65

UNIVERSIDAD NACIONAL AUTONOMA
DE MEXICO

FACULTAD DE CIENCIAS

ANALISIS DE GRAFICAS MARCADAS ESTOCASTICAS MEDIANTE TECNICAS DE REDUCCION DE VARLANZA

T E S I S

QUE PARA OBTENER EL TÍTULO DE

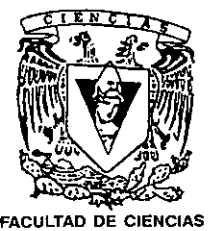
A C T U A R I O

P R E S E N T A

SERAFIN MARTINEZ JARAMILLO

DIRECTOR DE TESIS

SERGIO RAJSBAUM GORODEZKY



FACULTAD DE CIENCIAS
UNAM

UNIVERSIDAD NACIONAL
AVENOMA DE
MEXICO

M. EN C. ELENA DE OTEYZA DE OTEYZA
Jefa de la División de Estudios Profesionales de la
Facultad de Ciencias
Presente

Comunicamos a usted que hemos revisado el trabajo de Tesis:

"Análisis de Gráficas Marcadas Estocásticas mediante Técnicas de Reducción de Varianza"

realizado por Serafín Martínez Jaramillo

con número de cuenta 8638731-4 , pasante de la carrera de Actuaría

Dicho trabajo cuenta con nuestro voto aprobatorio.

Atentamente

Director de Tesis
Propietario Dr. Sergio Rajsbaum Gorodezky

Propietario Dra. Hanna Oktaba

Propietario M. en I. María de Luz Gasca Soto

Suplente M. en C. Ma. Guadalupe Elena Ibarguengoitia González

Suplente M. en I. Víctor Rafael Pérez Pérez

Consejo Departamental de Matemáticas

M. en C. José Antonio Flores Díaz

Unas palabras para la ocasión:
¿en serio?
sí, en serio
ha pasado mucho tiempo ¿no?
sí, ha pasado mucho tiempo
¿y la espera?
terminó
¿entonces?
entonces, palabras.

A mis padres y a Bili

Agradecimientos

Para empezar una lista se debería ordenar de alguna forma los elementos, aunque el tal orden no lo sea tanto. Con gusto empezaré a declarar mi agradecimiento y reconocimiento a las siguientes personas, una vez aclarado que no me es posible poner todos los nombres de un solo golpe y que no puedo poner en una lista a todas las maravillosas personas con las que he compartido un poco de vida-espacio-tiempo.

El mayor agradecimiento lo merecen mis padres, Serafín y Juana, por esa gran batalla que libraron por sus hijos. Gracias Bilianche por el enorme apoyo, la dulzura y alegía que me das. Gracias a mis hermanos por ser como son y por esas criaturas espléndidas que les acompañan. Gracias a las familias Kabadjovi por esos diálogos y ese regocijo que transmiten. Gracias a ese gran grupo de hermanos que he tenido la fortuna de conocer a través de mi vida: gracias Miguel Angel, gracias Miguel Angel, gracias Luis, gracias Ramón, gracias Javier y gracias Jesús. Un agradecimiento muy grande para Ara y Mari, seres maravillosos y generosos que, mi buena fortuna me permitió conocer.

Gracias a la tremenda FEFA, porque estudiar y ser miembro de tan reconocida federación, resultó doblemente placentero. Un reconocimiento a todos los profesores que me ayudaron a entender y también a los que no lo hicieron. A la UNAM por lo que es y representa. Gracias por la inagotable paciencia que mostraron siempre Sergio y Hanna; y también gracias a los generosos sinodales que permiten se realice el milagroso acto de agradecer.

Gracias en especial a Bili, Luis y U por sus valiosos comentarios sobre el trabajo. Gracias a mis compañeros de trabajo, de los que he aprendido mucho y que todos los días me manifiestan su cortesía y solidaridad. Gracias a la Dra. Catherine McGeoch y al Dr. Jöern Jannek por la información y la ayuda que me proporcionaron.

Contenido

Lista de Figuras

Descripción del trabajo

En las ciencias de la computación existen numerosos campos de interés, dos cuya importancia es primordial son: el análisis de algoritmos y los sistemas distribuidos.

El objetivo de este trabajo es presentar como llevar a cabo un análisis experimental mediante la simulación de un modelo de gran utilidad en el estudio de los sistemas distribuidos: las gráficas marcadas estocásticas.

El análisis que se propone es mediante unas técnicas que son ampliamente usadas en el área de la simulación digital. Estas técnicas son conocidas como *Técnicas de reducción de varianza.*

Como resultado del trabajo se obtuvo una serie de propuestas sobre como simular dos casos particulares de gráficas marcadas estocásticas: el ciclo dirigido y la gráfica completa. Se eligieron estos casos debido a que existen resultados teóricos exactos de su comportamiento asintótico.

Debido a la complejidad que representa elaborar un programa de simulación para este modelo, se revisaron algunas herramientas de simulación de redes de Petri. La decisión sobre que herramientas se deberían usar se basó principalmente en la posibilidad de controlar la generación de los números aleatorios que conducirían la simulación. Para lograr la generación de números aleatorios con distribuciones de acuerdo con los modelos de estas gráficas marcadas en particular, se revisará muy brevemente el método de la transformación inversa para la generación de las variables aleatorias de entrada.

En el Capítulo 1 se presenta brevemente dos temas: los sistemas distribuidos y el análisis experimental de algoritmos. Este capítulo está tomado principalmente de algunos trabajos de Catherine McGeoch sobre métodos experimentales de análisis de algoritmos [McGeoch 1986], [McGeoch 1992], [McGeoch 1995].

En la primera parte del Capítulo 2 se presenta el modelo de gráficas marcadas y se enuncian algunos resultados teóricos. Los teoremas enunciados en esta primera parte pertenecen a un trabajo en el que además se presentan algunos resultados sobre relaciones entre marcajes y una solución al problema de marcaje máximo [Even 1971].

En la siguiente parte, se hace referencia al modelo que será utilizado para realizar el análisis del desempeño de las gráficas marcadas estocásticas. La parte de los resultados teóricos para el desempeño de la red es tomada de [Rajsbaum, Sidi 1994], donde se presentan los resultados del análisis hecho para el cálculo de la *tasa de la red*, para los casos del ciclo dirigido y de la gráfica completa, suponiendo tiempos de procesamiento y de transmisión aleatorios.

En el primera parte del Capítulo 3 se presentan cuatro técnicas de reducción de varianza, se analizan teóricamente y se presentan algunos ejemplos de cada una. La presentación del desarrollo teórico de las técnicas de reducción de varianza está tomado en su mayoría del libro de Law y Kelton [Law 1991]. En el trabajo de Catherine Mc Geoch [McGeoch 1992] se encuentran descritas brevemente otras técnicas de reducción de varianza que podrían ser de utilidad.

En el Capítulo 4 se lleva a cabo la descripción de las herramientas de simulación de las redes de Petri, que se revisaron con el objetivo de llevar a cabo la simulación de las gráficas marcadas estocásticas. La información presentada en este capítulo se tomó principalmente de la documentación de las herramientas. En la segunda parte del Capítulo 4 se evaluan la posibilidad de usar este análisis experimental para la estimación del desempeño de la red , en los casos del ciclo dirigido y la gráfica completa.

En el Capítulo 5 se presenta las conclusiones sobre el trabajo y posibles líneas de investigación.

Capítulo 1

Conceptos preliminares

Debido a que el objetivo de este trabajo es aplicar algunas técnicas de simulación para el análisis de un modelo que facilita el estudio de algoritmos y mecanismos de cómputo distribuidos, se describirá brevemente qué son los algoritmos y cómo se les puede clasificar. Además se presentará algunos conceptos básicos de los sistemas distribuidos.

1.1 Algoritmos

Un algoritmo es un método detallado para solucionar un problema. La búsqueda y diseño de algoritmos eficientes es una tarea muy importante en las ciencias de la computación.

Un sistema al que se hace referencia frecuentemente cuando se desea clasificar y caracterizar la eficiencia de los algoritmos, es la notación asintótica. Aunque se han aplicado estudios experimentales a la investigación de la ejecución de los algoritmos, se le ha dado poca importancia al método experimental. Esto es desafortunado, puesto que este tipo de análisis puede proporcionar ideas nuevas acerca del problema y, además, proporciona una manera de mejorar mucho los resultados obtenidos mediante una experimentación.

Cuando se realiza el análisis teórico de un algoritmo, se trata de encontrar cotas expresadas por medio de funciones que relacionen el tamaño de la entrada con alguna medida combinatoria del costo de la ejecución (independiente de la máquina).

Las notaciones O, Ω y Θ son muy útiles en el análisis de algoritmos y a continuación se procederá a definirlas.

Denótese con N el conjunto de todos los enteros no negativos. En la teoría de complejidad se está interesado en funciones de N a N, tales como n^2, 2^n y $n^3 - 2n + 5$. Se usa n como el argumento estándar de tales funciones. Aunque se denoten las funciones usadas para definir la complejidad de un algoritmo en una forma que admita valores no enteros o negativos, tales como $\sqrt{n}$, $\log n$ y $\sqrt{n} - 4\log^2 n$ siempre se pensarán como enteros no negativos. Esto es, cualquier función denotada como $f(n)$ realmente significa $max\{\lceil f(n) \rceil, 0\}$.

Sean f y g funciones de N a N. Se dice que $f(n) = O(g(n))$ ($f(n)$ es O de $g(n)$ o f es del orden de g) si existen enteros positivos c y n_0 tales que para todo $n \geq n_0$, $f(n) \leq cg(n)$. El significado informal de $f(n) = O(g(n))$ es que f crece tanto como g o menos. Se dice que $f(n) = \Omega(g(n))$ si sucede lo opuesto, esto es, si $g(n) = O(f(n))$. Finalmente $f(n) = \Theta(g(n))$ significa que f y g tienen precisamente la misma tasa de crecimiento.

Por ejemplo, es posible afirmar que un algoritmo A requiere no más de $O(n^2)$ comparaciones para ordenar una lista de n elementos. Si esta afirmación se cumple para todas las listas posibles de tamaño n, entonces tenemos una cota para el peor caso. Alternativamente, se podría establecer una distribución de probabilidad sobre el conjunto de posibles entradas de tamaño n y demostrar que el número esperado de comparaciones no es más de $O(n \log n)$; esta es una cota del caso promedio del algoritmo.

Estas cotas analíticas proporcionan un sistema de clasificación que ha sido útil para predecir el tiempo de ejecución del algoritmo en la práctica, sin importar qué arquitectura de computadora o lenguaje de programación se usa. Es un hecho que cualquier tasa polinomial será sobrepasada en algún momento por cualquier tasa exponencial. Más aún, rara vez aparecen en la práctica tasas extremas de crecimiento, tales como n^{80} y $2^{n/100}$. Los algoritmos polinomiales típicamente tienen exponentes pequeños y constantes multiplicativas razonables, mientras que los algoritmos exponenciales son en verdad intratables.

Sin embargo, existen cómputos eficientes que no son polinomiales y cómputos polinomiales que no son eficientes en la práctica. De hecho existe un problema importante que proporciona ejemplos para ambos casos: la programación lineal. Un algoritmo ampliamente usado para este problema básico, el *método simplex*, se sabe que tiene complejidad exponencial en el peor caso, pero tiene un desempeño

consistentemente espléndido en la práctica; de hecho, su desempeño esperado es probablemente polinomial. En contraste, el primer algoritmo polinomial descubierto para este problema, el *algoritmo del elipsoide*, parece demasiado lento en la práctica.

La notación $O(f(n))$ puede tiene un defecto: solo examina cómo se desempeña el algoritmo en las situaciones menos favorables. El desempeño exponencial del peor caso de un algoritmo puede deberse a una fracción estadísticamente insignificante de las entradas, aunque el algoritmo pueda desempeñarse satisfactoriamente *en promedio*. Seguramente un análisis del desempeño del caso promedio de un algoritmo sería más informativo que uno del peor caso. Desafortunadamente, en la práctica difícilmente se conoce la distribución de la entrada de un problema (esto es, la probabilidad con la que cada posible instancia pueda ser una entrada al algoritmo) y así un análisis verdaderamente informativo del caso promedio es imposible. Además, si se desea solucionar únicamente una instancia particular, y el algoritmo tarda mucho en dicha instancia, saber que se está en una excepción estadísticamente insignificante es de poca ayuda.

Cuando los métodos analíticos fallan, pueden utilizarse los experimentos computacionales para estudiar el caso promedio de la ejecución de un algoritmo, usando instancias del problema generadas aleatoriamente. El proceso parece bueno: implementar el algoritmo, generar las instancias, y usar métodos estadísticos apropiados para analizar los resultados. Pero no hay ninguna garantía de éxito. En la práctica se ha mostrado que puede ser muy difícil obtener estimaciones seguras de formas funcionales a partir de los datos experimentales. Además, los estudios basados en la experimentación pueden producir de igual manera resultados inconclusos o engañosos.

La elaboración del experimento correcto puede proporcionar buenas ideas sobre el problema e igualmente estimular nuevos resultados. Pero , ¿de qué manera se puede diseñar el experimento correcto, aquel que dará resultados libres de errores, precisos y no ambiguos acerca de la ejecución del algoritmo?

Una buena técnica experimental para llevar a cabo este análisis es la simulación. El algoritmo representa el modelo de simulación del cual se desea cuantificar μ el valor esperado de alguna medida de ejecución, en término de algunos parámetros, los cuales describen las propiedades de distribución de la entrada.

Para poder lograr que en el desarrollo de nuestro estudio de simulación se obtenga una buena calidad en los resultados, es necesario atender dos factores:

- El primero es la varianza observada. Una meta común de los estudios expe-

rimentales es estimar la media de alguna cantidad promediando sobre varios ensayos aleatorios. La confiabilidad de esta estimación depende de la varianza en los resultados. Si fuera posible de alguna manera reducir la varianza de una variable aleatoria de salida *sin perturbar su esperanza*, se podría obtener mayor precisión o, alternativamente, lograr con menos ensayos la precisión deseada. El uso apropiado de alguna de estas *técnicas de reducción de varianza* (*Variance-Reduction Techniques*), puede hacer la diferencia entre un proyecto de simulación inoperante y uno útil.

- El segundo factor es la eficiencia de la simulación, qué tan rápido calcula los resultados un programa de simulación. La eficiencia de dicho programa puede influir en las decisiones sobre el tamaño del problema más grande, el número de ensayos, y el número y rango de los parámetros considerados. También la eficiencia de la simulación puede reducir la varianza permitiendo más ensayos por unidad de tiempo, ya que la varianza es inversamente proporcional al número de puntos muestrales independientes. Recíprocamente, una reducción en la varianza puede proporcionar una aceleración en la simulación cuando se necesitan pocos ensayos para obtener la precisión deseada.

En un estudio realizado por Catherine McGeoch [McGeoch 1992], la aplicación de *técnicas de reducción de varianza* produce una disminución en la varianza de un factor de 1000 en un caso, dando un gran mejoramiento en la precisión de los resultados experimentales. Además, en el mismo estudio la complejidad del programa de simulación mejora de $\Theta(mn/Hn)$ a $\Theta(m + n \log n)$ (donde m es típicamente más grande que n), dando un programa de simulación mucho más rápido y por lo tanto más datos por unidad de tiempo de cómputo.

En otro trabajo de Catherine McGeoch [McGeoch 1995], se proporcionan algunas sugerencias para mejorar la experimentación sobre algoritmos.

En dicho trabajo se hace una distinción entre una experimentación para comparar programas y una experimentación para estudiar propiedades de algoritmos.

La diferencia radica principalmente en que en un estudio de simulación se tiene el control completo sobre el ambiente experimental, es decir, se puede diseñar experimentos para probar determinadas hipótesis o se puede cambiar las medidas de desempeño. En contraste, cuando se estudian aplicaciones se debe tener cuidado de que las pruebas no afecten el resultado y se sacrifica cierto grado de control sobre el experimento.

A continuación se enumeran y describen brevemente los pasos que se deben seguir para lograr un buen trabajo experimental:

- Planeación de los experimentos. Se realizan experimentos computacionales cuando algunas preguntas no pueden ser respondidas directamente. Estas preguntas abiertas sugieren cuales medidas de desempeño son interesantes y que parámetros experimentales deben incorporarse, sin embargo otros factores también son clave en la etapa de planeación:
  - *¿Qué medir?.* Las medidas de desempeño pueden ser sugeridas por las preguntas abiertas que motivaron la investigación.
  - *Modos de experimentación.* Los experimentos computacionales pueden desarrollarse para responder varios tipos de preguntas acerca del desempeño.
    * En un estudio de dependencia la meta es descubrir relaciones funcionales entre los parámetros y las medidas de desempeño.
    * Un estudio sobre la robustez observa las propiedades de la distribución observadas sobre varios ensayos aleatorios, en lugar de observar sólo el comportamiento promedio.
    * Los estudios de sondeo involucran analizar el programa de simulación y estudiar los componentes de las medidas de desempeño.
- El estudio piloto. Un buen primer paso es implementar un programa de simulación básico y un generador de entradas basado en suposiciones simples acerca del modelo. Este *programa piloto*, desarrollado independientemente del programa de simulación completo tiene varios usos.
  - La implementación piloto puede usarse para una exploración preliminar y para determinar el alcance y las dimensiones del proyecto de simulación completo.
  - Un programa piloto pudede usarse como implementación de respaldo para validar los resultados experimentales. El programa piloto puede ser portable a otros ambientes de cómputo y ejecutado con generadores de números aleatorios alternativos para evaluar el impacto de los factores ambientales sobre los resultados.

