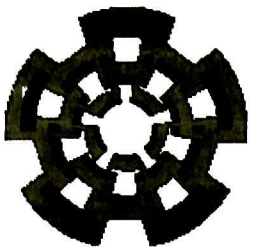
XX(80141.1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Scheduling en Sistemas de Eventos Discretos Modelados con Redes de Petri

Tesis que presenta
Carlos Alberto Ramírez García

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal., Enero del 2000

Scheduling en Sistemas de Eventos Discretos Modelados con Redes de Petri

Tesis de Maestría en Ciencias Ingeniería Eléctrica

Por:

Carlos Alberto Ramírez García

Ingeniero Electrónico en Computación
Centro de Enseñanza Técnica Industrial 1989-1993

Becario del CONACYT, expediente no. **125283**

Directores de Tesis
Dr. Antonio Ramírez Treviño

CINVESTAV del IPN Unidad Guadalajara, Enero del 2000

Agradecimientos

A Dios por darme la oportunidad de vivir y terminar la maestría.

A mis Padres Esther y José de Jesús por sus consejos y apoyo incondicional durante toda mi formación académica.

A mis hermanos Víctor, Sandra, Osvaldo, Jesús, Javier y Eduardo por el apoyo y confianza que me brindaron.

A mi esposa Verónica por la comprensión y apoyo que siempre me ha brindado.

A mi asesor, Dr. Antonio Ramírez Treviño por su paciencia y por toda la ayuda que me brindo para llevar a buen fin el presente trabajo.

A los revisores de la tesis, Dra. Ofelia Begovich Mendoza y Dr. Luis Ernesto López Mellado por ayudarme a detectar y corregir los errores de este trabajo.

A mis compañeros de maestría Paty, Alejandra, Raquel, Rosa, Luis I., Chava, Arturo, Santiago, Fito, Luis P., David e Ivan por toda la ayuda que me brindaron a los largo de la maestría.

A Rosy, PRG, María Meda y todos mis compañeros de generación por la ayuda que me brindaron.

A los profesores del CINVESTAV (especialmente al Dr. Jacinto, Dra. Ofelia, Dr. Mellado, Dr. Treviño) y a todos aquellos que han ayudado a mi formación académica y personal.

A todos ellos muchas gracias.

Este trabajo fue apoyado por el proyecto de investigación 125283 del CONACYT.

Índice General

Capítulo 1

Panorama General

Resumen: En este capítulo se presenta brevemente un panorama general sobre la optimización de los sistemas de eventos discretos (SED) modelados con redes de Petri. Primero se presenta de manera intuitiva, donde se describe el problema de scheduling en forma general, mencionando las dificultades y limitaciones que se tienen para resolverlo y la importancia del problema. Después se define formalmente el problema y se introducen algunos conceptos básicos relacionados con scheduling, se mencionan los principales métodos con los que se ha abordado el problema, los trabajos y personas que han trabajado en este problema, y por último, se presenta la forma en que se aborda el problema en este trabajo.

1.1 Introducción

El presente trabajo trata sobre el problema de scheduling, el termino scheduling se puede traducir como calendarización. Pero es más común encontrar en la literatura relacionada al tema la palabra scheduling que la de calendarización, por este motivo, en este trabajo se utilizará el termino scheduling.

El problema de scheduling es muy común y se le puede presentar a una persona varias veces en su vida cotidiana. Muchas personas tratan con problemas de scheduling simples y los resuelven sin conocer la teoría de scheduling, aunque no siempre de forma óptima.

Por ejemplo, supongamos que una estilista tiene que atender a cuatro clientas que llegaron al mismo tiempo a su salón de belleza, desean una base, un tinte de pelo, un tratamiento facial y un corte de pelo respectivamente. La estilista sabe que tiene que realizar cuatro tareas distintas, conoce también el tiempo y recursos que requiere cada una de ellas.

Las tareas requieren un tiempo total y una secuencia de acciones y tiempos que son los siguientes:

- Base : 1 hora 35 min. (30 min. poniendo tubitos, 20 min. reposo, 15 min. neutralizar tubos, 10 min. reposo, 20 min. quitar tubos, secar pelo y peinar).
- Tinte de pelo : 1 hora 20 min. (30 min. aplicar tinte, 30 min. reposo 20 min. lavar pelo, secarlo y peinar).
- Tratamiento facial : 45 min. (10 min. aplicar mascarilla , 20 reposo, 15 retirar mascarilla, poner loción refrescante y crema humectante).
- Corte de pelo : 20 min.

Lo ideal es que la estilista realice el mayor número de tareas en el menor tiempo posible y que los clientes esperen lo menos posible sin ser atendidos.

La estilista no pensó mucho la solución, pero decidió hacer primero el corte, enseguida la base (1 hora 35 min.), después el tratamiento facial (45 min.), y por último el tinte (1 hora 20 min.). Conforme fue realizando los trabajos se dió cuenta que no podía empezar ningún otro trabajo sin primero terminar el anterior, por lo cual tardó 4 horas en terminar estas tareas. Necesitando así medio turno de su jornada de trabajo. Además algunas clientas se molestaron o desesperaron por la espera para ser atendidas. Esto quiere decir que la estilista seleccionó mal, el orden de realización de las tareas. En otras palabras, seleccionó un mal schedule. Después de varios meses de experiencias se dió cuenta que para un caso como este, el tiempo óptimo para realizar estas tareas es de 2 horas con 40 min.

En la figura 1.1 se muestra un diagrama de Gantt, donde se puede apreciar de forma gráfica, las tareas que tiene que realizar la estilista con sus respectivos tiempos. También se puede ver el schedule elegido por la estilista y el schedule óptimo.

Intuitivamente se puede ver que el problema de scheduling está relacionado con el problema de optimización y que es de naturaleza altamente combinatoria porque se tienen que analizar todas las posibles combinaciones en que se pueden realizar los trabajos.

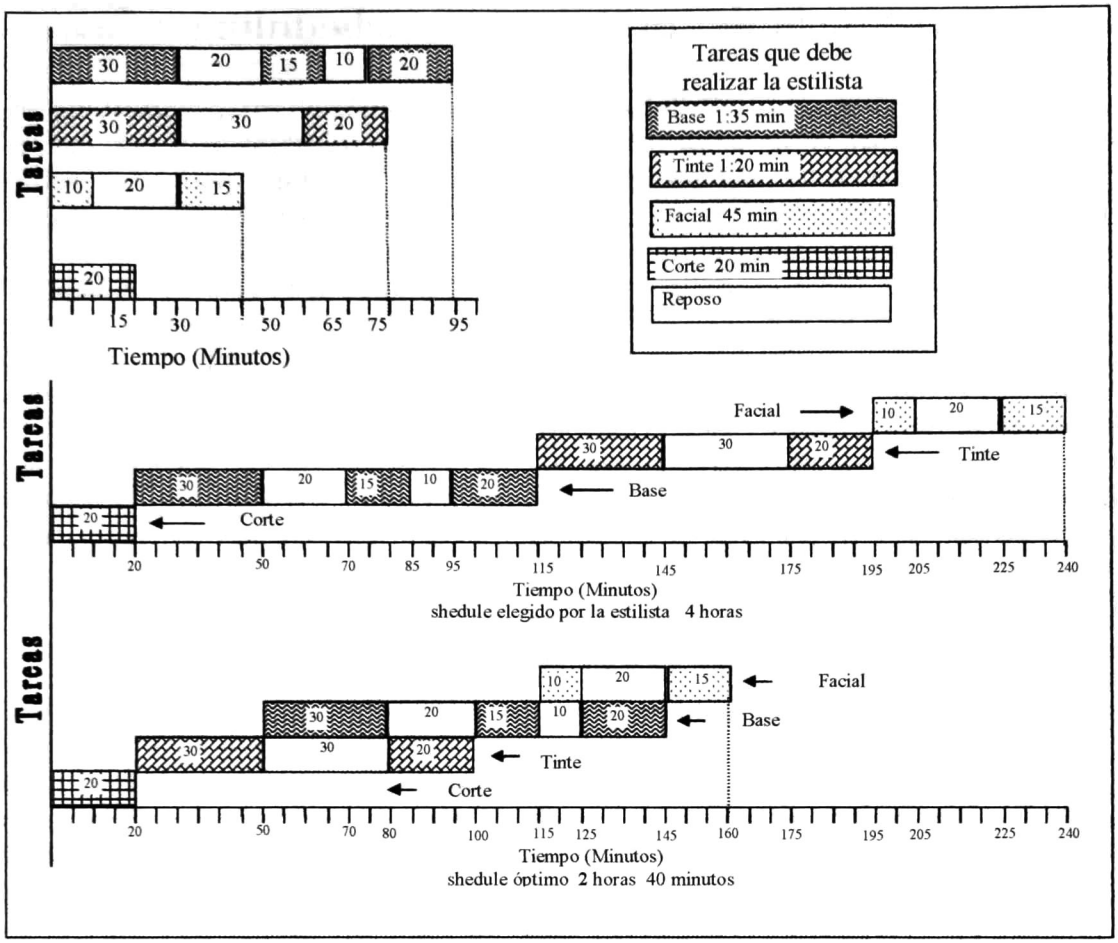


Figura 1.1: Diagrama de Gantt de tareas de la estilista

También se puede notar que se habla de "Tareas o trabajos", que tienen asociados ciertos parámetros (tiempos de realización y/o costos) y una relación de precedencia entre ellas (el orden en que se deben ejecutar las operaciones).

En este ejemplo son pocos los clientes (tareas) y sólo la estilista necesita los recursos, por lo cual pensándolo un poco más y mejor, ella puede calcular su schedule óptimo en poco tiempo. Pero ¿qué pasaría si a esta estilista la ponen a cargo de un salón más grande, con 5 empleadas (estilistas) y con casos en los cuales llegan 15 clientes o más (por ejemplo al abrir el salón)?. En este caso, aumentó la complejidad de su problema y tratar de obtener la solución óptima le llevaría mucho tiempo, tal vez todo un día, pero los clientes no pueden esperar tanto. Es aquí donde se requiere la teoría de scheduling para obtener soluciones óptimas en poco tiempo (si es posible).

1.2 Importancia del problema de scheduling

La teoría de scheduling es de gran importancia y beneficio en la optimización de muchos sistemas. Por ejemplo, en las industrias donde se fabrican productos, se manejan muchas máquinas, herramientas etc., obtener un buen schedule es importante porque nos permite asignar de manera adecuada todos los recursos. Es decir, si se tiene información sobre los procesos como tiempo de duración, recursos que utilizan etc, y de las máquinas, como cuáles son más rápidas o más lentas, cuáles fallan más, cuáles pueden trabajar más tiempo etc, se puede mimimizar la cantidad de piezas en los buffers (almacenes intermedios entre una máquina y otra), evitar los cuellos de botella, reducir al máximo los tiempos de producción etc, lo que permite cumplir con metas y objetivos en la producción que se traducen en ahorros, ganancias o beneficios para la empresa. En el caso de la estilista, en la figura 1.1 se puede apreciar que existe una diferencia de 80 minutos entre el schedule elegido por la estilista y el óptimo. Por tanto es importante obtener el schedule óptimo, ya que puede hacer las mismas tareas en menor tiempo, no estar de ociosa, y con el tiempo restante atender a más clientes y obtener mayores ganancias. En otras palabras la teoría de scheduling nos ayuda a optimizar el sistema.

