la limitante que muchas de estas técnicas resultan ser complejas y requieren de gran entendimiento del sistema. Otra limitante que existe es que muchas de las técnicas que se han utilizado para demostrar la convergencia de los algoritmo auto-estabilizantes sólo sirven para demostrar tal condición, dejando una incógnita en el tiempo que se requiere para llegar a la convergencia.

Demostrar que un algoritmo es auto-estabilizante es un reto interesante y las investigaciones preliminares que se han hecho respecto a este tema muestran diferentes enfoques. En su mayoría utilizan funciones para demostrar la propiedad de convergencia y algunas de estas investigaciones están orientadas a topologías simétricas. Una topología es la forma en que están dispuestos los enlaces de comunicación en el sistema y se dice que es simétrica cuando colocándose en cualquier nodo del sistema, éste tiene el mismo número de aristas que los demás nodos del sistema, es decir cada nodo "mira" lo mismo que cualquier otro nodo. Algunos ejemplos de este tipo de topología son los anillos, el hipercubo, grafos completos (aquéllos en los cuales entre cada par de nodos existe una arista), por nombrar algunos.

Debido a lo anterior, se puede ver que es deseable que las técnicas que normalmente se usan en topologías simétricas también se puedan extender a topologías asimétricas, por ejemplo una topología de árbol. La idea de utilizar estas técnicas es que sean útiles para demostrar no solo la convergencia del algoritmo, sino que den una cota en el tiempo de convergencia.

La técnica de acoplamiento (*Coupling*, como se usa en el campo de la Probabilidad Aplicada (Aldous, 1983; Lindvall, 2002)), además de demostrar la propiedad de convergencia de algoritmos auto-estabilizantes, ofrece una cota superior en el tiempo esperado de convergencia. El método de acoplamiento se utiliza para analizar la tasa de convergencia al equilibrio de una cadena de Markov en experimentos Monte Carlo (Sinclair, 1997). El tiempo de acoplamiento es el tiempo en que dos copias fieles de un proceso estocástico colisionan (Fribourg *et al.*, 2006). Esta técnica comprueba la convergencia de algoritmos auto-estabilizantes combinando dos medidas de tasa de convergencia de una cadena de Markov: el tiempo esperado de alcance (*expected hitting time*, en inglés), que es el tiempo esperado en el cual se ha alcanzado el estado legal a partir del peor estado inicial y el tiempo de absorción-ε (ε- *absorption time*, en inglés) que es el tiempo necesario para alcanzar el estado legal con una probabilidad de al menos $1 - \varepsilon$. Fribourg *et al.* (2006).

I.3 Objetivos

El objetivo principal de este trabajo de investigación es el de ampliar el uso de la técnica de acoplamiento en diversos algoritmos auto-estabilizantes, bajo topologías asimétricas.

También se propone aplicar esta técnica a algunos algoritmos auto-estabilizantes cuya propiedad de convergencia se haya demostrado por algún otro método.

I.4 Actividades Desarrolladas

Las actividades que se realizaron durante este trabajo de investigación son las siguientes.

1. Se hizo una revisión exhaustiva sobre las técnicas que se utilizan para demostrar la convergencia de algoritmos auto-estabilizantes representando la información encontrada por medio de una tabla informativa (ver página 23).

2. Se estudió el comportamiento del algoritmo de Herman (Herman, 1990) y se aplicó a una topología asimétrica.

3. Se observó el comportamiento del algoritmo del Dilema del Prisionero Iterado (Dyer *et al.*, 2002) y se hizo un cambio de topología de manera virtual para poder correr este algoritmo como si estuviera en la topología para la que originalmente fue diseñado.

4. Se analizó el protocolo de espacio mínimo (Dolev *et al.*, 1995) y se utilizó la técnica de acoplamiento para demostrar su convergencia y su tiempo esperado de convergencia.

5. Se analizó el algoritmo de Israeli y Jalfon (1990), se hacen cambios en el algoritmo, y se utiliza la técnica de acoplamiento para obtener un cota superior en el tiempo esperado de convergencia.

I.5 Contribuciones de esta Tesis

La principal contribución de esta tesis es la de ampliar el uso de la técnica de acoplamiento para demostrar la convergencia y el tiempo de convergencia de algoritmos auto-estabilizantes bajo topologías asimétricas. Para lograr esto, se definen algunas propiedades del comportamiento de algunos algoritmos auto-estabilizantes para que se pueda utilizar dicha técnica en ellos; como ejemplo se analizó el algoritmo de elección de líder de Herman (Herman, 1990) y el algoritmo del Dilema del Prisionero Iterado (Dyer *et al.*, 2002), y además, como otra contribución de este trabajo de investigación es que se modelan éstos dos algoritmos auto-estabilizantes en una topología de árbol para aplicarles la técnica de acoplamiento.

Adicionalmente, se utiliza la técnica de acoplamiento en el protocolo de espacio mínimo de Dolev *et al.* (1995) obteniendo una cota más ajustada en el tiempo de convergencia de este algoritmo que la que el autor señala.

Además se calcula la cota en el tiempo de convergencia del algoritmo de Israeli y Jalfon (1990); este algoritmo se modificó para aplicarle la técnica de acoplamiento.

I.6 Panorama General del Documento

En el Capítulo 2 se explican a detalle las técnicas que se utilizan para demostrar la propiedad de convergencia de los algoritmos auto-estabilizantes.

El Capítulo 3 muestra los conceptos relacionados con la técnica de acoplamiento de forma detallada.

En el Capítulo 4 se hace el cambio de topología en dos sistemas para que se puedan utilizar dos algoritmos auto-estabilizantes (el algoritmo de Herman (Herman, 1990) y el algoritmo del Dilema del Prisionero Iterado (Dyer *et al.*, 2002)) cuyas convergencias ya habian sido demostradas con la técnica de acoplamiento.

El Capítulo 5 explica el protocolo de espacio mínimo (Dolev *et al.*, 1995) y cómo se demuestra y se obtiene la cota en el tiempo esperado de su convergerncia a través de la técnica de acoplamiento.

El Capítulo 6 describe la demostración de convergencia del algoritmo de Israeli y Jalfon (1990) en el cual se utiliza la técnica de acoplamiento y se proporciona una cota superior en el tiempo esperado de convergencia.

El Capítulo 7 muestra las conclusiones que surgen en este trabajo de investigación, expone las aportaciones que se derivan del mismo, y proporciona una idea sobre posible trabajo futuro dentro de esta línea.

Capítulo II

Métodos de Comprobación de Convergencia

II.1 Introducción

Aquí se describen algunas de las técnicas básicas utilizadas para probar la convergencia de los algoritmos auto-estabilizantes.

Los métodos que se mencionan en este capítulo son los que se han encontrado en la mayor parte de la literatura donde se reportan algoritmos auto-estabilizantes y se demuestra su convergencia al estado legal.

Entre los métodos que se mencionan se encuentran las funciones variantes, el cual es un método clásico muy utilizado. Otros métodos están asociados a la teoría de control; éstos se han utilizado en el paradigma de auto-estabilización por su similitud con los circuitos de control. Otro de los métodos es el de escalones de convergencia, el cual es poco popular ya que el número de predicados que utiliza el sistema dificulta considerablemente su demostración. Posteriormente se describe el método de grafos de dependencia, el cual ofrece un panorama muy amplio del sistema que se está analizando, pero a su vez puede ser muy complejo de utilizar.

II.2 Funciones Variantes

Las funciones variantes es una estrategia clásica. La idea de esta técnica es usar una función que mapea el conjunto de todos los estados del sistema al conjunto de los números reales. Después se demuestra que esta función decrece (o crece) monotónicamente cada vez que los nodos ejecutan un paso y finalmente cuando la función alcanza cierto umbral, el sistema se encuentra en una configuración deseada.

Para demostrar la auto-estabilización, primero se demuestra que el sistema puede hacer una transición siempre y cuando no esté estabilizado y se utiliza la función variante que irá creciendo o decreciendo mientras no se haya llegado a la configuración deseada. Esto implica que el valor de la función es máximo o mínimo si el sistema está estabilizado.

Aunque aparece por primera vez en (Dijkstra, 1986), no se identifica a las funciones utilizadas con el nombre de funciones variantes; fue en (Kessels, 1988) donde se le reconoció como una técnica para comprobar la convergencia y se le asignó el nombre de funciones variantes. En (Huang, 1993) se menciona que una función variante depende de la configuración del sistema y se define en base al propio sistema. Cramer y Fuhrmann en (Cramer y Fuhrmann, 2005) utilizan funciones variantes para demostrar que el sistema se auto-estabiliza de manera local y global.

II.2.1 Ejemplo de Función Variante

A continuación se muestra un ejemplo de cómo funciona esta técnica. En (Huang, 1993), Huang presenta un algoritmo auto-estabilizante de elección de líder para anillos

uniformes. Se dice que un anillo de nodos es uniforme si todos los nodos son equivalentes desde el punto de vista lógico y están programados idénticamente. El anillo tiene n nodos, n es primo y además $n \geq 2$. Cada nodo tiene un identificador que toma valores del conjunto$\{0, 1, \ldots, (n-1)\}$. Sean a y b dos nodos, la función $g(a, b)$ denota un "espacio" entre a y b, la cuál se define como:

$$g(a,b) = \begin{cases} n & \text{si } a = b; \\ b - a \ mod \ n & \text{de otra manera} \end{cases}$$

La función $g(a, b)$ puede tomar valores del conjunto $\{1, 2, \ldots, n\}$. Los identificadores y espacios siempre siguen una dirección en sentido de las manecillas del reloj a través del anillo. El protocolo ajusta a los identificadores para que eventualmente se alcance una configuración estable, esto es, los espacios de cualquier par de nodos sucesivos tienen el mismo valor y todos los identificadores en tal configuración son distintos, y uno de ellos tiene el identificador 0, el cual se considera líder.

Este protocolo de elección de líder consiste de ciertas reglas para ajustar a los identificadores y a los espacios (para más detalle ver (Huang, 1993)). La Tabla I muestra un ejemplo de ejecución de este protocolo. El ejemplo es para $n = 5$, la configuración inicial de los identificadores es a, donde los números en los paréntesis "()" son espacios (el identificador subrayado enfatiza cuál nodo hará el movimiento), al aplicar ciertas reglas a la configuración a se obtiene la configuración b y así sucesivamente hasta llegar a la configuración f, la cual es una configuración estable.

En la Tabla I, en la configuración f existe un nodo único con identificador 0 y ter-

Tabla I. Ejemplo de ejecución de algoritmo de Huang

Configuración	
a	$0(2)2(5)\underline{2}(5)2(2)4(1),\ (G,B)=(59,1)$
b	$0(2)2(1)\underline{3}(4)2(2)4(1),\ (G,B)=(26,1)$
c	$0(2)2(2)\underline{4}(3)2(2)4(1),\ (G,B)=(22,4)$
d	$0(2)\underline{2}(3)0(2)2(2)4(1),\ (G,B)=(22,2)$
e	$\underline{0}(3)3(2)0(2)2(2)4(1),\ (G,B)=(22,0)$
f	$1(2)3(2)0(2)2(2)4(2),\ (G,B)=(22,0)$

mina el protocolo. Este nodo único es el que se elige líder.

Sea $F{\equiv}(G,B)$ una función variante definida sobre el sistema, la cual se compone de G y B que son funciones acotadas. La comparación de los valores de F es por orden lexicográfico, es decir, se considera dos secuencias de caracteres $a=[a_1,\ a_2,\ \ldots,\ a_n]$ y $b=[b_1,\ b_2,\ \ldots,\ b_m]$ se dice que $a<b$ si ambas secuencias tienen un prefijo común de tamaño i y el primer caracter diferente es $a_{i+1}<b_{i+1}$ ejemplo: $merced<mercurio$, porque $e<u$. Las funciones acotadas G y B están definidas como siguen. G representa la sumatoria del cuadrado de cada uno de los espacios, *i.e.* $G{\equiv}\sum g_l^2$. B se define al cumplirse las condiciones entre los espacios especificadas en (Huang, 1993). Cada vez que el sistema hace un movimiento, F decrece monotónicamente, por lo tanto el protocolo termina y se estabiliza. En la Tabla I se puede ver cómo F va disminuyendo monotónicamente, de $(59,1)\rightarrow(26,1)\rightarrow(22,4)\rightarrow(22,2)\rightarrow(22,0)\rightarrow(22,0)$. Como se menciona anteriormente F está definido como $F{\equiv}(G,B)$.

En la siguiente sección se explica otro método para la demostración de convergencia de algoritmos auto-estabilizantes. Este método se basa en la teoría de control, ya que los algoritmos auto-estabilizantes exhiben una analogía a la estabilización de circuitos de control usado en varios dominios de la ingeniería (Theel y Gartner, 1998).

II.3 Teoría de Control

La teoría de control más que ser un solo método, es un grupo de métodos utilizados en la estabilización de control de circuitos ya que se puede ver una analogía entre auto-estabilización y la teoría de control. Contrario al dominio de la auto-estabilización que es un área relativamente nueva de investigación en las ciencias de la computación, la teoría de control en el dominio de la ingeniería tiene más de 100 años investigándose y ofrece bases bien fundamentadas con criterios poderosos para razonar sobre la estabilidad en el análisis y diseño de sistemas dinámicos y su control.

II.3.1 Modelado de un Sistema Auto-estabilizante a un Circuito de Control

En (Theel y Gartner, 1998) se modela un sistema auto-estabilizante como un circuito de control (es aquel que tiene una entrada, un controlador, una salida y existe una retroalimentación de la entrada a la salida). La prueba de corrección en términos de estabilidad se puede obtener fácilmente. Adicionalmente, una demostración por medio de la teoría de control puede ser constructiva, es decir, indica cómo un circuito de control no estable se debe modificar para convertirlo en uno estable.

En (Theel y Gartner, 1998) se ejemplifica un sistema administrativo donde se tienen clientes, intermediarios y proveedores de un producto. Los proveedores venden paquetes de productos a los intermediarios y éstos a su vez, venden los productos a sus clientes. El objetivo es regular el número de paquetes que se almacenan en los intermediarios, de manera que en todo momento, todos los intermediarios tengan el mismo número

de paquetes, aunque éstos disminuyan el número de sus paquetes de productos por las ventas que realizan con los clientes.

Se modela el escenario de la aplicación en términos de un circuito de control discreto y lineal. La demostración de estabilidad para el circuito de control se obtiene de la siguiente manera:

1) Cuando el modelo es discretizado se obtienen las transformadas Z (modelo matemático que se emplea entre otras aplicaciones en el procesamiento de señales digitales, como son el análisis y proyectos de circuitos digitales) de todos los componentes del circuito de control.

2) Se deriva la transformada Z del comportamiento de la salida y entrada del circuito de control, basada en las transformadas Z del paso 1.

3) Para concluir la demostración, simplemente se muestra que todas las singularidades de la fórmula obtenida en el paso 2 caen dentro de un círculo de radio 1 alrededor del origen en el plano complejo z.

II.3.2 Funciones de Transferencia

En (Theel y Gartner, 1999) la tarea de estabilización es diseñar y verificar un algoritmo auto-estabilizante con base en un autómata A, que dado un objetivo específico w, garantiza que la carga local en A siempre estabiliza w una vez que los cambios dentro de las cargas locales ya no suceden. Formalmente esto se puede ver como las dos

propiedades de auto-estabilización: Cerradura y Convergencia.

Se definen las funciones de transferencia (cociente de la salida entre la entrada) del comportamiento de las entradas-salidas de A y las transformadas Z de esos comportamientos y funciones. Si todos los extremos de las transformadas Z de las funciones de transferencia caen en un círculo de radio 1 alrededor del origen del plano z, entonces el sistema entero se estabilizará para cualquier valor de w.

El ejemplo que se usa a continuación se obtiene de (Theel y Gartner, 1999). Se considera el sistema simple en la Figura 1. Consiste de un autómata que lee su entrada de un canal de comunicación *in* y escribe la salida a un canal *out*. El comportamiento entrada/salida es bien definido y del siguiente tipo: el sistema encapsula una forma de carga que se puede leer por el canal *out* dado como un número. Una vez que el sistema recibe una señal positiva sobre la línea de entrada, empieza a producir cargas y por lo tanto a incrementar la carga local. Por otra parte, si se recibe una señal negativa, el sistema empieza a destruir cargas y por lo tanto a decrementar las cargas internas. Al autómata lo puede influenciar el ambiente que ocasiona un cambio espóntaneo de carga local, indicado por los rayos en la Figura 1 . Las fuentes de tales cambios pueden ser fallas transitorias internas, mensajes corruptos vía canal *in* o una salida maliciosa alterada u otro cambio externo de carga local.

La tarea de estabilización actual es diseñar y verificar un algoritmo auto-estabilizante alrededor de A el cual, dado un valor objetivo fijo w, garantiza que la carga local en A siempre estabiliza a w una vez que los cambios espontáneos de la carga local de A causados por el ambiente se detienen.

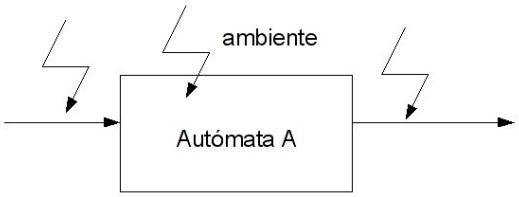

Figura 1. Autómata

Se usa un sensor/actuador simple como un acercamiento para combatir el problema. Al hacer esto, un sensor se agrega al canal de salida de A. Este componente se usa para "observar" el valor local de la carga al escuchar la salida. El valor observado se pasa a un actuador, el cual reside en el canal in de A. Se utiliza el sensor para influenciar la carga local al pasar la información a A vía in.

Al comparar el valor objetivo de w y la información dada por el sensor, ahora el actuador debe calcular una entrada para A, el cual eventualmente lleva a un estado estable del conteo de la carga igual a w.

Ahora se usan resultados de la teoría de control para derivar en una implementación del actuador. Para esto, se necesita alguna información del comportamiento salida-entrada de A. Con respecto al tiempo, el comportamiento entrada-salida se puede denotar como:

$$G_A(t) = \frac{out(t)}{in(t)} = \frac{1}{1 - z^{-1}} \tag{1}$$

A la fórmula 1 se le llama *la transformada z* del comportamiento entrada-salida de

A. La tarea de estabilización se reduce a derivar una "buena" función de transferencia, la cual produce que el sistema entero sea estable.

II.3.3 Segundo Método de Ljapunov

La idea básica de este método es identificar más fácilmente un tipo de función variante. Antes de aplicar el método de Ljapunov se deben investigar los estados de equilibrio del sistema, es decir, aquellos estados de los cuales el sistema no sale en ausencia de fallas una vez que se ha entrado a ellos.

Esta técnica se utiliza en (Theel, 2001) y generaliza el método, de tal manera que se pueda adoptar aún en los casos en que el sistema no sea lineal. Se presenta como ejemplo el algoritmo llamado *Un juego de cartas* visto en (Huang, 1993):

> Suponga un conjunto finito de jugadores sentados alrededor de una mesa. Inicialmente, cada jugador mantiene un número finito de cartas. El único movimiento de cada jugador es pasar una carta a su vecino izquierdo, si el vecino izquierdo tiene menos cartas que el jugador. El juego termina cuando ningún movimiento es posible.

Se hace un modelado del sistema, en el cual el método de Ljapunov se puede implementar y se aplica este modelo al Juego de Cartas. Los predicados constan de una condición y una acción, al cumplirse la condición, se ejecuta la acción y se hace un monitoreo constante de estos comandos. Estos comandos se modelan a través de un regulador R_i considerando que el predicado es parte de una *función de cambio*.

Se dice que la demostración de la propiedad de convergencia de los algoritmos auto-estabilizantes con Ljapunov se transforma a la identificación de una función de Ljapunov (Theel, 2001). Si tal función se puede identificar, entonces se garantiza en gran medida que existe una estabilidad asintótica uniforme. Esto significa que el sistema converge desde cualquier punto en el espacio de estados al estado de equilibrio. La propiedad de cerradura de los algoritmos auto-estabilizantes también se garantiza: una vez que se ha alcanzado un estado de equilibrio no se sale de este estado cuando no existen fallas que modifican al sistema. Por lo tanto se demuestra la auto-estabilización.

En la siguiente sección se puede encontrar el método de escalones de convergencia el cual se utiliza en (Dolev, 2000; Gouda y Multari, 1991), donde es posible demostrar la convergencia a través de los predicados del sistema.

II.4 Escalones de Convergencia

Es posible probar la convergencia de un algoritmo auto-estabilizante al probar que converge para satisfacer $k > 1$ condiciones $A_1, A_2, \ldots, A_k$ tal que, para cada $1 \leq i \leq k$, A_{i+1} es un *refinamiento* de A_i. El término *atractor* se usa comúnmente para tal condición A_i. Esto es, se demuestra que para un cierto punto de la ejecución, cada configuración satisface a la condición A_1. Cuando la condición A_1 se satisface, se demuestra que la condición A_2 se alcanza y así sucesivamente. De forma general, se demuestra que A_{i+1} se mantiene después de que A_i lo hizo. La meta es demostrar que el sistema alcanza una configuración de tal manera que la última condición A_k se mantiene, la cual es una condición para una configuración legal (Dolev, 2000).

En (Gouda y Multari, 1991) se define el método para verificar si un protocolo dado es R-estabilizante por alguna condición R-cerrada. El método está sugerido por el Teorema 1.

Teorema 1. Un protocolo es R-estabilizante si y solo si tiene una secuencia $(R_1, \ldots, R_n)$ de condiciones globales que satisfacen:

a) Límite: $R_1 =$ verdadero y $R_n = R$.

b) Cerradura: Cada R_i es cerrado.

c) Convergencia: Cada R_i converge a R_{i+1} para $i < n$.

Demostración del Teorema 1: Se demuestra en (Gouda y Multari, 1991) que se cumple la condición a); como esta condición se cumple, se verifica que se cumple la condición b); al hacer esta verificación, se prueba que se cumple la condición c); por lo tanto, el sistema es estable ya que la condición c) es una condición para una configuración legal .

□

II.4.1 Ejemplo de Escalones de Convergencia

El ejemplo que se muestra a continuación se utiliza en (Dolev, 2000) y es un protocolo de elección de líder en una red general de comunicación. Un algoritmo auto-estabilizante para esta tarea supone que cada nodo tiene un identificador único en el intervalo de 1 a n, donde n es una cota superior en el número de nodos en el sistema. Cada nodo P_i tiene un candidato a líder; al principio, el candidato es el mismo P_i. El nodo P_i

repetidamente comunica a sus vecinos el identificador x de su actual candidato. Siempre que P_i recibe un identificador y de un candidato de un vecino, si $x > y$, P_i cambia su candidato al nodo con identificador y.

La demostración de convergencia usa dos escalones de convergencia. El primer escalón de convergencia es una condición A_1 sobre el sistema de configuraciones, donde se verifica que ningún identificador flotante existe (un identificador flotante se usa para describir un identificador que aparece en la configuración inicial, cuando ningún nodo tiene este identificador). El segundo escalón de convergencia es una condición A_2 para una configuración segura (una condición que verifica que cada nodo escoje al identificador mínimo de un nodo en el sistema como el identificador de un líder).

En la siguiente sección se puede encontrar el método de grafos de dependencia que sirve para demostrar la convergencia de algoritmos auto-estabilizantes basándose en un grafo que surge de las relaciones entre las variables del sistema.