- Desarrollo y ejecución de los experimentos. Una vez que el alcance y la dimensión del proyecto son determinados por el estudio piloto, el siguiente paso es desarrollar un programa de simulación y probar el ambiente. Después de esto, las tareas de la ejecución de los experimentos y el análisis de los datos puede empezar.
  - Implementación del programa de simulación. Muchas cosas dependen de la eficiencia del programa de simulación. Las conclusiones que se obtengan en un estudio pueden depender mucho del tamaño del problema más grande. Además de lo anterior, el número de puntos de evaluación y el número de ensayos por cada uno de estos puntos están restringidos por la velocidad del programa de simulación.
  - Bases para el diseño experimental. ¿Cuáles serán los puntos de evaluación en un estudio de dependencia? Los métodos de diseño experimental proporcionan algunas reglas para la elección de los puntos de evaluación:
    * Si el diseño produce muchos puntos de evaluación, entonces es necesario eliminar algunos parámetros. Un desarrollo cuidadoso de de un diseño incompleto puede soportar un análisis estadístico riguroso.
    * Comprender parcialmente las relaciones funcionales entre los parámetros y el desempeño puede ayudar en el diseño experimental.
    * Es posible detectar más fácilmente comportamientos límite y cíclicos con un diseño que usa puntos de evaluación con espacios uniformes.
    * Una gran varianza en los datos hace difícil obtener estimaciones confiables de medias. La varianza puede reducirse haciendo más ensayos.
    * La experiencia sugiere que la calidad del experimento se mejora tomando los tamaños de problemas más grandes.
    * El análisis de los datos puede empezar con una transformación de los valores de los datos, frecuentemente tomando logaritmos. La transformación puede aplicarse a parámetros que tienen un crecimiento teórico no acotado o que acasionan grandes cambios en las medidas de desempeño.
- Estadísticas y análisis de los datos. Varias propiedades de los problemas algorítmicos sugieren que los métodos de análisis y exploración gráfica son muy útiles.

Las técnicas de graficación y visualización proporcionan los únicos medios prácticos para la manipulación de grandes cantidades de datos. También, se está interesado en desarrollar explicaciones de las observaciones y las técnicas gráficas y visuales permiten ver directamente los patrones producidos.

Los métodos de análisis exploratorio de datos son muy recomendados para descubrir la forma de la función que relaciona los parámetros de entrada con las medidas de desempeño.

Por el momento se considera terminada la presentación de esta metodología propuesta para el análisis experimental de algoritmos. Sin embargo, antes de terminar se debe mencionar que existen varios artículos sobre este tema [McGeoch, Moret 1999], [McGeoch 1997] y [Johnson 1996] e incluso Catherine McGeoch proporciona una bibliografía sobre la experimentación de algoritmos [McGeoch 1999]. En particular el artículo de Bernard M. E. Moret [Moret] es muy útil para evitar algunos de los errores que se cometen comúnmente en esta clase de trabajos.

1.2 Sistemas distribuidos

Un sistema es una unidad compleja formada de muchas partes organizadas para lograr un propósito común. Un sistema distribuido es un sistema el cual está esparcido sobre un área.

En el cómputo distribuido es posible tener múltiples nodos conectados; estos nodos pueden ser computadoras o pueden consistir de una gran variedad de elementos incluyendo sensores, dispositivos periféricos o microprocesadores trabajando conjuntamente con los sistemas de cómputo.

Las computadoras u otros elementos en un sistema de cómputo distribuido típicamente estarán dispersos aunque también pueden estar agrupados en distancias muy próximas. El grado de separación geográfica no es importante siempre y cuando los controles básicos instituídos en dicho sistema soporten la separación sin importar la distancia geográfica.

En un sistema de cómputo distribuido los componentes son entidades autónomas que exhiben control coordinado a través de un mecanismo global. Todos los sistemas distribuidos poseen capacidades de procesamiento global y local.

Sin embargo, el diseño y desarrollo de sistemas de cómputo distribuido son tareas complejas. La división de las aplicaciones entre múltiples sistemas de cómputo es una tarea difícil y laboriosa que demanda un análisis cuidadoso.

Es posible dividir el estudio de los sistemas distribuidos en dos áreas: fundamentos y sistemas de cómputo. En el área de fundamentos se estudian algoritmos y técnicas que permitan el desempeño del sistema de una manera eficiente y confiable. Por otro lado, en el área de sistemas de cómputo el objetivo es diseñar e implementar soluciones prácticas a los problemas de los sistemas distribuidos.

Para la representación de los sistemas distribuidos existe una gran variedad de modelos matemáticos; estos modelos deben describir tres aspectos fundamentales del sistema:

- La topología de la red.
- La sincronía.
- La tolerancia a fallas.

Existe una conexión muy estrecha con la investigación en teoría de gráficas y los algoritmos distribuidos, debido a que un sistema distribuido usualmente se modela por una gráfica que refleja su estructura de conexión. Los nodos de la gráfica son los procesos, y las aristas son los canales de comunicación. Algunas definiciones básicas y teoremas de teoría de gráficas se aplican también a los sistemas distribuidos.

La existencia de un canal de comunicación particular tiene la ventaja de que los procesos conectados pueden comunicarse directamente; pero tiene la desventaja de que usa recursos de los mismos. Así, el problema de escoger la topología de la red no es trivial. Es importante que la estructura en su totalidad sea conexa. Esto significa que cualquier proceso puede alcanzar a cualquier otro a través de una trayectoria de procesos intermedios.

En el diseño de redes tolerantes a fallas se considera la posibilidad de que un canal o proceso arbitrario falle, esto es, se remueva del sistema. Después de removerlo, el sistema de los procesos que quedan debe seguir siendo conexo y funcionar normalmente. A una subred del sistema con pocas líneas de comunicación pero con una buena confiabilidad, se le conoce como *certificado falso de conexidad*. Existen otros problemas relacionados con la topología de la red, por ejemplo encontrar un *árbol de expansión mínimo*, es decir, una subred que contenga a todos los procesos y tenga el mínimo número de aristas.

En los problemas de sincronización se requiere una relación de tiempo entre eventos en diferentes procesos. En su forma más simple se requiere que el evento b (en el proceso q) suceda después del evento a (en el proceso p). Esto puede resolverse de la siguiente manera: p manda un mensaje a q después de la ejecución de a, y q difiere la ejecución de b hasta recibir dicho mensaje.

En la sincronización de redes, en cada proceso p una secuencia de $a_{p,1}, a_{p,2}, \ldots$ de eventos deben ejecutarse de manera tal que un mensaje enviado a q en el evento $a_{p,i}$, es recibido por q antes del evento $a_{q,i+1}$. Este problema es fundamental para la implementación de sistemas síncronos en redes asíncronas. El evento $a_{p,i}$ representa la ejecución de la i-ésima ronda del algoritmo por el proceso p. Los mensajes en un sistema síncronos son recibidos siempre antes de la siguiente ronda. Los algoritmos para estos problemas se llaman *sincronizadores.*

En cuanto a la sincronización de relojes, se asume que cada proceso tiene un reloj local. El problema es mantener las lecturas del reloj tan cercanas como sea posible, con el objetivo de crear la noción de un reloj global. Las rutinas de sincronización de relojes son muy importantes en las redes de computadoras por varias razones. Una de las razones es que las aplicaciones de redes pueden requerir que los eventos estén fijos en el tiempo de manera exacta, por ejemplo en algoritmos de control concurrente usados en bases de datos distribuidas. Además, muchas técnicas para conseguir tolerancia a fallas se basan en la posibilidad de la sincronización de relojes.

Los componentes de un sistema distribuido están sujetos a fallas. Se habla de una falla cuando una componente muestra un comportamiento que se desvía de su especificación. En una situación ideal el sistema continuará funcionando como la hacía antes de la falla. Al menos las funciones realizadas por componentes correctos deben continuar en operación y proceder de una manera correcta. Lo impredecible de la ocurrencia y naturaleza de una o múltiples fallas, hace del análisis de la robustez de tolerancia a fallas y el diseño de sistemas distribuidos tolerantes a fallas una materia muy complicada.

Un error de transmisión puede acasionar que un mensaje sea recibido con un contenido diferente al que tenía cuando fue enviado. Se ha desarrollado una teoría matemática de detección de errores y códigos de corrección para hacer la comunicación robusta contra estos. Otros posibles errores en la comunicación incluyen la pérdida completa del mensaje.

Se consideran varios tipos de posibles fallas para los procesos y los canales que los conectan. Se dice que una falla es *transitoria* si ocurre únicamente una vez y la

componente recupera su operación normal después de que ha ocurrido. Se dice que una falla es *fail-stop* si el proceso detiene su actividad y no hace nada después de ésta. Por último, se dice que un proceso falla de manera *Bizantina* si muestra un comportamiento completamente arbitrario.

Como se mencionó anteriormente existe una conexión importante entre la teoría de gráficas y el estudio de algoritmos distribuidos. En particular las *gráficas marcadas estocásticas* [1] son un modelo útil para dicho estudio. La elección de estas gráficas para el desarrollo del presente trabajo está basado en las siguientes razones:

1. Es un modelo simple mediante el cual se pueden representar mecanismos básicos del cómputo distribuido (un sincronizador, por ejemplo).
2. Resultados teóricos de su desempeño.
3. El algoritmo para calcular su desempeño es de complejidad exponencial.
4. No existen algoritmos eficientes para calcular su desempeño.

El análisis del desempeño de estas gráficas se basa en el que se realiza para las redes de Petri estocásticas y debido al enorme tamaño del espacio de estados, las técnicas de solución quedan comprendidas en el campo de la simulación.

[1] Este modelo es una subclase del modelo *redes de Petri estocásticas*.

Capítulo 2

Técnicas de reducción de varianza

Las *técnicas de reducción de varianza* se han utilizado frecuentemente en el campo de la simulación digital y su objetivo es tratar de reducir la varianza de los valores muestrales obtenidos mediante la simulación de un modelo que representa el mundo real.

La simulación digital consiste en imitar mediante computadoras las operaciones de varios tipos de procesos del mundo real. A dichos procesos se les conoce como sistema y para estudiarlo se asumen ciertas características acerca de cómo trabaja. Estas suposiciones, las cuales toman la forma de relaciones matemáticas o lógicas, constituyen un modelo que se usa para tratar de entender cómo se comporta el sistema.

Si las relaciones que componen el modelo son lo suficientemente simples, es posible usar métodos matemáticos para obtener información exacta en preguntas de interés; esto se conoce como una solución analítica. Sin embargo, la mayoría de sistemas del mundo real son demasiado complejos para permitir soluciones analíticas y estos modelos deben estudiarse por medio de la simulación.

Puesto que el objetivo es conseguir, mediante la simulación, una buena aproximación del valor esperado de alguna medida de ejecución de un modelo (algoritmo) en particular, es muy importante aplicar técnicas estadísticas apropiadas para que los datos de salida de la simulación sean analizados, interpretados y usados apropiadamente.

Debido a que algunas simulaciones de tamaño considerable pueden requerir grandes cantidades de tiempo de cómputo y recursos de almacenamiento, es posible

que aplicar un análisis estadístico apropiado sea muy costoso (esto puede suceder debido a que a veces son necesarias múltiples repeticiones de nuestro modelo). Entonces es preciso usar todos los medios posibles para incrementar la eficiencia de la simulación.

Es claro que la "eficiencia" contempla una programación cuidadosa para facilitar la ejecución y minimizar los requerimientos de almacenamiento. Por ahora el objetivo será conseguir una eficiencia estadística, que va a estar calificada por las varianzas de las variables aleatorias de salida de la simulación. Si es posible de alguna manera reducir la varianza de una variable aleatoria de salida *sin perturbar su esperanza*, será posible obtener una mayor precisión para el mismo costo de simulación o, alternativamente, conseguir la precisión deseada con un menor costo. A las técnicas que nos permiten conseguir este objetivo se les conoce como *técnicas de reducción de varianza* , y aplicadas correctamente pueden ser la diferencia entre una simulación que por su costo es imposible de realizar y otra que nos puede ser de gran utilidad.

La aplicación de técnicas de reducción de varianza depende principalmente del modelo particular. Por lo tanto, es necesario entender bien el modelo para el uso apropiado de estas técnicas. Hay que señalar que es imposible saber de antemano qué tan grande será la reducción de varianza que se obtendrá; entonces lo que se puede hacer es realizar algunas corridas preliminares. Algunas técnicas de reducción de varianza por sí mismas incrementan el costo computacional, y esta pérdida en la eficiencia computacional debe compensarse con la eficiencia estadística obtenida.

En este capítulo se presentará principalmente cuatro tipos generales de técnicas de reducción de varianza que parecería que tienen la mayor oportunidad de lograr éxito en una gran variedad de simulaciones.

2.1 Números aleatorios comunes

La primera técnica de reducción de varianza que se considerará se conoce como *números aleatorios comunes (NAC)*; ésta técnica se usa cuando se desea hacer una comparación entre dos o más sistemas alternativos. A pesar de su sencillez, la técnica de NAC es probablemente la más popular de todas las técnicas de reducción de varianza.

La idea básica es que se debería comparar las configuraciones alternativas bajo

condiciones experimentales similares; así se podría estar seguro que cualquier diferencia observada en la ejecución se debe a las diferencias entre las configuraciones de los sistemas, y no a variaciones en las condiciones experimentales. En la simulación, estas condiciones experimentales son las variables aleatorias generadas que se usan para llevar el modelo a través del tiempo de la simulación.

El nombre de esta técnica emana de la posibilidad de usar, en muchas situaciones, los mismos números aleatorios básicos $U(0,1)$ para llevar a cabo cada una de las configuraciones alternativas. Sin embargo, son necesarias algunas técnicas de programación para la implementación apropiada de la técnica de NAC.

Para ver más detalladamente el análisis de la técnica de NAC, considérense dos configuraciones alternativas, donde X_{1j} y X_{2j} son observaciones de la primera y segunda configuraciones en la j-ésima repetición independiente, y se desea estimar $\xi = \mu_1 - \mu_2 = E(X_{1j}) - E(X_{2j})$. Si se hace n repeticiones de cada sistema y se hace $Z_j = X_{1j} - X_{2j}$ para $j = 1, 2, \ldots, n$, entonces $E(Z_j) = \xi$ y, así

$$\overline{Z}(n) = \frac{\sum_{j=1}^{n} Z_j}{n}$$

es un estimador insesgado de ξ. Puesto que las Z_j's son variables aleatorias independientes e idénticamente distribuidas,

$$Var[\overline{Z}(n)] = \frac{Var(Z_j)}{n} = \frac{Var(X_{1j}) + Var(X_{2j}) - 2Cov(X_{1j}, X_{2j})}{n}$$

Si las simulaciones de las dos diferentes configuraciones son hechas independientemente, es decir, con números aleatorios diferentes, X_{1j} y X_{2j} serán independientes, así la $Cov(X_{1j}, X_{2j}) = 0$. De otra manera, si podemos de alguna manera hacer las simulaciones de las configuraciones 1 y 2 de tal manera que X_{1j} y X_{2j} estén correlacionadas positivamente, entonces la $Cov(X_{1j}, X_{2j}) > 0$, y se reduce de esta manera la varianza de nuestro estimador $\overline{Z}(n)$. Así, cuando $\overline{Z}(n)$ es tomado en un experimento de simulación en particular, su valor estará más cercano a ξ.

La técnica de NAC es una técnica en la que se trata de inducir una correlación positiva usando (cuidadosamente) los mismos números aleatorios para simular todas las configuraciones. Esto es posible gracias a la naturaleza determinista de los generadores de números aleatorios. Si se generan los números aleatorios de una manera que sea imposible volver a generarlos (por ejemplo, tomando como semilla la raíz cuadrada del valor del reloj de la computadora), todo el posible beneficio de esta técnica, al igual que otras valiosas técnicas de reducción de varianza, podría perderse.