La optimización de los sistemas tiene dos etapas muy importantes:

Planificación: [Baker 91, Ramírez 93t] Aquí se define lo que se va hacer, cuántas y cuáles tareas se necesitan, el tiempo de duración, y los recursos que necesita cada una de ellas. También se presentan algunos problemas como seleccionar el tipo de piezas que se producirán; la arquitectura del sistema (distribución física y relaciones causales que debe tener el sistema); el grado de flexibilidad (el número óptimo, y los posibles procesados que deben tener las piezas para obtener los productos planeados, obteniéndose así las posibles rutas que pueden seguir las piezas dentro del sistema). Como resultado de esta etapa tenemos un plan, pero podemos obtener varios planes para conseguir el mismo objetivo, por lo cual se debe elegir el mejor de ellos. La planificación es la etapa previa a la de scheduling.

Scheduling : [Kligerman 86, Ramírez 93t] Es la etapa donde se decide el orden en el que se tienen que realizar las tareas u operaciones, la asignación de recursos y el calendario para su realización, de tal forma que cumpla con los criterios de optimización que se hayan planteado. El objetivo de esta etapa es obtener un "schedule" óptimo que cumple con la planificación.

1.3 Descripción del problema de scheduling

En la figura 1.2 se puede ver un sistema cualquiera que se quiere optimizar, por ejemplo un sistema de producción, de cómputo, de manufactura, industrial, de tráfico aéreo, un proceso químico o cualquier otro. En este trabajo el punto de partida es un modelo bajo la forma de una Red de Petri (RP) para iniciar su optimización. En la figura 1.2 no se muestra la RP, sólo se muestra un esquema que indica que en esa RP (modelo del sistema) tenemos un punto inicial y un punto final. Partiendo del punto inicial (estado inicial) se tiene varias trayectorias o caminos para llegar al punto final (estado final), siguiendo alguno de esos caminos tal vez se llegue a estados donde se inicien nuevas ramificaciones indicando

que a partir de ahí se tienen varios caminos alternativos para llegar al punto final. Al ir recorriendo esos caminos se irán efectuando una serie de operaciones, tareas, subprocesos o actividades que se tienen que realizar para llegar al estado o punto final en donde se obtiene como resultado la realización de un proceso o tarea, la fabricación de un producto o finalización de un programa de actividades o cualquier otra cosa dependiendo del sistema modelado. También se supone que se tienen todas las tareas u operaciones que se tienen que llevar a cabo, el tiempo que se necesita para realizarlas, los recursos necesarios y las posibles secuencias en las que se pueden realizar esas tareas u operaciones para llegar al punto final.

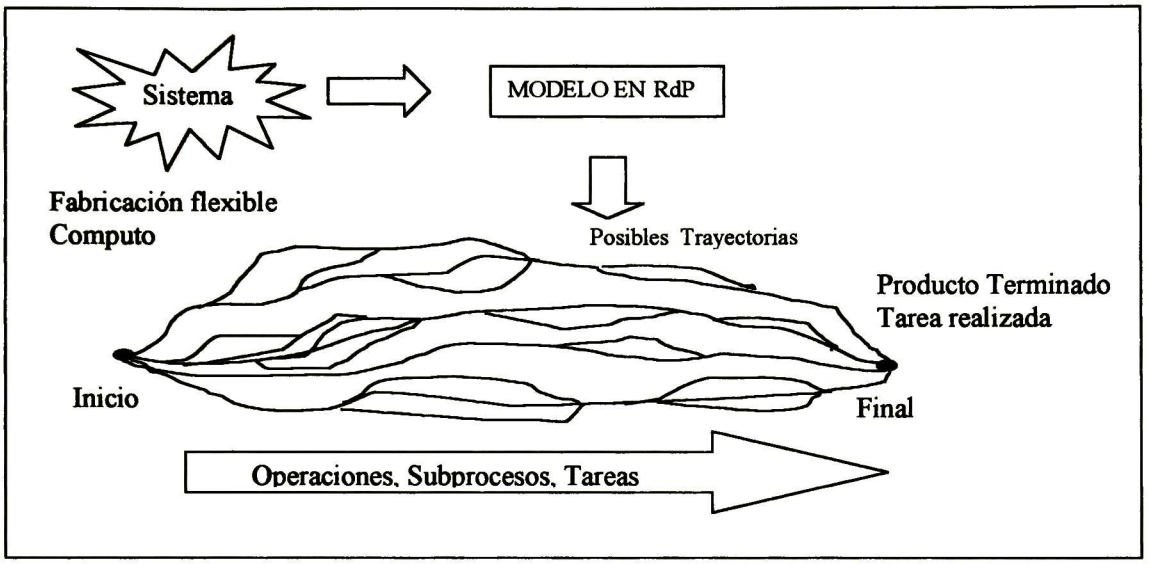


Figura 1.2: Esquema general para la descripción del problema de scheduling

El problema de optimización al que nos enfrentamos es encontrar una trayectoria óptima. Esta puede ser la que se realice en el menor tiempo posible o en la que se aprovechen al máximo los recursos o cualquier otra basándose en algún otro criterio de optimización.

Si el modelo en RP es muy pequeño, tal vez analizarlo y encontrar la solución óptima no sea tan difícil, pero si se incrementa su tamaño, analizarlo se vuelve muy complicado ya que se puede generar una explosión combinatoria de trayectorias. Por lo cual, obtener la solución óptima es algo muy costoso en tiempo de cálculo, ya que en los peores casos la complejidad de este problema es NP.

1.4 Scheduling en sistemas de eventos discretos

Para optimizar un sistema, primero es necesario identificar que tipo de sistema estamos tratando. Enseguida se menciona la clasificación de los sistemas, de acuerdo al tipo de variables que los describen [Ho91]

Sistemas Dinámicos de Variables Continuas (SDVC) : Son sistemas cuyo estado evoluciona de manera continua. Las ecuaciones diferenciales son adecuadas para modelarlos.

El estudio de la optimización de estos sistemas está a cargo de la teoría de control óptimo, ya sea con el uso de máximos y mínimos, calculo variacional, programación dinámica, Pontryagin, etc.

Sistemas Dinámicos de Eventos Discretos (SDED): (o simplemente sistema de eventos discretos **SED**) es aquel que contiene un espacio de estados numerable, posiblemente infinito. El paradigma de las ecuaciones diferenciales no es adecuado para modelarlos, por lo que generalmente se usan lenguajes formales o teoría de la medida.

Existe una gran variedad de SED, y son con los que trataremos en esta memoria, como ejemplos podemos citar sistemas de control de tráfico, sistemas de información, sistemas de comunicaciones digitales, etc. Un SED, como el caso de una central de autobuses, su estado se puede representar por un vector (camiones en la central, camiones en viaje, camiones en taller, ...), la evolución del sistema se realiza por la activación de una serie de eventos (llegada o salida de camiones, ...) que hacen que su estado cambie. El sistema es dinámico porque la salida depende del evento que se realiza (entrada al sistema) y del estado anterior. Además su espacio de estados y conjunto de eventos es numerable y posiblemente infinito. Además existe la presencia de cambios abruptos en los eventos y por consecuencia en los estados, por lo que la representación de estos sistemas no es posible hacerla mediante ecuaciones diferenciales.

En esta memoria sólo se consideran SED en los que influye el tiempo, y para su análisis nos auxiliamos de la teoría de scheduling y también desarrollamos algunas técnicas para realizar la optimización de este tipo de sistemas. Como herramienta formal para modelar los SED utilizamos las redes de Petri (RP) porque nos permite representar de forma sencilla y clara el paralelismo, sincronización, exclusión mutua, relaciones causales, decisiones y otras características que presentan los SED. También nos permiten analizar las propiedades cualitativas y cuantitativas de los SED como vivacidad, ausencia de bloqueos, limitación, etc. Para ser más exactos trabajamos con una extensión de las RP denominadas Redes de Petri Temporizadas, donde se toma en cuenta el tiempo.

1.4.1 Definición del problema de scheduling en SED

El problema de scheduling [Bellman 82, Lee 92] consiste en la asignación de recursos a tareas y en el establecimiento de inicio de las Tareas. En SED existen algoritmos que optimizan diferentes criterios. Por ejemplo, algunos algoritmos minimizan el tiempo de ejecución otros minimizan el número de productos en un almacén (inventario mínimo), etc. La solución al problema de scheduling es una secuencia de ternas (R, T, τ) llamado schedule. Donde T es la tarea, R son los recursos asignados y τ es el tiempo de inicio de una tarea respecto a una referencia.

Un **schedule factible** es aquél que se puede realizar físicamente y que satisface las restricciones impuestas. Al algoritmo que encuentra un schedule factible que minimiza o maximiza la función objetivo se le denomina scheduler.

Formalmente el problema de scheduling se define como sigue:

Definición 1.1 ***El problema de scheduling consiste de:***

- ***1.- Conjuntos y secuencias:*** *Determinar los siguiente conjuntos:*

  $W = \{(x, y, z) \mid x \in R, \quad y \in T, \quad z \in \mathbb{R}^+\}$; *donde* R *es el conjunto de los recursos,* T *es el conjunto de las tareas. El número real* z *representa el tiempo de inicio de la tarea* y. *Donde* $z \in \mathbb{R}^+ \cup \{0\}$.

  Un schedule $s = w_1, w_2, ...w_k, ...$ *es una secuencia de elementos* $w_i \in W$, *y* $\omega = \{s \mid s$ *es un scheduler factible* $\}$.

- ***2.- Función objetivo:*** *Encontrar la función* f *que describa el criterio de optimización.* $f : \omega \longrightarrow \mathbb{R}$

- ***3.- Optimización:*** *Encontrar, al menos, un elemento* $s_i^* \in w$ *tal que:*

  $f(s_i^*) = \min_{s_i \in w} f(s_i)$ *(caso de minimización).*

  $f(s_i^*) = \max_{s_i \in w} f(s_i)$ *(caso de maximización).*

1.5 Antecedentes Generales

El problema de scheduling en SED es un problema de optimización y de naturaleza altamente combinatoria. Cuando se quiere resolver un problema de scheduling surgen principalmente dos problemas. Uno de ellos es seleccionar la función objetivo, es decir, qué criterio de optimización se eligirá. El otro problema es la intratabilidad. Es decir, el tiempo necesario para encontrar la solución óptima suele ser muy grande. Básicamente existen dos tipos de métodos para obtener la solución [Dyke 91]; métodos estructurales, los cuales transforman el modelo del sistema en otro equivalente del que ya se conoce la solución, y los métodos operativos, que hacen uso de técnicas de investigación de operaciones, Inteligencia Artificial (IA), por mencionar algunos. Enseguida se describen brevemente 3 métodos operativos que abordan los problemas de diferente forma:

Métodos Constructivos: Obtienen schedules óptimos, pero no utilizan herramientas de modelado, éstos obtienen propiedades y reglas para cada sistema en particular lo que dificulta la aplicación general de estos métodos y por lo general se aplican a sistemas donde sólo existe una máquina.

Métodos basados en Técnicas de Investigación de Operaciones: Aquí los sistemas se representan con grafos o ecuaciones adecuadas para programación lineal. En esta área se tienen problemas bien caracterizados, para los cuales existen métodos de solución. En el caso que el sistema no pertenece a ese tipo de problemas, es modificado a una forma conocida; en estos casos se espera obtener una solución buena (no precisamente óptima). Cuando este método se aplica a sistema de fabricación flexible surgen serios inconvenientes si se tienen un gran número de variables. Además, generalmente están mal condicionados, por ejemplo, no se conocen las características de las piezas, y los problemas resultan de programación entera. Por lo cual son muy difíciles de tratar.