II.5 Grafos de Dependencia

Un grafo de dependencia es un grafo dirigido cuyos nodos corresponden a las variables en los estados locales y cuyas aristas corresponden a las relaciones de dependencia entre estas variables locales. Un sistema de iteración es un modelo de computación paralela, en el cual un vector de estados se actualiza constantemente por un conjunto de funciones no determinísticas. Se presenta en (Motteler y Sidhu, 1993) el problema de los filósofos comensales de Dijkstra como un ejemplo directo de un sistema de iteración. Los grafos de dependencia de un sistema de iteración capturan la relación

existente entre las variables del sistema (Arora *et al.*, 1993).

El grafo de dependencia es un grafo dirigido cuyos nodos corresponden a las variables en V (un vector de n variables) y cuyas aristas dirigidas corresponden a las *dependencias* en la relación; esto es, el grafo de dependencia contiene un nodo por cada elemento de V y una arista dirigida (y, x) si la función de actualización de la variable x depende de y. Al formar el grafo de dependencia se determina cada componente fuertemente conectado de G, que es el subgrafo maximal en donde existe un camino directo entre cada par de nodos.

Para demostrar la convergencia es suficiente con demostrar que cada componente fuertemente conectado de un sistema de iteración es convergente.

La Tabla II muestra una descripción resumida de estos métodos con algunas de sus ventajas y desventajas.

En este capítulo, se describen brevemente las técnicas de comprobación de convergencia de los algoritmos auto-estabilizantes. En el siguiente capítulo se describe con detalle la técnica de acoplamiento y se utilizan algunos ejemplos para clarificar la descripción.

Tabla II. Descripción de técnicas de comprobación de convergencia

Técnica:	Ventajas:	Desventajas:	Usada en:	Tipos de algoritmos:
Funciones variantes	Una vez construida la función, es fácil determinar la convergencia	Construir la función es una tarea complicada. No se sabe qué pasa cuando las funciones se definen y la topología cambia	Anillos, grafos completos	De ruteo en red de sensores, elección de líder y protocolos de comunicación
Funciones de transferencia	Fundamentada en la teoría de control	Está limitada a comportamientos lineales	Anillos	Balanceo de cargas en canales de comunicación
Segundo método de Ljapunov	Existe más información sobre cómo construir la función Ljapunov	Un algoritmo puede ser algorítmicamente simple, pero es sumamente complicado para demostrar lo esencial del método	Anillos	Elección de líder
Escalones de convergencia	La demostración de convergencia se divide en un gran número de demostraciones pequeñas	Poco utilizado ya que se vuelve muy complejo al incrementar el número de predicados	Grafos arbitrarios	Para crear clusters en redes Ad hoc, elección de líder
Grafos de dependencia	Ofrece un panorama muy amplio del sistema	Las relaciones entre las variables locales pueden ser muy complejas	Grafos arbitrarios	Máximo común divisor, encontrar el camino más corto en un grafo
Acoplamiento	Ofrece una cota en el tiempo de convergencia	Limitada a topologías simétricas	Anillos, grafos completos	Elección de líder, exclusión mutua

Capítulo III

Acoplamiento

En este capítulo se discute con más detalle en qué consiste y cómo funciona la técnica de acoplamiento para demostrar la convergencia de algoritmos auto-estabilizantes.

III.1 Definiciones

El método de acoplamiento es una técnica poderosa que ofrece una cota superior en el tiempo esperado en el que una cadena de Markov llega a su distribución estacionaria. Tradicionalmente, la técnica de acoplamiento ha sido una herramienta estándar en la teoría de la probabilidad (usada por ejemplo en (Aldous, 1983; Lindvall, 2002)) y recientemente ha rendido frutos significativos en ciencia de la computación teórica y se utiliza en (Bubley y Dyer, 1997; Dyer y Frieze, 2001; Hayes, 2003).

El método de acoplamiento utiliza cadenas de Markov, donde las cadenas de Markov sirven de herramienta para analizar el comportamiento de determinados tipos de procesos estocásticos, esto es, procesos que evolucionan de forma no determinística a lo largo del tiempo en torno a un conjunto de estados. Una cadena de Markov representa un sistema que varía su estado a lo largo del tiempo siendo cada cambio una transición del sistema. Dichos cambios no están predeterminados, aunque sí lo está la probabilidad del próximo estado en función de los estados anteriores, probabilidad que es constante a lo largo del tiempo.

Formalmente, para definir una cadena de Markov finita hace falta determinar los siguientes elementos (Torrents *et al.*, 2002):

a) Un conjunto de *estados* del sistema. Los cuales son una caracterización de la situación en la que se halla el sistema en un momento dado. Dicha caracterización puede ser tanto cuantitativa como cualitativa.

b) El concepto de *transición.* El cambiar de un estado a otro. El sistema que se modela como la cadena de Markov, por tanto, es una variable que cambia con el tiempo, a ese cambio se le llama transición.

c) Una ley de *probabilidad condicional*, que defina la probabilidad del nuevo estado en función de los estados anteriores.

Ahora bien, un conjunto de estados finitos $S = \{s_1, s_2, \ldots, s_r\}$, se llama cadena de Markov si existe una sucesión de variables aleatorias $\{X_t\}_{t \in T}$ (donde T es el tiempo) definidas sobre el conjunto S, tales que la probabilidad de que la cadena de Markov se encuentre en algún estado en el tiempo $t+1$, depende únicamente del estado en que se encuentra en el tiempo t. El proceso inicia en uno de esos estados y se mueve sucesivamente de un estado a otro. Si la cadena actualmente está en el estado s_i, entonces en el siguiente paso se mueve al estado s_j con una probabilidad denotada por p_{ij} y esta probabilidad no depende en cuál estado estaba la cadena antes del estado actual.

Las probabilidades p_{ij} se llaman *probabilidades de transición.* La probabilidad de que la cadena se mantenga en el mismo estado está dad por p_{ii}. Una distribución de probabilidad inicial, definida sobre S, especifica el estado inicial. Normalmente esto ocurre especificando un estado particular como el estado inicial.

En la cadena de Markov la manera más fácil de expresar la ley de probabilidad condicional de la cadena es mediante una matriz de probabilidades de transición P o de forma más sencilla, matriz de la cadena. Dicha matriz es cuadrada con tantas filas y columnas como estados tiene la cadena y los elementos de la matriz representan las probabilidades de transición p_{ij}.

A continuación se muestra un ejemplo de una cadena de Markov para clarificar los conceptos comentados anteriormente. Este ejemplo se tomó del libro de Kemeny y Snell (1976).

La tierra de Oz nunca tiene dos días soleados. Si tiene un día soleado, tiene la misma probabilidad que el siguiente día llueve o nieva. Si un día nieva o llueve, existe un 50% de probabilidad que el siguiente día sea igual. Si cuando nieva o llueve el clima cambia, existe un 50% de que el siguiente día sea soleado. Para estudiar el problema primero se definen los estados y se encuentran las probabilidades de transición. Para simplificar se denota a L como un día con lluvia, S un día soleado y N como un día con nieve, se tiene:

$P_{SS} = 0$ de un día soleado a un día soleado.

$P_{SL} = \frac{1}{2}$ de un día soleado a un día con lluvia.

$P_{SN} = \frac{1}{2}$ de un día soleado a un día con nieve.

$P_{LL} = \frac{1}{2}$ de un día con lluvia a un día con lluvia.

$P_{LS} = \frac{1}{4}$ de un día con lluvia a un día soleado.

$P_{LN} = \frac{1}{4}$ de un día con lluvia a un día con nieve.

$P_{NN} = \frac{1}{2}$ de un día con nieve a un día con nieve.

$P_{NL} = \frac{1}{4}$ de un día con nieve a un día con lluvia.

$P_{NS} = \frac{1}{4}$ de un día con nieve a un día soleado.

Por conveniencia las transiciones de probabilidad se representan por medio de una matriz de transición.

$$P = \begin{array}{c} \\ L \\ S \\ N \end{array} \begin{array}{c} \begin{array}{ccc} L & S & N \end{array} \\ \left\{ \begin{array}{ccc} .5 & .25 & .25 \\ .5 & 0 & .5 \\ .25 & .25 & .5 \end{array} \right\} \end{array} \tag{2}$$

Las entradas del primer renglón de la Matriz 2 del ejemplo representan las probabilidades para varios tipos de clima que siguen después de un día de lluvia. Similarmente las entradas en el segundo y tercer renglón representan las probabilidades para varios tipos de clima siguientes a un día soleado y un día nevado, respectivamente. Se observa que en la matriz los elementos deben ser positivos y además al sumar cada fila, el resultado que se obtiene es 1; a esta matriz se le llama matriz estocástica.

Una cadena de Markov también puede representarse por medio de grafos dirigidos, con pesos, de la forma (V, A, W) donde los nodos (los elementos de V) son los estados, una arista $a = (u, v)$ está en A si $p_{ij} > 0$, pudiendo darse el caso de auto-lazos, que es cuando un arista empieza y termina en el mismo nodo, y a cada arista se asigna el peso W que es p_{ij}.

El ejemplo del clima en la tierra de Oz es un ejemplo típico de una cadena de Markov: (veáse la Figura 2).

Cuando se desea conocer la probabilidad de que el sistema se encuentre en un

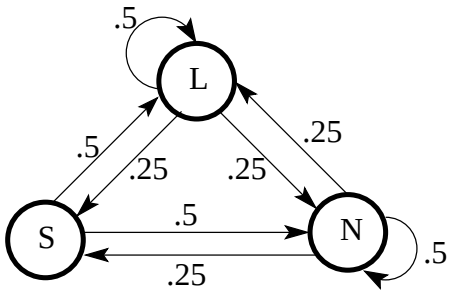

Figura 2. Grafo del clima de la tierra de Oz

estado determinado al llevar funcionando un tiempo indefinidamente largo, se dice que se desea conocer la probabilidad estacionaria. El estudio de las probabilidades estacionarias puede entenderse como el estudio del comportamiento a largo plazo de las cadenas de Markov. Dichas probabilidades se denotan como π_{ij} y la matriz de probabilidades de estado estable como P^*.

Puesto que las potencias de P definen las probabilidades en un número cualquiera de transiciones, se puede aproximar al estudio de las probabilidades viendo qué ocurre si se calculan las potencias elevadas de algunas matrices P. Se puede ver que el comportamiento se estabiliza cuando se pasa por cada uno de los estados con una frecuencia regular, independientemente de las transiciones ocurridas.

Cuando se eleva a una potencia la cadena de Markov y las probabilidades de estado estable son independientes del estado inicial se tiene una cadena regular y si además ninguna de las probabilidades vale cero entonces se tiene una cadena ergódica (Torrents *et al.*, 2002). Cada cadena de Markov finita tiene al menos un conjunto ergódico, se dice que una cadena de Markov es finita si el número de estados en la cadena es un número finito. Además dos conjuntos ergódicos distintos son disjuntos, esto debibo a que cada uno está fuertemente conectado (Fribourg *et al.*, 2006).

Ahora bien, el tiempo de mezclado de una cadena de Markov es el tiempo en el cual la cadena de Markov esta "cercana" a su distribución estacionaria. Más precisamente, un resultado fundamental de las cadenas de Markov es que una cadena ergódica tiene una distribución estacionaria única π, y a pesar de su estado inicial, el tiempo t de la distribución π, en donde la cadena converge, tiende a infinito. El tiempo de mezclado se refiere a varias formalizaciones de la idea: ¿cuán grande debe de ser t hasta que la distribución de probabilidad del vector de estados sea aproximadamente π?.

Dentro del contexto de cadenas de Markov, un conjunto de configuraciones legales es necesariamente un conjunto ergódico. Una cadena de Markov A_{MC} que representa a un algoritmo A es convergente con respecto a un conjunto $\mathcal{L}$ de configuraciones legales si, iniciando de cualquier configuración inicial, el sistema garantiza que alcanza una configuración de $\mathcal{L}$ con un número finito de transiciones. Dado un conjunto $\mathcal{L}$ de configuraciones, A es *auto-estabilizante con respecto a* $\mathcal{L}$ si:

1. $\mathcal{L}$ es fuertemente conectado
2. $\mathcal{L}$ es cerrado y
3. A_{MC} es convergente con respecto a $\mathcal{L}$.

En el contexto probabilístico de las cadenas de Markov, la propiedad de convergencia 3 se tiene que garantizar con probabilidad 1. Formalmente A_{MC} es convergente con respecto a $\mathcal{L}$ (*con probabilidad 1*) si $\forall x \ _{y \in \mathcal{L}} \ \mathbf{P}^t(x, y) \to 1$ cuando $t \to \infty$.

Tomando en cuenta lo anterior, al simular un algoritmo auto-estabilizante como una cadena de Markov y determinar el tiempo de mezclado de esa cadena, se puede

determinar la convergencia del algoritmo simulado. Para determinar el tiempo de mezclado es necesario usar un método para probar la propiedad de auto-estabilización de A con respecto a $\mathcal{L}$. El método que auxilia en encontrar el tiempo de mezclado es el de acoplamiento y usa dos medidas diferentes de convergencia: el *tiempo esperado de alcance* y el *tiempo de absorción-ε* para determinar el tiempo de mezclado.

En la técnica de acoplamiento, el tiempo de convergencia se representa por el tiempo esperado de alcance. Esto es, el tiempo esperado que requiere A_{MC} para alcanzar $\mathcal{L}$, a partir del peor escenario inicial de A (llamada la peor configuración en el resto de este documento). Dada una cadena de Markov A_{MC} y un conjunto $\mathcal{L}$ de configuraciones legales, el tiempo esperado de alcance de $\mathcal{L}$ es:

$$\mathbf{H}_{\mathcal{L}} = max_{x \in \Omega}\ E[\mathbf{H}_{x\mathcal{L}}] \tag{3}$$

donde $E[.]$ denota esperanza y

$$\mathbf{H}_{x\mathcal{L}} = min\{t : X_t \in \mathcal{L} | X_0 = x\} \tag{4}$$

El tiempo esperado de alcance se ve como una analogía al concepto de diámetro en un grafo, ya que el diámetro en un grafo es la distancia más larga de un vértice para llegar a otro, es decir, aquel recorrido que contiene el mayor número de aristas entre dos vértices. La técnica de acoplamiento además utiliza una medida diferente de convergencia, llamada tiempo de absorción-ε, que proporciona el tiempo después del cual se alcanza $\mathcal{L}$ con alta probabilidad. Dada una cadena de Markov A_{MC} y un

conjunto ergódico $\mathcal{L}$, el tiempo de absorción-ε para $\mathcal{L}$ es:

$$\Theta_{\mathcal{L}}(\varepsilon) = max_{x\in\Omega}\ \theta_{x\mathcal{L}}(\varepsilon) \tag{5}$$

donde

$$\theta_{x\mathcal{L}}(\varepsilon) = min\{t:\ Pr(X_t \in \mathcal{L}) \geq 1-\varepsilon | X_0 = x\} \tag{6}$$

Entonces $\theta_{x\mathcal{L}}(\varepsilon)$ es el mínimo número de pasos en el cual A_{MC} alcanza $\mathcal{L}$ con probabilidad de al menos $1-\varepsilon$. Esta noción es similar a la noción del tiempo de mezclado, que mide el número de pasos después de los cuales la cadena A_{MC} esta ε-cercana a su distribución estacionaria. Una cota superior del tiempo de mezclado se calcula comúnmente encontrando el tiempo de acoplamiento.

Una vez introducidos algunos conceptos asociados a las cadenas de Markov y a la técnica de acoplamiento, se da una explicación con más detalle de la técnica de acoplamiento en la siguiente sección.

III.2 Acoplamiento

Se considera una cadena de Markov A_{MC} que está caracterizada por una secuencia de estados, que toman sus valores del espacio Ω de estados de la cadena. El método de acoplamiento es un método probabilístico básico para medir el "tiempo de acuerdo" entre componentes de un proceso estocástico.

Un acoplamiento es una cadena de Markov sobre $\Omega \times \Omega$ definiendo un proceso estocástico $(X_t, Y_t)_{t=1}^{\infty}$ con las propiedades:

1. Cada uno de los procesos(X_t) y (Y_t) es una copia fiel de A_{MC} (dadas configura-

ciones iniciales $X_0 = x$ y $Y_0 = y$), es decir, cada uno de los procesos es la misma cadena de Markov con diferentes configuraciones iniciales.

2. Si $X_t = Y_t$, entonces $X_{t+1} = Y_{t+1}$.

La propiedad 1 asegura que al ver cada proceso por separado, éstos simulan la cadena original. Como el acoplamiento los actualiza simultaneamente y tienden a acercarse de acuerdo a alguna noción de distancia. Una vez que el par de configuraciones concuerdan, la propiedad 2 asegura que seguirán concordando desde ese momento en adelante.

En otras palabras un acoplamiento consiste en definir dos procesos aleatorios que están separados según un criterio de distancia y en cada iteración estos dos procesos se van acercando de tal manera que llegan a un punto donde ya no existe distancia alguna entre estos dos procesos. En la Figura 3 se representa un acoplamiento entre variables que toman sus valores en $\mathbb{R}^2$ para ejemplificar la idea del método.

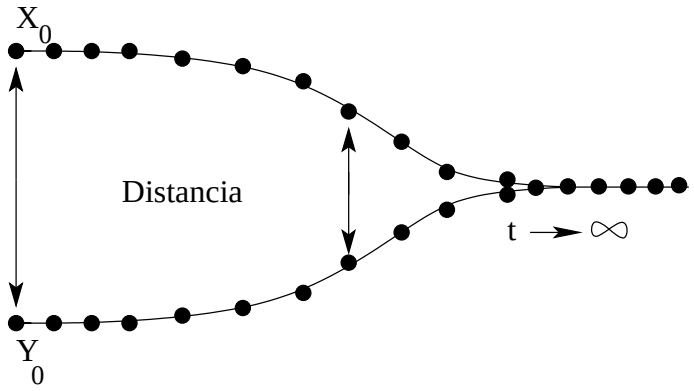

Figura 3. Acoplamiento

Dado un acoplamiento (X_t, Y_t), el tiempo esperado de acoplamiento es:

$$T = max_{x\in\Omega, y\in\Omega} E(T_{x,y}) \tag{7}$$

donde

$$T_{x,y} = min\{t : X_t = Y_t | X_0 = x, Y_0 = y\} \tag{8}$$

El tiempo de acoplamiento se usa normalmente como una cota superior en el tiempo de mezclado. Se demuestra que el tiempo de acoplamiento da una cota superior para el tiempo de mezclado. El Teorema 3 se toma de (Fribourg *et al.*, 2006).

Teorema 2. Dada una cadena de Markov A_{MC} y un conjunto ergódico $\mathcal{L}$, si existe un acoplamiento de tiempo esperado finito T, entonces:

1. A es auto-estabilizante con respecto a $\mathcal{L}$.

2. El tiempo de mezclado $H_{\mathcal{L}}$ es menor que o igual a $T : H_{\mathcal{L}} \leq T$.

Demostración del Teorema 2.

1. Por *reductio ad absurdum*. Suponga que existen dos conjuntos ergódicos no vacíos $\mathcal{L}_1$ y $\mathcal{L}_2$ con dos elementos $X_0 = x \in \mathcal{L}_1$ y $Y_0 = y \in \mathcal{L}_2$. Entonces, para toda $t > 0$, $X_t \in \mathcal{L}_1$ y $Y_t \in \mathcal{L}_2$ ya que $\mathcal{L}_1$ y $\mathcal{L}_2$ son cerrados. Por lo tanto para toda $t > 0$, $X_t \neq Y_t$ ya que $\mathcal{L}_1$ y $\mathcal{L}_2$ son disjuntos. Por lo tanto $T_{x,y}$ es infinito. Asi que T, contradice la suposición.

2. Se demuestra que $H_{\mathcal{L}} \leq T$. Recuerde que: $H_{x\mathcal{L}} = min\{t : X_t \in \mathcal{L} | X_0 = x\}$ y $T_{x,y} = min\{t : X_t = Y_t | X_0 = x, Y_0 = y\}$. Suponga ahora que $y \in \mathcal{L}$. Entonces $Y_t \in \mathcal{L}$

ya que $\mathcal{L}$ es cerrado. Por lo tanto: $H_{x\mathcal{L}} \leq T_{xy}$ para toda $x \in \Omega,\ y \in \mathcal{L}$. Y al tomar las esperanzas, entonces la máxima de los dos lados: $H_{\mathcal{L}} \leq T$.

□

Dos criterios suficientes de auto-estabilización. Por el Teorema 2, al encontrarse una cota superior en el tiempo de acoplamiento T se permite a la vez probar la auto-estabilización y obtener una cota superior en el tiempo esperado de alcance. Al seguir resultados clásicos sobre el tiempo esperado de alcance, se dan dos condiciones suficientes para acotar el tiempo de acoplamiento. En cada caso, esto provee adicionalmente una cota superior no solo para el tiempo esperado de alcance, sino también para el tiempo de absorción-ε. El Teorema 3 se toma de (Fribourg *et al.*, 2006).

Teorema 3. Dada una cadena de Markov A_{MC} y un conjunto ergódico $\mathcal{L}$, suponga que existe un acoplamiento (X_t,Y_t) y una función δ sobre $\Omega \times \Omega$ la cual toma valores en $\{0, 1, \ldots, B\}$ tal que:

$\delta\left(X_t, Y_t\right) = 0$ si y solo si $X_t = Y_t$ y

existe $\beta < 1$ tal que, para toda (X_t, Y_t) :

$$E\left[\delta\left(X_{t+1}, Y_{t+1}\right)\right] \leq \beta\delta\left(X_t, Y_t\right) \tag{9}$$

La ecuación 9 dice que se tiene que demostrar que la distancia entre los dos procesos decrece monotónicamente.

Entonces $\mathcal{L}$ es el conjunto ergódico único y A es auto-estabilizante con respecto a $\mathcal{L}$. Además:

1. El tiempo esperado de alcance satisface:

$$H_{\mathcal{L}} \leq \frac{B}{1-\beta} \tag{10}$$

2. El tiempo de absorción-ε satisface:

$$\Theta_{\mathcal{L}}(\varepsilon) \leq \frac{ln(B/\varepsilon)}{1-\beta} \tag{11}$$

El teorema que se presenta a continuación se toma de Fribourg *et al.* (2006).

Teorema 4. Dada una cadena de Markov A_{MC} y un conjunto ergódico $\mathcal{L}$, suponga que existe un acoplamiento (X_t, Y_t) y una función δ en $\Omega \times \Omega$ que toma valores en $\{0, 1, \ldots, B\}$ tal que:

$\delta\left(X_t, Y_t\right) = 0$ si y solo si $X_t = Y_t$ y

existe $\alpha > 0$ tal que, para toda (X_t, Y_t) con $X_t \neq Y_t$:

$$E\left[\delta\left(X_{t+1}, Y_{t+1}\right)\right] \leq \delta\left(X_t, Y_t\right) \tag{12}$$
$$\wedge Pr\left(\delta\left(X_{t+1}, Y_{t+1}\right) \neq \delta\left(X_t, Y_t\right)\right) \geq \alpha$$

La ecuación 12 dice que se tiene que demostrar que la distancia entre los dos procesos no crece y que existe una probabilidad no nula que cambie.

Entonces $\mathcal{L}$ es un conjunto ergódico único y A es auto-estabilizante con respecto a $\mathcal{L}$. Además:

1. El tiempo esperado de alcance satisface:

$$H_{\mathcal{L}} \leq B^2/\alpha \tag{13}$$

.