Desafortunadamente, no hay una prueba completamente general de que la técnica de NAC realmente funciona, es decir, de que siempre reducirá la varianza. Aún si funciona, usualmente no se sabe de antemano qué tan grande será la reducción de varianza que se obtendrá. La eficacia de la técnica de NAC depende por completo de los modelos particulares a compararse, y presupone que el analista crea que los modelos diferentes responderán similarmente a valores grandes o pequeños de las variables aleatorias que determinan a los modelos.

En la Fig. 2.1 se ilustra el principio de cuándo esta técnica podría funcionar o ser contraproducente. En esta figura el eje de las x muestra posibles valores de un U_k en particular usado para un propósito específico en las dos simulaciones; por ejemplo, estos U_k podrían usarse para generar el tiempo de disparo de un nodo en una simulación de una gráfica marcada estocástica. Las curvas indican cómo podrían reaccionar los resultados de las simulaciones (manteniendo todas las demás cosas igual) para posibles valores de U_k. En cualquiera de las dos simulaciones de las gráficas superiores, ambas X_{1j} y X_{2j} reaccionan monotónicamente en la misma dirección a U_k, y se esperaría que la técnica de NAC induzca la correlación positiva deseada, y así se reduzca la varianza. Sin embargo, en las dos gráficas inferiores, X_{1j} y X_{2j} reaccionan en direcciones opuestas a U_k, entonces NAC podría inducir una correlación negativa y ser contraproducente, haciendo la $Cov(X_{1j}, X_{2j}) < 0$ y provocando un incremento en la varianza.

Usualmente, se usan números aleatorios para generar variables de otras distribuciones, las cuales son usadas para llevar a cabo los modelos de simulación. Para dar a la técnica de NAC la mejor oportunidad de que funcione, se debe primero asegurar que las variables generadas reaccionan monotónicamente a las U_k's en este paso intermedio de generación de variables; entonces se debe suponer que las medidas de ejecución reaccionan monotónicamente a las variables generadas. Para el problema particular de las gráficas marcadas estocásticas, se puede usar el método de la transformación inversa, ya que este método garantiza la monotonicidad de las variables de entrada generadas por los números aleatorios. Aunque para algunas distribuciones este método es lento, su ineficiencia computacional es compensada por su eficiencia estadística [Law 1991].

De ser posible, un pequeño estudio piloto puede dar una idea preeliminar sobre la eficacia de la técnica de NAC para las configuraciones alternativas. En el caso de dos configuraciones, hacer n repeticiones de cada una, usando NAC, para obtener las observaciones de salida X_{1j} y X_{2j} para $j = 1, 2, \ldots, n$. Sean $S_1^2(n)$ y $S_2^2(n)$

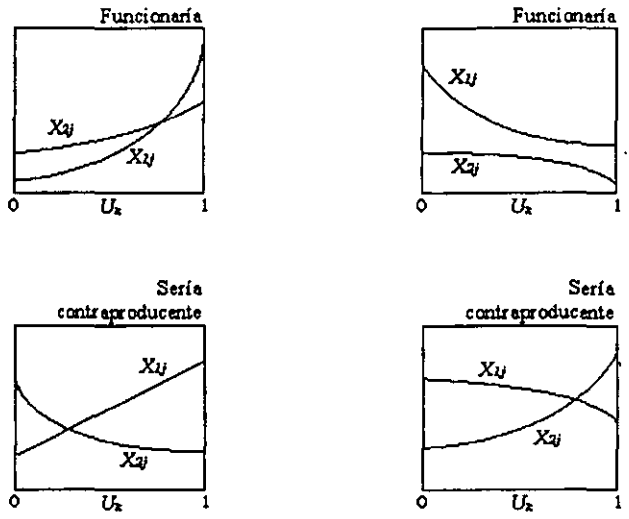


Figura 2.1: Respuestas de los modelos para que NAC funcione (primer renglón) y sea contraproducente (segundo renglón).

las varianzas muestrales de las X_{1j} y X_{2j} respectivamente, y sea $S_Z^2(n)$ la varianza muestral de las diferencias $Z_j = X_{1j} - X_{2j}$; puesto que las corridas fueron hechas usando NAC, S_Z^2 es un estimador insesgado de la varianza de Z_j bajo NAC. Sin importar el hecho de que se use NAC, $S_1^2(n)$ es insesgado para la $Var(X_{1j})$ y $S_2^2(n)$ lo es para $Var(X_{2j})$, entonces $S_1^2(n)+S_2^2(n)$ es un estimador insesgado de la varianza de Z_j si se hubieran hecho las corridas sin NAC. Entonces, si NAC está funcionando, esperaríamos observar que $S_Z^2(n) < S_1^2(n) + S_2^2(n)$, y la diferencia permite estimar qué tanto está reduciendo la varianza de un Z_j. Por supuesto que cualquier esfuerzo de programación extra que NAC requiera para su implementación debe hacerse para el estudio piloto, sin importar si la técnica finalmente va a utilizarse.

Debe considerarse que NAC es una herramienta útil cuando el analista se en-

cuentra en la tarea de comparar dos o más configuraciones alternativas. Nótese que además de una posible reducción en la varianza, esta técnica da un factor constante de mejoramiento en la eficiencia de la simulación, ya que el costo de la generación de la entrada es partido a la mitad.

2.1.1 Sincronización

Para implementar la técnica de NAC apropiadamente se deben sincronizar los números aleatorios a través de las diferentes configuraciones del sistema en una repetición particular. Idealmente, un número aleatorio usado para un propósito específico en una configuración es usado exactamente para el mismo propósito en todas las otras configuraciones. Por ejemplo, si un U_k específico se usa en la primera de dos gráficas alternativas para generar un tiempo de disparo particular, entonces también debe usarse en la segunda configuración para generar el mismo tiempo de disparo (en lugar de usarlo para un tiempo de transmisión o algún otro tiempo de disparo); de otra manera podría perderse el beneficio de la técnica de NAC o peor aún el método podría ser contraproducente.

La dificultad de mantener una sincronización apropiada depende en general de la estructura del modelo, sus parámetros y de los métodos usados para generar las variables de entrada usadas en la simulación. Algunas ideas útiles para conseguir una buena sincronización en una simulación son:

- Si existen varios flujos de números aleatorios disponibles, o si hay varios generadores de números aleatorios independientes, se puede dedicar un generador (o flujo) a producir los números aleatorios para cada tipo particular de variable aleatoria de entrada.
- El método de la transformación inversa para la generación de variables aleatorias puede facilitar la sincronización ya que siempre necesita exactamente un número aleatorio para producir cada valor de la variable aleatoria deseada. Más aún, este método transforma monotónicamente los números aleatorios.
- Desperdiciar algunos números aleatorios en cierto punto de la simulación.

Algunas veces puede ser imposible lograr una sincronización completa de los modelos en estudio. Del mismo modo, el esfuerzo extra de programación, tiempo de cómputo, o costos de almacenamiento necesarios para una sincronización completa,

podrían hacer perder la importancia de reducir la varianza. Entonces, podría considerarse la sicronización de algunas variables de entrada y generar las otras de manera independiente a través de las configuraciones. En el análisis final, el beneficio de usar NAC y el grado de sincronización dependen de la situación particular.

2.2 Variables antitéticas

La técnica de *variables antitéticas* (VA), es una técnica de reducción de varianza que se aplica para la simulación de un único sistema. Como en NAC, se trata de inducir una correlación entre corridas separadas, pero ahora el objetivo es de inducir una correlación negativa.

La idea principal es hacer pares de corridas del modelo, tales que una observación pequeña en una de las corridas en un par tienda a equilibrarse por una observación grande en el otro; es decir, las dos observaciones están correlacionadas negativamente. Entonces, si se usa el promedio de las dos observaciones en el par como un punto básico para el análisis, tenderá a estar más cercano a la esperanza común μ de una observación (la cual queremos estimar) de lo que estaría si las dos observaciones en el par fueran independientes.

En su forma más simple, la técnica de VA trata de inducir esta correlación negativa usando números aleatorios complementarios para llevar a cabo las dos simulaciones en un par. Esto es, si U_k es un número aleatorio particular usado para un propósito particular(por ejemplo, generar el i-ésimo tiempo de disparo) en la primera corrida, se usa $1 - U_k$ para este mismo propósito en la segunda corrida. Es válido usar $1 - U_k$ en lugar de simplemente uno generado directamente por el generador de números aleatorios, ya que $U \sim U(0,1)$ implica que $1 - U \sim U(0,1)$ también.

Un punto importante es que el uso de un U_k en una repetición y su complemento $1 - U_k$ en la repetición apareada deben estar sincronizados; es decir, usados para el mismo propósito; de otra manera el beneficio de la técnica de VA podría perderse o, peor aún, el uso de la técnica podría ser contraproducente. Las ideas propuestas en la sección anterior pueden ser también de utilidad para lograr una sincronización apropiada para los números aleatorios. Es posible también el uso parcial de VA (como para NAC) generando algunas entradas antitéticamente y otras independientemente dentro de un par si la sincronización completa es demasiado compleja, o si

no se observa un camino claro a seguir para poder usar VA para todas las entradas.

Para entender el funcionamiento de esta técnica primero se verán las bases teóricas de la misma. Supóngase que se hacen n pares de corridas de la simulación y resultan las observaciones $(X_1^{(1)}, X_1^{(2)}), \ldots, (X_n^{(1)}, X_n^{(2)})$, donde $X_j^{(1)}$ es de la primera corrida (usando las "U") del j–ésimo par, y $X_j^{(2)}$ es de la corrida antitética (usando los "$1 - U$", propiamente sincronizados) del j–ésimo par. Ambas $X_j^{(1)}$ y $X_j^{(2)}$ son observaciones legítimas del modelo de simulación, entonces $E(X_j^{(1)}) = E(X_j^{(2)}) = \mu$. Además, cada par es independiente de cualquier otro par; es decir, para $j_1 \neq j_2$, $X_{j_1}^{(l_1)}$ y $X_{j_2}^{(l_2)}$ son independientes, a pesar de que l_1 y l_2 sean iguales. Es importante notar que el número de repeticiones es $2n$. Para $j = 1, 2, \ldots, n$, sea $X_j = (X_j^{(1)} + X_j^{(2)})/2$, y sea el promedio de las X_j, $\overline{X}(n)$, el estimador (insesgado) puntual de $\mu = E(X_j^{(l)}) = E(X_j) = E[\overline{X}(n)]$. Entonces ya que las X_j son variables independientes e idénticamente distribuidas (IID),

$$Var[\overline{X}(n)] = \frac{Var(X_j)}{n} = \frac{Var(X_j^{(1)}) + Var(X_j^{(2)}) + 2Cov(X_j^{(1)}, X_j^{(2)})}{4n}$$

Si las dos corridas fueran independientes, esto provocaría que la $Cov(X_j^{(1)}, X_j^{(2)}) = 0$. Por otro lado, si es posible de alguna manera inducir una correlación negativa entre $X_j^{(1)}$ y $X_j^{(2)}$, entonces $Cov(X_j^{(1)}, X_j^{(2)}) < 0$, lo cual reduce la $Var[\overline{X}(n)]$, y este es el objetivo de VA.

Al igual que la técnica de NAC, tampoco se puede estar seguro que la técnica de VA funcionará y su eficacia y factibilidad son tal vez más dependientes del modelo que la técnica de NAC. En general, no es posible saber qué tan grande será la reducción de varianza lograda. Un estudio piloto como el que se propuso para NAC podría ser útil para asegurar cuando la técnica de VA es una buena idea en un caso específico.

El requerimiento fundamental que un modelo debe satisfacer para que VA funcione, es que su respuesta a un número aleatorio usado para un propósito particular sea monotónica, en cualquier dirección. Podría ocurrir una situación contraproducente de VA, por ejemplo, si una respuesta del modelo fuera grande para U_k pequeñas, menor para U_k cerca de 0.5, y nuevamente fuera grande para U_k grandes. Al igual que para la técnica de NAC, se sugiere el método de la transformación inversa para generar las variables aleatorias de entrada para asegurar la monotonicidad requerida en este paso intermedio de generación de variables aleatorias.

Debido a las similitudes entre las técnicas de NAC y VA, parece razonable la idea de usarlas conjuntamente cuando van a compararse varias configuraciones alternativas. En un principio parecería que se podría obtener una mayor reducción de varianza aplicando la técnica de VA para cada configuración separadamente y usando la técnica de NAC para las diferentes configuraciones. Sin embargo, haciendo un examen más cercano, encontramos que si ambos NAC y VA funcionan propiamente, es decir, que induzcan las correlaciones del signo deseado, ciertas covarianzas cruzadas (específicamente, la covarianza entre la primera corrida de un par antitético de corridas para la configuración 1 y la segunda corrida del par antitético correspondiente para la configuración 2, y viceversa) introduce la expresión relevante de la varianza con el signo equivocado. Así, de ningún modo es una buena idea, combinar VA con NAC para la comparación de dos configuraciones alternativas de un sistema.

2.3 Variables de control

De la misma forma que para las técnicas de NAC y VA, el método de *variables de control* (VC) intenta tomar ventaja de la correlación entre ciertas variables aleatorias para obtener una reducción de varianza. Dependiendo de la técnica específica de VC usada, esta correlación podría surgir naturalmente durante el curso de la simulación, o podría inducirse usando NAC en una simulación auxiliar.

En principio, intuitivamente el uso de la técnica de VC parece aceptable. Sea X una variable aleatoria de salida y asúmase que se quiere estimar $\mu = E(X)$. Supóngase que Y es otra variable aleatoria involucrada en la simulación que se piensa que está correlacionada con X (positiva o negativamente), y que nosotros sabemos el valor de $\nu = E(Y)$. En el caso que X y Y estén correlacionados, si se ejecuta una simulación y se nota que $X > \nu$(lo cual se puede afirmar con seguridad ya que se conoce ν), se podría sospechar que X está también por encima de su esperanza μ (aunque no se puede saber esto con seguridad a menos que la correlación entre Y y X fuera perfecta), y de acuerdo con esto ajustar a X hacia abajo por algún monto. Por otro lado, si $Y < \nu$, se sospecharía que también $X < \mu$ y entonces habría que ajustarla hacia arriba. De esta manera, se usa el conocimiento que se tiene sobre la esperanza de Y para empujar a X (hacia arriba o hacia abajo) de acuerdo con su esperanza μ, reduciendo así su variabilidad respecto de μ de una corrida a la

siguiente. Se dice entonces que Y es una *variable de control* para X ya que es usada para ajustar a X, o controlarla parcialmente.

A diferencia de las técnicas de NAC y VA, el éxito de VC no depende de que la correlación sea de un signo en particular. Si Y y X estuvieran correlacionadas negativamente, simplemente se ajustaría X hacia arriba si $Y > \nu$ y hacia abajo si $Y < \nu$.

Para llevar a cabo la idea anterior, se debe cuantificar el monto del ajuste de X hacia arriba o hacia abajo. Es conveniente expresar este monto en término de la desviación, $Y - \nu$, de Y con su esperanza. Sea a una constante (que se determinará después) que es del mismo signo que la correlación entre X y Y. Se usa a para balancear (aumentar o disminuir) la desviación $Y - \nu$ para llegar a un ajuste en X y así se define el estimador "controlado"

$$X_C = X - a(Y - \nu)$$

Nótese que si X y Y están correlacionados positivamente, entonces $a > 0$, se ajustaría X hacia abajo cuando $Y > \nu$ y hacia arriba cuando $Y < \nu$, tal como es deseado; lo opuesto es cierto cuando Y y X están correlacionados negativamente, en cuyo caso $a < 0$.

Puesto que $E(X) = \mu$ y $E(Y) = \nu$, se cumple que para cualquier número real a, $E(X_C) = \mu$; esto es, X_C es un estimador insesgado para μ que podría tener varianza más pequeña que X. Específicamente,

$$Var(X_C) = Var(X) + a^2 Var(Y) - 2aCov(X, Y) \tag{2.1}$$

entonces X_C tiene menor varianza que X si y sólo si

$$2aCov(X, Y) > a^2 Var(Y)$$

lo cual puede o no ser cierto, dependiendo de la elección de Y y a. En muchos trabajos de VC, sólo se consideran los casos especiales $a = 1$ (si se cree que $Cov(X, Y) > 0$) o $a = -1$ (si se cree que $Cov(X, Y) < 0$), pero esto requiere la condición más fuerte de que $| Cov(X, Y) | > Var(Y)/2$ para que pueda conseguirse una reducción de la varianza. Así, simplemente haciendo $a = \pm 1$ se coloca todo el peso de lograr éxito sobre la selección de la Y; permitiendo otros valores para a es posible hacerlo mejor.