Métodos basados en Inteligencia Artificial: Los sistemas se representan describiendo su comportamiento y sus relaciones relevantes. Se trata de reducir el espacio de búsqueda. Generalmente se trata de que el comportamiento cumpla con las restricciones y que sea bueno, más que obtener el comportamiento óptimo. La principal desventaja de esta aproximación es que no hay una herramienta formal para obtener las propiedades de los modelos obtenidos, además, no se conoce que tan lejos o cerca estamos de la solución óptima.

Desafortunadamente no siempre se obtiene la solución aplicando sólo un método, sino que se combinan varios métodos para obtener la solución, y además, esta solución generalmente es una aproximación al valor óptimo y su exactitud depende del modelo y algoritmos utilizados.

En esta memoria se usan las redes de Petri porque nos proporcionan más elementos para modelar, analizar y diseñar los sistemas que los métodos mencionados.

Además existen una serie de trabajos y artículos que sirven como base para este trabajo. Por ejemplo, en [Silva 85] se describen los conceptos básicos sobre RP, como analizar y estudiar las propiedades dinámicas y estructurales de las RP, clasificación de las RP, entre otros temas.

También se tienen trabajos como [Ramchandani 74] que introduce las redes de Petri temporizadas para evaluar el desempeño (performance) de los sistemas; el de [Magott 84] que presenta un problema de programación lineal para evaluar cotas del desempeño de los sistemas modelados con RPT; el de [Magott 87] que estudia la complejidad del análisis del desempeño; [Ramamoorthy 80] que estudia el comportamiento óptimo de los sistemas y obtiene cotas alcanzables para sistemas que pueden ser modelados con grafos marcados; [Carlier 84, Carlier 88] también estudia el desempeño de los sistemas, y obtiene algunas propiedades que deben cumplir los schedules óptimos, como por ejemplo que debe pertenecer a un subconjunto de schedules que el denomina los schedules activos, los cuales no permiten ociosidad en el sistema (por lo menos un evento debe estar sucediendo).

En [Chretienne 83] se presenta que en grafos marcados "disparar las transiciones tan pronto como estén habilitadas" permite obtener el schedule óptimo. También en [Carlier 84, Carlier 88] se presenta el grafo de estados diligentes, este contiene sólo schedules activos. En [Onaga 91] se muestra un algoritmo que encuentra un schedule bueno para cualquier tipo de redes. En [Campos 90] se presenta cómo calcular el tiempo de ciclo de una máquina de estados resolviendo un problema de programación lineal, y en [Ramírez 93t] se muestra que en máquinas de estados "Disparar las transiciones de las T-componentes tan pronto como están habilitadas" permite obtener un schedule óptimo.

En [Magott 87] se trabajó con un una subclase un poco más grande que incluye a las dos anteriores denominadas RP de Libre Elección y demostró que obtener el schedule óptimo en este tipo de redes es un problema NP-Completo.

En [Ramírez 93t] también se trabajó en RP de Libre Elección clasificándolas en FCA*, FCB* y FCC* y describe cómo obtener, para algunas, un schedule óptimo, y para otras un schedule que no es óptimo pero que se considera bueno (cuasi-óptimo). Mostró que uno de los problemas para el cálculo eficiente del schedule óptimo es la forma en que se relacionan los que él llama pares distribución-conjunción con los lugares de selección. También mostró que para

redes tipo FCA* se puede encontrar el schedule óptimo en tiempo polinomial. Para redes tipo FCB* mostró que son muy difíciles de analizar y el cálculo de tiempo mínimo de ciclo es equivalente al problema de balanceo de contenedores (bin packing problem) [Garey 79]. Aquí el cálculo del schedule bueno se realiza dividiendo el problema general en subproblemas y dependiendo de las condiciones finales del algoritmo presentado se puede saber si se obtuvo el óptimo. En las redes FCC* muestra la forma de obtener un schedule óptimo en tiempo polinomial. También introduce los conceptos de optimalidad local y sistemas lugar lugar para tratar de obtener el schedule óptimo de otro tipo de RP.

1.5.1 Optimalidad Local en scheduling y su ventaja

En esta memoria se utiliza la filosofía "Divide y Venceras" para tratar de resolver el problema de scheduling de un sistema. Para entender la idea que se quiere aplicar es necesario introducir la siguiente definición.

Definición 1.2 *Si un sistema puede ser dividido en subsistemas y el desempeño óptimo de los subsistemas garantiza el desempeño óptimo de todo el sistema, entonces el sistema posee la propiedad de* ***Optimalidad Local*** *(OL) con respecto a esa división.*

Dicho de otra forma, se trata de estudiar los sistemas de eventos discretos en RP y determinar cuándo es posible dividir el sistema en subsistemas y hacer que optimizaciones locales en los subsistemas permitan alcanzar el óptimo global, lo anterior se ilustra en la figura 1.3.

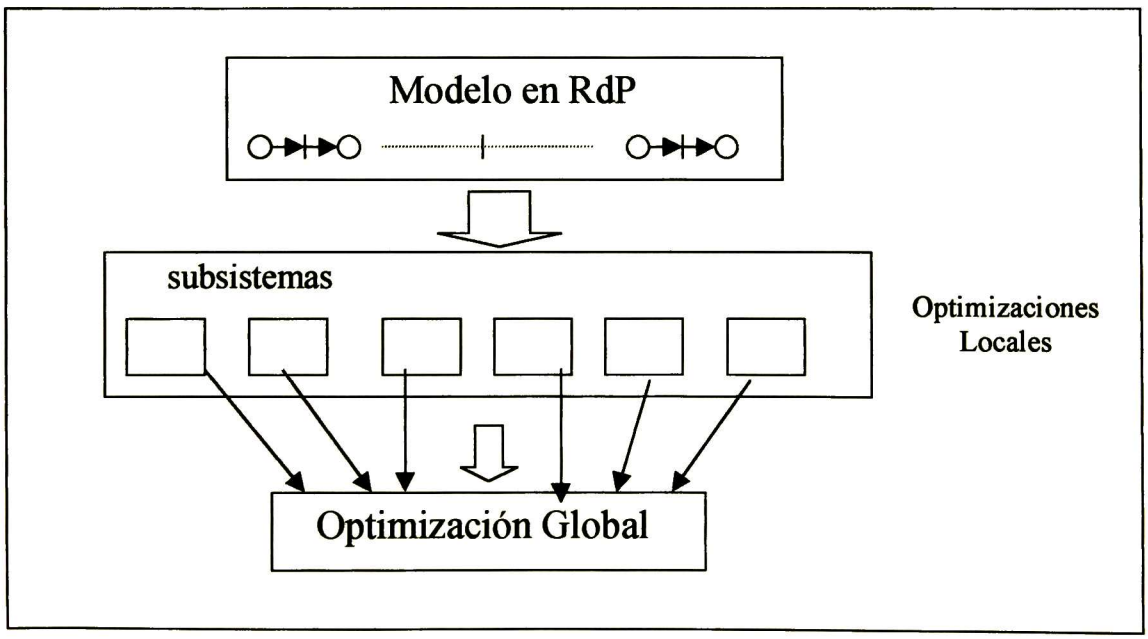


Figura 1.3: Esquema general de optimizaciones locales

La ventaja que se obtiene al optimizar un sistema por partes es la siguiente:

Se sabe que en general, la complejidad del problema de scheduling es 2^n donde n es el número de secuencias, pero si se pudiera dividir en k subsistemas de forma que tengamos :

forma global dividiendo en subsistemas

$$2^n \geq 2^{n_1} + 2^{n_2} + \ldots\ldots\ldots + 2^{n_k} \qquad \text{donde : } n_1 + n_2 + \ldots\ldots n_k = n$$

Entonces, se podría obtener la solución en menor tiempo.

Por ejemplo, si existe un algoritmo de scheduling que en el peor de los casos requiere 2^n microsegundos para resolver el problema (n es el numero de secuencias) y suponiendo que se tiene una computadora que realiza una comparación en un microsegundo, entonces resolver un sistema donde la cardinalidad de w es 10 tomará un tiempo menor que 0.001 segundos; pero si la cardinalidad de w es 60 (problema pequeño) debido a una explosión combinatoria de alternativas, el problema se resolverá en 366 siglos aproximadamente. Por lo cual se puede decir, que en general, el tiempo necesario para encontrar un schedule óptimo no puede ser representado por un polinomio en la cardinalidad de w, o dicho de otra forma el problema es NP.

Si la red de Petri de un sistema posee la propiedad de optimalidad local, el sistema se puede dividir en subsistemas, entonces por medio de optimizaciones locales de cada subsistema y juntando esas optimizaciones locales obtenemos la optimización global. En la figura 1.4, se muestra un ejemplo para un caso en el que w = 50 y si el sistema se pudiera dividir en 5 subsistemas cada uno con w= 10. Entonces tendríamos que cada subsistema se puede optimizar en 2^{10} osea en 0.001 segundos. la variable t indica el tiempo en el que el sistema se divide en subsistemas.

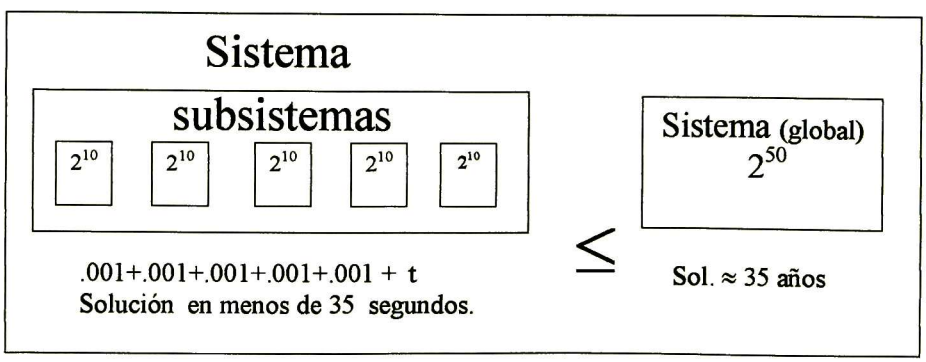


Figura 1.4: Ventaja en tiempo de optimizaciones locales

Como se puede ver si el sistema posee la propidedad de optimalidad local, podemos obtener la solución óptima en mucho menor tiempo (35 segundos contra 35 años) o en el peor de los casos en el mismo tiempo.

1.6 Metas y Objetivos

Los objetivos de este trabajo son los siguientes:

- Determinar las condiciones necesarias y/o suficientes bajo las cuales un sistema de eventos discretos modelado en redes de Petri posee la propiedad de optimalidad local.
- Proponer esquemas y algoritmos de scheduler que exploten la propiedad de optimalidad local de los sistemas de eventos discretos modelados en redes de Petri que la posean.

1.7 Organización de la tesis

Este trabajo está organizado de la siguiente manera:

- El capítulo dos presenta a las redes de Petri, la cual es una herramienta formal para el modelado y análisis de los SED. En esta sección se presenta la definición de RP, terminología, conceptos importantes, algunas de sus propiedades, su clasificación, metodos de análisis y una extensión de las RP denominadas redes de Petri temporizadas. También se presentan los principales resultados relacionados con la obtención del schedule óptimo para las diferentes subclases de RP.
- El capítulo tres presenta la forma en que se obtiene el schedule óptimo utilizando el grafo diligente junto con el algoritmo A*.
- En el capítulo cuatro se propone otra forma de obtener el schedule óptimo combinando las técnicas vistas en el capítulo tres junto con los conceptos de optimalidad local y pares inevitables.
- En el capítulo cinco se presenta un ejemplo, donde se aplican los conceptos mencionados en el capítulo anterior para la obtención del schedule óptimo.
- Finalmente se presentan las conclusiones y trabajo futuro.