2. El tiempo de absorción-ε satisface:

$$\Theta_{\mathcal{L}}(\varepsilon) \leq \left\lceil e\frac{B^2}{\alpha} \right\rceil \left\lceil ln\left(\frac{1}{\varepsilon}\right) \right\rceil \tag{14}$$

Por lo tanto encontrando un acoplamiento (X_t, Y_t) y una función δ tal que 9 o 12 se mantenga para probar que A es auto-estabilizante y que proporciona una cota superior en dos medidas diferentes de convergencia.

A continuación se muestra un ejemplo de cómo funciona el método de acoplamiento, el cual se utiliza en Sinclair (1997).

III.2.1 Ejemplo de Acoplamiento

El espacio de estados (conjunto de configuraciones) es $\Omega = \{0,1\}^n$, el tamaño de todos los vectores booleanos es n, se denota a un vector $x \in \{0,1\}^n$ como $(x_1, \ldots, x_n)$. La función de peso w es constante, entonces la distribución π es uniforme, es decir $\pi(x) = 2^{-n}$ para toda $x \in \{0,1\}^n$. Las configuraciones x y y son adyacentes si y solo si difieren en exactamente una posición. La cadena de Markov *Metrópolis* en este caso es simplemente una caminata aleatoria por el vecino más cercano en los vértices del hipercubo de dimensión n. Un hipercubo es una topología muy utilizada en computadoras paralelas. Ver Figura 4, como ejemplo de un hipercubo de dimensión 4.

Para evitar complicaciones relacionadas con la periocidad, se agrega una probabilidad de permanencia de $\frac{1}{2}$ para cada estado, esto es, en cada paso de la cadena de Markov, con probabilidad $\frac{1}{2}$ no se hace nada, o con probabilidad $\frac{1}{2}$ se hace un movimiento.

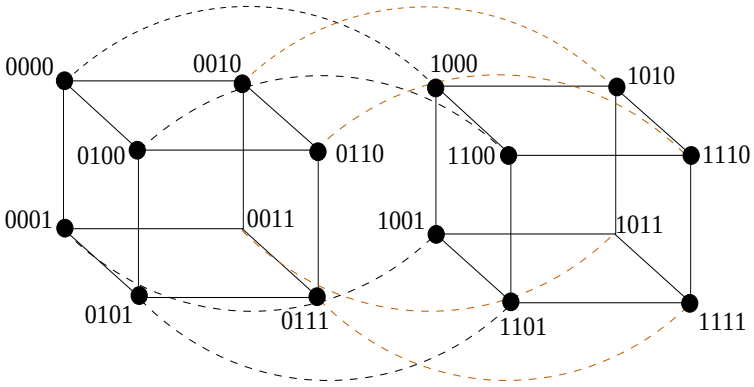

Figura 4. Ejemplo de hipercubo de dimensión 4.

En resumen, la cadena de Markov hace transiciones de cualquier estado $x \in \Omega$ como sigue:

- Escoger una posición $i \in \{1, \ldots, n\}$ uniformemente al azar.
- Con probabilidad $\frac{1}{2}$, girar el i-ésimo bit de x (es decir reemplazar x_i, por $1 - x_i$).
- Con probabilidad $\frac{1}{2}$, no hacer nada.

Primero, se necesita escoger un acoplamiento conveniente, ¿cómo se puede definir una distribución común a dos copias de este proceso para acoplarlos rápidamente?. Una forma es hacer que ambos procesos escojan el mismo bit aleatorio en cada paso, para que así los bits tiendan a acercarse.

Ahora se observa el acoplamiento de la siguiente manera. Si el par de procesos (X_t, Y_t) está en el estado $(x, y) \in \Omega \times \Omega$ se hace lo siguiente:

- Escoger una posición $i \in \{1, \ldots, n\}$ uniformemente al azar.

- Escoger un valor $b \in \{0, 1\}$ uniformemente al azar.
- Asignar $x_i = b$ y $y_i = b$.

Por lo tanto, ambas copias del proceso escogen la misma posición i y el mismo nuevo valor b para el bit asociado. Debe ser claro que éste es un acoplamiento: cada copia vista en aislamiento evoluciona exactamente como la cadena original, por lo tanto la Propiedad 1 del acoplamiento se satisface. El par de procesos nunca causan que los bits que concuerdan desacuerden, por lo tanto la propiedad 2 del acoplamiento también se satisface.

Sea δ el número de posiciones de bits en las cuales X_t y Y_t difieren. Por lo tanto δ es un proceso que toma valores enteros en el intervalo $[0, \ldots, n]$ y $\delta = 0$ si y solo si $X_t = Y_t$. La cantidad $T_{x,y}$ es el tiempo requerido para que δ alcance el cero, dado que $X_0 = x$ y $Y_0 = y$.

¿Cómo cambia δ con el tiempo? La idea principal es que, tan pronto como una posición del bit $i \in \{1, 2, \ldots, n\}$ se elige, los valores x_i, y_i concuerdan y esto persiste todo el tiempo. Esto implica que δ es monotónicamente decreciente: más precisamente, esto implica que, si $\delta_t = d$, entonces:

$$\delta_{t+1} = \begin{cases} d-1 & \textit{con probabilidad } \frac{d}{n} \\ d & \textit{de otra manera} \end{cases} \tag{15}$$

Por lo tanto, para valores iniciales cualesquiera x, y el tiempo $T_{x,y}$ es estocásticamente dominada (Motwani y Raghavan, 1995) por la variable aleatoria

$T_n + T_{n-1} + \cdots + T_1$, donde T_d es el tiempo para que δ decrezca de d a $d-1$. Pero por la ecuación 15, T_d es sólo el número de lanzamientos de una moneda no balanceada con la probabilidad de cara de $\frac{d}{n}$ hasta que la primera cara aparezca. Por lo tanto $E(T_d) = \frac{n}{d}$ y entonces $E(T_{xy}) \leq \sum_{d=1}^{n} E(T_d) \sim n(\ln n + \gamma)$ como $n \to \infty$, donde γ es la constante de Euler (Motwani y Raghavan, 1995). Por la Desigualdad de Markov, se tiene $Pr[T_{xy} > eE(T_{xy})] \leq e^{-1}$ y por lo tanto $\tau \leq max_{x,y} eE(T_{xy}) \leq en(\ln n + O(1))$.

El tiempo de mezclado de la cadena de Markov del hipercubo se acota por arriba por:

$$\tau_x(\epsilon) \leq en(\ln n + O(1)) \left\lceil \ln \epsilon^{-1} \right\rceil$$

III.3 Refinamiento de la Técnica de Acoplamiento

Como se menciona en (Randall, 2003), a menudo es díficil medir el cambio esperado en la distancia entre dos configuraciones arbitrarias. El método de camino de acoplamiento introducido por Bubley y Dyer (1997) simplifica el acercamiento demostrando que solo el par de configuraciones que son "cercanas" se necesitan considerar. El método de camino de acoplamiento implica definir un acoplamiento (X_t, Y_t) considerando un *camino*, o secuencia $X_t = Z_0, Z_1, \ldots, Z_n = Y_t$ entre X_t y Y_t donde Z_i satisface ciertas condiciones.

III.3.1 Camino de Acoplamiento

El siguiente Lema se toma de (Dyer y Greenhill, 1998).

Lema 1. Sea δ una métrica de valor entero definida en $\Omega \times \Omega$ que toma valores en $\{0, \ldots, B\}$. Sea U un subconjunto de $\Omega \times \Omega$. Para todo $(X_t, Y_t) \in \Omega \times \Omega$, existe un

camino $X_t = Z_0, Z_1, \ldots, Z_r = Y_t$ entre X_t y Y_t tal que $(Z_i, Z_{i+1}) \in U$ para $0 \leq i \leq r$ y $\sum_{i=0}^{r-1} \delta(Z_i, Z_{i+1}) = \delta(X_t, Y_t)$. Suponga que existe un acoplamiento $(X, Y) \mapsto (X', Y')$ de la cadena de Markov A sobre todos los pares $(X, Y) \in U$ y una constante $\beta \leq 1$ tal que, para todo $(X, Y) \in U$:

$$E\left[\delta\left(X', Y'\right)\right] \leq \beta\delta(X, Y) \tag{16}$$

Entonces este acoplamiento se puede extender a un acoplamiento de A sobre todos los pares $(X, Y) \in \Omega \times \Omega$ que también satisface 16.

Se dice que dos configuraciones X y Y son adyacentes si $(X, Y) \in U$. La ventaja del Lema 1 es que permite verificar la desigualdad 16 solamente sobre el conjunto U de pares adyacentes en lugar del espacio entero $\Omega \times \Omega$. A continuación se muestran dos ejemplos utilizados en (Fribourg *et al.*, 2006) para aclarar la idea.

III.3.2 Ejemplo: Aplicación al Algoritmo de Herman

Se considera el algoritmo de Herman de exclusión mutua (Herman, 1990). La topología es un grafo cíclico (anillo) de N nodos y un calendarizador síncrono (que selecciona en cada iteración a todos los nodos). El conjunto de estados locales es $Q = \{0, 1\}$ y el número de nodos es impar. En cada iteración, el estado de cada nodo $x(i)$, $1 \leq i \leq N$ se cambia a $x'\left(i\right)$ de la siguiente forma:

- Si $x\left(i\right) \neq x\left(i-1\right)$ entonces $x'(i) = \neg x(i)$.
- Si $x\left(i\right) = x\left(i-1\right)$ entonces:

$$x'(i) = \begin{cases} 0 & \text{con probabilidad } 1/2 \\ 1 & \text{con probabilidad } 1/2 \end{cases}$$

(Cuando i=1, $(i-1)$ es el nodo N. Como es usual, $\neg 0$ es 1 y $\neg 1$ es 0).

La siguiente demostración se toma de Fribourg *et al.* (2006). Para el algoritmo de Herman y N impar, existe un subconjunto de $\Omega \times \Omega$ y una métrica δ sobre $\Omega \times \Omega$ que toma valores en $\{0,...,N\}$ y satisface la desigualdad 16 y un acoplamiento tal que:

- $\forall (X_t, Y_t) \in U \; E[\delta(X_{t+1}, Y_{t+1})] \leq \delta(X_t, Y_t) \; y$
- $\forall (X_t, Y_t) \in \Omega \times \Omega \; (con \; X_t \neq Y_t)$:

$$Pr(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq 1/2 \tag{17}$$

Demostración de 17:

Subconjunto U y métrica δ. Se define δ como la distancia Hamming: $\delta(X_t, Y_t)$ es el número de posiciones en las cuales X_t y Y_t difieren. El par (X_t, Y_t) pertenece a U si y solo si $\delta(X_t, Y_t) = 1$.

Acoplamiento. El acoplamiento se define en orden para forzar a X_t y Y_t a que realicen la misma acción, cuando ambos tienen que ejecutar una acción aleatoria. En otras palabras, para toda i $(1 \leq i \leq N)$:

Si $X_t(i) = X_t(i-1)$ y $Y_t(i) = Y_t(i-1)$ entonces:

$$X_{t+1}(i) = Y_{t+1}(i) = \begin{cases} 0 & \text{con probabilidad } 1/2 \\ 1 & \text{con probabilidad } 1/2 \end{cases}$$

Demostración de $E[\delta(X_{t+1}, Y_{t+1})] = \delta(X_t, Y_t)$ *sobre* U. Considere un par $(X_t, Y_t) \in U$ y sea ℓ la posición de desacuerdo entre X_t y Y_t. Para aclarar ideas considere el siguiente vector:

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} v_1 & v_2 & \dots & v_{\ell-2} & 0 & \mathbf{0} & 0 & v_{\ell+2} & \dots & v_N \\ v_1 & v_2 & \dots & v_{\ell-2} & 0 & \mathbf{1} & 0 & v_{\ell+2} & \dots & v_N \end{pmatrix}$$

donde todas las v_i están en $\{0,1\}$, los símbolos en negritas corresponden a las posiciones ℓ (los otros casos son similares). Después de un paso, el estado de todos los nodos en la posición $1, \dots, N$ se actualiza y se obtiene:

$$\begin{pmatrix} X_{t+1} \\ Y_{t+1} \end{pmatrix} = \begin{pmatrix} v'_1 & v'_2 & \dots & v'_{\ell-2} & v'_{\ell-1} & ? & ? & v'_{\ell+2} & \dots & v'_N \\ v'_1 & v'_2 & \dots & v'_{\ell-2} & v'_{\ell-1} & \mathbf{0} & 1 & v'_{\ell+2} & \dots & v'_N \end{pmatrix}$$

donde '?' significa "0 con probabilidad 1/2 y 1 con probabilidad 1/2". Note que, para $1 \leq i \leq \ell-1$ y $\ell+2 \leq i \leq N$, $X_{t+1}(i) = Y_{t+1}(i) = v'_i$ gracias al acoplamiento. Entonces X_{t+1} y Y_{t+1} coinciden en todas partes, excepto quizás en las posiciones ℓ o $\ell+1$ y se obtiene:

$$\delta(X_{t+1}, Y_{+1}) = \begin{cases} 0 \text{ con probabilidad } 1/4, \\ 1 \text{ con probabilidad } 1/2, \\ 2 \text{ con probabilidad } 1/4. \end{cases}$$

Por lo tanto $E[\delta(X_{t+1}, Y_{t+1})] = \delta(X_t, Y_t)$ para toda $(X_t, Y_t) \in U$.

□

Demostración de $Pr(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq 1/2$. Se denota a q como el número de tokens en desacuerdo (un token de desacuerdo es una posición i tal que

$X_t(i-1) = X_t(i) \neq Y_t(i-1) = Y_t(i))$ y por p el número de zonas de posiciones contiguas de desacuerdo. Se identifican las tres fuentes de posibles evoluciones del conjunto de posiciones de desacuerdos después de un paso:

1. Gracias al acoplamiento, cada token de desacuerdo $X_t(i-1) = X_t(i) \neq Y_t(i-1) = Y_t(i)$ evoluciona en una nueva posición de acuerdo $X_{t+1}(i) = Y_{t+1}(i)$ con probabilidad 1.

2. Cada primera posición en una zona de desacuerdo, por ejemplo i, tal que $X_t(i-1) = Y_t(i-1)$ y $X_t(i) \neq Y_t(i)$ puede evolucionar en una posición de acuerdo con probabilidad $1/2$. Se denota por r el número de tal i $(0 \leq r \leq p)$.

3. Cada primera posición en una zona de acuerdo, por ejemplo i, tal que $X_t(i-1) \neq Y_t(i-1)$ y $X_t(i) = Y_t(i)$ puede evolucionar en una posición de desacuerdo con probabilidad $1/2$. Se denota por s el número i $(0 \leq r \leq p)$. En la Figura 5 se muestra la evolución de una zona de desacuerdo.

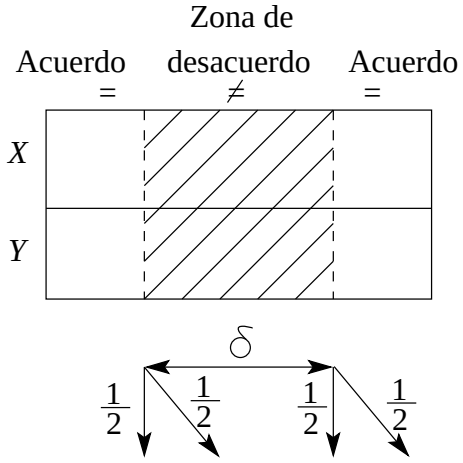

Figura 5. Zona de desacuerdo

Se tiene: $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t) - q - r + s$. Por lo tanto el evento $\delta(X_{t+1}, Y_{t+1}) = \delta(X_t, Y_t)$ corresponde a todos los casos donde $q + r = s$. Si $q < p$

tal que un evento puede ocurrir (probabilidad 0). De otra manera, su probabilidad es:

$$\begin{aligned} \frac{1}{4^p}\sum_{r=0}^{p-q}\binom{p}{r}\binom{p}{q+r} &= \frac{1}{4^p}\sum_{r=0}^{p-q}\binom{p}{r}\binom{p}{q+r} \\ &\leq \frac{1}{4^p}\binom{2p}{p-q} \\ &\leq \frac{1}{2^{2p}}\binom{2p}{p} \\ &\leq \frac{1}{2} \end{aligned} \tag{18}$$

Por lo tanto $Pr(\delta(X_{t+1}, Y_{t+1}) \neq \delta(X_t, Y_t)) \geq \frac{1}{2}$.

□

Ya que $\delta(X_t, Y_t)$ toma valores en $\{0, \ldots, N\}$, se establecen las cotas; el siguiente corolario se toma de Fribourg *et al.* (2006).

Corolario 1. Para N impar, el algoritmo de Herman es auto-estabilizante con respecto al conjunto de configuraciones $\mathcal{L}$ con un solo token. Además:

1. El tiempo esperado de alcance satisface:

$$\mathbf{H}_{\mathcal{L}} \leq 2N^2$$

2. El tiempo de absorción$-\varepsilon$ satisface:

$$\theta_{\mathcal{L}}(\varepsilon) \leq 2eN^2 \left\lceil ln(\frac{1}{\varepsilon}) \right\rceil$$

III.3.3 Ejemplo: Aplicación al Algoritmo del Dilema del Prisionero Iterado

Se considera el algoritmo del Dilema del Prisionero Iterado (Dyer *et al.*, 2002). La topología es un grafo cíclico (anillo) de N vértices y un calendarizador central aleatorio, es decir, se escoge un vértice al azar para que realice alguna acción. El conjunto de estados es $Q = \{-, +\}$. En cada paso un vértice $i,\ 1 \leq i \leq N$, se escoge uniformemente al azar, y los valores $x(i)$ y $x(i+1)$ se cambian a $x'(i)$ y $x'(i+1)$, respectivamente como sigue:

- Si $x(i) = x(i+1)$, entonces $x'(i) = x'(i+1) = +$,
- Si $x(i) \neq x(i+1)$, entonces $x'(i) = x'(i+1) = -$.

(Cuando $i = N,\ (i+1)$ es el nodo 1).

La siguiente demostración se toma de Fribourg *et al.* (2006). Para el algoritmo del Dilema del Prisionero, existe un subconjunto U de $\Omega \times \Omega$ tomando valores en $\{0, \ldots, 11N\}$ y un acoplamiento definido en U tal que, para todo $(X_t, Y_t) \in U$:

$$E\left[\delta\left(X_{t+1}, Y_{t+1}\right)\right] \leq \left(1 - \frac{1}{18N}\right) \delta\left(X_t, Y_t\right) \tag{19}$$

Demostración de 19:

Pares adyacentes. Un par (X, Y) pertenece a U si y solo si X y Y coinciden en todas las posiciones excepto en k posiciones contiguas, donde $(1 \leq k \leq 5)$. Sea $\delta(X, Y) = a_k$, donde a_k es una constante positiva que se determina después. Por convención, sea $a_0 = 0$. La función δ en U se extiende a todo el espacio $\Omega \times \Omega$ según lo que se explica

más adelante. Considere $(X, Y) \in \Omega \times \Omega$ tal que X y Y difieren solamente en ℓ posiciones contiguas.

Se tiene: $\ell = 5m + r$ para alguna $m \geq 0$ y $0 \leq r \leq 4$. La función δ se define como: $\delta(X, Y) = \sum_{p=1}^{n} m_p a_5 + a_{r_p}$. Suponga que X y Y no concuerdan en n zonas separadas de posiciones contiguas W_p $(1 \leq p \leq n)$. Sean m_p y r_p el cociente y el resto de la longitud de W_p dividida por 5 ($|W_p| = 5m_p + r_p$ *con* $0 \leq r_p \leq 4$). Se demuestra después que, para valores apropiados de a_k $(1 \leq k \leq 5)$, la función δ es una métrica que satisface las condiciones requeridas por la desigualdad 19.

Acoplamiento. El acoplamiento $(X, Y) \mapsto (X', Y')$ se define tal que, en cada paso, la posición escogida uniformemente al azar coincide para X y Y.

Demostración de $E[\delta(X', Y')] \leq \beta\delta(X, Y)$: Considere un vector $(X_t, Y_t) \in U$ con k posiciones contiguas de desacuerdo. Sea i la primera posición de desacuerdo. El vector es de la forma:

$$\begin{pmatrix} X_t \\ Y_t \end{pmatrix} = \begin{pmatrix} \gamma_1 \cdots \gamma_{i-2}\gamma_{i-1}\gamma_i \cdots \gamma_{i+k-1}\gamma_{i+k} \cdots \gamma_N \\ \gamma_1 \cdots \gamma_{i-2}\gamma_{i-1}\neg\gamma_i \cdots \gamma_{i+k-1}\gamma_{i+k} \cdots \gamma_N \end{pmatrix}$$

donde γ_ℓ están en $\{-, +\}$. Suponga que la posición seleccionada j es tal que $1 \leq j \leq i-2$ o $i+k \leq j \leq N$. Entonces $X_t(j) = Y_t(j)$ y $X_t(j+1) = Y_t(j+1)$, por lo que $X_{t+1}(j) = Y_{t+1}(j)$ y $X_{t+1}(j+1) = Y_{t+1}(j+1)$ y la zona de desacuerdo no se modifica. Suponga ahora que la posición seleccionada j es igual a $i-1$. Por lo tanto,

después de un paso, se tiene:

$$\begin{pmatrix} X_{t+1} \\ Y_{t+1} \end{pmatrix} = \begin{pmatrix} \gamma_1 \cdots \gamma_{i-2}\gamma'_{i-1}\gamma'_i \cdots \gamma_{i+k-1}\gamma_{i+k} \cdots \gamma_N \\ \gamma_1 \cdots \gamma_{i-2}\neg\gamma'_{i-1}\neg\gamma'_i \cdots \gamma_{i+k-1}\gamma_{i+k} \cdots \gamma_N \end{pmatrix}$$

donde $\gamma'_{i-1} = \gamma'_i = +$ si $\gamma_{i-1} = \gamma_i$ y $\gamma'_{i-1} = \gamma'_i = -$ de otra manera. Esto significa que la zona de desacuerdo ha progresado en la posición a la izquierda. Un caso simétrico existe para $j = i + k - 1$. Se dice que j es una "frontera externa". Un caso simple de análisis demuestra que, para valores apropiados de $a_k\,(1 \leq k \leq 5)$ existe β con $\beta \leq 1 - \frac{1}{18N}$ tal que $E\left[\delta\left(X', Y'\right)\right] \leq \beta\delta\left(X, Y\right)$. Además, para estos valores de a_k, el valor máximo B de δ sobre $\Omega \times \Omega$ es tal que $B \leq 11N$.

□

Por lo tanto por el Teorema 4 (ver página 35) se obtiene:

Para el algoritmo del Dilema del Prisionero Iterado A, se tiene:

1. A es auto-estabilizante con respecto al conjunto $\mathcal{L} = \{(+)^N\}$.
2. El tiempo esperado de alcance satisface: $H_{\mathcal{L}} \leq 198N^2$.
3. El tiempo de absorción-ε satisface: $\Theta_{\mathcal{L}(\varepsilon)} \leq 18Nln\left(\frac{11N}{\varepsilon}\right)$.

En este capítulo se explica con detalle la técnica de acoplamiento y se utilizan algunos ejemplos para clarificar las definiciones que aquí se mencionan. En el siguiente capítulo se hace un cambio de topología, en los dos algoritmos auto-estabilizantes (que se mostraron aquí como ejemplos) que con la técnica de acoplamiento ya se demostró su convergencia.