Para encontrar el "mejor" valor de a dada una Y, es posible ver el lado derecho de la Ec. 2.1 como una función $g(a)$ de a e igualar su derivada a cero, es decir

$$\frac{dg}{da} = 2aVar(Y) - 2Cov(X,Y) = 0$$

y resolver para el valor óptimo (el cual minimiza la varianza)

$$a^* = \frac{Cov(X,Y)}{Var(Y)} \tag{2.2}$$

una condición suficiente para que a^* sea el mínimo de $g(a)$, es que $d^2g/da^2 = 2Var(Y)$ sea positivo, lo cual se cumple. Una de las implicaciones de la ecuación 2.2 es que si Y está fuertemente correlacionada con X, esto es, $\mid Cov(X,Y) \mid$ es grande, el valor de a^* se incrementa, y entonces se están haciendo ajustes más drásticos a X ya que se siente más confianza de lo que la desviación de Y hacia ν dice acerca de la desviación que podría haber de X hacia μ. También, si Y por sí misma es menos variable, esto es $Var(Y)$ es pequeña, se podría obtener un valor de a^* mayor (y un mayor ajuste a X) ya que se tiene mayor confianza en la precisión del valor observado de Y.

Sustituyendo a^* de la Ec. 2.2 en el lado derecho de la Ec. 2.1, se obtiene que el estimador de mínima varianza *controlado* (o *ajustado*) X_C^* sobre todas las elecciones de a tiene varianza

$$Var(X_C^*) = Var(X) - \frac{[Cov(X,Y)]^2}{Var(Y)} = (1 - \rho_{XY}^2)Var(X)$$

donde ρ_{XY} es la correlación entre X y Y. Así, usando el valor óptimo a^* para a, el estimador óptimamente controlado X_C^* nunca puede ser más variable que la X sin control, y tendrá de hecho menor varianza si Y está del todo correlacionada con X. Más aún, la fuerte correlación entre X y Y, logra la mayor reducción de varianza(en el extremo, como $\rho_{XY} \to \pm 1$, se ve que la $Var(X_C^*) \to 0$). Intuitivamente, esto dice que si la correlación entre Y y X fuera cercana a ser perfecta (± 1), se podría controlar X casi exactamente hacia μ todo el tiempo, eliminando así prácticamente toda su varianza.

En la práctica las cosas no son tan fáciles. Dependiendo de la fuente y naturaleza de la variable de control Y, podría no conocerse la $Var(Y)$, y no conocerse

la $Cov(X,Y)$, haciendo imposible la tarea de encontrar el valor exacto de a^*. De acuerdo con esto, se han desarrollado varios métodos para estimar a^* a partir de las corridas de la simulación. A continuación se presenta uno de los métodos más simples.

El siguiente método simplemente reemplaza la $Cov(X,Y)$ y la $Var(Y)$ en la Ec. 2.2 por sus estimadores muestrales. Supóngase que se hacen n repeticiones independientes para obtener n observaciones IID $X_1, X_2, \ldots, X_n$ en X y las n observaciones IID $Y_1, Y_2, \ldots, Y_n$ en Y. Sean $\overline{X}(n)$ y $\overline{Y}(n)$ las medias muestrales de las X_j's y Y_j's respectivamente, y sea $S_Y^2(n)$ la varianza muestral insesgada de las Y_j's. La covarianza entre X y Y está estimada por

$$\hat{C}_{XY}(n) = \frac{\sum_{j=1}^{n}[X_j - \overline{X}(n)][Y_j - \overline{Y}(n)]}{n-1}$$

y el estimador para a^* es entonces

$$\hat{a}^*(n) = \frac{\hat{C}_{XY}(n)}{S_Y^2(n)}$$

para llegar al estimador puntual final para μ,

$$\overline{X}_C^*(n) = \overline{X}(n) - \hat{a}^*(n)[\overline{Y}(n) - \nu]$$

Se debe notar que puesto que la constante a^* se ha reemplazado por la variable aleatoria $\hat{a}^*(n)$, la cual generalmente no es independiente de $\overline{Y}(n)$ (habiéndose calculado de los mismos datos de salida de la simulación), no es posible simplemente tomar las esperanzas a través de los factores en el segundo término de $\overline{X}_C^*(n)$. Desafortunadamente, entonces, $\overline{X}_C^*(n)$, a diferencia de X_C y X_C^* en general será sesgado para μ.

En algunos casos existen realmente dos o más variables de control diferentes que se pueden combinar para obtener una única variable de control. Sin embargo, no es fácil decidir entre sustraer una de la otra o dividir una entre la otra, o más aún restar a una un múltiplo de otra. En modelos más complejos existen muchas potenciales variables de control disponibles, y será difícil tomar la mejor manera de juntarlas todas en una sola. Más aún, si se estuvieran combinando de una manera razonable, podría no estar usándose su información para nuestro mejor provecho. Por ejemplo,

se podría construir la variable de control $Y = Y^{(1)} - Y^{(2)}$; entonces el estimador controlado es

$$X_C = X - a(Y - \nu) = X - a(Y^{(1)} - \nu^{(1)}) - a(Y^{(2)} - \nu^{(2)})$$

donde $\nu^{(1)} = E(Y^{(1)})$. En este caso, se están forzando ambas variables para entrar al ajuste usando el mismo coeficiente, a, el cual podría no ser el mejor uso de su información. Una modificación lógica podría ser: permitir a las dos variables de control tener diferentes pesos, y redefinir

$$X_C = X - a_1(Y^{(1)} - \nu^{(1)}) - a_2(Y^{(2)} - \nu^{(2)})$$

Se podría entonces encontrar (y estimar) los pesos a_1 y a_2 que minimicen la $Var(X_C)$, como se hizo antes cuando se tenía sólo una variable de control.

Esta idea se generaliza fácilmente al caso donde se tienen m variables de control $Y^{(1)}, Y^{(2)}, \ldots, Y^{(m)}$ con sus respectivas esperanzas conocidas $\nu^{(1)}, \nu^{(2)}, \ldots, \nu^{(m)}$. El estimador (linealmente) controlado general es

$$X_C = X - \sum_{l=1}^{m} a_l(Y^{(l)} - \nu^{(l)})$$

donde las a_l's son números reales a determinarse (y estimarse). Tomando en cuenta la correlación no sólo entre X y las variables de control sino también entre las variables de control, se obtiene

$$\begin{aligned} Var(X_C) &= Var(X) + \sum_{l=1}^{m} a_l^2 Var(Y_l) - 2\sum_{l=1}^{m} a_l Cov(X, Y_l) \\ &\quad +2\sum_{l_1=1}^{m}\sum_{l_2=1}^{l_1-1} a_{l_1} a_{l_2} Cov(Y_{l_1}, Y_{l_2}) \end{aligned} \tag{2.3}$$

Tomando las derivadas parciales del lado derecho de la Ec. 2.3 con respecto a cada una de las a_l e igualándolas a cero, se llega a un conjunto de m ecuaciones lineales por resolver para los m pesos que minimizan la varianza. Como en el caso de una sola variable de control, estos pesos óptimos deben estimarse, y la introducción de sesgo en el controlador estimado es posible nuevamente.

A continuación se mencionará algunos puntos importantes para encontrar y seleccionar variables de control. Como se ha visto, una buena variable de control debe

de estar fuertemente correlacionada con la variable aleatoria de salida X para que pueda dar mucha información sobre X y lograr un buen ajuste para ésta. También resulta atractiva una variable de control que por sí misma tenga poca varianza. Para encontrar dicha variable de control se puede proceder por un análisis de la estructura del modelo, o a través de experimentaciones iniciales. Con estos objetivos presentes, se han sugerido tres fuentes generales de variables de control:

- *Internas.* Variables aleatorias de entrada, o funciones simples de ellas (tales como promedios), se usan frecuentemente como variables de control. Sus esperanzas generalmente serán conocidas, y un análisis simple en el modelo puede sugerir cómo están correlacionadas con la variable aleatoria de salida. Es importante señalar que las variables de control internas deben generarse de todas maneras en el curso de la simulación; así no agregan casi nada al costo de la simulación.

- *Externas.* Es posible que se esté haciendo una simulación, puesto que no se puede calcular $\mu = E(X)$ analíticamente. Tal vez si se altera el modelo asumiendo algo adicional que lo simplifica, se podría calcular la esperanza ν de la variable aleatoria de salida Y del modelo simplificado. Mientras se podría no querer hacer estas simplificaciones en el modelo original ya que podrían dañar la validez del mismo, Y podría servir como una variable de control para X. Entonces se simularía el modelo simplificado junto con el modelo actual, usando NAC, y esperando que Y esté correlacionada con X, presumiblemente de manera positiva. Como este enfoque no es de bajo costo ya que involucra una segunda simulación para obtener la variable de control, cosa que no sucede con las VC internas; entonces, la correlación entre X y Y debe ser mayor en las VC externas para compensar la que se obtendría en el caso que la Y fuera una VC interna.

- *Uso de múltiples estimadores.* En algunas situaciones se podrían tener varios estimadores insesgados $X^1, X^2, \ldots, X^k$ para μ, donde las X^i's podrían ser o no independientes. Si $b_1, \ldots, b_k$ son cualesquiera números reales (no necesariamente positivos) que suman 1, entonces

$$X_C = \sum_{i=1}^{k} b_i X^{(i)}$$

es también insesgado para μ. Puesto que $b_1 = 1 - \sum_{i=2}^{k} b_i$, se puede expresar X_C como

$$\begin{aligned} X_C &= (1 - \sum_{i=2}^{k} b_i)X^{(1)} + \sum_{i=2}^{k} b_i X^{(i)} \\ &= X^{(1)} - \sum_{i=2}^{k} b_i (X^{(1)} - X^{(i)} \end{aligned}$$

entonces es posible ver $Y_i = X^{(1)} - X^{(i)}$, para $i = 2, 3, \ldots, k$, como $k - 1$ variables para $X^{(1)}$.

Como se observó anteriormente, puede haber un número muy grande de posibles variables de control para un modelo complejo. Sin embargo, no necesariamente es una buena idea usarlas todas, puesto que la reducción de varianza que puedan conseguir está acompañada por las contribuciones de varianza asociadas con la necesidad de estimar las a_i's optimas.

2.4 Esperanza condicional

La técnica de reducción de varianza *esperanza condicional* (EC), trata de explotar alguna propiedad especial de un modelo para reemplazar una estimación de una cantidad por su valor analítico exacto. Removiendo esta fuente de variabilidad, se espera que la variable aleatoria de salida final sea más estable, aunque no hay garantía de éxito; nuevamente, es posible utilizar estudios pilotos que comparen la técnica de EC con una simulación correcta para indicar cuándo se logrará una reducción de la varianza y qué tan grande será la misma.

Sea X una variable aleatoria de salida, cuya esperanza μ se quiere estimar. Supóngase que hay alguna otra variable aleatoria Z tal que, dado cualquier posible valor particular z para Z, es posible calcular analíticamente la esperanza condicional $E(X \mid Z = z)$. Nótese que $E(X \mid Z = z)$ es una función (conocida) determinista del número real z, pero $E(X \mid Z)$ es una variable aleatoria. Entonces, condicionando sobre Z, se ve que $\mu = E(X) = E_Z[E(X \mid Z)]$ (la esperanza exterior es denotada por E_Z ya que es tomada con repecto a la distribución de Z) y que la variable aleatoria $E(X \mid Z)$ es también insesgada para μ. Por ejemplo, si Z es discreta con función de

masa de probabilidad $p(z) = P(Z = z)$, entonces

$$E_Z[E(X \mid Z)] = \sum_z E(X \mid Z = z)p(z)$$

donde se asume que las $p(z)$ son conocidas.

A continuación se define la varianza condicional como:

Definición 2.1

$$Var[X \mid Z = z] = E[X^2 \mid Z = z] - (E[X \mid Z = z])^2$$

Esta definición sirve para la demostración del siguiente teorema:

Teorema 2.1

$$Var[X] = E_Z[Var(X \mid Z)] + Var_Z(E[X \mid Z])$$

Demostración

$$\begin{aligned} E_Z[Var(X \mid Z)] &= E_Z[E[X^2 \mid Z]] - E_Z[(E[X \mid Z])^2] \\ &= E[X^2] - (E[X])^2 - E_Z[(E[X \mid Z])^2] + (E[X])^2 \\ &= Var(X) - E_Z[(E[X \mid Z])^2] + (E_Z[E[X \mid Z]])^2 \\ &= Var(X) - Var_Z(E[X \mid Z]) \end{aligned}$$

Entonces ahora se puede concluir el siguiente resultado:

$$Var_Z[E(X \mid Z)] = Var(X) - E_Z[Var(X \mid Z)] \leq Var(X) \tag{2.4}$$

indicando que si se observa la variable aleatoria $E(X \mid Z)$ (como calculada a partir de una observación z en Z), en lugar de observar X directamente, se obtendrá una menor varianza. En otras palabras, se sugiere que se simule para obtener una observación aleatoria de z en Z (puesto que su distribución no es conocida), sustituyamos esta observación en la fórmula conocida $E(X \mid Z = z)$, y se use ésta como una observación básica.

Aquí, el objetivo es especificar una variable aleatoria Z tal que:

- Z puede generarse de una manera fácil y eficiente, puesto que se tiene que simular.

- $E(X \mid Z = z)$ como una función de z puede calcularse analíticamente y eficientemente para cualquier posible valor de z que Z pueda tomar.
- $E_Z[Var(X \mid Z)]$ es grande; así, reduciendo mucho la $Var(X)$ en la Ec. 2.4. $E_Z[Var(X \mid Z)]$ es la varianza condicional media de X sobre los posibles valores de Z, y puesto que se tiene una fórmula para $E(X \mid Z = z)$, nunca se tiene que simular X dado Z, es decir, su varianza no afecta.

Aquí se da por terminada la descripción de algunas técnicas de reducción de varianza, aunque existen otras técnicas que también pueden ser útiles. Estas técnicas sólo son descritas brevemente en la siguiente sección en el contexto de los algortimos, si se desea una explicación más detallada sobre las mismas se puede consultar el trabajo de Catherine McGeoch [McGeoch 1992] donde además se presentan algunos ejemplos de aplicación de estas técnicas.

2.5 Otras técnicas de reducción de varianza

- *Atajos en la simulación.* Ahora se considerará el problema de acelerar el programa de simulación. La idea clave es que se desea simular un algoritmo, no necesariamente implementarlo. Puede ser posible aplicar mejoramientos al algoritmo para permitir más ensayos por unidad de tiempo de cómputo, sin cambiar las propiedades de distribución de las variables de salida. Mientras que no es como las técnicas de reducción de varianza (porque la varianza permanece igual en cada corrida), los atajos en la simulación pueden reducir la varianza permitiendo más ensayos por unidad de tiempo.

  Podría obtenerse una aceleración de la simulación explotando el conocimiento que no estaría disponible en una aplicación.

- *Particionamiento.* Esta técnica puede ser útil cuando una simulación produce dos variables en secuencia, donde la primera representa algún "estado" y la segunda es un estimador del costo esperado de ese estado. Cuando la generación de un estado es cara, esta técnica reduce mucho el monto de tiempo de cómputo requerido para obtener estimaciones del costo esperado. El método de esperanza condicional extiende esta idea para el calculo directo del costo esperado del estado; *particionando* debe considerarse cuando el cómputo directo no es factible.

- *Estratificación.* La técnica de *estratificación* reduce la varianza distinguiendo clases o "estratos" de entrada y generando instancias de la entrada de tal manera que el número de instancias que ocurran en cada clase corresponda exactamente a su esperanza.

- *Postestratificación.* Un método para "corregir" un estimador usando información auxiliar es mediante *postestratificar* el experimento. Se habla de "corregir" una medida de salida de acuerdo con su esperanza y con la variación en la distribución de la entrada.

Capítulo 3

Gráficas marcadas estocásticas

El modelo conocido como gráficas marcadas estocásticas sirve para modelar algunos mecanismos básicos del cómputo distribuido. Para este modelo existen resultados teóricos de su desempeño; sin embargo, el algoritmo para el cálculo de dicho desempeño es de complejidad exponencial.

En principio se enunciarán algunos teoremas básicos del modelo de gráficas marcadas y posteriormente mediante el modelo de gráficas marcadas estocásticas se calculara *la tasa de una red*, la cual puede verse como el número promedio de pasos computacionales ejecutadas por un vértice en la red por unidad de tiempo.

3.1 Gráficas marcadas

Existen muchos modelos de estructura de gráficas que se han usado y sugerido para sistemas de procesamiento concurrente. Estas estructuras difieren en la generalidad y extensión de acuerdo con las propiedades que se desea modelar y analizar. Las gráficas marcadas pueden derivarse como un caso especial de las Redes de Petri.

3.1.1 Marcajes vivos y seguros de una gráfica dirigida

Asúmase que se tiene una gráfica dirigida $G(V, E)$, donde V es el conjunto de vértices y E es el conjunto de aristas. La notación $a \xrightarrow{e} b$ significa que la arista e sale del vértice a y entra al vértice b. Se asigna un número $M(e)$ de *fichas* (un entero no negativo) a cada arista e. La función M es llamada *marcaje* de la gráfica. Se dice

que un vértice está *habilitado* si el número de fichas en cada una de sus aristas de entrada es positivo. El *disparo* de un vértice habilitado consiste en quitar una ficha de cada una de sus aristas de entrada y sumar una ficha a cada una de sus aristas de salida. Puesto que el número de aristas de entrada y de salida no es necesariamente el mismo, el número total de fichas en la gráfica puede incrementarse o disminuir a través de un disparo.