Capítulo 2

Herramienta de Modelado y Análisis

Resumen: En este capítulo se introducen las redes de Petri (RP) como herramienta de modelado y análisis para el problema de scheduling, mencionando las ventajas que proporcionan las redes de Petri con respecto a otras herramientas de modelado. Se presentan los conceptos básicos sobre RP, sus principales propiedades dinámicas y estructurales, su clasificación, y se mencionan algunos métodos de análisis de las propiedades de las RP.

También se presenta una extensión de las RP, denominadas Redes de Petri Temporizadas (RPT), las cuales se utilizan para modelar y analizar sistemas temporizados. Por último, se presentan los principales resultados que existen actualmente para el problema de scheduling referentes a algunas subclases de redes de Petri.

2.1 Introducción

Para comprender y analizar un sistema es necesario obtener un modelo que describa su dinámica y comportamiento. Existen varias herramientas para modelar y analizar SED´s, algunos representan actividades, por ejemplo el PERT/GERT que por medio de un grafo describen las actividades que se realizan en el sistema (en el grafo los arcos y los vértices representan las actividades y los sucesos respectivamente), otros representan estados y cómo cambia el sistema de un estado a otro, por ejemplo, las redes de colas, tablas de estados, automatas finitos (AF), grafos reducidos, y RP por mencionar algunos, por ejemplo, en el caso de AF éstos describen por medio de un grafo, todos los posibles estados en los que se puede éncontrar el sistema, y los eventos que pueden hacer que el sistema cambie de estado. Los AF permiten representar lenguajes regulares.

En esta memoria utilizamos las RP porque son una herramienta formal que permite modelar adecuadamente los SED, es decir, permite describir de forma sencilla y clara el paralelismo, sincronización, exclusión mutua, relaciones causales, decisiones y otras características de estos sistemas. Además posee una representación gráfica y una teoría matemática que permite analizar la estructura de las RP obtenidas y las propiedades cualitativas o cuantitativas de los sistemas modelados, por ejemplo, vivacidad, ausencia de bloqueos, limitación, etc.

Las RP describen las actividades y estados del sistema, su representación gráfica permite ver en cierta forma cómo se utilizan los recursos del sistema, son más flexibles que otras herramientas ya que permiten realizar fácilmente modificaciones locales y descripción por refinamientos sucesivos, además es una herramienta independiente de cualquier tecnología de implementación.

En el caso de lenguajes, las RP permiten describir lenguajes regulares, algunos libres de contexto y otros sensitivos a contexto, también se puede decir que los AF son un subconjunto de la RP.

En este capítulo se presenta, la definición formal de una RP, los conceptos y terminología básica, su representación gráfica y matricial, su dinámica o evolución, algunas de sus propiedades, tipos de redes de petri y algunas extenciones de las RP, principalmente las redes de Petri temporizadas, en fin, se presentan los conceptos necesarios para entender las RP y la forma en que se utlizan en el problema de scheduling.

2.2 Redes de Petri (RP)

2.2.1 Estructura y terminología básica.

Definición 2.1 *Una Red de Petri generalizada [Silva 85] se representa por una cuádrupla* $\mathcal{N} =< P, T, PRE, POST >$

donde:

- $P = \{p_1, \ldots, p_n\}$; es un conjunto finito y no vacío de lugares.
- $T = \{t_1, \ldots t_n\}$; es un conjunto finito y no vacío de transiciones.
- $P \cap T = \emptyset$; los lugares y transiciones son conjuntos disjuntos y $P \cup T \neq \emptyset$ (se debe tener por lo menos un elemento).
- $PRE : P \times T \rightarrow \mathbb{N} \cup \{0\}$; es la función de PRE-incidencia. Indica los arcos dirigidos de los lugares a las transiciones.
- $POST : P \times T \rightarrow \mathbb{N} \cup \{0\}$; es la función de POST-incidencia[1]. Indica los arcos dirigidos de las transiciones a los lugares.
- $M : P \rightarrow \mathbb{N} \cup \{0\}$; es una función de Marcado que representa el número de marcas que contienen los lugares.

Gráficamente una RP es un grafo dirigido bipartito que tiene dos tipos de nodos, los Lugares (representados por círculos), y las Transiciones (representadas por segmentos de linea), unidos por medio de arcos. Generalmente los lugares representan los estados del sistema y las transiciones los eventos. Los lugares sólo se conectan con transiciones y viceversa.

Existe un arco que va del lugar p_i a la transición t_j sii $PRE(p_i, t_j) \neq 0$. Análogamente, existe un arco que va de la transición t_k al lugar p_i sii $POST(p_i, t_k) \neq 0$. Cada arco se etiqueta con un número entero natural $PRE(p,t)$ o $POST(p,t)$, que se denomina peso del arco. Un arco no etiquetado se considera de peso unitario.

También se puede representar a la RP por $\mathcal{N} =< P, T, F >$ donde P y T tienen el mismo significado que la representación anterior, y F representa los arcos de la red, tanto los de entrada como los de salida. Un arco $(p_i, t_j) \in F$ si y sólo si $PRE(p_i, t_j) = 1$; y un arco $(t_i, p_j) \in F$ si y sólo si $POST(p_j, t_i) = 1$. En esta memoria se utilizan las dos representcaiones indistintamente.

Para ver cómo se modela un sistema con RP, tomemos como referencia un sistema sencillo como el de dos carros que van y vienen sincronizadamente (ver figura 2.1). Supongamos que los carros C_1 y C_2 están inicialmente sobre los contactos E y F, respectivamente. Al presionar el botón "A" los carros C_1 y C_2 se desplazan simultáneamente hacia la derecha. El inicio del regreso hacia la izquierda lo realizarán simultáneamente cuando ambos carros se encuentren en el extremo derecho (sobre los contactos B y C)

De la RP de la figura 2.1 podemos ver que

P = {P_1,P_2,P_3,P_4,P_5,P_6,P_7,P_8} y T = {A,B,C,D,E,F},

las funciones PRE y POST son :

PRE(P_1,A)=PRE(P_2,A)=PRE(P_3,B)=PRE(P_4,C)=PRE(P_5,D)=1.

PRE(P_6,D)=PRE(P_7,E)=PRE(P_8,F)=1.

POST(P_1,E)=POST(P_2,F)=POST(P_3,A)=POST(P_4,A)=POST(P_5,B)=1.

[1] Anteriormente, la función POST se representaba: POST: TxP→ $\mathbb{N} \cup \{0\}$, primero se nombraba la transición y despues el lugar, POST(t,p).

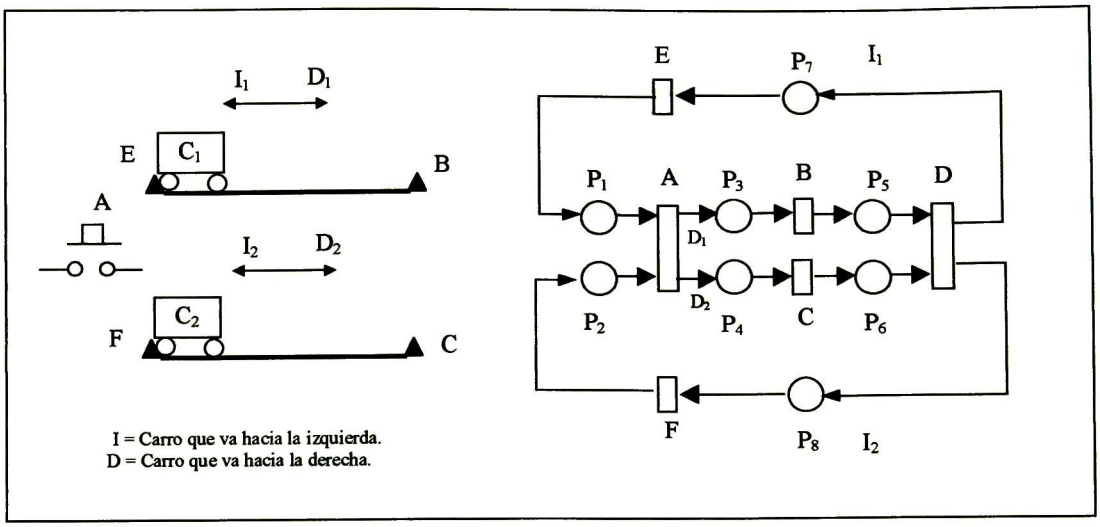


Figura 2.1: Sistema de carros que van y vienen, y su respectivo modelo en RP.

$POST(P_6,C)=POST(P_7,D)=POST(P_8,D)=1.$

Definición 2.2 *Una red es ordinaria si sus funciones de incidencia sólo pueden tomar valores 0 y 1:*

$$PRE(p,t) \in \{0,1\}; \qquad POST(p,t) \in \{0,1\};$$

Definición 2.3 *Una red es pura si ninguna transición contiene un lugar que sea simultáneamente de entrada y de salida :*

$$\forall t_j \in T \quad \forall p_i \in P \quad PRE(p_i,t_j) \;\; POST(t_j,p_i) = 0.$$

La RP de la figura 2.1 es una red ordinaria y pura.

2.2.2 Representación matricial de una RP.

Una RP se representa matricialmente por medio de dos matrices. Sea $\mid P \mid = n$ (número de lugares en la RP) y sea $\mid T \mid = m$ (número de transiciones en la RP).

Se le denomina matriz de incidencia previa a la matriz

$$\mathbf{C}^- = [c_{ij}^-]_{n \times m} \;\; donde \;\; c_{ij}^- = PRE(p_i,t_j).$$

Se le denomina matriz de incidencia posterior a la matriz

$$\mathbf{C}^+ = [c_{ij}^+]_{n \times m} \quad donde \quad c_{ij}^+ = POST(p_i, t_j).$$

Es decir, en las matrices de incidencia las filas (i) representan los lugares y las columnas (j) las transiciones, y cada elemento (i, j) indica la incidencia que el lugar i tiene sobre la transición j.

La representación matricial de una red pura se simplifica definiendo una única matriz, C, denominada matriz de incidencia:

$$\mathbf{C} = \mathbf{C}^+ - \mathbf{C}^- \quad \text{donde : } \mathbf{c}_{ij} = \begin{cases} POST(p_i, t_j) & \text{si no es nulo} \\ -PRE(p_i, t_j) & \text{si no es nulo} \\ 0 \text{ en cualquier otro caso.} \end{cases}$$

En la matriz **C** un elemento positivo indica incidencia posterior y uno negativo indica incidencia previa. Un elemento nulo en **C** indica que la transición y lugar correspondientes no están conectados directamente a través de un arco.