Capítulo IV

Transformación de Topología

En este capítulo se discute cómo se puede cambiar la topología simétrica de dos algoritmos auto-estabilizantes de Herman (1990); Dyer *et al.* (2002) con la finalidad de utilizar estos algoritmos en una topología asimétrica y ampliar el uso de éstos para resolver un mayor número de problemas. Ambos problemas se describen brevemente en el Capítulo III. Aquí por claridad, se da una descripción más profunda.

IV.1 Algoritmo de Herman

En (Herman, 1990) se describe un algoritmo aleatorio (un algoritmo aleatorio es aquel que toma de decisiones basadas en números generados al azar) que hace circular de manera justa un token, que denota a un líder, en un anillo con un número impar de nodos idénticos; los nodos operan de manera síncrona (todos los nodos actúan a la vez) y la comunicación se lleva a cabo en un sentido sobre el anillo. La función normal del algoritmo es circular un token entre los nodos del anillo. Si el estado inicial del anillo es anormal, es decir, el número de tokens difiere de uno, entonces la ejecución del algoritmo resulta en la convergencia a un estado normal con un token.

Sea n el número de nodos en el anillo. Se requiere que n sea impar para obtener un estado de un token. Si n es par entonces el algoritmo llega a un estado sin tokens. El estado local de cada nodo i lo define una variable binaria x_i. El estado del anillo

se puede representar por un vector de n bits. Suponga que x es tal vector; se agregan subíndices a x para hacer referencia a un elemento individual del vector. En la Figura 6 se muestra un ejemplo de una configuración inicial del sistema.

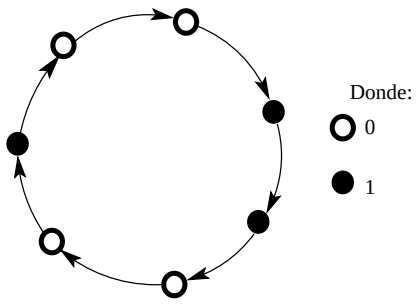

Figura 6. Configuración inicial de algoritmo de Herman

Se define a un token como un par consecutivo (x_{i-1}, x_i) con valores iguales en el anillo. La Figura 6 es una configuración con tres tokens. El conjunto de estados de cada nodo (o estado local) es $Q = \{0, 1\}$. En cada paso el estado de cada nodo $x(i)$ $(1 \leq i \leq N)$ se cambia a $x'(i)$ como sigue:

Si $x(i) \neq x(i-1)$ entonces $x'(i) = \neg x(i)$.

Si $x(i) = x(i-1)$ entonces $x'(i) = 0$ con probabilidad $\frac{1}{2}$ o 1 con probabilidad $\frac{1}{2}$.

Esto quiere decir, en cada iteración cada nodo i lee el estado de su vecino anterior $i-1$ y compara el estado de éste con su propio estado; si los dos estados son iguales, entonces i toma una decisión basada en la generación de un bit aleatorio y con igual probabilidad el estado del nodo puede tomar valor de 1 o 0. Si los valores de los estados son diferentes, entonces el estado del nodo i toma el valor contrario al que tiene en ese instante. En la Figura 7 se muestra un ejemplo de una configuración con un token.

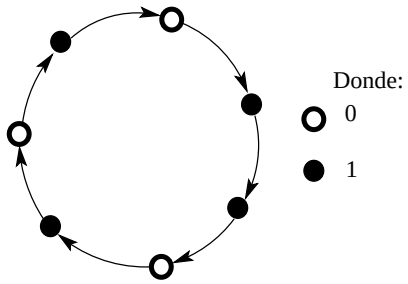

Figura 7. Configuración con un solo token

Como se explica anteriormente, este algoritmo se diseñó para ejecutarse sobre una topología simétrica (anillo). A continuación se propone un mecanismo para usar este algoritmo sobre una topología asimétrica, un ejemplo de ésta es un árbol.

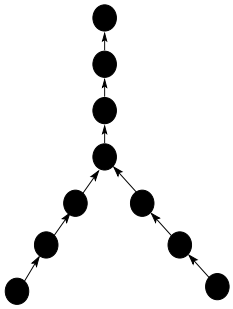

Figura 8. Ejemplo de topología de árbol

Para utilizar el algoritmo de Herman en un árbol como el de la Figura 8, en esta investigación se propone que la topología del árbol original simule a un anillo lógico virtual donde el número de nodos es impar, ya que es una restricción del algoritmo. Para hacer ésto, se hace una representación lógica de un anillo con los nodos del árbol y las líneas de comunicación que existen entre éstos de la siguiente manera. Note que el anillo lógico virtual debe tener un número impar de nodos; sin embargo, el número de nodos en el árbol no tiene restricción, puede ser par o impar.

Suponga un árbol enraizado $T = (V_t, E_t)$ (un árbol enraizado es aquel donde cada nodo sabe quién es su nodo padre), donde V_t representa el número de nodos del árbol de la topología original y E_t representa las aristas (líneas de comunicación) del árbol original. Suponga que $C = (V_c, E_c)$ es el anillo lógico virtual que se obtiene a partir del árbol de la topología original; V_c representa a los nodos en el anillo lógico virtual y es un número impar y E_c representa a las aristas en el anillo lógico virtual. Para convertir el número de nodos del árbol a un número de nodos impar se reemplaza a todo $v \in V_t$ por un nodo en V_c, es decir, se coloca un nodo (negro) en el anillo lógico virtual por cada nodo en el árbol original, como lo muestra la Figura 9.

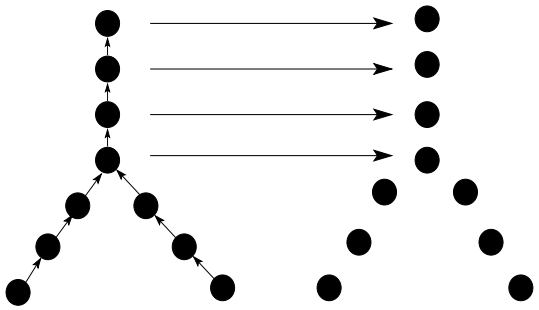

Figura 9. Nodos negros que representan nodos del árbol

Ademas, para cada $e \in E_t$ se agrega un nodo (blanco) en V_c, como lo muestra la Figura 10. Con esta conversión se obtiene un número de nodos impar en el anillo lógico virtual porque se incluyen $|V_t|$ nodos más $|V_t| - 1$ aristas convertidas en nodos:

$$|V_c| = |V_t| + |V_t| - 1 \tag{20}$$

la Ecuación 20 deja como resultado $|V_c| = 2|V_t| - 1$.

Para simplificar la notación, se representa cada nodo del árbol original como nodos

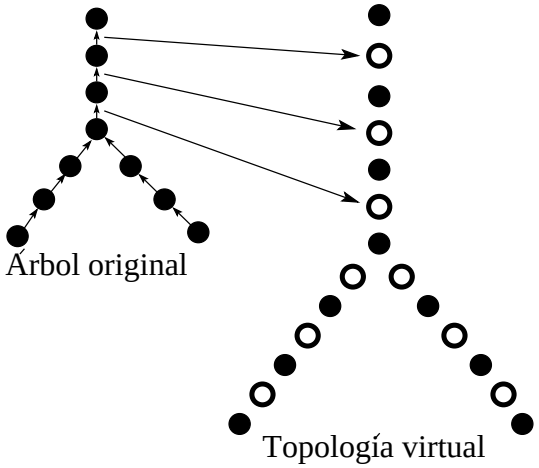

Figura 10. Nodos blancos que representan aristas del árbol

de color negro en el anillo lógico virtual y cada arista del árbol como nodos blancos en el anillo lógico virtual.

Como se menciona anteriormente en este trabajo de investigación se propone que la topología del árbol original simule a un anillo lógico virtual. En donde al tener el número de nodos en el anillo lógico virtual se colocan dos aristas entre cada par de nodos y se hace una búsqueda en profundidad desde el nodo que representa a la raíz en el anillo lógico virtual y se crean tres reglas para la asignación del sentido del ciclo, que a continuación se mencionan.

1. Para un nodo negro interno i (aquel que tiene al menos un hijo).

El sucesor del nodo i, denotado por $suc(i)$, es el descendiente más izquierdo que sea negro. En la Figura 11 se muestra un ejemplo de la primera regla de orientación del ciclo.

2. Para un nodo negro hoja k diferente de la raíz (aquel nodo que no tiene hijos).

El sucesor de k, denotado por $suc(k)$, es el nodo j tal que j es el primer nodo blanco

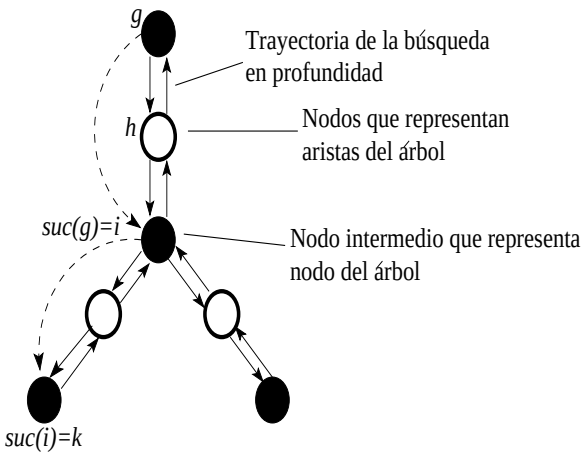

Figura 11. Ejemplo que ilustra el sucesor de un nodo negro interno

después de k siguiendo la trayectoria generada por la búsqueda en profundidad. En la Figura 12 se muestra un ejemplo de esta regla.

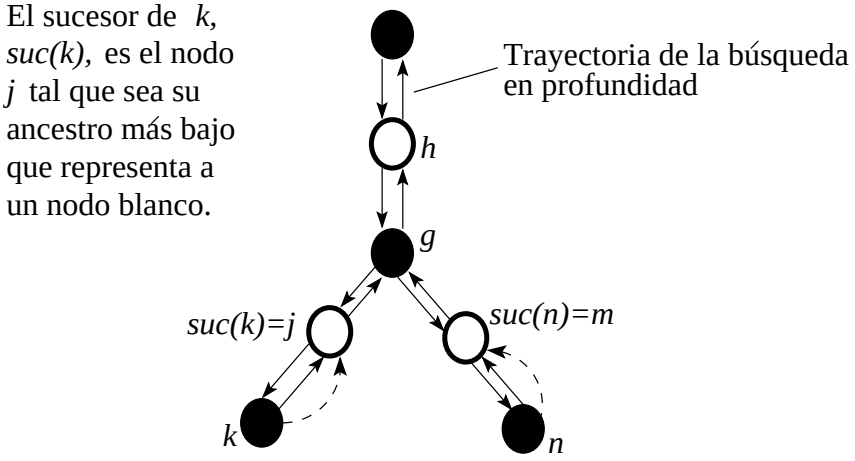

Figura 12. Ejemplo que ilustra al sucesor de un nodo hoja diferente de la raíz.

3. Para un nodo blanco i, sea j el siguiente nodo blanco en la trayectoria de una búsqueda en profundidad. El sucesor de i, $suc(i)$ es:

- Si j es un ancestro de i, entonces $suc(i) = j$.
- Si j no es un ancestro de i, entonces $suc(i) = k$, donde k es el siguiente nodo

negro después de j al seguir la trayectoria de búsqueda en profundidad.

- Si el siguiente nodo de i es la raíz, entonces $suc(i) = raíz$.

Un ejemplo de la regla 3 se observa en la Figura 13. Conforme a las reglas anteriores se determina el sentido del anillo y se obtiene el anillo lógico virtual, donde el número de aristas en el conjunto E_c es:

$$|E_c| = |V_c| = (2|V_t| - 1) \tag{21}$$

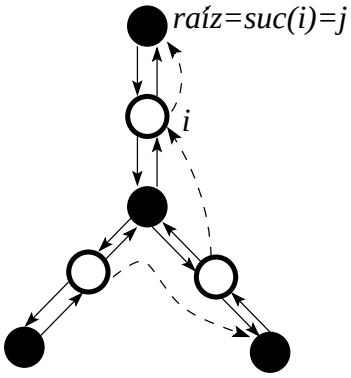

Figura 13. Ejemplo que ilustra el sucesor de un nodo blanco.

Ahora se tiene un número de nodos impar en un anillo lógico virtual donde se puede ejecutar el algoritmo de Herman en este anillo lógico virtual (ver Figura 14). Hay que recordar que el algoritmo de Herman se ejecuta de manera síncrona, es decir, todos los nodos a la vez realizan la comparación y cambio de su estado, pero al hacer el cambio de topología y definir el sentido del anillo lógico virtual se observa que el algoritmo tarda más en ejecutarse. Lo anterior debido a que cuando se hacen las lecturas en el anillo lógico virtual algunos nodos blancos tardan más en leer a su sucesor, como lo muestran las Figuras 12 y 13.

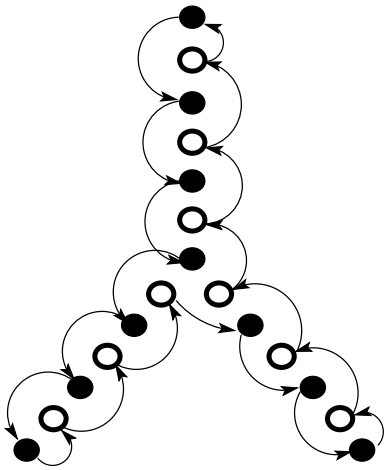

Figura 14. Anillo lógico virtual

El cambio de topología tiene un costo adicional de 1 paso por cada lectura realizada por el algoritmo de Herman, ya que los sucesores de algunos nodos blancos se encuentran a 2 nodos de distancia de sus sucesores. La ventaja del cambio que se hizo es que ya se puede ejecutar el algoritmo de Herman en un árbol con cualquier número de nodos ya sea par o impar. Al ejecutarse el algoritmo de Herman en el anillo lógico virtual existe otro costo adicional por el incremento de nodos virtuales, ya que el trabajo que realizan los nodos blancos en realidad es trabajo simulado que realiza un nodo en el árbol, para determinar cuál nodo negro realiza el trabajo de un nodo blanco se toma en cuenta el grado mínimo entre un par de nodos negros, *i.e.* entre cada par de nodos vecinos del árbol aquel nodo que tenga el menor grado realiza el trabajo del nodo blanco. Cuando el grado del nodo, en un par de nodos negros, es igual se lanza una moneda para determinar quien realiza el trabajo (ver Figura 15). Por lo tanto, el costo adicional al ejecutar el algorito de Herman depende del máximo grado del conjunto de los grados mínimos entre cada par de nodos.

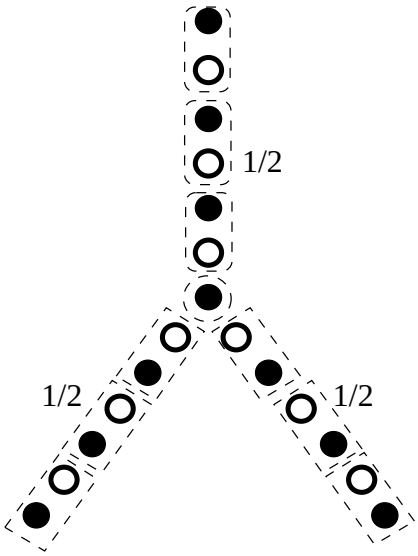

Figura 15. Trabajo de nodos negros

IV.2 Algoritmo del Dilema del Prisionero Iterado

El Dilema del Prisionero Iterado es un problema de teoría de juegos (área de la matemática aplicada que utiliza modelos para estudiar interacciones en estructuras formalizadas de incentivos y llevar a cabo procesos de decisión). El problema del Dilema del Prisionero Iterado se enuncia de la siguiente manera:

La policía arresta a dos sospechosos (Pedro y Juan). No hay pruebas suficientes para condenarlos y, tras haberlos separado, los visita a cada uno y les ofrece el mismo trato. Si Pedro confiesa y Juan no, Juan será condenado a la pena total, cinco años, y Pedro será liberado. Si Pedro calla y Juan confiesa, Pedro recibirá esa pena y será Juan quien salga libre. Si ambos confiesan, ambos serán condenados a tres años. Si ambos lo niegan, todo lo que podrán hacer será encerrarlos durante un año por un cargo menor. En la Tabla III se resume el Dilema del Prisionero Iterado como:

Tabla III. Dilema Prisionero Iterado

	Pedro confiesa	Pedro lo niega
Juan confiesa	Ambos condenados a 3 años	Pedro condenado a 5 años
Juan lo niega	Juan condenado a 5 años	Ambos condenados 1 año

El objetivo del problema es el de que todos los prisioneros colaboren entre sí, pero cada prisionero de modo independiente trata de aumentar al máximo su propia ventaja sin importarle el resultado del otro jugador. Las técnicas de análisis de la teoría de juegos estándar, por ejemplo determinar el equilibrio de Nash, pueden llevar a cada prisionero a decidir traicionar al otro, pero curiosamente ambos prisioneros obtendrían un resultado mejor si colaborasen. Desafortunadamente (para los prisioneros), cada prisionero está incentivado individualmente para defraudar al otro, incluso tras prometerle colaborar. Éste es el punto clave del dilema.

En (Dyer *et al.*, 2002) se estudian los juegos de cooperación y uno de estos juegos es una versión restringida del algoritmo del Dilema del Prisionero Iterado. Se tiene una población de $n \geq 4$ nodos situados en los vértices de un grafo conectado $G = (V, E)$. Cada nodo tiene un estado $X(i) \in \{-1, 1\}$ donde el *cooperar* se denota como 1 y el *no cooperar* como -1. En cada etapa se escoge un par $\{i, j\} \in E$ uniformemente al azar y se reemplaza el signo de los nodos $X(i)$ y $X(j)$ por el producto de los signos de $X(i)$ y $X(j)$. El proceso tiene una memoria de un solo paso porque cada nodo i recuerda su estado actual $X(i)$ pero no recuerda ningún estado que tenía antes. En la Figuras 16 y 17 se muestra un estado inicial del sistema y un estado absorbente del sistema (un estado s_i de una cadena de Markov se dice absorbente si es imposible abandonar el estado s_i) que se puede ver como un estado final, respectivamente.

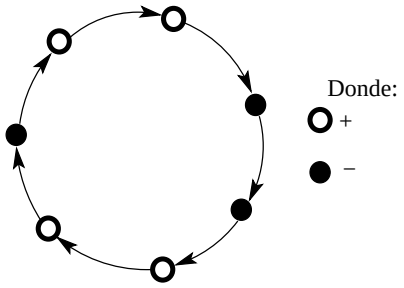

Figura 16. Configuración inicial en algoritmo del Dilema del Prisionero Iterado

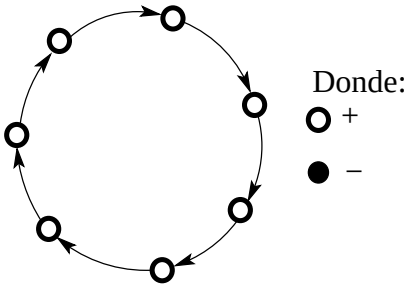

Figura 17. Configuración final en algoritmo del Dilema del Prisionero Iterado

El estado X^* con $X^*(i) = 1$ para toda $v \in V$ es un estado absorbente y existe una secuencia de movimientos que pueden transformar a X a X^*, para cada $X \in \{1, -1\}^V$.

En (Fribourg *et al.*, 2006) se presenta un algoritmo auto-estabilizante y se ofrece una cota superior en el tiempo esperado de alcance. Se define el sistema como un conjunto de N nodos en un anillo y un calendarizador central aleatorio (es decir, se escoge a un nodo a la vez para que lea la información de su vecino y tome acciones con esa información). El conjunto de estados es $Q = \{-, +\}$. En cada paso, un nodo $i(1 \leq i \leq N)$ se escoge uniformemente al azar y los valores $x(i)$ y $x(i+1)$ se cambian a $x'(i)$ y $x'(i+1)$, respectivamente, como sigue:

Si $x(i) = x(i+1)$, entonces $x'(i) = x'(i+1) = +$

Si $x(i) = \neg x(i+1)$, entonces $x'(i) = x'(i+1) = -$

El tiempo esperado de alcance que se obtiene en (Fribourg *et al.*, 2006) es de

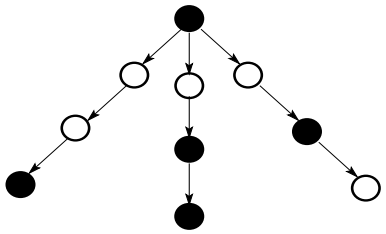

Figura 18. Topología de árbol

$H_{\mathcal{L}} \leq 198N^2$.

Este problema también se ejecuta en una topología simétrica, pero pensando en problemas con topología asimétrica por ejemplo un árbol, se considera realizar un cambio en la topología para ampliar el uso de este algoritmo. En la Figura 18 se muestra un ejemplo de una topología asimétrica.

Para utilizar el algoritmo del Dilema del Prisionero Iterado (Dyer *et al.*, 2002) en este trabajo de investigación se propone cambiar la topología de árbol a un anillo lógico virtual. La única restricción que existe en este algoritmo es que el número de nodos sea mayor a 4, ya que así se garantiza que se llega al estado convergente, el cual es el estado absorbente.

Para convertir la topología del árbol original a un anillo lógico virtual se hace un cambio de los nodos de la topología original y de las aristas que representan las líneas de comunicación entre los nodos.

Suponga un árbol de entrada enraizado $T = (V_t, E_t)$, donde V_t es el número total de nodos del árbol y E_t las aristas que representan las líneas de comunicación entre

los nodos. Sea $C = (V_c, E_c)$ el anillo lógico virtual que se forma a partir del árbol de entrada, donde V_c es el número total de nodos que se obtienen de la transformación de la topología, y E_c es el número de aristas que conforman al anillo lógico virtual.

Al hacer el cambio de topología de un árbol a un anillo lógico virtual donde existe una arista del árbol se coloca un par de aristas en el anillo lógico virtual y se le asigna un sentido a las aristas, el sentido de la arista izquierda es hacia abajo y el sentido de la arista derecha es hacia arriba, como lo muestra la Figura 19.

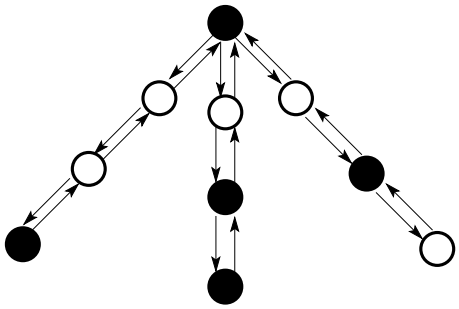

Figura 19. Sustitución de aristas en el árbol

A continuación se realiza un búsqueda en profundidad por medio de un tour de Euler (recorrido del grafo pasando por las aristas solo una vez), como lo muestra la Figura 20. Se "eliminan" los nodos del árbol y cada vez que se une una arista con otra, es decir, cuando el final de una arista toca el inicio de otra arista se coloca un nodo y así se obtiene un ciclo, como lo muestran las Figuras 21 y 22.