Considérese el número de fichas en un circuito dirigido simple C; este número, $\langle M \mid C \rangle$, es la suma de fichas en las aristas del circuito y se le conoce como *conteo de fichas*. El siguiente lema es una consecuencia directa de la definición de disparar.

Lema 3.1 *El conteo de fichas de un circuito dirigido no cambia por el disparo de un vértice.*

Ejemplo 3.1 *En la Figura 3.1(i) se muestra una gráfica marcada. Existe únicamente una ficha en este marcaje y está en la arista que va del vértice a al b. El vértice b está habilitado. Después de disparar el vértice b, las dos aristas que van de b al vértice a tienen una ficha en cada una de ellas y a está habilitado. Después de disparar el vértice a nuevamente se regresa al marcaje original. Así, el número de fichas en la gráfica cambia de 1 a 2 y nuevamente toma el valor de 1; pero el conteo de fichas en cada uno de los dos circuitos simples sigue siendo 1.*

Se dice que un marcaje es *vivo* si todo vértice está habilitado o puede ser habilitado mediante una secuencia de disparos.

Teorema 3.1 *Un marcaje es vivo si y sólo si el conteo de fichas de todo circuito dirigido es positivo.*

Demostración

Si el conteo de fichas de algún circuito dirigido es cero, ningún vértice en este circuito puede ser habilitado; ya que el conteo de fichas no cambia si los otros vértices se disparan (Lema 3.1), ningún vértice en este circuito puede habilitarse mediante disparos.

Ahora asúmase que el conteo de fichas de todo circuito dirigido es positivo. Sea v cualquier vértice de la gráfica. Considérense las aristas libres de fichas que entran a v. Si no hay, el vértice está habilitado. Si no, considérense los vértices de los cuales emanan estas aristas. Si cada uno de aquéllos está inmediatamente

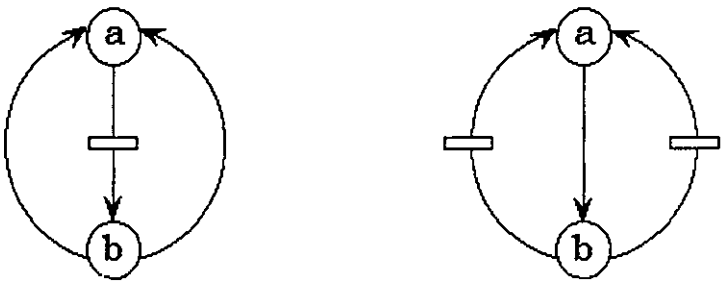


Figura 3.1: Ejemplo de una gráfica marcada

habilitado, entonces v estará habilitado después de que cada uno de ellos se dispare. Si algunos no lo están, considérense las aristas libres de fichas que entran a ellos, etc. Si se continúa con este retroceso, se está seleccionando una subgráfica de G, la cual consiste de v, las aristas libres de fichas que entran a v, los vértices de los cuales estas aristas emanan, las aristas libres de fichas que entran a ellos, etc. El proceso debe terminar puesto que G es finita. Ahora, esta subgráfica debe estar libre de circuitos puesto que no hay circuitos dirigidos libres de fichas. Así, la subgráfica debe tener al menos un vértice que no tiene aristas de entrada que pertenezcan a la subgráfica. Este vértice está habilitado en el marcaje presente de G. Después de dispararlo, la subgráfica de retroceso libre de fichas de v es reducida en un vértice. Repitiendo este proceso, es posible habilitar a v.

■

Corolario 3.1 *Un marcaje que está vivo sigue vivo después de disparar.*

Demostración

Puesto que el conteo de fichas de los circuitos no cambia con los disparos (Lema 3.1), y si un marcaje está vivo, entonces el conteo de todos los circuitos dirigidos es positivo (Teorema 3.1); este conteo seguirá siendo positivo después de disparar. Por el Teorema 3.1 el marcaje permanece vivo.

■

Se dice que un marcaje es *seguro* si ninguna arista tiene asignada más de una ficha, y ninguna secuencia de disparos trae dos o más fichas a una arista.

Teorema 3.2 *Un marcaje es seguro si y sólo si toda arista en la gráfica está en un circuito dirigido con conteo de fichas de 1.*

Demostración

Si para toda arista existe un circuito del cual esta arista forma parte, con exactamente una ficha en éste, entonces por el Lema 3.1 el conteo de fichas de este circuito sigue siendo 1 y nunca habrá dos o más fichas en esta arista.

Asúmase que existe una arista e, $a \xrightarrow{e} b$, tal que todo circuito dirigido que pasa a través de ésta tiene conteo de fichas de 2 o más. Se quiere demostrar que mediante una secuencia propia de disparos se puede poner dos fichas en e. Si no hay fichas en e, se retrocede a la subgráfica libre de fichas, empezando en el vértice a, como en la demostración del Teorema 3.1. De la misma forma que la demostración del Teorema 3.1, es posible habilitar el vértice a y dispararlo. Esto pone una ficha sobre e. Se repite esta construcción. Nuevamente la subgráfica libre de fichas de retroceso desde a no incluye a b, ya que esto implicaría la existencia de un circuito con conteo de fichas de 1 a través de e. Así, es posible disparar a nuevamente sin disparar b y, por lo tanto, colocar una segunda ficha sobre e. Por lo tanto, el marçaje inicial no es seguro.

■

Corolario 3.2 *Si una gráfica tiene un marcaje vivo y seguro, entonces para toda arista de la gráfica es posible encontrar un circuito que pasa a través de la arista.*

Esta es una consecuencia inmediata del Teorema 3.2. Esta consecuencia elimina la posibilidad de tener sumideros, fuentes o cualquier arista de separación en una gráfica, a la que se le puede asignar con un marcaje vivo y seguro.

Teorema 3.3 *Si la gráfica subyacente*[1] *de una gráfica dirigida $G(V, E)$ es conexa, y si a G se le puede asignar un marcaje vivo y seguro, entonces G es fuertemente conexa.*

Demostración

Asúmase que la gráfica subyacente de G es conexa y que a G se le puede asignar un marcaje vivo y seguro. Si G no es fuertemente conexa, entonces existen dos vértices a y b tales que no existe una trayectoria dirigida de a a b en G. Sea A el conjunto de todos los vértices, incluyendo a, que son alcanzables desde a. Puesto que $A \neq V$ y puesto que la gráfica subyacente es conexa, debe existir un vértice a' en A y un vértice b' en $V - A$ que están conectados por una arista e. La arista e debe estar orientada $b' \xrightarrow{e} a'$, o b' pertenecería a A. Sin embargo, el Corolario 3.2 implica la existencia de una trayectoria dirigida de a' a b', lo cual contradice el hecho de que $b' \in V - A$. Así, la gráfica debe ser fuertemente conexa.

■

En resumen, los Teoremas 3.1, 3.2 y 3.3 dicen que un marcaje dado de una gráfica es vivo y seguro si y sólo si ningún circuito dirigido está libre de fichas y a través de toda arista existe un circuito con conteo de fichas de 1. También si existe un marcaje vivo y seguro la gráfica es fuertemente conexa. Pero ¿existe un marcaje vivo y seguro para toda gráfica fuertemente conexa?. La respuesta es el contenido del siguiente teorema.

Teorema 3.4 *Para toda gráfica dirigida finita y fuertemente conexa existe un marcaje vivo y seguro.*

Demostración

Se puede encontrar un marcaje vivo simplemente poniendo una ficha en cada arista. Por el Teorema 3.1 este marcaje es vivo. Ahora se puede usar la técnica desarrollada en las pruebas de los Teoremas 3.1 y 3.2 para cambiar el marcaje hasta

[1] Esto significa la gráfica que resulta de G ignorando las direcciones.

que éste sea seguro, sin cambiar la cualidad de estar vivo. Asúmase que para una arista dada el mínimo conteo de fichas de los circuitos dirigidos que pasan a través de ella es $k > 1$. Es posible escribir una secuencia de disparos que traerá k fichas a la arista. Quitando $k - 1$ de ellas ningún circuito se vuelve libre de fichas. Esto puede repetirse mientras existan aristas para las cuales no hay ningún circuito cuyo conteo de fichas sea 1.

■

En el trabajo de Even, Commoner, Holt y Pnueli [Even 1971] se mencionan algunos otros resultados sobre éstas gráficas. Por ejemplo:

- En una gráfica fuertemente conexa, un marcaje M puede derivarse mediante una secuencia de disparos de un marcaje M' si y sólo si tienen conteos de fichas para los circuitos idénticos, es decir, para todo C $\langle M' \mid C\rangle = \langle M \mid C\rangle(\langle M \mid C\rangle$ es el conteo de fichas que el marcaje M coloca en el circuito C).
- En una gráfica fuertemente conexa si un marcaje M' puede generar a M (a través de una secuencia de disparos), entonces M puede generar a M'.

3.2 Gráficas marcadas estocásticas

Las gráficas marcadas son una subclase de las redes de Petri, también llamadas redes de Petri libres de decisión o gráficas de eventos. Existen dos tipos de eventos en una gráfica marcada: el procesamiento realizado en los vértices, y el retardo de transmisión de las fichas a través de las aristas. La duración de los eventos puede expresarse por un número no negativo o por una variable aleatoria. Para el caso de las cotas que se usarán en este trabajo se considera que las duraciones de los eventos son variables aleatorias independientes, exponenciales e idénticamente distribuidas.

Las gráficas marcadas junto con el factor del tiempo permiten la evaluación del desempeño de sistemas concurrentes. Por ejemplo, modelos de cómputo paralelo, sistema de cómputo distribuido, sistemas de manufactura, algoritmos distribuidos.

La complejidad de un algoritmo distribuido se calcula usualmente suponiendo que los tiempos de procesamiento y retardos de transmisión son iguales a una constante que representa una cota superior de estos tiempos. En el trabajo desarrollado por Sergio Rajsbaum y Moshe Sidi [Rajsbaum, Sidi 1994] se estudia el efecto en la

tasa de la red de que los tiempos de procesamiento y retardos de transmisión sean aleatorios y se comparan los resultados con el caso determinista del estudio de Simon Even y Sergio Rajsbaum [Even, Rajsbaum] en el que los tiempos de procesamiento así como los retardos de transmisión son constantes o acotados.

La medida de desempeño principal es la tasa de cómputo $R(v)$, es decir, el número promedio de pasos computacionales (disparos) de un vértice v por unidad de tiempo. Se sabe que para gráficas fuertemente conexas, $R(v)$ es la misma para todo vértice v. Se demostró por Michael K. Molloy [Molloy 1982] que la tasa puede calcularse analizando una cadena de Markov. Sin embargo este método de cálculo de la tasa es muy ineficiente porque, en general, el tamaño de la cadena de Markov es muy grande. Por ejemplo, el número de estados de la cadena que corresponde a la gráfica marcada completa con una ficha en cada arista es $2^{|n|} - 1$.

A continuación, usando ecuaciones de recurrencia se derivan cotas calculables en la tasa de gráficas marcadas fuertemente conexas. El material que aquí se presenta proviene principalmente de los trabajos de Sergio Rajsbaum y Moshe Sidi [Rajsbaum, Sidi 1994] y de Sergio Rajsbaum [Rajsbaum 1991]; se presentará algunos resultados que se utilizarán para obtener las cotas de la tasa de la red para el ciclo dirigido y la gráfica completa.

3.2.1 El modelo

Sea $G = (V, E)$ una gráfica finita dirigida y fuertemente conexa. Un marcaje s, es una función de E a los enteros no negativos y que representa el estado de la gráfica, donde $s(e)$ es el número de fichas en la arista e. Una gráfica marcada $GM = (G, s_0)$ consiste de una gráfica G y un marcaje inicial s_0. Se asume que GM está libre de *deadlock*, es decir, que cada vértice dispara un número infinito de veces. Esto es equivalente a asumir que todo ciclo tiene al menos una ficha en s_0.

En una gráfica marcada estocástica $s(e)$ representa el número total de fichas en la arista e, las fichas que viajan a través de e más las fichas almacenadas en un *buffer* FIFO[2] (primero en entrar, primero en salir) al final de e. La operación de disparo de un vértice v empieza en el momento en el cual hay al menos una ficha en el *buffer* de cada arista de entrada. Después de un *tiempo de procesamiento*, instantáneamente la primera ficha del *buffer* de cada arista de entrada se remueve y se envía una ficha a través de cada arista de salida de v. En este momento termina el disparo.

[2]First In, First Out.

Inicialmente todos los vértices se encuentran en estado de reposo, en el cual no hay fichas en tránsito y no se realizan cálculos. Una vez que un vértice deja su estado de reposo, opera en fases que se describen a continuación. Asúmase que en un tiempo arbitrario $t(v)$, el vértice v deja el estado de reposo y entra en su primer tiempo de procesamiento PS_0. Entonces, v permanece en PS_0 por $\tau_0(v)$ unidades de tiempo y transita a su primer estado de espera, WS_0. De ahora en adelante, sean PS_k y WS_k, $k \geq 0$, el estado de procesamiento y espera para la k-ésima fase.

Las reglas de transición entre los estados son las siguientes: Si un vértice v transita del estado PS_k a WS_k, manda una ficha en cada una de sus ligas de salida. Estas fichas se denotan por M_k. Cuando v manda sus M_k fichas, se dice que v ha completado su k-ésimo paso de procesamiento.

Si un vértice v está en el estado WS_k y ha recibido una ficha en cada una de sus aristas de entrada, borra una ficha de cada una de sus aristas de entrada, transita al estado PS_{k+1}, permanece ahí por τ_{k+1} unidades de tiempo y entonces transita al estado WS_{k+1}. De otra manera (al menos una de sus aristas de entrada no ha recibido fichas aún), el vértice v permanece en el estado WS_k hasta que recibe una ficha de cada uno de sus vecinos de entrada, y entonces opera como fue descrito anteriormente.

Los estados PS_k, $k \geq 0$, corresponden a los pasos de cómputo durante la k-ésima fase. Los tiempos de procesamiento, $\tau_k(v)$, corresponden al tiempo que le toma al vértice v completar el k-ésimo paso computacional. Los tiempos de procesamiento $\tau_k(v), k \geq 0, v \in V$, son variables aleatorias positivas de valores reales definidas sobre un espacio de probabilidad. Los estados WS_k corresponden a la espera de v hasta la terminación de la k-ésima fase de todos sus vecinos de entrada.

Para $k \geq 0$, sea $t_k^G(v)$ el k-ésimo tiempo de terminación, es decir, el tiempo al cual el vértice v envía las fichas M_k en la red G. Sea $N^G(v)$ (o simplemente $N(v)$)el conjunto de entrada de un vértice v en G, el conjunto de vértices en G que tienen una arista a v, incluyendo a v, esto es $N(v) \triangleq \{u : u \to v \in E\} \bigcup \{v\}$. Con esta notación, la operación del vértice $v \in V$ es como sigue. Una vez que v ha enviado la ficha M_k en el tiempo $t_k(v)$, espera hasta que todos los vértices con una arista hacia él envíen la ficha M_k ,y empieza el $(k+1)$-ésimo paso computacional; esto es, después del máximo de $t_k(u), u \in N(v)$, inicia el $(k+1)$-ésimo paso computacional, el cual toma $\tau_{k+1}(v)$ unidades de tiempo, y entonces envía M_{k+1}. De este modo, la

evolución del modelo puede describirse por las siguientes recursiones:

$$\begin{aligned} t_0(v) &= t(v) + \tau_0(v) \\ t_{k+1}(v) &= max_{u \in \mathcal{N}(v)}\{t_k(u)\} + \tau_{k+1}(v),\ k \geq 0 \end{aligned} \tag{3.1}$$

Los tiempos de terminación tienen una interpretación en la teoría de gráficas. Para un vértice v, sea $S_k(v)$ el conjunto de todas las trayectorias dirigidas de longitud k que terminan en v, suponiendo que cada vértice tiene un arco hacia él mismo (una arista $v \to v$). Para $k = 0$, la única trayectoria de longitud 0 que termina en v consiste de v. Para una trayectoria $P_k = v_0 \to v_1 \to \cdots \to v_k (= v)$, sea $T(P_k) \triangleq t(v_0) + \sum_{i=0}^{k} \tau_i(v_i)$, y $T(S_k(v)) \triangleq \{T(P) : P \in S_k(v)\}$. Así, $T(S_k(v))$ es un conjunto de variables aleatorias; cada una es la suma de $k+1$ variables aleatorias. Nótese que estas variables aleatorias no son independientes, aún si las $\tau_i(v)$ son independientes. El cálculo de $t_k(v)$ se puede hacer de la siguiente manera:

Teorema 3.5 *Para todo* $v \in V, k \geq 0$, $t_k(v) = max\ T(S_k(v))$.