La representación matricial de la RP de la figura 2.1 es :

$\mathbf{C}^+$

	A	B	C	D	E	F
P_1	0	0	0	0	1	0
P_2	0	0	0	0	0	1
P_3	1	0	0	0	0	0
P_4	1	0	0	0	0	0
P_5	0	1	0	0	0	0
P_6	0	0	1	0	0	0
P_7	0	0	0	1	0	0
P_8	0	0	0	1	0	0

$\mathbf{C}^-$

	A	B	C	D	E	F
P_1	1	0	0	0	0	0
P_2	1	0	0	0	0	0
P_3	0	1	0	0	0	0
P_4	0	0	1	0	0	0
P_5	0	0	0	1	0	0
P_6	0	0	0	1	0	0
P_7	0	0	0	0	1	0
P_8	0	0	0	0	0	1

$\mathbf{C}$

	A	B	C	D	E	F
P_1	−1	0	0	0	1	0
P_2	−1	0	0	0	0	1
P_3	1	−1	0	0	0	0
P_4	1	0	−1	0	0	0
P_5	0	1	0	−1	0	0
P_6	0	0	1	−1	0	0
P_7	0	0	0	1	−1	0
P_8	0	0	0	1	0	−1

Definición 2.4 *Sea* $\mathcal{N} =< P, T, PRE, POST >$ *una RP donde* $p \in P$ *y* $t \in T$ *podemos definir los siguiente conjuntos:*

- Conjunto de lugares de entrada de una transición t : $\quad {}^\bullet t = \{p \in P | C^-(p,t) > 0\}$
- Conjunto de lugares de salida de una transición t : $\quad t^\bullet = \{p \in P | C^+(t,p) > 0\}$
- Conjunto de transiciones de entrada de un lugar p : $\quad {}^\bullet p = \{t \in T | C^+(t,p) > 0\}$
- Conjunto de transiciones de salida de un lugar p : $\quad p^\bullet = \{t \in T | C^-(p,t) > 0\}$

Por ejemplo en la figura 2.1 si tomamos como referencia la transición A y el lugar P_4 entonces :

${}^\bullet A = \{P_1, P_2\}$ $\qquad A^\bullet = \{P_3, P_4\}$

${}^\bullet P_4 = \{A\}$ $\qquad P_4^\bullet = \{C\}$

2.2.3 Marcado y dinámica de la RP

Una marca se representa gráficamente por un punto en el interior de los lugares, y generalmente representa disponibilidad de recursos o datos.

Definición 2.5 *El marcado M de una RP es una función de P en $\mathbb{N} \cup \{0\}$, es decir, la asignación de un número entero no negativo (número de marcas) a cada lugar.*

En la figura 2.1 $M(P_1) = M(P_2) = 1$, $M(P_3) = M(P_4) = M(P_5) = M(P_6) = M(P_7) = M(P_8) = 0$.

El marcado representa el estado en el que se encuentra la RP o sistema en un determinado instante. Si $|P| = n$ entonces un marcado se puede representar matricialmente por un vector de n elementos $M(p_i)$ denominado vector de marcado.

Para la figura 2.1 el vector de marcado es $\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ que se le denomina marcado inicial.

La evolución del marcado le proporciona a la RP marcada una dinámica que le permite describir el comportamiento, y modelar evoluciones simultáneas de sistemas discretos.

Definición 2.6 *Una RP marcada es el par $\langle \mathcal{N}, M_0 \rangle$. en el que $\mathcal{N}$ es una RP y M_0 un marcado inicial.*

Definición 2.7 *Una transición $t \in T$ está habilitada por el marcado M sii cada uno de sus lugares de entrada posee al menos $PRE(p,t)$ marcas. Es decir, se exige que $\forall p \in {}^\bullet t$ $M(p) \geq PRE(p,t)$*

En la figura 2.1 sólo hay una transición habilitada que es la transición A.

Regla de evolución del marcado. Disparar una transición habilitada t es la operación que consiste en eliminar $C^-(p,t)$ marcas a cada lugar $p \in {}^\bullet t$ y añadir $C^+(p,t)$ marcas a cada lugar $p \in t^\bullet$

$Mi \xrightarrow{t} Mj$ significa que t está habilitada por M_i y que al disparar t a partir de M_i se alcanza M_j.

Es decir, al disparar t se obtiene:

$$M_j(p) = M_i(p) + C^+(p_j, t_i) - C^-(p_j, t_i), \qquad \forall p \in P.$$

En la figura 2.1 si se presiona el botón A, en su RP correspondiente equivale a disparar la transición A, y la RP del marcado inicial pasa al marcado $\begin{bmatrix} 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}^T$ Es decir, se quitaron las marcas de P_1,P_2 y se añadieron a P_3,P_4.

Definición 2.8 *Una secuencia de disparos σ aplicable a partir del marcado M_0 se representa por una secuencia de transiciones tal que el disparo de cada transición conduce a un marcado que habilita la transición siguiente de la secuencia. Si $M_0 \xrightarrow{t_i} M_1 \xrightarrow{t_j} M_2 \ldots \xrightarrow{t_r} M_q$, se dice que la secuencia $\sigma = t_i, t_j, \ldots\ldots t_r$ es aplicable a partir de M_0. La anterior evolución de marcado se puede reducir escribiendo : $M_0 \xrightarrow{\sigma} M_q$ (lo que significa que M_q es alcanzable desde M_0).*

Definición 2.9 *El conjunto de secuencias disparables a partir de M_0 es un lenguaje :*

$$L(< \mathcal{N}, M_0 >) = \left\{ \sigma \mid M_0 \xrightarrow{\sigma} M \right\}$$

El lenguaje que genera la RP de la figura 2.1 es $\overline{(A(BC + CB)D(EF + FE))}^*$

Definición 2.10 *Se le llama vector característico asociado a una secuencia de disparos σ al vector $\bar{\sigma} \in \mathbb{N}^m (|T| = m)$, cuya i-ésima componente es el número de ocurrencias del disparo de t_i en la secuencia σ.*

Por ejemplo si: σ = ABCDEFABCDEFAB su vector característico es:

$$\bar{\sigma} = \begin{bmatrix} 3 & 3 & 2 & 2 & 2 & 2 \end{bmatrix}^T$$

Definición 2.11 *El conjunto de vectores característicos de las secuencias disparables a partir de M_0 es:*

$$\bar{L}\,(< \mathcal{N}, M_0 >) = \left\{ \bar{\sigma} \mid M_0 \xrightarrow{\sigma} M \right\}$$

Definición 2.12 *Un marcado M es alcanzable a partir del marcado inicial M_0 sii existe una secuencia de disparos aplicable a partir de M_0 que transforma M_0 en $M : M_0 \xrightarrow{\sigma} M$.*

En la RP de la figura 2.1 el marcado $\begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^T$ no es alcanzable a partir de M_0 porque no existe una secuencia σ que nos lleve a él. Por otro lado, el marcado $\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \end{bmatrix}^T$ si es alcanzable, porque existen secuencias que nos lleven a él, por ejemplo: $\sigma = AB$, $\sigma = ABCDEFAB$, etc.

Definición 2.13 *El conjunto de los marcados alcanzables a partir de M_0 es:*

$$R(< \mathcal{N}, M_0 >) = \left\{ M \mid (\exists \sigma \in L(< \mathcal{N}, M_0 >)) \wedge (M_0 \xrightarrow{\sigma} M) \right\}$$

Definición 2.14 *El **grafo de alcanzabilidad** (grafo de estados) asociado a la RP marcada $< \mathcal{N}, Mo >$ es el grafo $G(M, E)$ donde M es el conjunto finito de nodos, cada nodo representa un marcado (estado) alcanzable a partir de Mo y E es el conjunto finito de arcos, cada arco representa el disparo de una transición. Existe un arco etiquetado t_k que va desde el nodo que representa M_i al que representa M_j sii al disparar t_k a partir de Mi se alcanza $M_j : M_i \xrightarrow{t_k} M_j$.*

Definición 2.15 *Una red marcada $\langle \mathcal{N}, M_0 \rangle$ es **sana** (binaria) sii:*

$$M(p) \leq 1 \forall p \in P \; y \;\; \forall M \in R(\langle \mathcal{N}, M_0 \rangle)$$

2.2.4 Ecuación de estado de una RP.

Sea C la matriz de incidencia de una red pura y marcada. A partir de la definición de C y de la regla de evolución del marcado se puede escribir:

(1) $M_k = M_{k-1} + C \cdot V_k$

(2) $(M_0 \xrightarrow{\sigma} M_k) \implies M_k = M_0 + C \cdot \vec{\sigma}$,

donde :

- M_k es el marcado que se obtiene al realizar el $k - ésimo$ disparo;
- V_k es el vector de disparos cuyas componentes son nulas salvo la $i - ésima$ si t_i es la transición disparada en el k-ésimo lugar : $V_k(i) = 1, \quad V_k(j) = 0, \quad \forall j \neq i$.
- $\vec{\sigma} : T \rightarrow \mathbb{N}$ es un vector de enteros no negativos, llamado el *vector de disparos.* $\vec{\sigma}\,(t)$ representa el número de veces que la transición t aparece en σ.

La primera ecuación tiene la forma de la ecuación de estados de un sistema dinámico lineal e invariante discretizado en el tiempo. La segunda ecuación integra la evolución desde M_0 hasta M_k.

Se pueden plantear diversos análisis sobre el comportamiento de la RP utilizando técnicas derivadas del álgebra lineal, pero también se tiene dos limitaciones:

1) No todo vector $\bar{\sigma} \in \mathbb{N}^m$ es admisible;

2) El vector $\bar{\sigma}$ no define unívocamente la secuencia σ por lo que se ha perdido una información fundamental para estudiar la actividad de la red.

2.2.5 Conceptos básicos para el análisis estructural de las RP.

T-semiflujos :

Todos los vectores X tal que $C \;\; X = 0, \;\; X \geq 0$ son llamados **T-semiflujos.**

Si $X(\mathrm{i}) > 0$ entonces se le llama componente repetitiva, y la RP se dice que es repetitiva. Esto se puede ver si hacemos $V_k = X$ y lo sustituimos en la ecuación de estado:

$M_{k+1} = M_0 + C \cdot V_k.$

$M_{k+1} = M_0 + C \cdot X.$ como $C \cdot X. = 0$ se reduce a:

$M_{k+1} = M_0$

En otras palabras un T-semiflujo es un conjunto de transiciones cuyo disparo nos lleva nuevamente al marcado inicial.

P-semiflujos

Todos los vectores Y tal que $Y^T \cdot C = 0, \quad Y \geq 0$ son llamados **P-semiflujos.**

Si $Y(\mathrm{i}) > 0$ entonces se le llama componente conservativa, y la RP se dice que es conservativa. Esto se puede ver si multiplicamos la ecuación de estado por Y^T, la cual quedaría de la siguiente forma :

$Y^T M_{k+1} = Y^T M_0 + Y^T C \cdot V_k.$ como $Y^T \cdot C. = 0$ se reduce a:

$Y^T M_{k+1} = Y^T M_0$

En otras palabras un P-semiflujo es un conjunto de lugares donde el número de marcas se conserva (circuitos formados por lugares donde el número de marcas permanece acotado).

El **soporte de un t-semiflujo** X se representa por $\|X\|$; se define como: $\|X\| = \{t_i \mid X(t_i) \neq 0\}$. El vector característico del soporte es: $\left\|\vec{X}\right\| = [x_i]$

$$x_i = \begin{cases} 1 \text{ si } t_i \in \|X\| \\ 0 \text{ otro caso} \end{cases}$$

$\|Y\|$ es el soporte de un P-semiflujo y se define de manera similar.

Un P-semiflujo (T-semiflujo) $Y_i(X_i)$ es de **soporte mínimo** si y sólo si no existe otro P-semiflujo (T-semiflujo) Y_i' (X_i') tal que $\|Y_i'\| \subset \|Y_i\|$ $(\|X_i'\| \subset \|X_i\|)$. Un P-semiflujo (T-semiflujo) es *canónico* si el máximo común divisor de sus componentes es 1. Un P-semiflujo (T-semiflujo) es mínimo si y sólo si es canónico y de soporte mínimo.