Ya se tiene un anillo lógico virtual a partir del árbol de entrada, en el cual se puede implementar el algoritmo del Dilema del Prisionero Iterado. Al cambiar la topología de un árbol de entrada a un anillo lógico virtual se reemplaza cada nodo v por $|d(v)|$

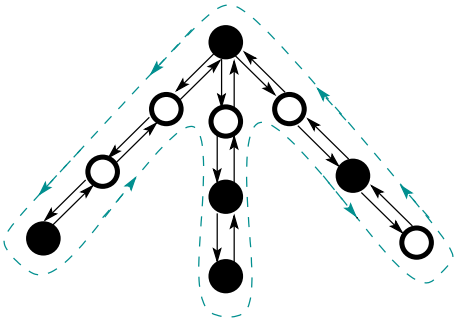

Figura 20. Tour de Euler

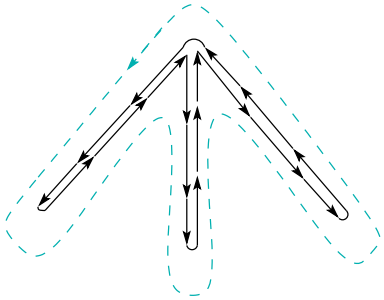

Figura 21. Unión de aristas

nodos, donde $|d(v)|$ es el grado del nodo, por lo que el número total de nodos en el anillo lógico virtual es $V_c = 2(|V_t| - 1)$. El número de aristas es también el mismo, $E_c = 2(|V_t| - 1)$.

Como se puede observar en la Figura 22, el número de nodos y aristas se incrementa prácticamente al doble con respecto a la topología original, por lo tanto hay un incremento en el trabajo que realizan los nodos físicos de la topología original. Así que el trabajo realizado por cada nodo físico v es realmente $|d(v)|$.

Nota: Este problema es un problema de tamaño más grande y el análisis de este problema se puede ver como una cota superior para el análisis del original.

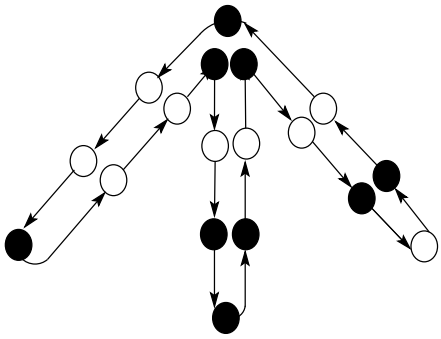

Figura 22. Colocación de nodos en el anillo

Al igual que cuando se hizo el cambio de topología para implementar el algoritmo de Herman existe un sobrecosto, al hacer el cambio de topología para implementar el algoritmo del Dilema del Prisionero Iterado también existe un sobrecosto por el cambio de topología de un árbol a un anillo, este sobrecosto se debe al incremento en el número de nodos y aristas.

Con esto se termina la discusión de cómo se puede cambiar la topología de un algoritmo que está diseñado para correr en una configuración cíclica para que pueda implementarse en otras topologías. También se observó que un cambio de topología depende de las restricciones del algoritmo para llegar a su estado estable.

Al cambiar la topología se crea un incremento en el número de nodos y/o de aristas y es por eso que puede existir un incremento en el trabajo que realiza cada nodo en la topología original y por lo tanto un sobrecosto constante en la ejecución del algoritmo. En el siguiente capítulo se utiliza la técnica de acoplamiento en el Protocolo de Espacio Mínimo de Dolev *et al.* (1995).

Capítulo V

Técnica de Acoplamiento en el Protocolo de Espacio Mínimo

En este capítulo se utiliza la técnica de acoplamiento en el protocolo de espacio mínimo propuesto en (Dolev *et al.*, 1995). Este algoritmo se utiliza para la elección de un líder en un sistema distribuido uniforme.

V.1 Descripción del Protocolo de Espacio Mínimo

En (Dolev *et al.*, 1995) se discute el protocolo de espacio mínimo, el cual es un algoritmo de elección de líder en un sistema de nodos uniformes.

Un sistema distribuido uniforme consiste de n nodos. Los nodos son anónimos (no tienen identificadores) y son iguales con respecto a los demás nodos (no existe una característica que haga que se diferencien entre ellos). Cada procesador se comunica con otros nodos usando un registro de solo escritura y un registro de multi-lectura, es decir cada procesador puede modificar solamente su registro y puede leer el registro de los demás pero no puede modificarlo.

Se puede ver a cada procesador como un CPU cuyo programa se compone de pasos atómicos. Un paso atómico de un procesador consiste de un cálculo interno seguido por una acción de terminación (acción que termina el paso atómico).

El algoritmo propuesto en (Dolev *et al.*, 1995) emplea el esquema del juego de calendarizador-suerte (*sl-game*), para analizar protocolos aleatorios distribuidos, usando información completa de dos jugadores de un juego. Este juego se denota como una tripleta $G = (PR, I, F)$ donde PR es un protocolo, I es un conjunto de configuraciones iniciales y F un conjunto de configuraciones finales. En el contexto de auto-estabilización I es el conjunto de todas las posibles configuraciones y F es el conjunto de configuraciones que son legales.

Los jugadores de G se llaman calendarizador o adversario (*scheduler*, en inglés) y suerte (*luck*, en inglés), cuyas metas son opuestas. La meta del calendarizador es prevenir que PR alcance una configuración en F; mientras que la de suerte es ayudar a que PR alcance esa configuración.

Los estados de G son configuraciones del sistema del protocolo PR (similarmente a cualquier algoritmo auto-estabilizante); cada turno de G inicia en alguna configuración c y en él, el calendarizador escoge el siguiente procesador para realizar un paso atómico (un lanzamiento de moneda se considera como una acción interna y no termina el paso atómico que la contiene). Si durante este paso, el procesador seleccionado lanza una moneda, entonces suerte puede intervenir y forzar el resultado del lanzamiento. Cuando el paso atómico se completa, se alcanza una nueva configuración c' del sistema, desde la cual se inicia un nuevo turno. Cada ejecución de G corresponde naturalmente a una ejecución del protocolo PR. La ejecución de G termina cuando el sistema alcanza una configuración en F.

En el protocolo para elección de líder (Dolev *et al.*, 1995), cada procesador se comu-

nica con los otros nodos usando un registro binario de solo escritura y de multi-lectura llamado *leader* donde $leader_i$ denota el registro *leader* del procesador P_i.

El sistema inicia en cualquier posible combinación de valores binarios de los registros *leader*, el protocolo eventualmente fija todos los registros *leader* en 0 excepto uno. El único procesador cuyo valor de *leader* es 1 se denomina líder. En el protocolo, cada procesador repetidamente lee todos los registros *leader*, de los otros nodos.

Con base en el resultado de las lecturas y en el valor de su registro *leader,* un procesador puede actualizar su propio registro *leader* con el resultado del lanzamiento de una moneda booleana. El pseudo código de este algoritmo se presenta en la siguiente sección.

El protocolo se estabiliza con a lo más $2^{O(n)}$ número esperado de pasos de acuerdo al esquema calendarizador-suerte (Dolev *et al.*, 1995). Ésta es una cota bastante holgada para este problema, su utilidad en realidad es para demostrar la convergencia y no para encontrar el tiempo de convergencia.

V.2 Aplicando la Técnica de Acoplamiento en el Protocolo de Espacio Mínimo

El protocolo inicia en una configuración arbitraria donde cada procesador tiene un registro *leader* y un vector con copias de los registros *leader* de los demás nodos. En cada iteración un procesador i lee el registro *leader* de un procesador j, el procesador i cambia en su vector interno el valor del registro *leader* de j y determina si él mismo

es candidato a líder o no.

Pseudo-código del Protocolo de Espacio Mínimo

do forever

for$j := 1$ to n $(j \neq i)$ **do** $leader_i[j] :=$ **read**$(leader_j)$;

if $(leader_i = 0$ **and**$\{\forall j \neq i, |leader_i[j] = 0\})$ **or**

$(leader_i = 1$ **and**$\{\exists j \neq i, |leader_i[j] = 1\})$ **then**

write $leader_i := random(\{0, 1\})$;

end

Note cómo en caso de empate (si $leader_i = 0$ y todos los demás también son ceros, o si $leader_i = 1$ y existe un 1 en algún otro procesador), el procesador i tiene la opción de mantenerse en la pelea por el liderato (si su moneda sale 1) o salirse del concurso (si su moneda sale 0).

Para garantizar que todos los nodos lean a todos los demás nodos, el procesador i lee al procesador j, en la siguiente iteración el procesador i lee al procesador $j+1$ $(mod\ n)$, y así sucesivamente hasta completar un ciclo de lecturas de todos los nodos.

Se considera una configuración legal cuando existe un líder y todos los demás nodos saben quién es el líder. El principal cambio que se le hizo a este protocolo en este trabajo de investigación para poder implementarle la técnica de acoplamiento, es que en el protocolo de espacio mínimo original el calendarizador escogía a un procesador y éste efectuaba una acción atómica ya sea leer o modificar su propio registro, pero en esta propuesta se escoge al azar un procesador para que realice dos acciones atómicas,

leer y modificar su propio registro (si es el caso).

Al utilizar el método de acoplamiento para demostrar la convergencia y establecer una cota en el tiempo de convergencia de este protocolo, se inicia con dos configuraciones arbitrarias X y Y y se define δ que es una función de distancia entre estas dos configuraciones. En esta ocasión δ es la distancia Hamming, la cual cuenta el número de elementos en donde las dos configuraciones difieren.

Como el protocolo de espacio mínimo tiene más de un conjunto ergódico (conjunto global estable), se usa el Teorema 5 descrito en (Fribourg *et al.*, 2006).

Teorema 5. Dada una cadena de Markov A y dos conjuntos cerrados disjuntos $\mathcal{L}_0$ y $\mathcal{L}_1$ ($\mathcal{L}$ es $\mathcal{L} = \mathcal{L}_0 \uplus \mathcal{L}_1$, donde cada $\mathcal{L}_i$ es un conjunto ergódico cerrado disjunto y $\uplus$ representa la unión de esos conjuntos), suponga que existe un acoplamiento (X_t, Y_t) y una función de distancia δ' sobre el conjunto de todas las configuraciones que toma valores del conjunto $\{0, 1, \ldots, B\}$ tal que:

$$\delta'(X_t, Y_t) = 0 \text{ si y solo si } X_t = Y_t, \tag{22}$$

$$\delta'(X_t, Y_t) = B \Rightarrow \delta'(X_{t+1}, Y_{t+1}) = B, \tag{23}$$

$$(\delta'(X_t, Y_t) = B \wedge Y_t \in \mathcal{L}_0) \Rightarrow X_t \in \mathcal{L}_1, \tag{24}$$

Existe un $\alpha > 0$ tal que, para todo (X_t, Y_t) con $0 < \delta'(X_t, Y_t) < B$:

$$E[\delta'(X_{t+1}, Y_{t+1})] \leq \delta'(X_t, Y_t)$$
$$\wedge \Pr(\delta'(X_{t+1}, Y_{t+1}) \neq \delta'(X_t, Y_t)) \geq \alpha \tag{25}$$

Entonces el protocolo es auto-estabilizante con respecto a $\mathcal{L}$. Además:

1. El tiempo esperado de alcance satisface: $\mathbf{H}_{\mathcal{L}} \leq B^2/\alpha$.

2. El tiempo de absorción-ε satisface:

$\Theta_{\mathcal{L}}(\varepsilon) \leq \left\lceil e\frac{B^2}{\alpha} \right\rceil \left\lceil \ln\left(\frac{1}{\varepsilon}\right) \right\rceil$.

Nota: El Teorema 5 se puede extender para cuando existen más de dos conjuntos ergódicos.

A continuación, el autor de este trabajo de tesis demuestra que se cumplen las 4 condiciones del Teorema 5 en el protocolo de espacio mínimo de Dolev *et al.* (1995) para así demostrar que el algoritmo converge y establecer una cota superior en el tiempo de convergencia.

Primero, se propone la siguiente función de distancia δ' la cual es:

$$\delta' = \begin{cases} \delta & \text{Si ambos } X_t \text{ y } Y_t \text{ no están en conjuntos ergódicos} \\ \delta \times \left(\frac{N^2+1}{2N}\right) & \text{de otra manera} \end{cases}$$

donde δ es la distancia de Hamming y N es el número de nodos en el grafo.

Note que la distancia δ' puede tomar valores del conjunto $(0, 1, 2, \ldots, N^2+1)$. El valor de N^2+1 se obtiene cuando ambos X_t y Y_t están en distintos conjuntos ergódicos.

Demostración de 22.

$$\delta'(X_t, Y_t) = 0 \text{ si y solo si } X_t = Y_t$$

Sean $\{x_{1,1}, \ldots, x_{1,n}, \ldots, x_{n,1}, \ldots x_{n,n}\}$ y $\{y_{1,1}, \ldots, y_{1,n}, \ldots, y_{n,1}, \ldots y_{n,n}\}$ los vectores de X_t y Y_t, si cada elemento del vector de X_t es igual a su correspondiente elemento de Y_t entonces la distancia es cero.

En caso contrario si $\delta' = 0$, de acuerdo a la definición de distancia forzosamente cada elemento de X_t y Y_t deben ser iguales y por consiguiente $X_t = Y_t$.

□

Demostración de 23.

$$\delta'(X_t, Y_t) = B \Rightarrow \delta'(X_{t+1}, Y_{t+1}) = B$$

Por la definición de la distancia δ', la distancia es $\delta' = B$ sólo cuando ambos X_t y Y_t están en conjuntos ergódicos diferentes. Como los conjuntos ergódicos son cerrados, entonces $\delta'(X_{t+1}, Y_{t+1}) = B$.

□

Demostración de 24.

$$(\delta'(X_t, Y_t) = B \wedge Y_t \in \mathcal{L}_0) \Rightarrow X_t \in \mathcal{L}_1,$$

El conjunto de valores que puede tomar la distancia Hamming δ es $\{0, \ldots, N^2\}$. Si

la distancia δ' es N^2+1, por la definición de δ', necesariamente X_t y Y_t deben estar en conjuntos ergódicos distintos, entonces si $Y_t \in \mathcal{L}_0$ quiere decir que $X_t \in \mathcal{L}_1$.

□

Demostración de 25.

La demostración se hace por inducción sobre el parámetro n (número de nodos).

Caso base:

El caso base se considera cuando $n = 2$.

Para demostrar 25 con $n = 2$, los dos nodos tienen sus registros $leader_i(j)$ $\forall j,\ 1 \leq j \leq 2$ (para efectos de simplificar esta demostración esos registros se cambian por $x_{1,1}$ $x_{1,2}$ y $x_{2,1}$ $x_{2,2}$). Simultáneamente se supone la existencia de otro sistema Y cuyos registros son $y_{1,1}$ $y_{1,2}$ y $y_{2,1}$ $y_{2,2}$.

En el caso base, al iniciar el protocolo se encuentra en dos configuraciones (X y Y) al azar, con una distancia inicial $\delta(X_t, Y_t)$ entre éstas. Al ejecutar el protocolo se escoge un procesador al azar para que realice las operaciones atómicas de leer y modificar su registro, si es el caso, de la siguiente forma. Cuando se escoge el procesador 1 en X para que ejecute los pasos atómicos, éste lee del registro $x_{2,2}$ (que es el registro líder del procesador 2) y actualiza el valor del registro $x_{1,2}$ y de acuerdo a lo que tiene en el registro $x_{1,1}$ determina si lanza una moneda o no, para modificar el registro $x_{1,1}$, lo mismo con Y. A su vez cuando se escoge el procesador 2 en X para que realice operaciones atómicas, este lee del registro $x_{1,1}$ (registro líder del procesador 1), actualiza el registro $x_{2,1}$ y modifica o no el registro $x_{2,2}$ (lo cual depende de si lanza una moneda o no y del resultado del lanzamiento), lo mismo para Y. Al ejecutar el protocolo, se

genera una nueva distancia $\delta(X_{t+1}, Y_{t+1})$, la cual puede ser diferente o igual a la distancia inicial $\delta(X_t, Y_t)$. Esta nueva distancia depende de los cambios que se realizaron en las 4 variables que se actualizaron durante la ejecución del protocolo; que corresponden a las dos variables de X y a las dos variables de Y, ya que el valor de la variable que no se considera en la ejecución de las operaciones atómicas no ofrece ningún cambio en la distancia. Cuando se realiza una acción aleatoria, el resultado siempre es el mismo para ambas configuraciones, es decir, al escoger un procesador en X para que realice las operaciones atómicas, el mismo procesador se elige en Y. También el resultado de lanzamiento de la moneda es el mismo para ambas configuraciones. A continuación, las variables que tienen debajo una flecha son las variables involucradas cuando el procesador 1 ejecuta las operaciones atómicas.

$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$		$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$
$\uparrow$	$\uparrow$		$\uparrow$		$\uparrow$	$\uparrow$		$\uparrow$

Similarmente se muestran las variables involucradas cuando se escoge el procesador 2 para que realice operaciones atómicas.

$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$		$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$
$\uparrow$		$\uparrow$	$\uparrow$		$\uparrow$		$\uparrow$	$\uparrow$

A continuación se muestra un ejemplo de lo anterior y cómo se están cumpliendo las condiciones de 25, en el caso base. Suponiendo la siguiente configuraciones inicial del sistema.

$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$		$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$	$\delta(X_t, Y_t)$
1	0	0	1		0	1	1	0	4

Siguiendo la ejecución del protocolo y escogiendo al procesador 1 para que realice las operaciones atómicas, se obtienen dos posibles resultados.

$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$	$\delta(X_{t+1}, Y_{t+1})$	Comentarios
1	0	0	1	0	1	1	0	4	Inicio
			↑				↑		
1	**1**	0	1	0	**0**	1	0		$x_{1,2} \leftarrow x_{2,2}\,(y_{1,2} \leftarrow y_{2,2}\,)$
	↑				↑				
?	1	0	1	?	0	1	0		$x_{1,1}\,,\ y_{1,1} \leftarrow$ Res. moneda
↑				↑					
0	1	0	1	0	0	1	0	3	Resultado 1
1	1	0	1	1	0	1	0	3	Resultado 2

Durante la ejecución del algoritmo el registro $x_{2,1}(y_{2,1})$ no cambia. Se escoge a un procesador para que realice operaciones atómicas, en este ejemplo se escoge al procesador 1 para que realice estas acciones. Los registros del vector interno del procesador 2 se mantienen fijos. El procesador 1 lee el registro $x_{2,2}$ $(y_{2,2})$, esto se puede ver representado con el primer renglón de flechas. Después de esto, el registro $x_{1,2}(y_{1,2})$ se modifica con el valor leído del registro $x_{2,2}(y_{2,2})$, esto se visualiza en el segundo renglón que contiene flechas. A continuación el procesador 1 determina si lanza un volado conforme su propio valor $x_{1,1}(y_{1,1})$ y el valor del registro $x_{1,2}$ $(y_{1,2})$ actualizado, esto se puede ver en el tercer renglón que contiene flechas. En este ejemplo sí se hace un lanzamiento de moneda ya que el valor del registro $x_{1,2}$ $(y_{1,2})$ es un 1(0) y el valor del registro $x_{1,1}(y_{1,1})$ también es un 1(0). Se generan dos posibles resultados, uno donde el volado da como resultado un 0 y otro donde el resultado del volado es un 1, el registro $x_{1,1}(y_{1,1})$ toma

el valor del resultado del lanzamiento de moneda. Note que la distancia entre ambas configuraciones disminuye de 4 a 3.

En forma similar si se hubiera escogido al procesador 2 para que realice las operaciones atómicas se obtienen dos posibles resultados.

$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$	$\delta(X_{t+1}, Y_{t+1})$	Comentarios
1	0	0	1	0	1	1	0	4	Inicio
↑				↑					
1	0	**1**	1	0	1	**0**	0		$x_{2,1} \leftarrow x_{1,1}\,(y_{2,1} \leftarrow y_{1,1}\,)$
		↑				↑			
1	0	1	?	0	1	0	?		$x_{2,2}\,,\ y_{2,2} \leftarrow$ Res. moneda
			↑				↑		
1	0	1	0	0	1	0	0	3	Resultado 3
1	0	1	1	0	1	0	1	3	Resultado 4

Note que siguiendo la ejecución en la misma forma como el caso anterior, para el presente caso la distancia también se reduce a 3.

Como se puede ver en el ejemplo, una configuración inicial puede cambiar a cuatro posibles configuraciones en un paso.

El ejemplo anterior se repitió para las 256 posibles configuraciones iniciales. Para cada configuración se calculó la distancia promedio en la siguiente configuración, la probabilidad de cambio y se hizo una clasificación que se muestra en la Tabla IV.

Tabla IV. Resultados en el caso base

Distancia inicial	Distancia promedio siguiente configuración	Probabilidad de cambio de la distancia	Número de configuraciones iniciales
4	3.5	0.5	16
3	2.75	.625	64
2	1.91	.75	96
1	1	.625	64
0	0	0	16
Total	.128	.625	256

En la primera columna de la Tabla IV se muestran las distancias iniciales de las configuraciones del protocolo. La segunda columna muestra cómo en promedio la distancia decrece en las configuraciones siguientes (después que el protocolo ejecuta un paso) y en el último renglón se observa que la distancia decrece en promedio 0.128. La tercera columna de la tabla muestra la probabilidad de cambio de la distancia de las configuraciones siguientes (después que el protocolo se ejecuta un paso) y se puede ver que en total la probabilidad de cambio es de 0.625 para el caso base, el cual es mayor a $\frac{1}{2}$. La cuarta y última columna muestra el número de configuraciones iniciales por cada distancia. De los resultados anteriores se puede ver que las condiciones de 25 se cumplen para el caso base .

En el Apéndice A se muestra una tabla con todos los posibles casos para $n = 2$ donde se cumplen las condiciones de 25 con $\alpha = \frac{1}{2}$. A continuación se establece la hipótesis inductiva.

Hipótesis inductiva:

Las condiciones de 25 se satisfacen para cualquier número de nodos $n \leq k$.

Paso inductivo:

Dado que la hipótesis inductiva se cumple $\forall n \leq k$, se demuestra que también se cumple para $n = k + 1$.

En el caso cuando $n = k + 1$ también se consideran 4 variables que son sujetas a cambio, a continuación se muestra un ejemplo.

$$\begin{array}{cccccccccccccccc} x_{i,i} & \cdots & x_{i,k+1} & \cdots & \cdots & \cdots & x_{k+1,k+1} & & y_{i,i} & \cdots & y_{i,k+1} & \cdots & \cdots & \cdots & y_{k+1,k+1} \\ \uparrow & & \uparrow & & & & \uparrow & & \uparrow & & \uparrow & & & & \uparrow \end{array}$$

Los puntos suspensivos representan registros del sistema, pero para efectos de simplificación de la demostración se omiten. Estos registros no cambian durante la ejecución del protocolo durante el instante de tiempo que se está ejemplificando y por ende no modifican la distancia entre las configuraciones.

En el caso general, se escoge un procesador al azar para que ejecute operaciones atómicas (en este ejemplo el procesador i), el cual lee el registro de algún procesador seleccionado al azar (en este caso el procesador $k + 1$) y modifica su registro interno $x_{i,k+1}$ $(y_{i,k+1})$. Una vez modificado este registro, toma la decisión de lanzar o no una moneda para modificar su propio registro $x_{i,i}$ $(y_{i,i})$. A diferencia del caso base, en el caso general, la decisión de lanzar o no la moneda también depende de los valores de los demás registros del vector interno del procesador i y se supone que éstos pueden ser

0 o 1 con igual probabilidad. Si $x_{i,i} = 0$ $(y_{i,i}) = 0$ existe una probabilidad de $\left(\frac{1}{2}\right)^{n-2}$ de que todos los demás registros del vector interno sean 0 y por lo tanto el procesador i tiene que lanzar una moneda. Ahora bien, si $x_{i,i} = 1$ $(y_{i,i} = 1)$, existe una probabilidad de $1 - \left(\frac{1}{2}\right)^{n-2}$ que exista al menos un 1 en los demás registros del vector interno, ocasionando un lanzamiento de moneda. La distancia de la siguiente configuración se determina contando entre las variables involucradas los registros que cambiaron en esa ocasión y como los registros representados por puntos suspensivos se mantienen igual, éstos no afectan al determinar la nueva distancia.