Demostración

Por inducción sobre k. Para $k = 0$ la única trayectoria de longitud 0 hacia v es v, es decir, $S_0(v) = \{v\}$ y $T(S_0(v)) = \{t(v_0) + \tau_0(v)\}$. Así,

$$t_0(v) = max\ T(S_0(v)) = t(v) + \tau_0(v).$$

Se supone que se cumple para $k \geq 0$. De la Ecuación 3.1 tenemos

$$t_{k+1}(v) = max_{u \in \mathcal{N}(v)}\ \{t_k(u)\} + \tau_{k+1}(v).$$

Por la hipótesis de inducción,

$$t_{k+1}(v) = max_{u \in \mathcal{N}(v)}\ \{max\ T(S_k(u)) + \tau_{k+1}(v)\},$$

Lo que da el resultado deseado

$$t_{k+1}(v) = max\ T(S_{k+1}(v)).$$

■

3.2.2 Las medidas de ejecución

Existen varias medidas de ejecución, unas de las más importantes son los tiempos de terminación $t_k(v), k \geq 0, v \in V$. Una medida de ejecución de interés es el conteo de procesos $N_t^G(v)$ (o simplemente $N_t(v)$), asociada con cada vértice v y definida por

$$N_t(v) \triangleq sup\ \{k : t_k(v) \leq t\},$$

esto es, $N_t(v)$ es el número de pasos computacionales (menos 1) completados por v hasta el tiempo t, o el índice más alto de una ficha M_k que ha sido enviada por v hasta el tiempo t. De la misma manera, $N_t \triangleq \sum_{v=1}^{n} N_t(v)$ denota el número total de pasos computacionales (menos n) ejecutados en la red hasta el tiempo t. La siguiente afirmación indica que ningún vértice puede avanzar (en términos de pasos computacionales ejecutados) mucho más adelante que cualquier otro vértice.

Afirmación 3.1 *Sea d el diámetro de una gráfica G dirigida y fuertemente conexa. Entonces para todo $u, v \in V$ y $t \geq 0$, $|N_t(u) - N_t(v)| \leq d$.*

Demostración

Denótese por l la longitud de una trayectoria dirigida simple de u a v. Un argumento inductivo simple sobre l demuestra que el hecho de que la última ficha enviada por u hasta el tiempo t es $M_{N_t(u)}$, implica que $N_t(v) \leq N_t(u) + l$. Así, $N_t(v) - N_t(u) \leq l \leq d$. El mismo argumento para una trayectoria simple de v a u demuestra que $N_t(u) - N_t(v) \leq d$.

■

Otra medida de ejecución importante es la *tasa de cómputo*, $R^G(v)$, (o simplemente $R(v)$) del vértice v en la red G y está definida por

$$R(v) \triangleq lim_{t\to\infty} \frac{N_t(v)}{t}$$

siempre que el límite existe. Nótese que $R(v)$ es un número, no una variable aleatoria. De manera similar, la tasa de cómputo de la red está definida por

$$R \triangleq lim_{t\to\infty} \frac{N_t}{t}$$

La Afirmación 3.1 implica que para todo $u, v \in V$, $R(u) = R(v)$, y por lo tanto $R = nR(v)$.

3.2.3 Distribuciones de probabilidad generales

En esta sección se compara el desempeño de redes diferentes con distribuciones generales de los tiempos de procesamiento, se demuestra que agregar aristas a una gráfica con una topología arbitraria, hace más lenta la operación de cada vértice en la gráfica; en el trabajo de Sergio Rajsbaum y Moshe Sidi [Rajsbaum, Sidi 1994] se comparan las tasas de diferentes gráficas usando empotramientos en gráficas. Además se comparan gráficas con la misma topología pero diferentes distribuciones de los tiempos de procesamiento. En particular, se demuestra que el determinismo maximiza la tasa y la distribución exponencial la minimiza.

El siguiente teorema demuestra que agregar aristas a una red con una topología arbitraria retarda la operación de cada vértice en la red.

Teorema 3.6 *Sea una gráfica $G(V,E)$ y $E' \subset V \times V$ un conjunto de aristas dirigidas. Sea $H(V, E \cup E')$ la gráfica obtenida de G agregando las aristas E'. Asúmase que el vértice v, $1 \leq v \leq n$ despierta en G y H al mismo tiempo $t(v)$. Para toda realización de las variables aleatorias $\tau_k(v)$, $k \geq 0$, $1 \leq v \leq n$, se cumple la siguiente desigualdad*

$$t_k^G(v) \leq t_k^H(v),$$

para todo $k \geq 0$, $1 \leq v \leq n$.

Demostración

La demostración es por inducción sobre k. La base de la inducción es

$$t_0^G(v) = t(v) + \tau_0(v) = t_0^H(v).$$

La hipótesis de inducción es $t_k^G(v) \leq t_k^H(v)$. Se necesita demostrar que $t_{k+1}^G(v) \leq t_{k+1}^H(v)$. De la Ecuación 3.1 se tiene que

$$t_{k+1}^G(v) = max_{u \in \mathcal{N}^G(v)}\{t_k^G(u)\} + \tau_{k+1}(v),\ t_{k+1}^H(v) = max_{u \in \mathcal{N}^H(v)}\{t_k^H(u)\} + \tau_{k+1}(v).$$

Puesto que $\mathcal{N}^G \subset \mathcal{N}^H$, se tiene que

$$max_{u \in \mathcal{N}^G(v)}\{t_k^G(u)\} \leq max_{u \in \mathcal{N}^H(v)}\{t_k^H(u)\}$$

y por lo tanto $t_{k+1}^G \leq t_{k+1}^H$ para todo v.

■

Corolario 3.3 *Bajo las condiciones del (Teorema 3.6) tenemos que* $N_t^G(v) \geq N_t^H(v)$ *y* $R_t^G(v) \geq R_t^H(v)$ *(cuando existe el límite) para todo* $v \in V$. *También* $N_t^G \geq N_t^H$.

Es importante hacer las siguientes observaciones:

1. Nótese que no se hizo ninguna suposición acerca de las variables aleatorias $\tau_k(v)$. En particular no necesitan ser independientes.

2. Esta demostración implica que la variable aleatoria N_t^G es estocásticamente mayor que N_t^H, es decir $Prob\{N_t^G \geq \alpha\} \geq Prob\{N_t^H \geq \alpha\}$ para toda α.

3. Lo anterior implica que si se inicia con un circuito dirigido (la gráfica fuertemente conexa con el menor número de aristas) y se agregan aristas sucesivamente, se obtiene una gráfica completa sin incrementar la tasa.

Se puede usar la teoría de empotramientos en gráficas para comparar el comportamiento y tasas de gráficas diferentes. Un empotramiento de una gráfica G en una gráfica H se especifica por una asignación uno a uno de los vértices de G a los vértices de H : $\alpha : V_G \to V_H$ y un ruteo ρ de cada arista de G a una trayectoria diferente en H : $\rho : E_G \to Trayectorias(H)$. La dilatación de un empotramiento es el monto máximo que el ruteo ρ *estrecha* cualquier arista de G:

$$dilatacion(\alpha, \rho) = max_{u \to v \in E_G} longitud(\rho(u \to v))$$

También es posible comparar redes $G(V, E)$ y $H(V, E)$ con la misma topología pero que operan con distribuciones de las variables aleatorias $\tau_k(v)$ diferentes. Para esto se supone que los tiempos de procesamiento $\tau_k^G(v)$, $k \geq 0$, $v \in V$ son independientes y tienen media finita $E[\tau_k^G(v)] = \lambda_v^{-1}$.

Se dice que λ_v como la tasa potencial de v (esta sería la tasa de v si no tuviera que esperar fichas de sus vecinos de entrada). Los tiempos de procesamiento de H se distribuyen como en G excepto por un subconjunto $V' \subset V$ de vértices, para los cuales se supone que los tiempos de procesamiento son deterministas, es decir, $\tau_k^H(v) = \lambda_v^{-1}$, $v \in V'$, para $k \geq 0$. Sean $\tau_k^H(v) = \tau_k^G(v) = \tau_k(v)$, $k \geq 0$, $v \ni V'$ realizaciones específicas de las variables aleatorias en G. Se asume que los vértices se despiertan al mismo tiempo en ambas redes.

Teorema 3.7 *Bajo las condiciones anteriores se tiene*

$$t_k^H(v) \leq E[t_k^G(v)],$$

para todo vértice v y $k \geq 0$. La esperanza se toma sobre las distribuciones respectivas de los tiempos de procesamiento de los vértices de G en V'.

Demostración La demostración es por inducción sobre k: para el caso base $k = 0$ obsérvese que:

$$E[t_0^G(v)] = t_0^G = t(v) + \tau_0(v) = t_0(v)$$

, para $v \ni V'$

$$E[t_0^G(v)] = t(v) + \lambda_v^{-1} = t_0^H$$

, para $v \in V'$ La hipótesis de inducción es $t_k^H(v) \leq E[t_k^G(v)]$ y es necesario demostrar que $t_{k+1}^H(v) \leq E[t_{k+1}^G(v)]$ para toda $v \in V$

De la Ecuación 1 se tiene que

$$t_{k+1}^G(v) = max_{u \in N(v)} t_k^G(u) + \tau_{k+1}^G(v),$$

para $v \in V$. La desigualdad de Jensen implica:

$$E[t_{k+1}^G(v)] \geq max_{u \in N(v)} E[t_k^G(u)] + E[\tau_{k+1}^G(v)]$$

. Por hipoótesis de inducción

$$E[t_{k+1}^G(v)] \geq max_{u \in N(v)} t_k^H(u)] + E[\tau_{k+1}^G(v)] = t_{k+1^H(v)}$$

, porque $E[\tau_{k+1}^G(v)] = \tau_{k+1}(v)$ para $v \ni V'$ y $E[\tau_{k+1}^G(v)] = \lambda_v^{-1}$, para $v \in V'$

Cabe señalar que el teorema se cumple si los tiempos de procesamiento de los vértices de H en V' son deterministas, pero no necesariamente tienen que ser los mismos para todo k.

Distribuciones exponenciales

En esta sección se enuncia algunos resultados suponiendo que los tiempos de procesamiento $\tau_k(v)$, $k \geq 0$, $v \in V$ se distribuyen exponencialmente, son independientes y con media λ^{-1}. A continuación se presentan algunos resultados para topologías generales y se mencionan los cálculos para el ciclo dirigido y la gráfica completa.

Denótese con $d_+(v)$ el número de aristas que salen de v más 1 y $d_-(v)$ como el número de aristas que entran a v más 1, sean

$$\begin{aligned} \triangle_+ &= max_{v \in V} d_+(v) \qquad \triangle_- = max_{v \in V} d_-(v) \\ \delta_+ &= min_{v \in V} d_+(v) \qquad \delta_- = min_{v \in V} d_-(v) \end{aligned}$$

Lema 3.2 (Cota inferior) :

i) Para todo $k \geq 0$ existe un vértice $v \in V$ para el cual

$$E[t_k(v)] \geq max_{u \in V} t(u) + \lambda^{-1}[1 + k \log \delta_+]$$

ii) Para todo $k \geq 0$, y todo $v \in V$ se cumple lo siguiente

$$E[t_k(v)] \geq min_{u \in V} t(u) + \lambda^{-1}[1 + k \log \delta_-]$$

Lema 3.3 (Cota superior) :

i) Para todo $k \geq 1$, para todo v,

$$E[t_{k-1}(v)] \leq max_{u \in V} t(u) + \frac{4}{\lambda}(1 + k \log \Delta_-)$$

ii) Para todo $k \geq 1$, y para todo v,

$$E[t_{k-1}(v)] \leq max_{u \in V} t(u) + \log \mid V \mid + \frac{4}{\lambda}(1 + k \log \Delta_+)$$

El siguiente teorema se obtiene usando las cotas para los valores esperados de los $t_k(v)$ y nos da las cotas superior e inferior de la tasa de la red.

Teorema 3.8

$$\frac{\lambda}{4 \log min(\Delta_+, \Delta_-)} \leq R_v \leq \frac{\lambda}{\log max(\delta_+, \delta_-)}$$

Como resultado del (Teorema 3.8) se obtienen las siguientes cotas para la tasa del ciclo dirigido C_n ($\Delta = \delta = 2$) y para la gráfica completa K_n ($\Delta = \delta = n$)tomando a n como el número de vértices:

$$0.36\lambda \leq R^{C_n}(v) \leq \lambda$$

$$\frac{\lambda}{4 \log n} \leq R^{K_n}(v) \leq \frac{\lambda}{\log n}$$

Para calcular los valores exactos de las tasas de C_n y K_n se define una cadena de Markov asociada con la gráfica. Esta cadena se puede denotar como $X(t) = (X_1(t), X_2(t), \ldots, X_m(t))$, donde $X_i(t)$ es el número de fichas guardadas en el buffer de la arista i en el tiempo t, y m es el número de aristas en la gráfica. El estado en el cual $X_i(0) = 1, 1 \leq i \leq m$ se denota por s_0.

Es posible demostrar que el número de estados,denotado por N, en la cadena de Markov es finito. Esto ocurre porque una transición de la cadena no cambia el número total de fichas en un circuito de la gráfica (Lema 3.1). Si la gráfica es fuertemente conexa la cadena de Markov es irreductible, es decir, que todos los estados de la cadena se comunican entre sí. Por lo tanto, las probabilidades límite $P_i, 1 \leq i \leq N$ de los estados s_i de la cadena existen, son todas positivas y la suma es igual a 1. Sin embargo N puede ser exponencial sobre n; por lo tanto no es factible calcular directamente la tasa resolviendo la cadena de Markov. Mediante un enfoque combinatorio es posible solucionar esta cadena para dos clases de gráficas.

Sea G_X el diagrama de transición de la cadena de Markov X. Considérese un árbol de búsqueda por capas BFS de G_X con raíz en s_0. El nivel $L(v)$ de un vértice v es la distancia de s_0 a v. Así, $L(s_0) = 0$. Denótese con L_i, $i \geq 0$ al conjunto de vértices en el nivel i y con L el número de niveles de G_X.

El siguiente teorema presenta el valor exacto de la tasa del ciclo dirigido y nos dice que cada vértice en C_n trabaja al menos a la mitad de su tasa potencial, sin importar el valor de n.

Teorema 3.9 *La tasa $R(v)$ de un vértice en C_n es*

$$R(v) = \lambda \left[1 - \frac{n-1}{2n-1}\right] \sim \frac{\lambda}{2}$$

$$N = \frac{(2n-1)!}{n!(n-1)!}$$

y la probabilidad límite de cada estado es $1/N$ donde N es el número de estados en la cadena asociada.

Demostración

Si M es el número de estados en los cuales hay al menos una ficha en cada arista, entrando a un vértice v, entonces la tasa de ejecución será M/N veces la tasa de disparo esperada. Esto se produce porque v será habilitado cuando tenga más de

cero fichas en su arista de entrada y, puesto que todos los estados tiene la misma probabilidad, el porcentaje de tiempo que está habilitado es simplemente M/N.

El número de maneras de poner n objetos en k lugares es

$$P(n,k) = \frac{(n+k-1)!}{n!(k-1)!}$$

En el caso del ciclo $N = P(n,n)$ y $M = N - P(n,n-1)$. Así

$$\frac{M}{N} = 1 - \frac{n-1}{2n-1}$$

■

Sea K_n una gráfica completa con n procesadores. Un estado está en el nivel l, $0 \leq l \leq n-1$, si es alcanzable desde s_0 por el disparo de l vértices. Se denota con $P(l)$ a la probabilidad límite de un estado en el nivel l.

Teorema **3.10** *La tasa de un vértice en K_n es*

$$R(v) = \lambda / \sum_{i=1}^{n} \frac{1}{i} \sim \frac{\lambda}{\log n}$$

Demostración

Considérese una cadena de Markov T. La raíz de T, s_0, es el estado en el cual cada arista tiene una ficha. Un estado s tendrá un hijo por cada uno de los vértices habilitados en el estado s; un hijo de s corresponde al estado al que se llega de s por el disparo de uno de los vértices habilitados en el estado s.

En T el número de estados en el nivel l es $n!/(n-l)!$, porque cada vez que un vértice dispara no puede disparar nuevamente hasta que el resto de los vértices ha disparado. Así, el número N_T de estados en T es

$$N_T = \sum_{i=1}^{n} \frac{n!}{i!}$$

El número de estados en los cuales un vértice dado está habilitado en el nivel l, $hab(l)$ (las aristas del nivel l al nivel $l+1$), es

$$hab(l) = \frac{1}{n} \frac{n!}{(n-l-1)!}$$

porque que en el nivel l hay $n!/(n-l-1)!$ vértices habilitados, y por simetría cada vértice es habilitado el mismo número de veces en cada nivel.