2.3 Algunas propiedades dinámicas y estructurales de las RP

- **Vivacidad:** Una transición t es viva para un marcado inicial dado M_0 sii existe una secuencia de disparos a partir de cualquier marcado M, sucesor de M_0, que comprenda

a t:

$$\forall M \in R(< \mathcal{N}, M_0 >) \quad \exists \sigma : M \xrightarrow{\sigma} M' \text{ tal que } \quad t \subset \sigma.$$

Una RP marcada es **viva** para M_0 sii todas sus transiciones son vivas para M_0. Una RP viva no puede bloquearse porque todas sus transiciones pueden llegar a dispararse. La proposición contraria no es cierta. Para caracterizar esta situación, se dice que una RP marcada es **parcialmente viva** para M_0, si tomando como punto de partida cualquier marcado alcanzable a partir de M_0, existe al menos una transición disparable y otra no viva. Una RP $\mathcal{N}$ es **estructuralmente viva** si existe un M_0 finito para el cual la RP marcada es *viva*.

- **Limitación:**Un lugar p es k-limitado para M_0 sii existe un entero k tal que $M(p) \leq k$ para cualquier marcado $M \in R(< \mathcal{N}, M_0 >)$. Se denomina límite del lugar p al menor entero k que verifica la desigualdad anterior.

  Una red marcada es $k-$**limitada** para M_0 sii todos sus lugares son k-limitados para $M_0 : \forall p \in P$ y $\forall M \in R(\mathcal{N}, M_0), M(p) \leq k$. Una RP es estrulturalmente limitada si es limitada para cualquier marcado inicial y finito.

  La propiedad de limitación determina la finitud del número de estados, si el sistema no es limitado entonces el sistema tiene infinidad de estados.

- Una RP es **Conservativa**, sii existe un vector $Y \in (\mathbb{N}^+)^n$ tal que $Y^T \ C = 0$ (n es el número de lugares de la RP). Es decir, si su matriz de incidencia admite un vector anulador izquierdo tal que todos sus elementos sean positivos.

- Una RP es **Repetitiva**, sii existe un vector $X \in (\mathbb{N}^+)^m$ tal que $C \cdot X = 0$ (m es el número de transiciones de la RP). Es decir, si su matriz de incidencia admite un vector anulador derecho tal que todos sus elementos sean positivos.

- Un RP posee un comportamiento globalmente **Cíclico**, para marcado inicial M_0 si existe una secuencia de disparos que permite alcanzar el marcado inicial M_0 a partir de cualquier marcado M alcanzable a partir de M_0 :

$$\forall M \in R(< \mathcal{N}, M_0 >), \quad \exists \sigma \text{ tal que } M \xrightarrow{\sigma} M_0.$$

- **Conflictividad:** Se dice que en una RP existe **conflicto estructural** cuando un lugar posee más de una transición de salida. Se dice que dos transiciones, t_i y t_j, están en conflicto efectivo para M_0 sii:

  $\exists M \in R(< \mathcal{N}, M_0 >)$, que habilita a t_i y t_j; y al disparar t_i (t_j) el marcado obtenido no habilita la transición t_j (t_i)

2.4 Métodos de análisis de propiedades de la RP

Mencionamos anteriormente que existen muchos sistemas que pueden ser optimizados, y que para ello se tiene que obtener un modelo que describa su dinámica. También mencionamos que elegimos las RP porque existen SED en los que las actividades se realizan en secuencia, y otros en los cuales existen tareas o actividades que se realizan en paralelo, y que las RP son adecuadas para modelar dichas actividades. Ahora, también sabemos que la mayoría de los sistemas son grandes y complejos, y que obtener y comprender su modelo es difícil para quien va a realizar la optimización. Además se obtienen modelos con una gran cantidad de elementos donde es probable la presencia de errores. Para evitar pasar a la etapa de realización con un modelo erróneo, es necesario introducir una etapa para realizar un estudio cualitativo (una validación funcional) del modelo y ver si el modelo cumple con una serie de propiedades (la mayoría independientes de las funciones que realiza el sistema) que caracterizan su buen funcionamiento. Estas propiedades pueden ser ausencia de bloqueos, finitud del conjunto de estados, ausencia de conflictos etc. También se hace una verificación funcional donde se examina la descripción del sistema para comprobar si cumple con las especificaciones del sistema.

Para analizar la validez de la Red de Petri obtenida existen varios tipos de análisis:

- Enumeración.
- Transformación.
- Estructural.
- Simulación.

Los tres primeros se denominan métodos estáticos. Su aplicación a las RP consideradas como grafos conducen a resultados exactos. Los métodos de simulación se denominan métodos dinámicos y permiten cierta confianza en la descripción, pero no verifican las propiedades de la RP. En seguida se mencionan los aspectos principales de los dos métodos de análisis mas comunes:

Análisis por enumeración: Se basa en la construcción del llamado grafo de alcanzabilidad (más adelante se describirá como se construye este grafo de estados) que representa individualizadamente los marcados de la RP y el disparo de sus transiciones, es decir, se basan en la simulación exhaustiva de las posibles evoluciones de la RP. Si la RP es limitada el grafo es finito y se puede verificar vivacidad, ciclicidad, limitación, conflictividad y exclusión mutua. Si la RP no es limitada entonces hace que el grafo no sea finito, y por lo tanto es imposible construirlo.

Análisis estructural: Permite demostrar una serie de propiedades de la RP, casi independientemente del marcado inicial de ésta (se basa en la estructura de la RP). Dicho de otro modo, esta técnica permite de forma eficiente el análisis de una RP para diferentes marcados iniciales. En este tipo de análisis se puede distinguir dos subgrupos, que son, los metodos basados en álgebra lineal (basados en la ecuación de estado), y el metodo de cerrojos y trampas [Silva 85].

2.5 Clasificación de RP

En esta sección se presentan algunas de las principales subclases de RP, esta clasificación se realiza basándose en restricciones sobre la estructura de las RP. Para determinar a que clase pertenece alguna RP se presentan las siguientes definiciones [Silva 85] :

- Un **Grafo o Máquina de Estados (ME)** es una RP tal que:

$$\forall t \in T \qquad |{}^\bullet t| = 1 \qquad y \qquad |t^\bullet| = 1$$

  Es decir, es aquella RP en la que toda transición tiene exactamente un lugar de entrada y uno de salida. Si se trabaja con RP binarias entonces en este caso sería como trabajar con automatas finitos y los sistemas se modelarían como si fueran secuenciales. Se pueden modelar alternativas pero no concurrencia.

- Un **Grafo Marcado (GM)** es una red de Petri tal que:

$$\forall p \in P \qquad |{}^\bullet p| = 1 \qquad y \qquad |p^\bullet| = 1$$

  Es decir, es aquella red en la que todo lugar tiene exactamente una transición de entrada y una de salida. Este tipo de redes puede modelar paralelismo y sincronización, pero no alternativas.

- Una **Red de Libre Elección (RPLE)** es una red tal que:

$$\forall p \in P,\ si\ |p^\bullet| > 1,\ \text{entonces}\ \forall t_k \in p^\bullet \quad |{}^\bullet t_k| = 1$$

  Es decir, si dos transiciones t_i y t_j tienen un lugar de entrada p en común, entonces p es el único lugar de entrada de t_i y de t_j. En este tipo de redes es posible elegir libremente la transición que se disparará.

- Una **Red de Petri Simple (RPS)** es una aquella en la que toda transición tiene como máximo un lugar de entrada compartido con otras transiciones.

2.6 Redes de Petri Temporizadas.

Para analizar el problema de scheduling con redes de Petri, utilizamos una extensión de las mismas denominadas Redes de Petri Temporizadas (RPT) en las que se introduce el tiempo para realizar el análisis de sistemas temporizados. Este tipo de modelo se utiliza generalmente para la evaluación del desempeño (performance) de los sistemas, calcular tiempos de ciclos y algunas otras propiedades de los sistemas temporizados.

Definición 2.16 *Una RP temporizada [Ramchandani 74] [Ramírez 93a] RPT* $= \langle \mathcal{N}, M_0, D \rangle$ *donde* D *es una función que asigna un número real no negativo* d_i *a cada transición de la red,* $D : T \rightarrow \mathbb{R}^+ \quad D(t_i) = d_i$ *se llama duración de la transición* t_i *(tiempo de ejecución) e indica el tiempo que tarda la transición* t_i *en ser disparada.*

Existen dos formas de trabajar en cuanto a la asociación del tiempo con la RP, un modelo asocia el tiempo a las transiciones y otro asocia tiempos a los lugares [Sifakis 78], en esta memoria trabajamos con el primero por ser más natural, ya que las transiciones expresan las actividades del sistema, y sin olvidar que el disparo de t_i dura d_i unidades de tiempo.

En la RPT una marca puede tener cualquiera de los siguientes dos atributos: *disponibles y no disponibles*. Para el marcado inicial todas las marcas están disponibles y su estado a no disponibles cambia de acuerdo a la siguiente regla de disparo.

2.6.1 Regla de disparos en una RPT

Una transición puede dispararse si está habilitada; una transición está habilitada si $M(p_i) \geq PRE(p_i, t_i) \quad \forall p_i \in {}^\bullet t_i$, donde $M(p_i)$ son marcas *disponibles*; el disparo de una transición t_i congela $PRE(p, t_i)$ marcas de cada lugar de entrada p (o sea, no desmarca los lugares, pero si las etiqueta como *no disponibles*, de forma tal, que ninguna otra transición las puede utilizar en su disparo), pero no es sino hasta d_i unidades de tiempo más tarde cuando deposita $POST(p, t_i)$ marcas disponibles en el lugar de salida respectivo p. El marcado M_0 contiene sólo marcas diponibles.

Como se mencionó anteriormente en la sección de propiedades de las RP, las transiciones en **conflicto** [Ramírez 93t] son aquellas que poseen un predecesor común. Se dice que las transiciones están en conflicto bajo un marcado M si todas ellas están habilitadas, pero sólo una se puede disparar. Las transiciones participantes en el conflicto tienen el problema de que el disparo de una de ellas impide el disparo de las otras transiciones del conflicto. Para solucionar el problema, se restringe la evolución del sistema agregándole un número a las transiciones del conflicto para indicar la proporción de disparos entre dichas transiciones. Por ejemplo, supongamos que tenemos dos transiciones t_1 y t_2 donde t_1 se dispara 2 veces por cada 5 de t_2. Sea σ una secuencia disparable y $\#(\sigma, x)$ el número de veces que aparece el símbolo x en las secuencia σ, entonces:

$$\lim_{long(\sigma)\to\infty} \frac{\#(\sigma,t_1)}{\#(\sigma,t_2)} = \frac{2}{5}$$

En las RPT las transiciones t_i de un conflicto tienen asociado un número r_i (tiempo de ejecución). Siguiendo la observación anterior, para un par de transiciones pertenecientes al mismo conflicto:

$$\frac{\#(\sigma, t_i)}{\#(\sigma, t_j)} = \frac{r_i}{r_j}; \qquad r_i \cdot \#(\sigma, t_j) = r_j \cdot \#(\sigma, t_i) \qquad long(\sigma) \to \infty \tag{2.1}$$

Además, como la red es limitada, las secuencias de ejecución infinitas tienen que ser generadas por un t-semiflujo, así pues, podemos calcular dicho t-semiflujo de la siguiente forma:

$$\begin{pmatrix} C \\ R \end{pmatrix} \cdot v_k = 0 \tag{2.2}$$

donde R es la matriz que resume las restricciones introducidas por el sistema de ecuaciones (2.1).