Para el caso donde $n = k + 1$ se pueden tener algunos de los siguientes casos con una cierta probabilidad, dependiendo de la configuración inicial:

1. Ambas configuraciones lanzan una moneda.
2. Ninguna de las configuraciones lanza una moneda.
3. La configuración X lanza una moneda, pero la configuración Y no lanza una moneda.
4. La configuración Y lanza una moneda, pero la configuración X no lanza una moneda.

Conforme a lo anterior y el estudio de los posibles casos cuando $n = k + 1$ se determina que se cumplen las condiciones de 25 se cumplen con $\alpha = \frac{1}{2}$. El valor de α se determina por medio de una fórmula que se obtuvo con el estudio del caso general.

A continuación se muestra un ejemplo en el que solo se expresa la distancia de los registros que se conocen, ya que los registros que están representados por los puntos suspensivos no cambian y por lo tanto no influyen en la distancia:

La configuración inicial del sistema es la siguiente (se supone que la distancia inicial es R).

$x_{1,1}$	$\cdots$	$x_{1,k+1}$	$\cdots$	$x_{k+1,k+1}$	$y_{1,1}$	$\cdots$	$y_{1,k+1}$	$\cdots$	$y_{k+1,k+1}$	$\delta(X_t,Y_t)$
1		1		0	0		0		0	R

Al ejecutar el protocolo, el calendarizador escoge a un procesador al azar para que ejecute las operaciones atómicas, en este ejemplo se escoge al procesador 1 para que realice estas acciones y el procesador del cual lee es el procesador $k+1$. Por lo que $x_{i,k+1} \leftarrow x_{k+1,k+1}$ $(y_{i,k+1} \leftarrow y_{k+1,k+1})$.

$x_{1,1}$	$\cdots$	$x_{1,k+1}$	$\cdots$	$\cdots$	$x_{k+1,k+1}$	$y_{1,1}$	$\cdots$	$y_{1,k+1}$	$\cdots$	$\cdots$	$y_{k+1,k+1}$
1		**0**			0	0		**0**			0
		$\uparrow$			$\uparrow$			$\uparrow$			$\uparrow$

En la siguiente operación atómica el procesador que leyó tiene que decidir si lanza la moneda o no y determinar el valor de su registro, en este caso $x_{1,1}(y_{1,1})$, quedando las siguientes posibilidades:

1. La configuración X lanza una moneda y la configuración Y también lanza la moneda.

Lo anterior se da porque en la configuración X, en el vector del procesador 1, puede existir al menos otro 1, además del que ya hay en $x_{1,1}$. Esto pasa con una probabilidad de $1-\left(\frac{1}{2}\right)^{n-2}$. En la configuración Y el registro del procesador 1 contiene un 0 y con una probabilidad de $\left(\frac{1}{2}\right)^{n-2}$ este lanza un moneda, ya que pudiera ser que los demás

registros tuvieran puros 0. La configuración siguiente puede ser alguna de estas:

$x_{1,1}$	$\cdots$	$x_{1,k+1}$	$\cdots$	$x_{k+1,k+1}$	$y_{1,1}$	$\cdots$	$y_{1,k+1}$	$\cdots$	$y_{k+1,k+1}$	$\delta(X_{t+1},Y_{t+1})$
0		0		0	**0**		0		0	$R-2$
1		0		0	**1**		0		0	$R-2$
$\uparrow$					$\uparrow$					

2. Ninguna de las dos configuraciones lanza una moneda.

Esto sucede en la configuración X porque el registro $x_{1,1} = 1$ y los demás nodos solo contienen 0, esto pasa con una probabilidad $\left(\frac{1}{2}\right)^{n-2}$. En la configuración Y ésto ocurre con una probabilidad de $1-\left(\frac{1}{2}\right)^{n-2}$, ya que ésta es la probabilidad de que exista al menos un 1 en los demás registros del vector y porque el registro $y_{1,1} = 0$. La configuración en el siguiente paso puede ser como sigue:

$x_{1,1}$	$\cdots$	$x_{1,k+1}$	$\cdots$	$x_{k+1,k+1}$	$y_{1,1}$	$\cdots$	$y_{1,k+1}$	$\cdots$	$y_{k+1,k+1}$	$\delta(X_{t+1},Y_{t+1})$
1		0		0	**0**		0		0	$R-1$
$\uparrow$					$\uparrow$					

3. La configuración X lanza una moneda y la configuración Y no lanza moneda.

Que la configuración X lance una moneda puede ocurrir con una probabilidad de $1-\left(\frac{1}{2}\right)^{n-2}$, por lo explicado en el punto 1, que Y no lance una moneda ocurre con una probabilidad de $1-\left(\frac{1}{2}\right)^{n-2}$, por lo explicado en el punto 2. Las configuraciones

generadas que se producen cuando esto ocurre son las siguientes:

$x_{1,1}$	$\cdots$	$x_{1,k+1}$	$\cdots$	$x_{k+1,k+1}$	$y_{1,1}$	$\cdots$	$y_{1,k+1}$	$\cdots$	$y_{k+1,k+1}$	$\delta(X_{t+1},Y_{t+1})$
0		0		0	**0**		0		0	$R-2$
1		0		0	**0**		0		0	$R-1$
$\uparrow$					$\uparrow$					

4. La configuración X no lanza moneda y la configuración Y lanza una moneda.

Con una probabilidad de $\left(\frac{1}{2}\right)^{n-2}$, X no lanza la moneda por lo que se explica en el punto 2. Con una probabilidad de $\left(\frac{1}{2}\right)^{n-2}$, Y lanza una moneda por lo explicado en el punto 1. Las configuraciones generadas cuando esto ocurre son las siguientes:

$x_{1,1}$	$\cdots$	$x_{1,k+1}$	$\cdots$	$x_{k+1,k+1}$	$y_{1,1}$	$\cdots$	$y_{1,k+1}$	$\cdots$	$y_{k+1,k+1}$	$\delta(X_{t+1},Y_{t+1})$
1		0		0	**0**		0		0	$R-1$
1		0		0	**1**		0		0	$R-2$
$\uparrow$					$\uparrow$					

Como se ve en este ejemplo, la distancia en las siguientes configuraciones en promedio disminuye o se mantiene, cumpliendo la primera condición de 25 y la probabilidad de cambio de esta distancia es mayor a un medio, cumpliendo la segunda condición de 25.

Por lo tanto, el protocolo es auto-estabilizante con respecto a $\mathcal{L}$ (cuando existe un líder). Además, por las cotas dadas por la técnica de acoplamiento se tiene que:

1. El tiempo esperado de alcance satisface:

$$\mathbf{H}_{\mathcal{L}} \leq 2\left(N^2+1\right)^2 \tag{26}$$

2. El tiempo de absorción-ε satisface:

$$\Theta_{\mathcal{L}}(\varepsilon) \leq \left\lceil 2e\left(N^2+1\right)^2 \right\rceil \left\lceil \ln\left(\frac{1}{\varepsilon}\right) \right\rceil \tag{27}$$

A continuación se muestra la Figura 23 donde se puede observar el tiempo de convergencia del peor caso de Dolev *et al.* (1995) el cual se determina con la técnica de calendarizador-suerte y se obtiene $2^{O(N)}$. El autor de este trabajo de investigación con la técnica de acoplamiento obtiene una cota para el tiempo esperado de convergencia de la peor configuración de $2\left(N^2+1\right)^2$.

En el presente capítulo se utiliza el método de acoplamiento para demostrar la convergencia del protocolo de espacio mínimo de Dolev *et al.* (1995) y se ofrece una cota superior en el tiempo de convergencia de este algoritmo. En el siguiente capítulo se utiliza la técnica de acoplamiento en el algoritmo de Israeli y Jalfon (1990).

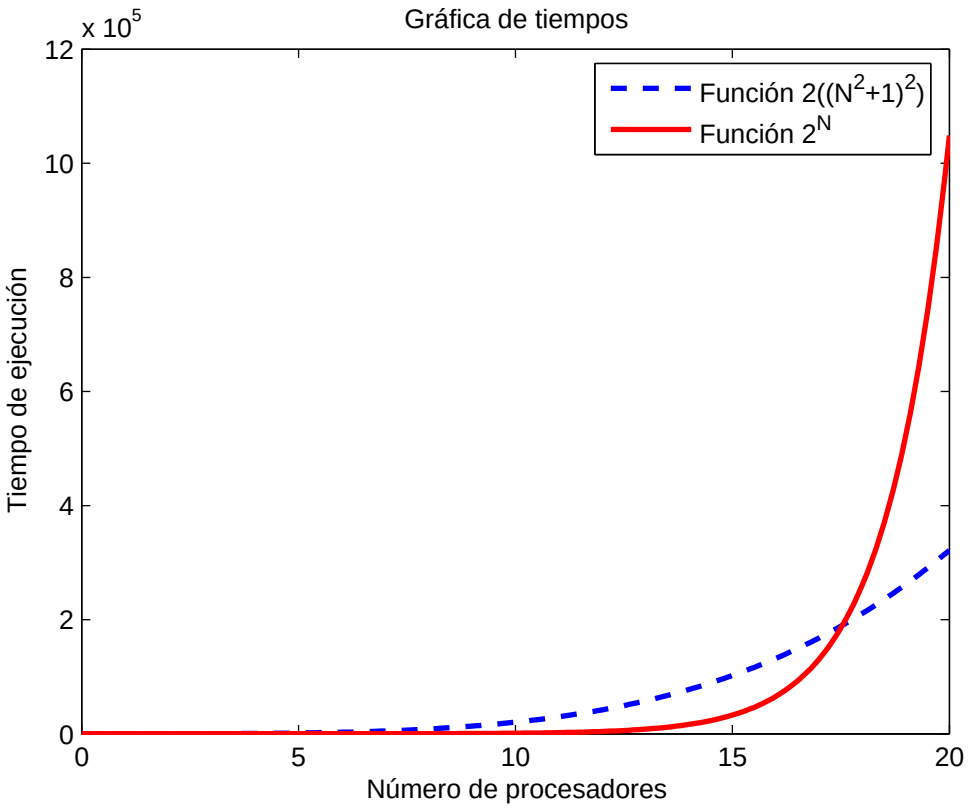

Figura 23. Tabla de tiempos

Capítulo VI

Técnica de Acoplamiento en el Algoritmo de Israeli-Jalfon

En este capítulo se usa la técnica de acoplamiento en el algoritmo de Israeli-Jalfon de exclusión mutua (Israeli y Jalfon, 1990) para que corra en una topología asimétrica. Este algoritmo se utiliza para el paso y eliminación de tokens en un sistema distribuido que está representado por un grafo arbitrario.

VI.1 Descripción del Algoritmo de Exclusión Mutua de Israeli-Jalfon

En (Israeli y Jalfon, 1990) se presenta un método modular para construir un protocolo uniforme auto-estabilizante de exclusión mutua. Un algoritmo de exclusión mutua se utiliza para regular el acceso a recursos que no se deben compartir simultáneamente. En (Israeli y Jalfon, 1990) el concepto de uniformidad difiere de otros documentos; en ese documento se dice que un protocolo es uniforme porque en él todos los nodos usan el mismo programa, a diferencia de otros trabajos en donde especifican que un sistema es uniforme si todos los nodos tienen las mismas características.

El protocolo de Israeli-Jalfon es simple, un nodo que mantiene un token se dice que está *privilegiado* y puede entonces entrar a su sección crítica, es decir, puede utilizar

el recurso por el cual está compitiendo. Una vez que el nodo ha terminado de utilizar su sección crítica, pasa el token a uno de sus vecinos. Siempre que dos tokens se junten en un solo nodo, uno de los tokens se elimina. La auto-estabilización de un token basado en un protocolo de exclusión mutua puede dividirse en dos subtareas llamadas: *auto-estabilización local,* en el cual el vecindario de cada nodo estabiliza a una configuración legal, y *auto-estabilización global,* la cual asegura que eventualmente existirá exactamente un token en todo el sistema.

Israeli y Jalfon (1990) utilizan dos esquemas para construir protocolos auto-estabilizantes uniformes de exclusión mutua para cualquier grafo:

- *Esquema de manejo de token.* Es un protocolo de auto-estabilización local el cual habilita toda la actividad del token. Permite el paso del token y previene situaciones de "puntos muertos" (cuando se detecta que ningún token está presente en el sistema). Este esquema constituye el nivel más bajo de los protocolos y asegura la auto-estabilización local.

- *Esquema de recorrido del grafo.* Este esquema determina en cada nodo que tiene un token a cuál de sus vecino le enviará el token. Este esquema sirve para obtener la auto-estabilización global y para asegurar un comportamiento justo del protocolo una vez que se ha alcanzado la auto-estabilización global.

Este método modular para construir protocolos auto-estabilizantes uniformes de exclusión mutua tiene varias ventajas. Ambos niveles de protocolos se pueden reemplazar

por esquemas mejores, una vez que se encuentren. La prueba de corrección y la complejidad se pueden analizar por separado para ambos niveles.

Un esquema de recorrido del grafo se usa para que cada nodo que tiene un token escoja a qué vecino enviar el token. Esto asegura a que si el sistema inicia en una configuración con más de un token, entonces eventualmente todos los tokens excepto uno se eliminan por colisión con otros tokens (auto-estabilización global). Una vez que se mantenga un token en el sistema, el esquema asegura que todos los nodos se privilegian un número infinito de veces (comportamiento justo). Es bien conocido que una caminata aleatoria es un esquema de recorrido del grafo. Para probar la auto-estabilización global es suficiente con demostrar que dos tokens que realizan una caminata aleatoria, en cualquier grafo finito, se encontrarán con alta probabilidad. La demostración se hace en (Israeli y Jalfon, 1990) considerando el siguiente juego de un solo jugador.

El juego se lleva a cabo en un grafo finito $G = (V, E)$, donde V son los nodos que representan a cada nodo y E son las aristas que representan las líneas de comunicación entre éstos. Un ejemplo del juego es cuando dos tokens residen inicialmente en dos distintos nodos de G. En cada vuelta, uno de los tokens (seleccionado por el jugador) se mueve desde su ubicación actual a un nodo vecino seleccionado uniformemente al azar. El juego termina cuando los dos tokens colisionan.

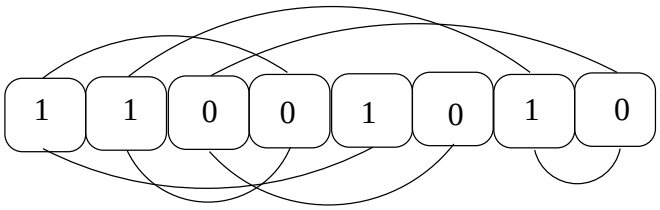

Figura 24. Modelado del sistema como un vector

VI.2 Aplicando la Técnica de Acoplamiento en el Esquema de Recorrido del Grafo de Israeli-Jalfon

El esquema de recorrido del grafo de Israeli-Jalfon inicia al mover uno de los tokens a través de alguno de los vecinos del nodo donde se encuentra en ese instante; se repite este paso sucesivamente hasta que los token colisionan en el mismo nodo y uno de ellos se elimina. Para aplicar la técnica de acoplamiento, el autor de este trabajo de tesis hace un cambio a este esquema; el cambio radica en que, en lugar de escoger un token para que se mueva a través de uno de los vecinos, se escoge un nodo para que envíe el token (si lo tiene) a uno de sus vecinos; si el nodo no tiene un token, éste no hace nada.

En el presente trabajo de investigación la demostración de convergencia del Esquema del Recorrido del Grafo de Israeli-Jalfon se hace de la siguiente manera.

Este protocolo se puede modelar como un vector de N registros, los cuales representan a los nodos del sistema y las aristas se representan con conexiones entre registros, un token se representa como un 1 en el registro y la no existencia de token se modela como un 0 en el registro. En la Figura 24 se tiene un ejemplo de 8 nodos con 4 tokens.

Al utilizar la técnica de acoplamiento se tienen dos vectores (X y Y) que representan

al mismo protocolo y una δ que es una función de distancia entre estos dos vectores, en este caso se usa la distancia Hamming que cuenta el número de registros en los cuales los vectores difieren. Se inicia en un estado donde los vectores tienen el mismo número de tokens y éstos pueden estar en el mismo o en diferente nodo. Se escoge el mismo registro en los dos vectores y se escoge también al mismo vecino de ese registro para enviarle el token; si en el vecino existe un token, se elimina uno de los tokens. A continuación un ejemplo sencillo.

Se tiene un sistema con 4 nodos con los siguientes estados iniciales.

$$\begin{matrix} X_t & 1 & 0 & 1 & 0 \\ Y_t & 1 & 1 & 0 & 0 \end{matrix} \quad \delta(X_t, Y_t) = 2$$

Después de una iteración se escoge al segundo registro para que envie el token al tercer registro y queda de la siguiente manera.

$$\begin{matrix} X_{t+1} & 1 & 0 & 1 & 0 \\ Y_{t+1} & 1 & 0 & 1 & 0 \end{matrix} \quad \delta(X_{t+1}, Y_{t+1}) = 0$$

En la siguiente iteración se escoge al tercer registro para que envie el token al primer registro y queda de la siguiente manera.

$$\begin{matrix} X_{t+2} & 1 & 0 & 0 & 0 \\ Y_{t+2} & 1 & 0 & 0 & 0 \end{matrix} \quad \delta(X_{t+2}, Y_{t+2}) = 0$$

Como se puede ver en cada iteración del sistema, la distancia δ se mantiene o disminuye, así mismo el sistema en cada vector converge a un solo token y queda en el

mismo nodo que el otro vector.

Para demostrar este esquema con el método de acoplamiento es necesario demostrar el Teorema 6 que se toma de (Fribourg *et al.*, 2006) que dice:

Teorema 6. Dada una cadena de Markov A y un conjunto ergódico $\mathcal{L}$, suponga que existe un acoplamiento (X_t, Y_t) y una función δ en el conjunto de configuraciones que toma valores en $\{0, 1, \ldots, B\}$ tal que:

$$\delta(X_t, Y_t) = 0 \text{ si y solo si } X_t = Y_t \text{ y} \tag{28}$$

existe una $\beta < 1$ tal que, para todo (X_t, Y_t):

$$E[\delta\left(X_{t+1}, Y_{t+1}\right)] \leq \beta\delta\left(X_t, Y_t\right) \tag{29}$$

Entonces $\mathcal{L}$ es el único conjunto ergódico y A es auto-estabilizante con respecto a $\mathcal{L}$. Además:

1. El tiempo esperado de alcance (hitting time, en inglés) satisface: $\mathrm{H}_{\mathcal{L}} \leq \frac{B}{1-\beta}$.
2. El tiempo de absorción-ε ($\varepsilon-$absorptiontime, en inglés) satisface: $\Theta_{\mathcal{L}}(\varepsilon) \leq \frac{\ln(B/\varepsilon)}{1-\beta}$.

El Teorema 6 quiere decir que una vez que los sistemas X y Y se han acoplado, la distancia entre ellos es 0; y que en promedio la distancia de la siguiente configuración se mantiene o decrece con respecto a la distancia entre las configuraciones en el tiempo

presente, con una cierta medida de reducción.

Demostración de 28.

$$\delta(X_t, Y_t) = 0 \text{ si y solo si } X_t = Y_t$$

Sean $\{x_1, x_2, \ldots, x_n\}$y $\{y_1, y_2, \ldots, y_n\}$ los vectores de X_t y Y_t, si cada elemento del vector de X_t es igual a su correspondiente elemento de Y_t entonces la distancia es cero.

En caso contrario si $\delta = 0$ forzosamente cada elemento de X_t y Y_t deben ser iguales y por consiguiente $X_t = Y_t$.

□

Demostración de 29.

Demostrar que se cumple la condición 29 en el esquema de recorrido del grafo. La demostración se hace tomando el mismo nodo fuente y el mismo nodo destino en los dos vectores.

La Tabla V muestra los posibles estados de los nodos fuente (F_x y F_y) y destino (D_x y D_y) en los dos vectores (X y Y) con su respectiva distancia inicial ($\delta(X_t, Y_t)$), y las probabilidades de que tal estado ocurra (P); ya que no es un grafo completo, las probabilidades de que cierto estado ocurra no es una probabilidad uniforme y depende del número de tokens en cada vector (ℓ_1 y ℓ_2 tokens del vector X y el vector Y, respectivamente), en cada iteración el número de tokens puede cambiar en una configuración pero en la otra configuración no cambia, es por eso, que se utiliza diferentes variables para denotar el número de tokens en cada configuración; las últimas dos columnas muestran

la distancia en la siguiente iteración ($\delta(X_{t+1,}Y_{t+1})$) y el incremento en la distancia ($\triangle\delta$).

La Tabla V es para el caso general donde el número de tokens es el mismo en los dos vectores pero están distribuidos de diversa manera a través de los nodos para este caso se obtiene una β.

Para obtener β primero se despeja de la condición 29 y se obtiene la desigualdad 30.

$$\beta \geq \frac{E[\delta(X_{t+1}, Y_{t+1})]}{\delta(X_t, Y_t)} \tag{30}$$

Sustituyendo el valor de $E[\delta(X_{t+1}, Y_{t+1})]$ en la desigualdad 30 se obtiene la desigualdad 31:

$$\beta \geq \frac{\delta(X_t, Y_t) + E[\triangle\delta]}{\delta(X_t, Y_t)} \tag{31}$$

Simplificando se tiene la desigualdad 32:

$$\beta \geq 1 + \frac{E[\triangle\delta]}{\delta(X_t, Y_t)} \tag{32}$$

Para el caso general se tiene:

$$\begin{aligned}
\mathrm{E}\left[\Delta\delta\right] &= -\frac{\left(1-\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1}{n-1}\right)(\ell_2)(\ell_2-1)}{n(n-1)} - \frac{\left(1-\frac{\ell_1}{n}\right)(\ell_1)(\ell_2)(\ell_2-1)}{n(n-1)^2} \\
&- \frac{(\ell_1)\left(1-\frac{\ell_{1-1}}{n-1}\right)(\ell_2)(\ell_2-1)}{n^2(n-1)} - \frac{(\ell_1)(\ell_1-1)\left(1-\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2}{n-1}\right)}{n(n-1)} \\
&- \frac{(\ell_1)(\ell_1-1)\left(1-\frac{\ell_2}{n}\right)(\ell_2)}{n(n-1)^2} - \frac{(\ell_1)(\ell_1-1)(\ell_2)\left(1-\frac{\ell_2-1}{n-1}\right)}{n^2(n-1)} \\
&- \frac{2(\ell_1)\left(1-\frac{\ell_1}{n}\right)(\ell_2)\left(1-\frac{\ell_2-1}{n-1}\right)}{n(n-1)} - \frac{2(\ell_2)\left(1-\frac{\ell_2}{n}\right)(\ell_1)\left(1-\frac{\ell_1-1}{n-1}\right)}{n(n-1)}
\end{aligned} \tag{33}$$

el cual depende del número de tokens y del número de nodos.