Sea P_l^T la probabilidad límite de un estado de T en el nivel l. Se puede demostrar que $P_l^T = (n-l-1)!/K$, donde

$$K = \sum_{l=0}^{n-1} \frac{n!}{(n-l)!} P_l^T = n! \sum_{i=1}^{n} \frac{l}{i}$$

Así se obtiene que el porcentaje del tiempo que un vértice está habilitado es

$$ut = \sum_{l=0}^{n-1} ut(l) = \frac{1}{\sum_{i=1}^{n} \frac{1}{i}}$$

donde $ut(l) = hab(l) P_l^T$ y su tasa es λut.

■

Corolario 3.4 *Para una red* K_n

$$N = 2^n - 1$$

$$P_l = \frac{l!(n-l-1)!}{n! \sum_{i=1}^{n} \frac{1}{i}}$$

Demostración

Es posible que dos estados de T correspondan al mismo estado s de K_n. De hecho, si un estado de T se alcanza desde s_0 disparando una secuencia de vértices de longitud k, entonces todas las permutaciones $k!$ de vértices en esta secuencia constituyen una secuencia de disparos válida, la cual lleva al mismo estado s. Así, la probabilidad límite de un estado s al nivel l es:

$$Pl = l! P_l^T = \frac{l!(n-l-1)!}{n! \sum_{i=1}^{n} \frac{1}{i}}$$

El número total de estados diferentes en el nivel l es $n!/l!(n-l-1)!$ y el número total de estados diferentes es:

$$\sum_{l=0}^{n-1} n!/l!(n-l)! = 2^n - 1$$

Corolario 3.5 *Asintóticamente, la tasa de una red de n vértices está entre* $\lambda n/2$ *y* $\lambda n/\log n$.

De esta forma termina el Capítulo 3 y ahora será necesario utilizar las técnicas de reducción de varianza para analizar las clases especiales de gráficas presentadas al final de este capítulo .

Para llevar a cabo el análisis de estas gráficas mediante simulación, se podría usar alguna herramienta de simulación para las redes de Petri, puesto que, como ya se ha mencionado, estas gráficas son una subclase de las redes de Petri.

Para ver con mayor detalle cómo se calculan las probabilidades límite de la cadena de Markov asociada con una red de Petri, pueden consultarse los artículos de Michael K. Molloy [Molloy 1982] y de M. Ajmone Marsan [Marsan].

Para los resultados mencionados en este capítulo se supone que los tiempos de transmisión de las fichas es insignificante; si se desea ver qué sucede cuando no se consideran como insignificantes los tiempos de transmisión, es posible consultar el trabajo de Sergio Rajsbaum y Moshe Sidi [Rajsbaum, Sidi 1994].

Capítulo 4

Herramientas para la simulación

Debido que las Gráficas Marcadas Estocásticas son una subclase de las redes de Petri Estocásticas es posible utilizar las técnicas para el análisis de las redes de Petri estocásticas en las gráficas marcadas estocásticas.

Las redes de Petri estocásticas son un formalismo conveniente para calcular el desempeño de un sistema concurrente o para determinar si dicho sistema es correcto. Las redes de Petri definidas originalmente por Karl Addam Petri, no consideraban el tiempo. Esta extensión surgió posteriormente para permitir el análisis cuantitativo de un sistema concurrente.

El análisis necesario para verificar si un sistema es correcto se basa en análisis de invariantes o generando la gráfica de alcance. Para realizar el análisis del desempeño es necesario calcular la distribución de probabilidad de la Cadena de Markov asociada, la cual es una técnica analítica exacta. Un estado de la cadena de Markov asociada con la red de Petri está dado por un marcaje de la red; cada marcaje diferente corresponde a un estado de la cadena de Markov. Sin embargo, el tamaño del espacio de estados crece de manera exponencial.

Otra técnica para el análisis del desempeño de una red de Petri es la simulación. En particular se proponen las *técnicas de reducción de varianza* para realizar el análisis de una red de Petri mediante simulación.

Como se mencionó en el Capítulo 2 para el uso de las técnicas de reducción de varianza, es preciso tener el control sobre la generación de los números aleatorios, ya sea para la sincronización en la técnica de NAC o para generar el par de variables deseadas en el caso de la técnica de VA.

En las siguientes secciones se describen las características de las herramientas consideradas en este trabajo.

4.1 Generación y sincronización de las variables aleatorias

Para la generación de los valores aleatorios que representan la distribución exponencial es recomendable utilizar el método de la transformación de la inversa debido a que asegura la monotonicidad en el comportamiento de los números aleatorios con respecto a la generación de las variables aleatorias.

Este método consiste en que una vez que se obtuvo el número uniforme $(0,1)$, el número exponencial se obtiene calculando $X = F^{-1}(U)$.

En el caso de la distribución exponencial sea X una variable aleatoria con distribución exponencial con media λ. Su función de distribución es:

$$F(x) = \begin{cases} 1 - e^{-x/\lambda} & \text{si } x \geq 0 \\ 0 & \text{en otro caso} \end{cases}$$

y así tenemos que

$$F^{-1}(u) = -\lambda \ln(1-u)$$

De este modo, para obtener la variable aleatoria exponencial primero generamos $U \sim U(0,1)$ calculamos $1-U$ y hacemos $X = -\lambda \ln(1-U)$.

En el caso de las variables aleatorias que se generan para las gráficas se escoge $\lambda = 1/2$. Así la función que genera números de acuerdo con una distribución exponencial con media $\lambda = 1/2$ queda:

$$X = -1/2 \ln(1-u)$$

En la presentación posterior del código Java de la definición del retardo de una transición, se puede observar que efectivamente este es el método que se sigue para la generación de los valores de la distribución exponencial.

También debe señalarse que el generador de números aleatorios juega un papel muy importante en la aplicación de las técnicas de reducción de varianza. Como se vió en el Capítulo 2 se necesitan ciertas características deseables en los pasos intermedios de la generación de los variables aleatorias como la monotonicidad, la

posibilidad de tener diferentes flujos de números aleatorios para poder obtener una buena sincronización y, por último, otra característica importante de un generador de números pseudo-aleatorios es que se pueda reproducir un flujo de números, de otra forma sería imposible la aplicación de las técnicas de reducción de varianza por las razones que se expresaron en su momento.

4.2 Herramientas de simulación de redes de Petri

Hay una gran variedad de herramientas y con caracteríticas diferentes: para diferentes plataformas, con algún costo o gratuitas, que permiten animación o simulación, mediante la cual se puede realizar un análisis cualitativo o cuantitativo, que modelan redes de Petri coloreadas o con jerarquía y colas.

Estas herramientas permiten o exigen diferentes grados de intervención del usuario; mientras que existen algunas que son muy abiertas y permiten mucha flexibilidad, existen también los que son muy rígidas pero que presentan mucha facilidad en su uso. Para este trabajo fue necesario inclinarse por una herramienta que permitiera mayor flexibilidad y sobre todo aceso a la generación de las variables aleatorias que conducirían la simulación.

La selección de una herramienta de redes de Petri adecuada puede ser un proceso largo y complejo debido a la gran variedad de éstas que existen. Afortunadamente existen algunos trabajos de evaluación de dichas herramientas que facilitan esta tarea. En particular el trabajo de Harald Störrle [Störrle 1998] es muy informativo sobre las características de algunas de las herramientas que se evaluaron. De esta forma no fue necesario realizar una investigación exhaustiva sobre las herramientas que se encuentran disponibles y sólo se consideraron las que quedaron mejor ubicadas en la evaluación antes mencionada.

Se tomaron en cuenta principalmente dos criterios para que estas herramientas fueran consideradas en este trabajo:

- El primer criterio que se consideró con este conjunto reducido de herramientas fue el costo, descalificando el uso de las que no fueran de distribución gratuita.
- El segundo criterio seguido para seleccionar alguna herramienta dentro de la nueva y reducida lista fue que permitieran tener cierto control sobre la generación de las variables aleatorias para poder utilizar algunas de las técnicas de reducción de varianza.

De esta forma algunas herramientas muy buenas y que se encuentran bien colocadas dentro de la evaluación no pueden utilizarse para la aplicación de las técnicas de reducción de varianza por las razones que se describen a continuación.

La herramienta de Design CPN está enfocada a la redes de Petri coloreadas y también con jerarquía. Esta herramienta es la mejor ubicada en la evaluación antes mencionada; sin embargo no permite tomar el control de la generación de las variables aleatorias. Cabe mencionar que la documentación que existe sobre esta herramienta en Internet es excelente, está muy bien organizada y es abundante. La única posibilidad que nos permite es el establecimiento del valor de la semilla del generador de números aleatorios, pero esto resulta insuficiente.

La herramienta HQPN-Tool, a pesar de que permite realizar el análisis cuantitativo y cualitativo de una red de Petri con colas y jerárquicas, no nos permite aplicar alguna de las técnicas de reducción de varianza porque la forma en que calcula las probabilidades límite es un método analítico exacto, es decir, no permite llevar a cabo una simulación de una red de Petri.

Acerca de la herramienta PEP se puede decir que el criterio que la descalificó para considerarse como candidata para usar alguna de las técnicas de reducción de varianza, es que tampoco permite tomar el control sobre la generación de las variables aleatorias. Sin embargo existe un gran interés por parte de los creadores de la herramienta en extender sus funciones y en recibir opiniones y sugerencias por parte de los usuarios.

De este modo, la herramienta que mejores oportunidades ofrece para la simulación de las gráficas marcadas estocásticas mediante técnicas de reducción de varianza es Moses, la cual es una evolución de la herramienta CodeSign (que es la que aparece originalmente en la evaluación). Esta herramienta presenta las mejores oportunidades para la aplicación de algunas técnicas de reducción de varianza y a continuación se describirán algunas de sus características.

4.2.1 La herramienta Moses

La herramienta de Moses está basada en Java2, por lo tanto es posible desarrollar modelos que podrían ser analizados y simulados en distintas plataformas. Este punto es muy importante, puesto que algunos autores que escriben sobre el análisis experimental de algoritmos, sugieren entre otras cosas precisamente esto: realizar los experimentos en distintas plataformas.

A continuación se mencionarán algunas de sus características:

- Permite el modelado visual de sistemas de eventos discretos
- Soporta notaciones visuales (formalismos) heterogéneas
- Permite la creación simple de nuevos formalismos
- Cuenta con un ambiente de simulación, validación y evaluación
- Genera código Java
- Permite crear una plataforma de experimentación para:
  - la verificación de algoritmos
  - realizar técnicas de simulación
  - desarrollo y especificación de formalismos
- Produce modelos ejecutables.
- Favorece la creación de componentes reutilizables
- Cuenta con manejo de tiempo
- Permite visualización del comportamiento y los resultados
- Cuenta con interfaces para simuladores externos
- Cuenta con entrada y salida para escenarios de prueba

Existen diferentes niveles de interacción con Moses, dependiendo de la clase de tareas que se deseen realizar y para ello cuenta con tres diferentes tipos de usuario:

- Usuario
- Metausuario
- Programador

Cada tipo de usuario tiene un conjunto de facilidades que puede utilizar para la simulación del sistema que está interesado en analizar. La organización de las facilidades que Moses proporciona mediante tipos de usuario, permite evitar que, por ejemplo, un usuario que sólo desea utilizar algunos de los formalismos ya definidos para modelar un sistema tenga que aprender muchas de las características que tiene la herramienta. Por otro lado, si alguien quiere definir un nuevo formalismo, no es necesario que tenga que programar en Java; para ello cuenta con un conjunto de lenguajes definidos para estos propósitos y, por último, si alguien desea agregar nuevas características o conectar algunos componentes con otros programas de simulación puede hacerlo. A continuación se presentan las facilidades que se permiten para cada tipo de usuario.

El *usuario* puede realizar las siguientes tareas:

- Crear componentes del modelo en una variedad de notaciones gráficas y textuales.
- Simular y animar los componentes.
- Evaluar los resultados de la simulación usando una variedad de herramientas externas e internas.

El *metausuario*

- Define la sintaxis visual de una notación visual o textual
  - la parte textual se define por el lenguaje de expresión
- define su semántica
- define sus reglas de animación

El *programador*

- Puede conectar los componentes de Moses con otros ambientes de simulación.
- Puede usar algún componente como una aplicación independiente.
- Pueden incorporarse componentes de terceros al ambiente de Moses.

En el caso de que se esté interesado en usar Moses para la simulación de las gráficas marcadas estocásticas usando técnicas de reducción de varianza se tendrá que interactuar con la herramienta a los niveles de un *usuario* y de un *programador*, ya que con el conjunto de facilidades que proporciona para un *usuario* no es suficiente para poder aplicar dichas técnicas.

Debido a que en la definición de las redes de Petri sólo las transiciones están habilitadas para alojar un objeto de retardo de transmisión, las gráficas marcadas se modelaron de la siguiente forma:

- Los vértices de la gráfica son las transiciones en la red de Petri.
- Las aristas de la gráfica tendrán tres elementos que las representan: una liga entre la transición y un lugar (que funciona como *buffer*), el lugar y una liga entre el lugar y la transición a la que debe conectarse.

Si se desea modelar el hecho de que las aristas de una gráfica marcada estocástica presenten un retardo de transmisión, es posible definir un retardo en los lugares de la red de Petri modificando el formalismo de las redes de Petri estocásticas que Moses proporciona.

En el papel de usuario sólo sería necesario definir las gráficas que van a ser analizadas, es decir, definir la estructura de las mismas. Las gráficas se pueden definir recursivamente aprovechando esta facilidad de la herramienta.

La posibilidad de definir una gráfica de manera recursiva es importante, ya que de no contarse con ella la tarea de definir las gráficas resultaría casi artesanal. Por ejemplo, para el caso de la gráfica completa con 15 transiciones se tendría una gráfica como en la Fig. 4.1.

Utilizando las facilidades de Moses se puede definir la gráfica completa de una forma que permita tener una gráfica con un gran número de vértices. Para ello, se define la gráfica en dos niveles: el nivel de nodo y un nivel superior que repetirá la creación de objetos de tipo nodo tantas veces como sean determinadas por sus parámetros de creación.

Esta comunicación entre niveles de representación de un problema es posible gracias a la existencia de los *puertos*, los cuales permiten la salida o entrada de fichas u otros elementos a través de los componentes de un formalismo determinado. De esta forma es posible tener comunicación entre una red de Petri y una red de procesos.

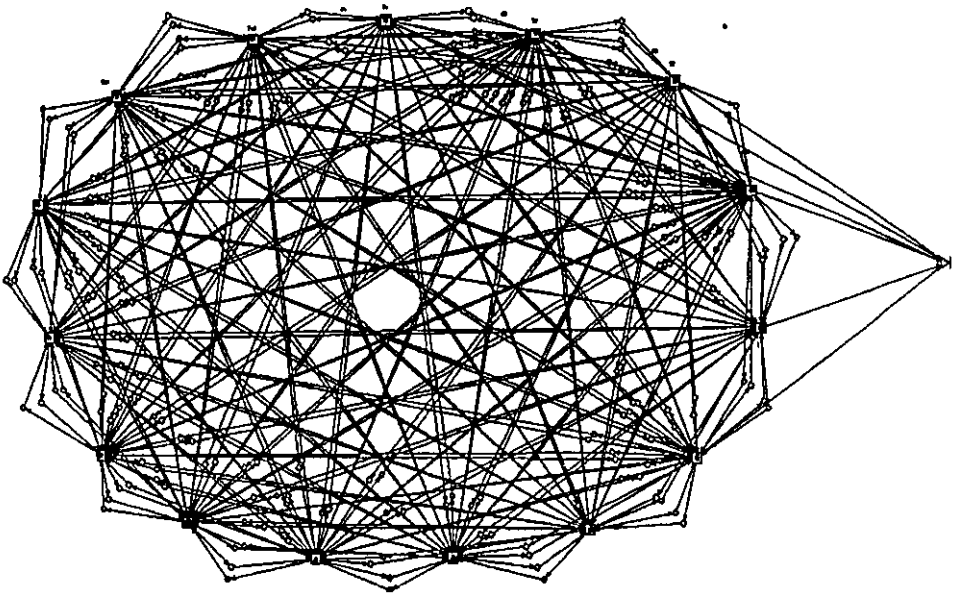


Figura 4.1: Gráfica completa con 15 transiciones.

Los elementos de un formalismo tienen un conjunto de atributos, los cuales pueden editarse. Por ejemplo, una transición tiene un atributo de retardo (*Delay*) y un atributo de función (*Function*) entre otros, un lugar tiene un atributo del número de fichas inicial (*Initial tokens*) y un atributo de nombre(*Name*).