El sistema de ecuaciones (2.2) tiene muchas soluciones, pero si se normaliza para una de sus transiciones t_k, se obtiene una condición más $v_k(k) = 1$ y si la red es una FRT entonces el sistema tiene una única solución [campos 90]. Al vector v_k que se obtiene se le llama *vector de ratios de visita (visit ratio).*

En el análisis se considera que los tiempos que están asociados a las transiciones son deterministas, también que las transiciones en conflicto tienen asociado un tiempo cero, a estas transiciones se les denomina inmediatas y sirven para que no se acoplen las decisiones de los conflictos. Enseguida se dan algunas definiciones que son de utilidad para el análisis de la vivacidad de una RPT

2.6.2 Conceptos básicos para el análisis estructural en la RP.

Definición 2.17 ***T-Componente.*** *Sea $\mathcal{N} = <P, T, F>$ una red de Petri. Un grafo marcado fuertemente conexo $\mathcal{N}' = <P', T', F'> \subseteq \mathcal{N}$ es una t-componente de $\mathcal{N}$ si y sólo si $P' =^{\bullet} T' = T'^{\bullet}$ En especial, una t-componente es mínima si $T' = \|X\|$, donde X es un t-semiflujo mínimo de $\mathcal{N}$*

Definición 2.18 ***P-Componente.*** *Sea $\mathcal{N} = <P, T, F>$ una red de Petri. Una máquina de estados fuertemente conexa $\mathcal{N}' = <P', T', F'> \subseteq \mathcal{N}$ es una p-componente de $\mathcal{N}$ si y sólo si $T' =^{\bullet} P' = P'^{\bullet}$. En especial, una p-componente es mínima si $P' = \|Y\|$, donde Y es un p-semiflujo mínimo de $\mathcal{N}$*

Definición 2.19 ***Ejecución de una t-componente.*** *Sea $\mathcal{N} = <P, T, F>$ una red, y $\mathcal{T} = (P', T', F') \subseteq \mathcal{N}$ una t-componente de N; $\exists M \in R(\langle \mathcal{N}, M_0 \rangle)$ y $\sigma = t_i t_j \ldots t_k \ldots$ con $t_i, t_j, \ldots \in T'$ es una secuencia disparable en $\langle \mathcal{T}, M \rangle$. Se dice que σ es una ejecución de la t-componente $\mathcal{T}$ si y sólo si $M \xrightarrow{\sigma} M$ y σ es de longitud mínima > 0. El hecho de que una t-componente $\mathcal{T}$ está siendo ejecutada al tiempo τ se denota por $\varepsilon(\mathcal{T}, \tau)$.*

2.6.3 Conceptos básicos de RPT para la búsqueda del Schedule óptimo.

Definición 2.20 *Sea $\mathcal{N} = \langle P, T, F \rangle$ una red de Petri, la* ***velocidad de disparo*** *(Throughput) de una transición $t_i \in T$ es el número de disparos de t_i por unidad de tiempo cuando el tiempo tiende a infinito.*

Definición 2.21 *El* ***tiempo de ciclo*** *de t_i es el inverso de su velocidad de disparo.*

Definición 2.22 *El* ***tiempo de ciclo*** *de una red con respecto a un t-semiflujo disparable X_i (de números enteros), es el tiempo que tarda en dispararse X_i una sola vez.*

Definición 2.23 *Un* ***schedule factible S*** *se define por una secuencia de pares ordenados (i_k, τ_k) , donde $\sigma = t_{i_1} t_{i_2} \ldots t_{i_k} \ldots$ es una secuencia de disparos realizable en la RP considerada compatible con las ratios de visita de las transiciones ($\bar{\sigma} = r \cdot v_k$, $r \in \mathbb{N}) t_{i_k}$, donde r es el tiempo de ejecución y τ_k es el tiempo de inicio de disparo de la transición t_{i_k} en σ*

Definición 2.24 *Un* ***schedule bueno*** *con respecto a una cota "c" es un schedule que se calcula en tiempo polinomial en $|P| + |T|$ y el tiempo de ciclo de la red que resulta de aplicar dicho schedule es menor que la cota "c"*

Definición 2.25 *El* ***Schedule óptimo*** *es un schedule factible cuya ejecución permite maximizar la velocidad de disparo de las transiciones del sistema.*

Definición 2.26 *Una secuencia s_n es K-periódica, con periódo r, si existe un entero N_0 tal que:*

$\forall n \geq N_0 \qquad n - N_0 = Kq + \rho (0 \leq \rho < K) \Rightarrow s_n = s_{N_0 + \rho + qr}$

La frecuencia de s_n es definida por K/r

2.7 Algunas propiedades y resultados de las subclases de RP

En la literatura relacionada con RP, se pueden encontrar algunos resultados que nos ayudan a obtener el schedule óptimo, enseguida se mencionan los más importantes.

2.7.1 Grafos Marcados fuertemente conexos

En el caso de GM, los P-semiflujos son los circuitos de la red, y la dimensión de la base de p-semiflujos es $|P| - |T| + 1$ y el rango de la matriz C es $|T| + 1$. En un GM sólo existe un T-semiflujo con todos sus elementos iguales a 1.

[Ramamoorthy 80] Si $Si(k)$ representa el tiempo de inicio del k-ésimo disparo de la transición t_i, entonces el tiempo de ciclo de la transición t_i es :

$$C_i = \text{Tiempo de ciclo} = \lim_{k \to \infty} \frac{Si(k)}{k}$$

Por ejemplo, si tenemos una transición t_3 que se dispara por cuarta vez a los 8 segundos, entonces $k = 4$ y $S_3(4) = 8$ segundos, entonces :

$$C_3 = \lim_{k \to \infty} \frac{8}{4} = 2 \; seg.$$

Teorema 1 (Ramamoorthy 80) *En grafos marcados fuertemente conexos todas la transiciones tienen la misma velocidad de disparo.*

Teorema 2 (Ramamoorthy 80) *El tiempo de ciclo de un grafo marcado es igual al tiempo de ciclo del circuito más lento.*

Tomando como base los resultados anteriores, se puede utilizar la regla [Chretienne 83]:

"En grafos marcados, disparando las transiciones tan pronto como están habilitadas se obtiene un schedule óptimo"

Esta regla se cumple debido a que no hay decisiones, por lo cual, el disparar una transición evita retardos en el circuito más lento y además, no puede llevar a una mala selección porque no hay recursos compartidos, lo que permite alcanzar el tiempo de ciclo óptimo.

En GM se puede obtener una cota inferior del tiempo de ciclo óptimo (π^*), buscando el p-semiflujo más lento [Campos 90], esto se realiza resolviendo el siguiente problema de programación lineal:

$$\pi = \max_{Y} \; Y^T \; C^- \cdot D$$
sujeto a:
$$Y^T \cdot C = 0$$
$$Y^T \; Mo = 1$$
$$Y \geq 0$$

Donde D es el vector que contiene los tiempos de duración de las transiciones.

Ejemplo 2.1 *Obtener el p-semiflujo más lento y el tiempo de ciclo de la RP de la figura 2.2 (a).*

Los p-semiflujos son: $Y_1 = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}$, $Y_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}$, $C^- = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ y $D = \begin{bmatrix} 1 \\ 2 \\ 4 \\ 4 \end{bmatrix}$

Multiplicando $Y_1^T \cdot C^-$ tenemos $\begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix} \cdot D = 1 + 2 + 0 + 4 = 7$

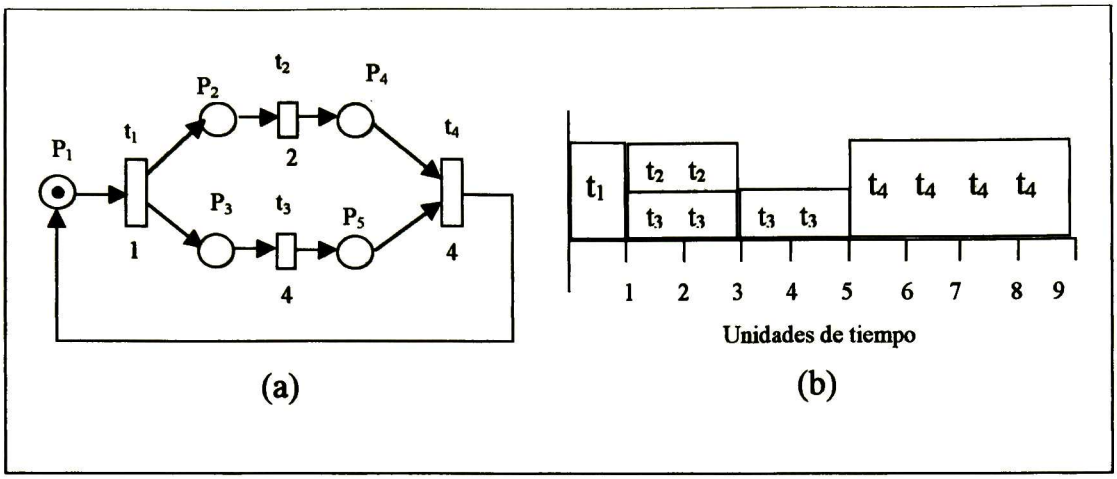


Figura 2.2: RP de ejemplo para obtener el p-semiflujo más lento y el tiempo de ciclo.

Multiplicando $Y_2^T \cdot C^-$ tenemos $\begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \cdot D = 1+0+4+4 = 9$

Como se puede ver el p-semiflujo más lento es Y_2^T = [1,1,0,1] y el tiempo de ciclo es 9. En esta red se ejecutan transiciones en paralelo, en este caso t_2 y t_3, en la figura 2.2 (b) se muestra un diagrama de Gantt donde se puede apreciar los instantes donde se ejecutan transiciones en paralelo, y que la ejecución óptima de ese sistema se puede obtener con el siguiente schedule, $S_1 = (t_1, 0), (t_2, 1), (t_3, 1), (t_4, 5)$, que en total consume 9 unidades de tiempo.

También es importante notar pueden existir diferentes schedules que nos permiten alcanzar el tiempo de ciclo óptimo, por ejemplo : $S_2 = (t_1, 0), (t_2, 2), (t_3, 1), (t_4, 5)$ y $S_3 = (t_1, 0), (t_2, 3), (t_3, 1), (t_4, 5)$ también cumplen con la ejecución óptima.

2.7.2 Máquinas de Estado fuertemente conexas

En este tipo de redes no existen sincronizaciones, todos los marcados son reversibles (incluyendo el marcado inicial), y el tiempo de ciclo es una suma ponderada de los tiempos de las transiciones (no hay operaciones de máximo, ya que no existen sincronizaciones).

El tiempo de ciclo de una ME se obtiene resolviendo el siguiente problema de programación lineal [Campos 90]:

$$\pi = \max_{Y} \; Y^T \; C^- \begin{bmatrix} d(t_1)\; V_k(t_1) \\ \vdots \\ d(t_m)\; V_k(t_m) \end{bmatrix}$$

sujeto a:
$Y^T \cdot C = 0$
$Y^T \; Mo = 1$

Para calcular el schedule, suponemos que se dispara un t-semiflujo que cumple con los ratios de visita, el cual proporciona la ponderación de la suma. Usando las siguientes propiedades posteriormente, podemos obtener el schedule óptimo.

Propiedades de las máquinas de estado.