Tabla V. Posibles estados iniciales de nodos fuente y destino

X_t		Y_t		$\delta(X_t, Y_t)$	P	$\delta(X_{t+1}, Y_{t+1})$	$\triangle\delta$
F_x	D_x	F_y	D_y				
0	0	0	0	0	$\left(1-\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2}{n-1}\right)$	0	0
0	0	0	1	1	$\left(1-\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(\frac{\ell_2}{n-1}\right)$	1	0
0	0	1	0	1	$\left(1-\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2-1}{n-1}\right)$	1	0
0	0	1	1	2	$\left(1-\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(\frac{\ell_2-1}{n-1}\right)$	1	−1
0	1	0	0	1	$\left(1-\frac{\ell_1}{n}\right)\left(\frac{\ell_1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2}{n-1}\right)$	1	0
0	1	0	1	0	$\left(1-\frac{\ell_1}{n}\right)\left(\frac{\ell_1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(\frac{\ell_2}{n-1}\right)$	0	0
0	1	1	0	2	$\left(1-\frac{\ell_1}{n}\right)\left(\frac{\ell_1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2-1}{n-1}\right)$	0	−2
0	1	1	1	1	$\left(1-\frac{\ell_1}{n}\right)\left(\frac{\ell_1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(\frac{\ell_2-1}{n-1}\right)$	0	−1
1	0	0	0	1	$\left(\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1-1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2}{n-1}\right)$	1	0
1	0	0	1	2	$\left(\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1-1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(\frac{\ell_2}{n-1}\right)$	0	−2
1	0	1	0	0	$\left(\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1-1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2-1}{n-1}\right)$	0	0
1	0	1	1	1	$\left(\frac{\ell_1}{n}\right)\left(1-\frac{\ell_1-1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(\frac{\ell_2-1}{n-1}\right)$	0	−1
1	1	0	0	2	$\left(\frac{\ell_1}{n}\right)\left(\frac{\ell_1-1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2}{n-1}\right)$	1	−1
1	1	0	1	1	$\left(\frac{\ell_1}{n}\right)\left(\frac{\ell_1-1}{n-1}\right)\left(1-\frac{\ell_2}{n}\right)\left(\frac{\ell_2}{n-1}\right)$	0	−1
1	1	1	0	1	$\left(\frac{\ell_1}{n}\right)\left(\frac{\ell_1-1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(1-\frac{\ell_2-1}{n-1}\right)$	0	−1
1	1	1	1	0	$\left(\frac{\ell_1}{n}\right)\left(\frac{\ell_1-1}{n-1}\right)\left(\frac{\ell_2}{n}\right)\left(\frac{\ell_2-1}{n-1}\right)$	0	0

En la técnica de acoplamiento el análisis se hace desde el peor caso; en este esquema el peor caso es cuando existe un token en cada vector, pero en nodos diferentes, lo cual genera una distancia 2. Para el peor caso se tiene la Tabla VI con los posibles estados iniciales que pueden existir en los nodos fuente y destino, que se escogen del conjunto total de nodos.

En el peor caso, al despejar β de la condición 29 se sustituye $\delta(X_t, Y_t) = 2$ y se obtiene la desigualdad 34:

$$\beta \geq 1 + \frac{E[\triangle\delta]}{2} \tag{34}$$

Tabla VI. Posibles estados iniciales del peor caso.

X_t		Y_t		$\delta(X_t, Y_t)$	P	$\delta(X_{t+1}, Y_{t+1})$	$\triangle\delta$
F_x	D_x	F_y	D_y				
0	0	0	0	0	$\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)$	0	0
0	0	0	1	1	$\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)\left(1-\frac{1}{n}\right)\left(\frac{1}{n-1}\right)$	1	0
0	0	1	0	1	$\left(1-\frac{1}{n}\right)\left(1-\frac{1}{n-1}\right)\left(\frac{1}{n}\right)(1)$	1	0
0	1	0	0	1	$\left(1-\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\left(1-\frac{1}{n}\right)(1)$	1	0
0	1	1	0	2	$\left(1-\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)(1)$	0	-2
1	0	0	0	1	$\left(\frac{1}{n}\right)(1)(1)\left(1-\frac{1}{n-1}\right)$	1	0
1	0	0	1	2	$\left(\frac{1}{n}\right)(1)(1)\left(\frac{1}{n-1}\right)$	0	-2

Sustituyendo la ecuación 34 por $E[\triangle\delta] = -2\left[\left(1-\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)+\left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\right]$ del peor caso, se obtiene la siguiente β que se específica en la desigualdad 35:

$$\beta \geq 1 - \left[\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)\left[1+\left(1-\frac{1}{n}\right)\right]\right] \tag{35}$$

que depende del número de nodos que tenga el sistema. Y como se tiene:

$$E[\delta(X_{t+1}, Y_{t+1})] = \delta(X_t, Y_t) + E[\triangle\delta] \tag{36}$$

entonces al sustituir $\delta(X_t, Y_t) = 2$ y $E[\triangle\delta] = -2\left[\left(1-\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)+\left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\right]$ en la ecuación 36 se obtiene la ecuación 37:

$$E[\delta(X_{t+1}, Y_{t+1})] = 2 - 2\left[\left(1-\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)+\left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\right] \tag{37}$$

Sustituyendo $E[\delta(X_{t+1}, Y_{t+1})]$ de la ecuación 37 y $\beta\delta(X_t, Y_t)$ de la ecuación 35 en

la condición 29 se obtiene la desigualdad 38:

$$\begin{aligned} 2 \;-\; & 2\left[\left(1-\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)+\left(\frac{1}{n}\right)\left(\frac{1}{n-1}\right)\right] \\ \leq\; & 2\left[1-\left[\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)\left[1+\left(1-\frac{1}{n}\right)\right]\right]\right] \end{aligned} \tag{38}$$

Como se puede apreciar en la desigualdad 37 la condición 29 se cumple, por lo tanto el sistema converge. Además:

1. El tiempo esperado de alcance para el esquema del recorrido del grafo satisface:

$$\mathbf{H}_{\mathcal{L}} \leq \frac{2}{\left[\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)\left[1+\left(1-\frac{1}{n}\right)\right]\right]} \tag{39}$$

2. El tiempo de absorción-ε para el esquema del recorrido del grafo satisface:

$$\Theta_{\mathcal{L}}(\varepsilon) \leq \frac{\ln(2/\varepsilon)}{\left[\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)\left[1+\left(1-\frac{1}{n}\right)\right]\right]} \tag{40}$$

En este capítulo se utiliza el método de acoplamiento en el Esquema de Recorrido del Grafo de (Israeli y Jalfon, 1990) y se obtiene una cota superior en el tiempo de convergencia del mismo. En el siguiente capítulo se mencionan las conclusiones que se obtienen de este trabajo de investigación.

Capítulo VII

Conclusiones y Trabajo a Futuro

VII.1 Conclusiones

El objetivo general de esta tesis es ampliar el uso de la técnica de acoplamiento para demostrar el tiempo de convergencia de los algoritmos auto-estabilizantes bajo topologías asimétricas. Entre las contribuciones de este trabajo, se observaron diferentes propiedades del comportamiento de algunos algoritmos auto-estabilizantes para utilizar en ellos la técnica de acoplamiento.

En el contexto de acoplamiento, la convergencia de los algoritmos auto-estabilizantes se demuestra modelando el sistema como una cadena de Markov ergódica. Para ello, se define una función de distancia entre las dos copias fieles (X y Y) de la misma cadena de Markov que representa al sistema. Esta función decrece conforme X y Y se acercan entre si hasta lograr acoplarse. Una vez que las copias fieles se acoplan, se comportan de la misma manera durante las siguientes iteraciones del sistema. También se cumple la propiedad de cerradura de los algoritmos auto-estabilizantes, i.e. una vez que ciertos elementos de X y Y son estables, siguen siéndolo indefinidamente. Esto también se puede ver como una cadena que representa la estabilidad total del sistema, ésta es monótona creciente, cada vez que el sistema alcanza cierta estabilidad, la cadena de estabilidad se incrementa, *i.e.* sea x una cadena que representa la estabilidad total de un sistema y se denota a x_i como el número de nodos que están estables dentro del sistema, en cada iteración se tiene que $x_i \leq x_{i+1}$.

Otra contribución de este trabajo de investigación es que se modelan dos algoritmos auto-estabilizantes en una topología de árbol para aplicarles la técnica de acoplamiento. Los algoritmos fueron el algoritmo de Herman de elección de líder (Herman, 1990) y el algoritmo del Dilema del Prisionero Iterado (Dyer *et al.*, 2002). Se observa que: para hacer un cambio de topología de un árbol a un anillo virtual, es necesario conocer las restricciones del algoritmo que se ejecutará en la topología virtual, como se muestra en el Capítulo IV. También se muestra que todo cambio de topología tiene asociado un costo adicional en el tiempo por ese cambio de topología y hay que valorar si es conveniente el cambio de topología para utilizar dicho algoritmo *versus* otro algoritmo con el mismo propósito (si existe). Sin embargo, para los algoritmos vistos el sobrecosto es constante.

Adicionalmente, se utiliza la técnica de acoplamiento en el protocolo de espacio mínimo de Dolev *et al.* (1995) obteniendo una cota más ajustada en el tiempo de convergencia de este algoritmo. El tiempo esperado de convergencia para el protocolo de espacio mínimo es $\mathbf{H}_{\mathcal{L}} \leq 2\left(N^2+1\right)^2$ que es mucho menor a la cota que se proporciona en (Dolev *et al.*, 1995), como se muestra en el Capítulo V. Un punto importante al usar la técnica de acoplamiento es definir la función distancia más conveniente.

Además en el Capítulo VI se calcula la cota en el tiempo de convergencia del esquema del recorrido del grafo de Israeli y Jalfon (1990). Este recorrido se modificó, de tal manera que los nodos eligen a quién enviar el token, y se obtuvo una cota en el tiempo de convergencia. La cota que se proporciona en el tiempo de convergencia en el esquema del recorrido del grafo es $\mathbf{H}_{\mathcal{L}} \leq \frac{2}{\left[\left(\frac{1}{n-1}\right)\left(\frac{1}{n}\right)\left[1+\left(1-\frac{1}{n}\right)\right]\right]}$. Esta cota es más ajustada a la cota exponencial que proporcionan Israeli y Jalfon (1990) para el caso general.

VII.2 Trabajo a Futuro

Aún falta mucho trabajo en relación al uso de la técnica de acoplamiento; algunos puntos que se prestan a investigación adicional y que se encontraron a raíz de este trabajo de investigación son:

1. El algoritmo de Israeli y Jalfon (1990) tiene dos esquemas, el esquema de recorrido del grafo y el esquema del manejo del token; como en esta tesis se utilizó la técnica de acoplamiento en el esquema del recorrido del grafo, un punto interesante sería el utilizar el método de acoplamiento en el esquema de manejo del token para determinar la cota en el tiempo de este esquema, y obtener una cota total en el tiempo del algoritmo.

2. Extender el uso de la técnica de acoplamiento en algoritmos auto-estabilizantes con diferentes propósitos, ya que en la mayor parte de la literatura la técnica de acoplamiento se utiliza en algoritmos auto-estabilizantes de elección de líder.

3. Determinar si la técnica de acoplamiento puede ser el método general que demuestre la convergencia de algoritmos auto-estabilizantes.

Referencias

Aldous, D. (1983). Random walks on finite groups and rapidly mixing markov chains. *SÃ©minaire de Probabilities XVII*, páginas 243–297.

Arora, A., Attie, P., Evangelist, M., y Gouda, M. (1993). Convergence of iteration systems. *Distributed Computing*, **7**(1): 43–53.

Boulinier, C., Petit, F., y Villain, V. (2004). When graph theory helps self-stabilization. En: *PODC '04: Proceedings of the twenty-third annual ACM symposium on Principles of Distributed Computing*, páginas 150–159, New York, NY, USA. ACM. ISBN 1-58113-802-4.

Boulinier, C., Petit, F., y Villain, V. (2008). Synchronous vs. asynchronous unison. *Algorithmica*, **51**(1): 61–80.

Bubley, R. y Dyer, M. (1997). Path coupling: A technique for proving rapid mixing in markov chains. En: *FOCS '97: Proceedings of the 38th Annual Symposium on Foundations of Computer Science*, página 223, Washington, DC, USA. IEEE Computer Society. ISBN 0-8186-8197-7.

Chen, Y., Datta, A. K., y Tixeuil, S. (2005). Stabilizing inter-domain routing in the internet. *Journal of High Speed Networks*, **14**(1): 21–37.

Cramer, C. y Fuhrmann, T. (2005). Self-stabilizing ring networks on connected graphs. University of Karlsruhe (TH), Fakultaet fuer Informatik, Technical Report 2005-5.

Dijkstra, E. W. (1974). Self stabilizing systems in spite of distributed control. *Communications of the ACM*, **17**(11): 643–644.

Dijkstra, E. W. (1986). A belated proof of self-stabilization. *Distributed Computing*, **1**(1): 5–6.

Dolev, S. (2000). *Self-Stabilization*. MIT Press.

Dolev, S. y Kat, R. I. (2004). Hypertree for self-stabilizing peer-to-peer systems. En: *NCA '04: Proceedings of the Network Computing and Applications, Third IEEE International Symposium*, páginas 25–32, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-2242-4.

Dolev, S. y Yagel, R. (2005a). Memory management for self-stabilizing operating systems. En: *Self-Stabilizing Systems*, Vol. 3764 de *Lecture Notes in Computer Science*, páginas 113–127. Springer Berlin/Heidelberg. ISBN 978-3-540-29814-4.

Dolev, S. y Yagel, R. (2005b). Self-stabilizing operating systems. En: *SOSP '05: Proceedings of the twentieth ACM symposium on Operating systems principles*, páginas 1–2, New York, NY, USA. ACM. ISBN 1-59593-079-5.

Dolev, S., Amos, I., y Moran, S. (1995). Analyzing expected time by scheduler-luck games. *IEEE Trans. Softw. Eng.*, **21**(5): 429–439.

Dyer, M. y Frieze, A. (2001). Randomly colouring graphs with lower bounds on girth and maximum degree. En: *FOCS '01: Proceedings of the 42nd IEEE symposium on Foundations of Computer Science*, página 579, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-1390-5.

Dyer, M. y Greenhill, C. (1998). A more rapidly mixing markov chain for graph colorings. En: *Proceedings of the Eighth International Conference on Random Structures and Algorithms*, páginas 285–317, New York, NY, USA. John Wiley & Sons, Inc.

Dyer, M., Goldberg, L. A., Greenhill, C., Istrate, G., y Jerrum, M. (2002). Convergence of the iterated prisoner's dilemma game. *Comb. Probab. Comput.*, **11**(2): 135–147.

Fribourg, L., Messika, S., y Picaronny, C. (2006). Coupling and self-stabilization. *Distrib. Comput.*, **18**(3): 221–232.

Gouda, M. G. y Multari, N. J. (1991). Stabilizing communication protocols. *IEEE Transactions on Computers*, **40**(4): 448–458.

Hayes, T. (2003). Randomly coloring graphs of girth at least five. En: *STOC '03: Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*, páginas 269–278, New York, NY, USA. ACM. ISBN 1-58113-674-9.

Herman, T. (1990). Probabilistic self-stabilization. *Information Processing Letters*, **35**(2): 63–67.

Huang, S.-T. (1993). Leader election in uniform rings. *ACM Transactions on Programming Languages and Systems*, **15**(3): 563–573.

Israeli, A. y Jalfon, M. (1990). Token management schemes and random walks yield self stabilizing mutual exclusion. *PODC '90: Proceedings of the eleventh annual ACM Symposium on Principles of Distributed Computing*, páginas 119–131.

Kemeny, J. G. y Snell, J. L. (1976). *Finite Markov Chains*. Springer-Verlag New York, Inc., Secaucus, NJ, USA. ISBN 0387901923.

Kersch, P., Szabo, R., Cheng, L., Jean, K., y Galis, A. (2008). Stochastic maintenance of overlays in structured p2p systems. *Computer Communications*, **31**(3): 603–619.

Kessels, J. L. W. (1988). An exercise in proving self-stabilization with a variant function. *Information Processing Letters*, **29**(1): 39–42.

Lindvall, T. (2002). *Lectures on the coupling method*. Dover publications.

Motteler, H. E. y Sidhu, D. (1993). Self-stabilization in iteration systems.

Motwani, R. y Raghavan, P. (1995). *Randomized algorithms*. Cambrigde University Press, New York, NY, USA. ISBN 0521474655.

Randall, D. (2003). Mixing. *Foundations of Computer Science, Annual IEEE Symposium on*, **0**: 4.

Schneider, M. (1993). Self-stabilization. *ACM Computing Surveys*, **25**(1): 45–67.

Sinclair, A. (1997). Convergence rates for monte carlo experiments. *Institute for Mathematics and Its Applications*, **102**: 1–18.

Theel, O. (2001). A new verification technique for self-stabilizing distributed algorithms based on variable structure systems and ljapunov theory. En: *HICSS '01: Proceedings of the 34th Annual Hawaii International Conference on System Sciences (HICSS-34)*, Vol. 9, página 9069, Los Alamitos, CA, USA. IEEE Computer Society. ISBN 0-7695-0981-9.

Theel, O. y Gartner, F. (1998). On proving the stability of distributed algorithms: self-stabilization vs. control theory. En: *Proceedings of SSCC '98, Durban, South Africa*, páginas 58–66.

Theel, O. y Gartner, F. (1999). An exercise in proving convergence through transfer functions. En: *ICDCS '99: Workshop on Self-stabilizing Systems*, páginas 41–47, Washington, DC, USA. IEEE Computer Society. ISBN 0-7695-0228-8.

Torrents, A. S., Guardiet, J. B. F., y Leyes, J. M. S. (2002). *Metodos Cuantitativos de Organizacion Industrial II*. Ediciones UPC, Barcelona, Espana. ISBN 8483017946.

Xu, Z., Hedetniemi, S. T., Goddard, W., y Srimani, P. K. (2003). A synchronous self-stabilizing minimal domination protocol in an arbitrary network graph. En: *Distributed Computing - IWDC 2003*, Vol. 2918 de *Lecture Notes in Computer Science*, páginas 26–32. Springer Berlin/Heidelberg. ISBN 978-3-540-20745-0.

Apéndice A

Tabla de Configuraciones

A continuación se muestra el Apéndice A, el cual contiene las configuraciones iniciales de los sistemas X y Y que representan a la misma cadena de Markov, además se muestra la distancia inicial e inmediatamente después de eso se pueden observar las posibles salidas para cada configuración inicial, la distancia de la siguiente configuración, la distancia promedio de la siguiente configuración y la probabilidad de cambio de las distancias inicial y de la siguiente configuración.

Distancia 4								Procesador	Salidas										
X				Y				que lee	X				Y				$\delta(X_{t+1},Y_{t+1})$	$\mathbf{E}\delta(X_{t+1},Y_{t+1})$	α
$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$		$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$			
0	0	0	0	1	1	1	1	P1	0	0	0	0	0	1	1	1	3	3	1
								P1	1	0	0	0	1	1	1	1	3		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	1	3		
0	0	0	1	1	1	1	0	P1	0	1	0	1	1	0	1	0	4	4	0
								P2	0	0	0	1	1	1	1	0	4		
0	0	1	0	1	1	0	1	P1	0	0	1	0	0	1	0	1	3	3	1
								P1	1	0	1	0	1	1	0	1	3		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	1	3		
0	0	1	1	1	1	0	0	P1	0	1	1	1	1	0	0	0	4	4	0
								P2	0	0	0	1	1	1	1	0	4		
0	1	0	0	1	0	1	1	P1	0	0	0	0	0	1	1	1	3	3	1
								P1	1	0	0	0	1	1	1	1	3		
								P2	0	1	0	0	1	0	1	0	3		
								P2	0	1	0	1	1	0	1	1	3		
0	1	0	1	1	0	1	0	P1	0	1	0	1	1	0	1	0	4	4	0
								P2	0	1	0	1	1	0	1	0	4		
0	1	1	0	1	0	0	1	P1	0	0	1	0	0	1	0	1	3	3	1
								P1	1	0	1	0	1	1	0	1	3		
								P2	0	1	0	0	1	0	1	0	3		
								P2	0	1	0	1	1	0	1	1	3		
0	1	1	1	1	0	0	0	P1	0	1	1	1	1	0	0	0	4	4	0
								P2	0	1	0	1	1	0	1	0	4		
																	Total	3.5	0.5

Distancia 3								Procesador	Salidas										
X				Y				que lee	X				Y						
$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$		$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$	$\delta(X_{t+1},Y_{t+1})$	$\mathbf{E}\delta(X_{t+1},Y_{t+1})$	α
0	0	0	0	1	1	1	0	P1	0	0	0	0	1	0	1	0	2	2.5	0.8
								P1	1	0	0	0	1	0	1	0	1		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	0	4		
0	0	0	0	1	1	0	1	P1	0	0	0	0	0	1	0	1	2	2.5	0.5
								P1	1	0	0	0	1	1	0	1	2		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	1	3		
0	0	0	0	1	0	1	1	P1	0	0	0	0	0	1	1	1	3	2.5	0.5
								P1	1	0	0	0	1	1	1	1	3		
								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	1	2		
0	0	0	0	0	1	1	1	P1	0	0	0	0	0	1	1	1	3	2.5	0.8
								P1	1	0	0	0	0	1	1	1	4		
								P2	0	0	0	0	0	1	0	1	2		
								P2	0	0	0	1	0	1	0	1	1		
0	0	0	1	1	1	1	1	P1	0	1	0	1	0	1	1	1	1	2.5	0.8
								P1	0	1	0	1	1	1	1	1	2		
								P2	0	0	0	1	1	1	1	0	4		
								P2	0	0	0	1	1	1	1	1	3		
0	0	0	1	1	1	0	0	P1	0	1	0	1	1	0	0	0	3	3.5	0.5
								P2	0	0	0	1	1	1	1	0	4		
0	0	0	1	1	0	1	0	P1	0	1	0	1	1	0	1	0	4	3.5	0.5
								P2	0	0	0	1	1	0	1	0	3		
0	0	0	1	0	1	1	0	P1	0	1	0	1	0	0	1	0	3	2.5	0.8
								P1	0	1	0	1	1	0	1	0	4		
								P2	0	0	0	1	0	1	0	0	2		
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	0	1	1	0	0	P1	0	0	1	0	1	0	0	0	2	2.5	0.8
								P1	1	0	1	0	1	0	0	0	1		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	0	4		
0	0	1	0	1	1	1	1	P1	0	0	1	0	0	1	1	1	2	2.5	0.5
								P1	1	0	1	0	1	1	1	1	2		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	1	3		