Aunque la definición de un ciclo dirigido es más sencilla, se tiene el mismo problema de la gráfica completa cuando se desea tener una estructura con un número muy grande de vértices, la tarea de definir la estructura puede resultar muy onerosa. Por ejemplo en la Fig. 4.2 puede verse en primera instancia el elemento básico para la construcción del ciclo; consta de un puerto de entrada, para recibir fichas, de dos puertos de salida, uno para enviar una ficha para su vecino de salida y otro para enviar las estadísticas ya sea a un archivo o si se desea a un graficador en línea con el que cuenta Moses.

Un atributo importante de la transición es el atributo *Delay*, el cual sirve para

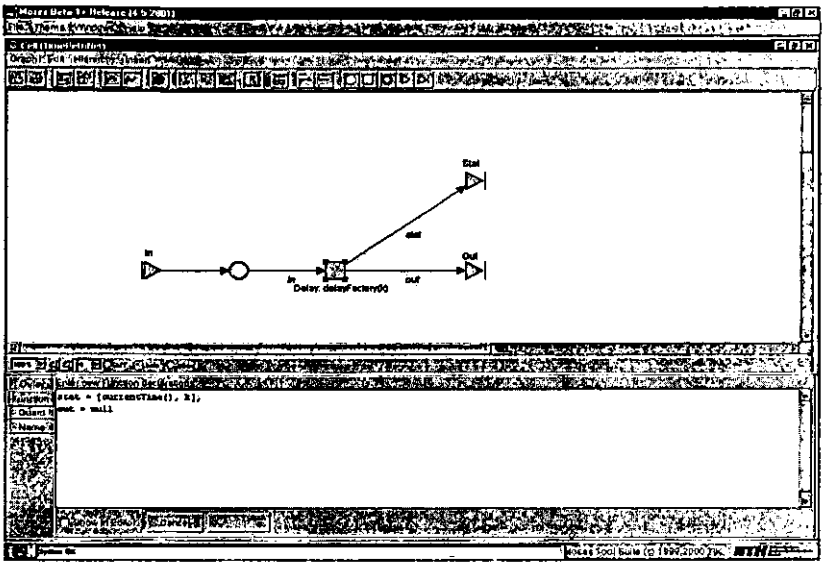


Figura 4.2: Definición recursiva del ciclo dirigido en Moses, elemento básico.

definir la función de distribución que se usará para generar los retardos de procesamiento, en nuestro caso se trata del parámetro *delayFactory* que será proporcionado en otro nivel superior de la definición del ciclo. Los parámetros de la simulación de un componente de Moses, se proporcionan a través de una clase del tipo *DinamicFiringDelay* en el componente de configuración de la simulación. Para poder emplear algunas de las técnicas de redución de varianza es posible definir alguna función de distribución mediante una clase de este tipo. Por ejemplo, en el caso de una distribución exponencial la clase de retardo del disparo es la siguiente:

```
package hades.PetriNet;
import java.util.*;
import hades.PetriNet.*;
public class DfdExponential extends AbstractDynamicFiringDelay {
```

```
    private double mean;
    public double   minFiringDelay() {
return 0;
    }
    public double   maxFiringDelay() {
return Double.POSITIVE_INFINITY;
    }
    public double   firingTime(double actTime, double now) {
double u = nextDouble();
return now - mean * Math.log(u);
    }
    public DfdExponential(double mean) {
this.mean = mean;
    }
   public DfdExponential(int mean) { this((double)mean); }
}
```

La definición tendría que hacerse en código del lenguaje java. De esta manera Moses proporciona un medio para extender las funciones que ofrece. Sin embargo, cabe mencionar que por el momento no se cuenta con la facilidad de establecer la semilla para generar el número aleatorio que servirá para generar el valor correspondiente a una distribución particular.

Como puede observarse se utiliza una notación de listas para definir el valor de las salidas de una transición, de tal forma que una ficha puede ser una lista con dos elementos, como el caso de la ficha que será liberada por la arista *stat*, o puede ser la lista nula. Es muy importante señalar que la lista que representa las fichas de salida o de entrada puede contener varios tipos de objetos, proporcionando un gran poder de modelado.

En el siguiente nivel de la representación del ciclo dirigido se tiene que se hace uso del formalismo de las redes de proceso, en este nivel se crearán tantos nodos del ciclo como se hayan solicitado mediante alguno de sus parámetros. Gracias a esta representación y a la posibilidad del uso de parámetros es posible definir redes con el tamaño y distribución de probabilidad que se desee.

En la Fig. 4.3 puede verse cómo el elemento básico es un objeto asociado a un proceso de la red de procesos y uno de sus atributos permite la creación de los elementos necesarios en función de su atributo *Parameters*.

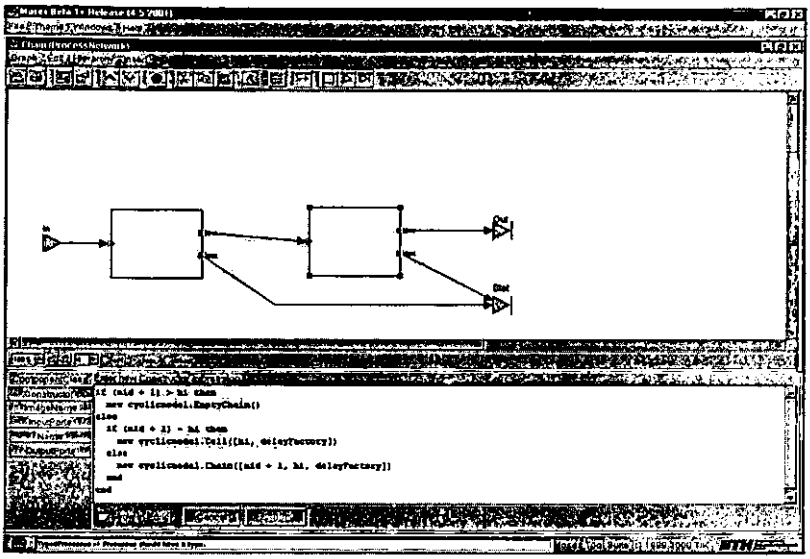


Figura 4.3: Definición recursiva del ciclo dirigido en Moses, elemento superior.

Además del ambiente de definición del modelo usando diferentes formalismos, Moses cuenta con un ambiente de simulación que permite varias facilidades.

Las características de la arquitectura de simulación son:

- Componentes de la simulación
  - componentes como objetos
  - jerarquía dinámica
  - firmas de componentes
- Contención y programación de eventos
  - flexible

  - facilita las estructuras de modelos dinámicos
  - reduce el control
- Simuladores eficientes en código duro para clases de modelos específicas
  - redes de procesamiento
  - redes de Petri
- DiscreteEventComponents
  - tienen entradas y salidas
    * MessageListeners, MessageProducers
    * Pueden tener firma
  - tienen un estado
  - se inicializan y pasan a un programador de eventos
- EventProcessors
  - ejecutan eventos y efectuan el cambio de estado
  - pueden programarse exactamente una vez
- Scheduler
  - arregla una cantidad de EventProcessor de acuerdo con su *timestamp*
  - inician la ejecución de primer EventProcessor

Además del ambiente de simulación, Moses permite el uso de expresiones en ciertas clases, por ejemplo, en el caso de las transiciones es posible tener funciones que tomen valores de sus aristas de entrada (esto es gracias que una ficha puede ser un objeto de cualquier clase y gracias a que las aristas tienen nombres que pueden usarse dentro de las expresiones)

El uso de expresiones en ciertas clases de la simulación agrega mucha expresividad a la herramienta. Existen varios lenguajes que se manejan dentro de la herramienta Moses y cada uno cumple con las siguientes funciones:

- Elan

  - cálculos dentro de los métodos
  - parte del procedimiento de verificación de sintaxis
  - configuración de la herramienta
- GTDL
  - especificación de la sintaxis
- OMA
  - definición semántica
  - configuración de la herramienta
- Otros
  - Scheme (cálculo dentro de los modelos)
  - Tcl/Tk/Jacl (configuración de la herramienta, cálculo)

A pesar de que no se cuenta por el momento con la facilidad de establecer la semilla para la generación de las variables aleatorias que conducen la simulación, uno de los autores de Moses está muy interesado en proporcionar esta característica.

Así se da por terminado este capítulo en el que se proporcionaron los elementos para realizar una simulación de dos casos particulares de gráficas marcadas estocásticas.

Capítulo 5

Conclusiones

En este capítulo se realiza una evaluación de la conveniencia de la aplicación de la experimentación para el análisis de las gráficas marcadas estocásticas y del uso de alguna herramienta de simulación de redes de Petri.

5.1 Simulación de las GME usando Moses

Como se pudo ver, utilizando las herramientas de simulación de redes de Petri se puede evitar la programación de un simulador particular para las gráficas marcadas. Sin embargo, existen algunas dificultades que hay que tomar en cuenta:

- Algunas de las herramientas son completamente cerradas y no permiten modificar o agregar algunas funciones.
- Existe la posibilidad de tener que realizar programación, para lo cual hay que comprender la filosofía de la herramienta que se está empleando.
- Puesto que se está evaluando el desempeño de las gráficas *per se*, es necesario que exista una forma eficiente para representar gráficas con un número grande de vértices.

Dentro del conjunto de herramientas disponibles Moses proporciona muchas facilidades no sólo para simular redes de Petri, sino que permite la simulación de sistemas complejos usando varios tipos de formalismos que inclusive se pueden usar

conjuntamente para modelar, simular y visualizar el comportamiento de algún sistema complejo.

Además, Moses cuenta con un componente mediante el cual es fácil visualizar la ejecución de algún objeto de un formalismo particular junto con la graficación, al tiempo de ejecución, de algunos valores numéricos de resultado. Esto, gracias a la existencia de *puertos* de entrada y salida que permiten el paso de objetos entre sus diferentes componentes.

También es posible crear un formalismo que represente a las gráficas marcadas de una forma natural y no se tenga que ajustar al modelo de redes de Petri. En cuanto a la representación gráfica de los elementos de los formalismos, Moses permite asignar las figuras que representan a cada uno de los objetos que son parte del mismo. Para que Moses pueda generar clases de un nuevo formalismo es necesario proporcionarle las reglas de sintaxis y de semántica, utilizando los lenguajes que se mencionaron en el Capítulo 4.

La aplicación de las técnicas de NAC y VA es posible de la siguiente manera: en el caso de la técnica de NAC hay que usar un flujo del generador de números aleatorios por cada una de las transiciones de la gráfica para conseguir una sincronización completa de los modelos a comparar y en el caso de las variables antitéticas es posible definir en Java un nuevo retardo exponencial antitético de disparo para las transiciones y usar la media aritmética en lugar de usar un retardo exponencial simplemente.

Una vez que se hubiera definido el formalismo de las gráficas marcadas, la aplicación de otras técnicas de reducción de varianza como Variables de Control o Esperanza Condicional podría llevarse a cabo. Sin embargo, con lo que se tiene actualmente también es posible realizarlo con el único inconveniente de que la representación de las gráficas diferentes del ciclo y la completa puede ser una tarea muy laboriosa.

Hay que recordar que a pesar que estas técnicas se han aplicado desde hace tiempo en el campo de la simulación, están demostrando su factibilidad en el análisis de algoritmos y para una mejor referencia pueden revisarse los trabajos de Catherine McGeoch y Bernard Moret.

Acerca de la simulación de redes de Petri en general, existe otra forma de abordar este problema. Existe un trabajo de Francois Baccelli y Miguel Canales [Baccelli, Canales] sobre la simulación paralela de las redes de Petri mediante las ecuaciones de recurrencia, lo que permite mejorar el tiempo de simulación con res-

pecto a la simulación de eventos discretos.

Otra tarea pendiente es analizar los resultados de simulaciones usando Moses en varias plataformas, esto gracias a que está basado en Java. Dichas comparaciones indican si el ambiente tiene influencia en la simulación. Para conseguir lo anterior basta copiar el repositorio o simplemente los componentes que se requieran.

De esta manera ya se ha construido un ambiente básico para la simulación de estos dos casos particulares de gráficas: el ciclo dirigido y la gráfica completa. Estos casos fueron fácilmente representados debido a su estructura, sin embargo tratándose de otros casos es posible que su representación resulte más complicada.

El control sobre el valor de la semilla del generador de números aleatorios está previsto en la herramienta Design CPN, pero no es posible ir más allá en el control de la generación de las variables aleatorias. En Moses, aunque no está actualmente implementado lo anterior, gracias a su flexibilidad es fácil lograrlo y la gran ventaja que tiene con respecto a otras aplicaciones de redes de Petri es que agregar funciones es sencillo.

Bibliografía

[Aho 1974] Alfred V. Aho, John E. Hopcroft, Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, 1974.

[Baccelli, Canales] François Baccelli, Miguel Canales, *Parallel Simulation of Stochastic Petri Nets using Recurrence Equations.*

[Bentley 1983] J. L. Bentley, D. S. Johnson, F. T. Leighton, C. C. McGeoch. *An Experimental Study of Bin Packing.* In Proceedings of the 21st Allerton Conference on Comunication, Control and Computing. Univ. of Illinois, Urbana Champaing, 1983.

[Cormen 1990] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[Csirik 1991] J. Csirik, D. S. Johnson. *Bounded Space On-line Bin Packing: Best is Better than First.* In Proceedings of the 2nd ACM-SIAM Symposium on Algorithms and Data Structures. ACM New York, 1991.

[Eppinger 1983] J. Eppinger. *An Empirical Study of Insertion and Deletion in Binary Trees.* Commun. ACM Vol. 26, No. 9, Sept 1983.

[Even 1971] F. Commoner, A.W. Holt, S. Even, A. Pnueli, *Marked Directed Graphs.* Journal of Computer and Systems Sciences, Academic Press, 1971.

[Even, Rajsbaum] F. Commoner, A.W. Holt, S. Even, A. Pnueli, *Marked Directed Graphs.* Journal of Computer and Systems Sciences, Academic Press, 1971.

[Johnson 1973] D. S. Johnson. *Near Optimal Bin Packing Algorithms.* Phd. Thesis, Project MAC TR-109. Massachusetts Institute of Technology, Cambridge Mass., 1973.

[Johnson 1996] David S. Johnson. *A Theoretician's Guide to the Experimental Analysis of Algorithms.* AT&T Research, 30 abril 1996.

[Knott 1975] G. D. Knott. *Deletion in Binary Storage Trees.* Tech. Rep. STAN-CS-75491. Phd. Thesis Stanford University. Stanford Cal., 1975.

[Law 1991] Averill M. Law, W. David Kelton, *Simulation Modeling and Analysis Second Edition.* McGraw-Hill International Editions, 1991.

[Marsan] M. Ajmone Marsan. *Stochastic Petri Nets: An Elementary Introduction.*

[McGeoch 1986] Catherine McGeoch. *Experimental Analysis of Algorithms.* Tech. Rep. CMV-CS-87-124. Phd. Thesis Dept. of Computer Science, Carnegie Mellon Univ., 1986.

[McGeoch 1992] Catherine McGeoch, *Analyzing Algorithms by Simulation: Variance Reduction Techniques and Simulation Speedups.* ACM Computing Surveys, Vol. 24, No. 2, Junio 1992.

[McGeoch 1995] Catherine McGeoch, *Toward an Experimental Method for Algorithm Simulation.* INFORMS Journal on Computing, Vol. 8, No. 1, Invierno 1996.

[McGeoch 1997] Catherine McGeoch, *How to Find Big-Oh in Your Data Set (and How Not To).* Junio 19 1997.

[McGeoch 1999] Catherine McGeoch, *A Bibliography of Algorithm Experimentation.* Augosto 27 1997.

[McGeoch, Moret 1999] Catherine McGeoch, *How to to Present a Paper on Experimental Work with Algorithms.* Septiembre 2 1999.

[Molloy 1982] Michael K. Molloy. *Performance Analysis Using Stochastic Petri Nets.* IEEE Transactions on Computers, Vol C31, No. 9, 1982.

[Moret] Bernard M. E. Moret, *Towards a discipline of Experimental Algorithmics.*

[Rajsbaum, Sidi 1994] Sergio Rajsbaum, Moshe Sidi. *On the Average Performance of Synchronized Programs on Distributed Networks.* IEEE Transactions on Parallel and Distributed Systems, Vol. 5 No. 9, Sept. 1994.

[Rajsbaum 1991] Sergio Rajsbaum. *Stochastic Marked Graphs.* Proc. 5th Int. Workshop on Petri Nets and Performance Modeling, Melbourne, pp95-101, IEEE Computer Society, 1991.

[Ross 1985] Sheldon M. Ross. *Introduction to Probability Models.* Academic Press, 1985.

[Störrle 1998] Harald Störrle *An Evaluation of High End Tools for Petri Nets* Junio 1998.

[Walker 1989] Henry M. Walker. *Computer Science 2.* Library of Congress Cataloging-in-Publicaton Data, 1989.