0	0	1	0	1	0	0	1	P1	0	0	1	0	0	1	0	1	3	2.5	0.5
								P1	1	0	1	0	1	1	0	1	3		
								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	1	2		
0	0	1	0	0	1	0	1	P1	0	0	1	0	0	1	0	1	3	2.5	0.8
								P1	1	0	1	0	0	1	0	1	4		
								P2	0	0	0	0	0	1	0	1	2		
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	1	1	1	0	1	P1	0	1	1	1	0	1	0	1	1	2.5	0.8
								P1	0	1	1	1	1	1	0	1	2		
								P2	0	0	0	1	1	1	1	0	4		
								P2	0	0	0	1	1	1	1	1	3		
0	0	1	1	1	1	1	0	P1	0	1	1	1	1	0	1	0	3	3.5	0.5
								P2	0	0	0	1	1	1	1	0	4		
0	0	1	1	1	0	0	0	P1	0	1	1	1	1	0	0	0	4	3.5	0.5
								P2	0	0	0	1	1	0	1	0	3		
0	0	1	1	0	1	0	0	P1	0	1	1	1	0	0	0	0	3	2.5	0.8
								P1	0	1	1	1	1	0	0	0	4		
								P2	0	0	0	1	0	1	0	0	2		
								P2	0	0	0	1	0	1	0	1	1		
0	1	0	0	1	0	1	0	P1	0	0	0	0	1	0	1	0	2	2.5	0.8
								P1	1	0	0	0	1	0	1	0	1		
								P2	0	1	0	0	1	0	1	0	3		
								P2	0	1	0	1	1	0	1	0	4		
0	1	0	0	1	0	0	1	P1	0	0	0	0	0	1	0	1	2	2.5	0.5
								P1	1	0	0	0	1	1	0	1	2		
								P2	0	1	0	0	1	0	1	0	3		
								P2	0	1	0	1	1	0	1	1	3		
0	1	0	0	1	1	1	1	P1	0	0	0	0	0	1	1	1	3	2.5	0.5
								P1	1	0	0	0	1	1	1	1	3		
								P2	0	1	0	0	1	1	1	0	2		
								P2	0	1	0	1	1	1	1	1	2		
0	1	0	1	1	0	1	1	P1	0	1	0	1	0	1	1	1	1	2.5	0.8
								P1	0	1	0	1	1	1	1	1	2		
								P2	0	1	0	1	1	0	1	0	4		
								P2	0	1	0	1	1	0	1	1	3		
0	1	0	1	1	0	0	0	P1	0	1	0	1	1	0	0	0	3	3.5	0.5
								P2	0	1	0	1	1	0	1	0	4		
0	1	0	1	1	1	1	0	P1	0	1	0	1	1	0	1	0	4	3.5	0.5

								P2	0	1	0	1	1	1	1	0	3		
0	1	1	0	1	0	0	0	P1	0	0	1	0	1	0	0	0	2	2.5	0.8
								P1	1	0	1	0	1	0	0	0	1		
								P2	0	1	0	0	1	0	1	0	3		
								P2	0	1	0	1	1	0	1	0	4		
0	1	1	0	1	0	1	1	P1	0	0	1	0	0	1	1	1	2	2.5	0.5
								P1	1	0	1	0	1	1	1	1	2		
								P2	0	1	0	0	1	0	1	0	3		
								P2	0	1	0	1	1	0	1	1	3		
0	1	1	0	1	1	0	1	P1	0	0	1	0	0	1	0	1	3	2.5	0.5
								P1	1	0	1	0	1	1	0	1	3		
								P2	0	1	0	0	1	1	1	0	2		
								P2	0	1	0	1	1	1	1	1	2		
0	1	1	1	1	0	0	1	P1	0	1	1	1	0	1	0	1	1	2.5	0.8
								P1	0	1	1	1	1	1	0	1	2		
								P2	0	1	0	1	1	0	1	0	4		
								P2	0	1	0	1	1	0	1	1	3		
0	1	1	1	1	0	1	0	P1	0	1	1	1	1	0	1	0	3	3.5	0.5
								P2	0	1	0	1	1	0	1	0	4		
0	1	1	1	1	1	0	0	P1	0	1	1	1	1	0	0	0	4	3.5	0.5
								P2	0	1	0	1	1	1	1	0	3		
1	0	0	0	1	1	1	1	P1	1	0	0	0	0	1	1	1	4	2.5	0.8
								P1	1	0	0	0	1	1	1	1	3		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	0	1	1	1	1	2		
1	0	0	1	1	1	1	0	P1	0	1	0	1	1	0	1	0	4	2.5	0.8
								P1	1	1	0	1	1	0	1	0	3		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	1	1	1	1	0	2		
1	0	1	0	1	1	0	1	P1	1	0	1	0	0	1	0	1	4	2.5	0.8
								P1	1	0	1	0	1	1	0	1	3		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	0	1	1	1	1	2		
1	0	1	1	1	1	0	0	P1	0	1	1	1	1	0	0	0	4	2.5	0.8
								P1	1	1	1	1	1	0	0	0	3		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	1	1	1	1	0	2		
																	Total	2.75	0.6

Distancia 2								Procesador	Salidas										
X				Y				que lee	X				Y						
$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$		$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$	$\delta(X_{t+1},Y_{t+1})$	$\mathbf{E}\delta(X_{t+1},Y_{t+1})$	α
0	0	0	0	0	0	1	1	P1	0	0	0	0	0	1	1	1	3	2	1
								P1	1	0	0	0	0	1	1	1	4		
								P2	0	0	0	0	0	0	0	1	1		
								P2	0	0	0	1	0	0	0	1	0		
0	0	0	0	0	1	0	1	P1	0	0	0	0	0	1	0	1	2	2	0.5
								P1	1	0	0	0	0	1	0	1	3		
								P2	0	0	0	0	0	1	0	1	2		
								P2	0	0	0	1	0	1	0	1	1		
0	0	0	0	1	0	0	1	P1	0	0	0	0	0	1	0	1	2	2	0
								P1	1	0	0	0	1	1	0	1	2		
								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	1	2		
0	0	0	0	1	1	0	0	P1	0	0	0	0	1	0	0	0	1	2	1
								P1	1	0	0	0	1	0	0	0	0		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	0	4		
0	0	0	0	1	0	1	0	P1	0	0	0	0	1	0	1	0	2	2	0.5
								P1	1	0	0	0	1	0	1	0	1		
								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	0	3		
0	0	0	0	0	1	1	0	P1	0	0	0	0	0	0	1	0	1	1	1
								P1	1	0	0	0	1	0	1	0	1		
								P2	0	0	0	0	0	1	0	0	1		
								P2	0	0	0	1	0	1	0	1	1		
0	0	0	1	1	0	0	0	P1	0	1	0	1	1	0	0	0	3	3	1
								P2	0	0	0	1	1	0	1	0	3		
0	0	0	1	0	1	0	0	P1	0	1	0	1	0	0	0	0	2	2	0.5
								P1	0	1	0	1	1	0	0	0	3		
								P2	0	0	0	1	0	1	0	0	2		
								P2	0	0	0	1	0	1	0	1	1		
0	0	0	1	0	0	1	0	P1	0	1	0	1	0	0	1	0	3	2	1
								P1	0	1	0	1	1	0	1	0	4		
								P2	0	0	0	1	0	0	0	0	1		
								P2	0	0	0	1	0	0	0	1	0		
0	0	0	1	1	1	0	1	P1	0	1	0	1	0	1	0	1	0	2	1
								P1	0	1	0	1	1	1	0	1	1		
								P2	0	0	0	1	1	1	1	0	4		

								P2	0	0	0	1	1	1	1	1	3		
0	0	0	1	1	0	1	1	P1	0	1	0	1	0	1	1	1	1	2	0.5
								P1	0	1	0	1	1	1	1	1	2		
								P2	0	0	0	1	1	0	1	0	3		
								P2	0	0	0	1	1	0	1	1	2		
0	0	0	1	0	1	1	1	P1	0	1	0	1	0	1	1	1	1	1	1
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	0	0	1	0	0	P1	0	0	1	0	0	0	0	0	1	1	1
								P1	1	0	1	0	1	0	0	0	1		
								P2	0	0	0	0	0	1	0	0	1		
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	0	1	0	0	0	P1	0	0	1	0	1	0	0	0	2	2	0.5
								P1	1	0	1	0	1	0	0	0	1		
								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	0	3		
0	0	1	0	1	1	1	0	P1	0	0	1	0	1	0	1	0	1	2	1
								P1	1	0	1	0	1	0	1	0	0		
								P2	0	0	0	0	1	1	1	0	3		
								P2	0	0	0	1	1	1	1	0	4		
0	0	1	0	1	0	1	1	P1	0	0	1	0	0	1	1	1	2	2	0
								P1	1	0	1	0	1	1	1	1	2		
								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	1	2		
0	0	1	0	0	1	1	1	P1	0	0	1	0	0	1	1	1	2	2	0.5
								P1	1	0	1	0	0	1	1	1	3		
								P2	0	0	0	0	0	1	0	1	2		
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	1	0	1	0	1	P1	0	1	1	1	0	1	0	1	1	1	1
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	1	1	0	0	1	P1	0	1	1	1	0	1	0	1	1	2	0.5
								P1	0	1	1	1	1	1	0	1	2		
								P2	0	0	0	1	1	0	1	0	3		
								P2	0	0	0	1	1	0	1	1	2		
0	0	1	1	1	0	1	0	P1	0	1	1	1	1	0	1	0	3	3	1
								P2	0	0	0	1	1	0	1	0	3		
0	0	1	1	0	1	1	0	P1	0	1	1	1	0	0	1	0	2	2	0.5
								P1	0	1	1	1	1	0	1	0	3		
								P2	0	0	0	1	0	1	0	0	2		
								P2	0	0	0	1	0	1	0	1	1		

0	0	1	1	1	1	1	1	P1	0	1	1	1	0	1	1	1	0	2	1
								P1	0	1	1	1	1	1	1	1	1		
								P2	0	0	0	1	1	1	1	0	4		
								P2	0	0	0	1	1	1	1	1	3		
0	1	0	0	0	1	1	1	P1	0	0	0	0	0	1	1	1	3	2	1
								P1	1	0	0	0	0	1	1	1	4		
								P2	0	1	0	0	0	1	0	1	1		
								P2	0	1	0	1	0	1	0	1	0		
0	1	0	0	1	1	1	0	P1	0	0	0	0	1	0	1	0	2	2	0.5
								P1	1	0	0	0	1	0	1	0	1		
								P2	0	1	0	0	1	1	1	0	2		
								P2	0	1	0	1	1	1	1	0	3		
0	1	0	0	1	1	0	1	P1	0	0	0	0	0	1	0	1	2	2	0
								P1	1	0	0	0	1	1	0	1	2		
								P2	0	1	0	0	1	1	1	0	2		
								P2	0	1	0	1	1	1	1	1	2		
0	1	0	0	1	0	0	0	P1	0	0	0	0	1	0	0	0	1	2	1
								P1	1	0	0	0	1	0	0	0	0		
								P2	0	1	0	0	1	0	1	0	3		
								P2	0	1	0	1	1	0	1	0	4		
0	1	0	1	1	1	1	1	P1	0	1	0	1	0	1	1	1	1	2	0.5
								P1	0	1	0	1	1	1	1	1	2		
								P2	0	1	0	1	1	1	1	0	3		
								P2	0	1	0	1	1	1	1	1	2		
0	1	0	1	1	0	0	1	P1	0	1	0	1	0	1	0	1	0	2	1
								P1	0	1	0	1	1	1	0	1	1		
								P2	0	1	0	1	1	0	1	0	4		
								P2	0	1	0	1	1	0	1	1	3		
0	1	0	1	1	1	0	0	P1	0	1	0	1	1	0	0	0	3	3	1
								P2	0	1	0	1	1	1	1	0	3		
0	1	0	1	0	1	1	0	P1	0	1	0	1	0	0	1	0	3	2	1
								P1	0	1	0	1	1	0	1	0	4		
								P2	0	1	0	1	0	1	0	0	1		
								P2	0	1	0	1	0	1	0	1	0		
0	1	1	0	1	1	1	1	P1	0	0	1	0	0	1	1	1	2	2	0
								P1	1	0	1	0	1	1	1	1	2		
								P2	0	1	0	0	1	1	1	0	2		
								P2	0	1	0	1	1	1	1	1	2		
0	1	1	0	1	1	0	0	P1	0	0	1	0	1	0	0	0	2	2	0.5

								P1	1	0	1	0	1	0	0	0	1			
								P2	0	1	0	0	1	1	1	0	2			
								P2	0	1	0	1	1	1	1	0	3			
0	1	1	0	1	0	1	0	P1	0	0	1	0	1	0	1	0	1	2	1	
								P1	1	0	1	0	1	0	1	0	0			
								P2	0	1	0	0	1	0	1	0	3			
								P2	0	1	0	1	1	0	1	0	4			
0	1	1	1	1	0	1	1	P1	0	1	1	1	0	1	1	1	0	2	1	
								P1	0	1	1	1	1	1	1	1	1			
								P2	0	1	0	1	1	0	1	0	4			
								P2	0	1	0	1	1	0	1	1	3			
0	1	1	1	1	1	0	1	P1	0	1	1	1	0	1	0	1	1	2	0.5	
								P1	0	1	1	1	1	1	0	1	2			
								P2	0	1	0	1	1	1	1	0	3			
								P2	0	1	0	1	1	1	1	1	2			
0	1	1	1	1	1	1	0	P1	0	1	1	1	1	0	1	0	3	3	1	
								P2	0	1	0	1	1	1	1	0	3			
1	0	0	0	1	0	1	1	P1	1	0	0	0	0	1	1	1	4	2	1	
								P1	1	0	0	0	1	1	1	1	3			
								P2	1	0	1	0	1	0	1	0	0			
								P2	1	0	1	0	1	0	1	1	1			
1	0	0	0	1	1	0	1	P1	1	0	0	0	0	1	0	1	3	2	0.5	
								P1	1	0	0	0	1	1	0	1	2			
								P2	1	0	1	0	1	1	1	0	1			
								P2	1	0	1	0	1	1	1	1	2			
1	0	0	0	1	1	1	0	P1	1	0	0	0	1	0	1	0	1	1	1	
								P2	1	0	1	0	1	1	1	0	1			
1	0	0	1	1	1	1	1	P1	0	1	0	1	0	1	1	1	1	1	1	
								P1	1	1	0	1	1	1	1	1	1			
								P2	1	0	1	0	1	1	1	0	1			
								P2	1	0	1	1	1	1	1	1	1			
1	0	0	1	1	1	0	0	P1	0	1	0	1	1	0	0	0	3	2	0.5	
								P1	1	1	0	1	1	0	0	0	2			
								P2	1	0	1	0	1	1	1	0	1			
								P2	1	0	1	1	1	1	1	0	2			
1	0	0	1	1	0	1	0	P1	0	1	0	1	1	0	1	0	4	2	1	
								P1	1	1	0	1	1	0	1	0	3			
								P2	1	0	1	0	1	0	1	0	0			
								P2	1	0	1	1	1	0	1	0	1			

1	0	1	0	1	1	1	1	P1	1	0	1	0	0	1	1	1	3	2	0.5
								P1	1	0	1	0	1	1	1	1	2		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	0	1	1	1	1	2		
1	0	1	0	1	1	0	0	P1	1	0	1	0	1	0	0	0	1	1	1
								P2	1	0	1	0	1	1	1	0	1		
1	0	1	1	1	1	0	1	P1	0	1	1	1	0	1	0	1	1	1	1
								P1	1	1	1	1	1	1	0	1	1		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	1	1	1	1	1	1		
1	0	1	1	1	1	1	0	P1	0	1	1	1	1	0	1	0	3	2	0.5
								P1	1	1	1	1	1	0	1	0	2		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	1	1	1	1	0	2		
1	1	0	0	1	1	1	1	P1	1	0	0	0	0	1	1	1	4	2	1
								P1	1	0	0	0	1	1	1	1	3		
								P2	1	1	1	0	1	1	1	0	0		
								P2	1	1	1	0	1	1	1	1	1		
1	1	0	1	1	1	1	0	P1	0	1	0	1	1	0	1	0	4	2	
								P1	1	1	0	1	1	0	1	0	3		1
								P2	1	1	1	0	1	1	1	0	0		
								P2	1	1	1	1	1	1	1	0	1		
																	Total	1.91	0.8

Distancia 1								Procesador	Salidas										
X				Y				que lee	X				Y						
$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$		$x_{1,1}$	$x_{1,2}$	$x_{2,1}$	$x_{2,2}$	$y_{1,1}$	$y_{1,2}$	$y_{2,1}$	$y_{2,2}$	$\delta(X_{t+1},Y_{t+1})$	$\mathbf{E}\delta(X_{t+1},Y_{t+1})$	α
0	0	0	0	0	0	0	1	P1	0	0	0	0	0	1	0	1	2	1.5	0.8
								P1	1	0	0	0	0	1	0	1	3		
								P2	0	0	0	0	0	0	0	1	1		
								P2	0	0	0	1	0	0	0	1	0		
0	0	0	0	0	0	1	0	P1	0	0	0	0	0	0	1	0	1	0.5	0.5
								P1	1	0	0	0	1	0	1	0	1		
								P2	0	0	0	0	0	0	0	0	0		
								P2	0	0	0	1	0	0	0	1	0		
0	0	0	0	0	1	0	0	P1	0	0	0	0	0	0	0	0	0	0.5	0.5
								P1	1	0	0	0	1	0	0	0	0		
								P2	0	0	0	0	0	1	0	0	1		
								P2	0	0	0	1	0	1	0	1	1		
0	0	0	0	1	0	0	0	P1	0	0	0	0	1	0	0	0	1	1.5	0.8
								P1	1	0	0	0	1	0	0	0	0		

								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	0	3		
0	0	0	1	0	0	1	1	P1	0	1	0	1	0	1	1	1	1	0.5	0.5
								P2	0	0	0	1	0	0	0	1	0		
0	0	0	1	0	1	0	1	P1	0	1	0	1	0	1	0	1	0	0.5	0.5
								P2	0	0	0	1	0	1	0	1	1		
0	0	0	1	1	0	0	1	P1	0	1	0	1	0	1	0	1	0	1.5	0.8
								P1	0	1	0	1	1	1	0	1	1		
								P2	0	0	0	1	1	0	1	0	3		
								P2	0	0	0	1	1	0	1	1	2		
0	0	1	0	0	0	1	1	P1	0	0	1	0	0	1	1	1	2	1.5	0.8
								P1	1	0	1	0	0	1	1	1	3		
								P2	0	0	0	0	0	0	0	1	1		
								P2	0	0	0	1	0	0	0	1	0		
0	0	1	0	0	1	1	0	P1	0	0	1	0	0	0	1	0	0	0.5	0.5
								P1	1	0	1	0	1	0	1	0	0		
								P2	0	0	0	0	0	1	0	0	1		
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	0	1	0	1	0	P1	0	0	1	0	1	0	1	0	1	1.5	0.8
								P1	1	0	1	0	1	0	1	0	0		
								P2	0	0	0	0	1	0	1	0	2		
								P2	0	0	0	1	1	0	1	0	3		
0	0	1	1	0	1	1	1	P1	0	1	1	1	0	1	1	1	0	0.5	0.5
								P2	0	0	0	1	0	1	0	1	1		
0	0	1	1	1	0	1	1	P1	0	1	1	1	0	1	1	1	0	1.5	0.8
								P1	0	1	1	1	1	1	1	1	1		
								P2	0	0	0	1	1	0	1	0	3		
								P2	0	0	0	1	1	0	1	1	2		
0	1	0	0	0	1	0	1	P1	0	0	0	0	0	1	0	1	2	1.5	0.8
								P1	1	0	0	0	0	1	0	1	3		
								P2	0	1	0	0	0	1	0	1	1		
								P2	0	1	0	1	0	1	0	1	0		
0	1	0	0	0	1	1	0	P1	0	0	0	0	0	0	1	0	1	0.5	0.5
								P1	1	0	0	0	1	0	1	0	1		
								P2	0	1	0	0	0	1	0	0	0		
								P2	0	1	0	1	0	1	0	1	0		
0	1	0	0	1	1	0	0	P1	0	0	0	0	1	0	0	0	1	1.5	0.8
								P1	1	0	0	0	1	0	0	0	0		
								P2	0	1	0	0	1	1	1	0	2		

								P2	0	1	0	1	1	1	1	0	3		
0	1	0	1	0	1	1	1	P1	0	1	0	1	0	1	1	1	1	0.5	0.5
								P2	0	1	0	1	0	1	0	1	0		
0	1	0	1	1	1	0	1	P1	0	1	0	1	0	1	0	1	0	1.5	0.8
								P1	0	1	0	1	1	1	0	1	1		
								P2	0	1	0	1	1	1	1	0	3		
								P2	0	1	0	1	1	1	1	1	2		
0	1	1	0	0	1	1	1	P1	0	0	1	0	0	1	1	1	2	1.5	0.8
								P1	1	0	1	0	0	1	1	1	3		
								P2	0	1	0	0	0	1	0	1	1		
								P2	0	1	0	1	0	1	0	1	0		
0	1	1	0	1	1	1	0	P1	0	0	1	0	1	0	1	0	1	1.5	0.8
								P1	1	0	1	0	1	0	1	0	0		
								P2	0	1	0	0	1	1	1	0	2		
								P2	0	1	0	1	1	1	1	0	3		
0	1	1	1	1	1	1	1	P1	0	1	1	1	0	1	1	1	0	1.5	0.8
								P1	0	1	1	1	1	1	1	1	1		
								P2	0	1	0	1	1	1	1	0	3		
								P2	0	1	0	1	1	1	1	1	2		
1	0	0	0	1	0	0	1	P1	1	0	0	0	0	1	0	1	3	1.5	0.8
								P1	1	0	0	0	1	1	0	1	2		
								P2	1	0	1	0	1	0	1	0	0		
								P2	1	0	1	0	1	0	1	1	1		
1	0	0	0	1	0	1	0	P1	1	0	0	0	1	0	1	0	1	0.5	0.5
								P2	1	0	1	0	1	0	1	0	0		
1	0	0	0	1	1	0	0	P1	1	0	0	0	1	0	0	0	0	0.5	0.5
								P2	1	0	1	0	1	1	1	0	1		
1	0	0	1	1	0	1	1	P1	0	1	0	1	0	1	1	1	1	0.5	0.5
								P1	1	1	0	1	1	1	1	1	1		
								P2	1	0	1	0	1	0	1	0	0		
								P2	1	0	1	1	1	0	1	1	0		
1	0	0	1	1	1	0	1	P1	0	1	0	1	0	1	0	1	0	0.5	0.5
								P1	1	1	0	1	1	1	0	1	0		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	1	1	1	1	1	1		
1	0	1	0	1	0	1	1	P1	1	0	1	0	0	1	1	1	3	1.5	0.8
								P1	1	0	1	0	1	1	1	1	2		
								P2	1	0	1	0	1	0	1	0	0		
									1	0	1	0	1	0	1	1	1		

1	0	1	0	1	1	1	0	P1	1	0	1	0	1	0	1	0	0	0.5	0.5
								P2	1	0	1	0	1	1	1	0	1		
1	0	1	1	1	1	1	1	P1	0	1	1	1	0	1	1	1	0	0.5	0.5
								P1	1	1	1	1	1	1	1	1	0		
								P2	1	0	1	0	1	1	1	0	1		
								P2	1	0	1	1	1	1	1	1	1		
1	1	0	0	1	1	0	1	P1	1	0	0	0	0	1	0	1	3	1.5	0.8
								P1	1	0	0	0	1	1	0	1	2		
								P2	1	1	1	0	1	1	1	0	0		
								P2	1	1	1	0	1	1	1	1	1		
1	1	0	0	1	1	1	0	P1	1	0	0	0	1	0	1	0	1	0.5	0.5
								P2	1	1	1	0	1	1	1	0	0		
1	1	0	1	1	1	1	1	P1	0	1	0	1	0	1	1	1	1	0.5	0.5
								P1	1	1	0	1	1	1	1	1	1		
								P2	1	1	1	0	1	1	1	0	0		
								P2	1	1	1	1	1	1	1	1	0		
1	1	1	0	1	1	1	1	P1	1	0	1	0	0	1	1	1	3	1.5	0.8
								P1	1	0	1	0	1	1	1	1	2		
								P2	1	1	1	0	1	1	1	0	0		
								P2	1	1	1	0	1	1	1	1	1		
																	Total	1	0.6
																	Decrece	***Cambio***	
								Total de totales									***0.128***	***0.625***	