- Los T-semiflujos son los circuitos de la red, la ecuación $C \cdot X = 0$ proporciona una base de t-semiflujos, los cuales generan una base de t-componentes mínimas y la dimensión de la base de T-semiflujos es $|T| - |L| + 1$.
- Existe una cobertura por t-componentes (GM) de la ME [Thiagarajan 84].
- En una ME sólo existe un P-semiflujo con todos sus elementos iguales a 1.
- El tiempo de ciclo óptimo cumpliendo con los ratios de visita, se calcula en tiempo polinomial [Campos 90].
- La ratio de visita v_k es una combinación lineal no negativa de una base β_T de t-semiflujos mínimos (Demostración [Ramirez 93t]), es decir, $v_k = \sum \alpha_i X_i$. con $\alpha_i > 0$.
- Si las t-componentes $\mathcal{T}_i$ inducidas por los t-semiflujos X_i encontrados en la característica anterior son ejecutados α_i veces respectivamente, entonces se alcanza el tiempo de ciclo óptimo.
- Si una $\mathcal{T}_i$ está siendo ejecutada y al marcarse el lugar de distribución p_k se habilita $\mathcal{T}_j$, entonces es posible ejecutar $\mathcal{T}_j$ completamente y terminar en el mismo lugar p_k en el que $\mathcal{T}_i$ fue abandonado. Esto se cumple porque p_k es un lugar compartido por ambas t-componentes; cuando p_k esta marcado, las t-componentes $\mathcal{T}_i$, $\mathcal{T}_j$ se pueden ejecutar. Si $\mathcal{T}_j$ se ejecuta, entonces acabará su ejecución dejando una marca en p_k porque $\mathcal{T}_j$ es un ciclo.

Tomando como base los resultados anteriores, se puede utilizar la en ME la siguiete regla de ejecución [Ramirez 93a]:

"Un schedule óptimo se obtiene disparando las transiciones de las t-componentes tan pronto como están sensibilizadas; y disparando cada t-componente ($\mathcal{T}_i$) α_i veces". Si una nueva t-componente se sensibiliza, entonces se debe empezar su ejecución. Una vez que una t-componente ha sido disparada α_i veces, entonces ésta no debe ser considerada nuevamente en el ciclo actual; la ejecución se continua con el resto de las t-componentes.

2.7.3 Redes Libre Elección (RLE)

En esta sección se presentan algunos conceptos y definiciones sobre las RLE necesarios para entender cómo se obtiene el schedule óptimo de este tipo de redes.

Los lugares y transiciones que tienen dos o más arcos de entrada se denominan de atribución y conjunción respectivamente. Los lugares y transiciones con dos o más arcos de salida de denominan de selección y distribución respectivamente.

Definición 2.27 *Si $\mathcal{N} = <P, T, F>$ es una RLE, entonces una t-componente de N es un grafo marcado fuertemente conexo $<P_1, T_1, F_1>$ que satisface : $P_1 \subseteq P$, y $F_1 = F \cap ((P_1 \times T) \cup (T \times P_1))$ y por tanto $T_1 = {}^\bullet P_1 = P_1^\bullet$; y una p-componente de $\mathcal{N}$ es una ME fuertemente conexa $<P_1, T_1, F_1>$ que satisface: $T_1 \subseteq T$, y $F_1 = F \cap ((P \times T_1) \cup (T_1 \times P))$ y por tanto $P_1 = {}^\bullet T_1 = T_1^\bullet$*

Las ratios de visita [Campos 90] son números racionales, ya que la matriz de incidencia y la matriz R obtenida de las proporciones de disparo entre transiciones están formadas por números enteros.

Definición 2.28 ***Ejecución cíclica de una RLE.*** *Sea $\mathcal{N} = < P, T, F >$ una RLE, $S = (t_{i_1}, \phi_1)(t_{j_2}, \phi_2), ...(t_{i_k}, \phi_k)...$ con $t_i, t_j ... \in T$, y $\sigma = t_{i_1}, t_{j_2}, ...t_{i_k}...$ la secuencia de transiciones que se genera al ordenar las transiciones t_{i_k} de acuerdo a su tiempo de inicio de disparo ϕ_k. Se dice que S es una ejecución cíclica de la red $\mathcal{N}$ si y sólo si S es un schedule factible en $< \mathcal{N}, M_o >$.*

Definición 2.29 ***Tiempo mínimo de ciclo*** *de una RLE Sea $\mathcal{N} = <P, T, F>$ una RLE, S_i una ejecución cíclica de la red y $\tau(S_i)$ el tiempo asociado a la ejecución de S_i. El problema de encontrar el mínimo tiempo de ciclo del sistema $< \mathcal{N}, M_o >$ consiste en determinar :*

$$\pi = \min_i \tau(S_i)$$

Definición 2.30 ***Tiempo de ejecución cíclico de una t-componente y de una transición.*** *Sea $\Sigma =< \mathcal{N}, M_o >$ una RLE temporizada marcada o sistema y $Q = \{\mathcal{T}_i \mid \mathcal{T}_i$ es una t-componente de $\mathcal{N}\}$. El tiempo de ejecución cíclica de un elemento $x \in Q \cup T$ es :*

$$\tau(x) = \begin{cases} \text{si } x = t_i \in T & d(x) \\ \text{si } x \in Q & \begin{cases} \max \; Y^T \cdot C^- \mid_x \cdot diag(D \mid_x) \\ \text{sujeto a:} \\ Y^T \cdot C^- \mid_x = 0 \\ Y^T \; Mo = 1 \\ Y \geq 0 \end{cases} \end{cases}$$

donde d(x) es el tiempo asociado a la transición x y $C^- \mid_x$ indica que se refiere a la matriz C^- de la t-componente x y $D \mid_x$ a los tiempos de las transiciones en la t-componente x.

Teorema 3 (Magott 87) *El cálculo del tiempo mínimo de ciclo en una RLE viva y sana es un problema NP completo.*

Para calcular una cota inferior del tiempo de ciclo en una RLE se utiliza la siguiente ecuación [Campos 90] :

$$\pi_k^* \geq \pi_k = \max\ (Y^T \cdot C^- \cdot diag(D)\ \ V_k$$
sujeto a:
$Y^T \cdot C = 0$
$Y^T \cdot Mo = 1$
$Y \geq 0$

donde $diag(D)$ es la matriz de dimensión $|T| \times |T|$; el elemento $diag(D)[i,i] = D(i)$ (el tiempo asociado a la transición t_i); y $diag(D)[i,j] = 0$ si $i \neq j$.

Definición 2.31 *Par Distribución-Conjunción en un GM. Sea* $\mathcal{N} =< P,T,F >$ *un grafo marcado,* $[t_k,t_i]$ *forman un par distribución-conjunción si y sólo si* $\mid t_k^\bullet \mid > 1$ *y* $\mid^\bullet t_k \mid > 1$ *y* $\forall Y^T \cdot C = 0$ *tal que si* $t_k \in {}^\bullet \parallel Y \parallel \cup \parallel Y \parallel^\bullet$, *se verifica que* $t_i \in {}^\bullet \parallel Y \parallel \cup \parallel Y \parallel^\bullet$

Definición 2.32 *Par Distribución-Conjunción en una RLE. En una RLE dos transiciones* t_k *y* t_h *forman un par distribución-conjunción si y sólo si son un par distribución-conjunción cualquier t-componente mínima de la red.*

Además en una RLE $\mathcal{N} =< P,T,F >$, un elemento x $\in P \cup T$ está entre un par distribución-conjunción $[t_k,t_h]$ si y sólo si existe un camino de t_k a x que no contiene a t_h y otro camino de x a t_h que no contiene a t_k, en cualquier t-componente mínima de la red. Si un lugar de selección p_d, se encuentra entre $[t_k,t_h]$ se denota por $\vdash p_d \dashv$

Proposición 2.1 *Cubrimiento por t-componentes [Thiagarajan 84]. Sea* $\mathcal{N} =< P,T,F >$ *una RLE y* $x \in P \cup T$. *Existe una t-componente* $\mathcal{N}' =< P',T',F' > \subseteq \mathcal{N}$ *tal que* $x \in P' \cup T'$

Proposición 2.2 *Sea* $\mathcal{N} =< P,T,F >$ *una RLE sana y viva, sea* $\mathcal{B}$ *una base de t-semiflujos mínimos; y* $\mathcal{T}$ *el conjunto de t-componentes inducidas por los t-semiflujos de* $\mathcal{B}$ *con* $\sum_{b_i \in \mathcal{B}} \alpha_i b_i = V_k$ *y* $\alpha_i \geq 0$.

$$Si\ \pi^* = \sum_i \alpha_i \tau_i(\mathcal{T}_i) \Longrightarrow\ \forall \tau \geq 0 \exists \mid \mathcal{E}(\mathcal{T}_i,\tau)$$

Es decir, si el tiempo de ejecución óptimo se calcula como la suma ponderada de los tiempos de ejecución de unas t-componentes mínimas cuya combinación lineal resulta en V_k, *entonces en cada instante* τ *existe una única t-componente* $\mathcal{T}_i$ *ejecutándose.*

Definición 2.33 *Una RLE es* A^* (FCA^*) *[Ramírez 93t] si no tiene lugares de selección entre pares distribución conjunción en ninguna de sus t-componentes mínimas.*

Proposición 2.3 *Sea* $\sum =< \mathcal{N}, M_o >$ *un sistema, con* $\mathcal{N}$ *sana y viva; sea* $\mathcal{X}_b$ *una base de t-semiflujos mínimos de* $\mathcal{N}$ *y* $\mathcal{T}_b$ *el conjunto de t-componentes inducidas por los t-semiflujos de* $\mathcal{X}_b$. $\mathcal{N}$ *es FCA* si y sólo si las t-componentes de* $\mathcal{T}_b$ *no tienen ningún lugar de selección entre pares distribución-conjunción.*

Corolario 2.1 *La caracterización de una FCA* [Ramírez 93t] se realiza en tiempo polinomial. Sólo es necesario encontrar una base de t-componentes mínimas (la cual se calcula en tiempo polinomial) y comprobar que al quitar de cualquier t-componente un lugar que sea de selección en la red original, entonces dicha t-componente es acíclica. Es decir, todos los ciclos de una t-componente de una FCA* pasan por los lugares de selección.*

Proposición 2.4 *En una red FCA*[Ramírez 93t], los pares distribución-conjunción son los mismos para cualquier conjunto de t-componentes inducidos por los t-semiflujos* $b_i \in \mathcal{B}_k$, *donde* $\mathcal{B}_k$ *es una base de t-semiflujos mínimos de la red.*

Proposición 2.5 *Sea* $\mathcal{N} =< P, T, F >$ *una red FCA*[Ramírez 93t]; y* $\mathcal{B}_k$ *una base de t-semiflujos mínimos de* $\mathcal{N}$ *y* $\mathcal{T}_k$ *el conjunto de t-componentes inducidas por los t-semiflujos de* $b_i \in \mathcal{B}_k$. *El tiempo de ciclo óptimo* π^* *se calcula como:*

$$\pi^* = \sum_{\mathcal{T}_{i_k} \in \mathcal{T}_k} \alpha_i \tau(\mathcal{T}_{i_k})$$

$$\text{donde} \sum_{b_{i_k} \in \mathcal{B}_k} \alpha_{i_k} b_{i_k} = v_j$$

En este capítulo se presentaron a las RP como herramienta de modelado y análisis, se definieron los conceptos básicos sobre RP, se presentaron sus principales propiedades, métodos de análisis y las principales clases de RP. También se presento a las RPT (una extensión de las RP), las cuales toman encuenta el tiempo. Además, se presentaron los principales resultados que existen relacionados al problema de scheduling sobre las subclases de RP tratadas en este capítulo.

En el siguiente capítulo, se presenta la forma en que se obtiene el schedule óptimo, partiendo de la construción del grafo diligente de la RP y la aplicación del algoritmo A*.

Capítulo 3

RP, Grafo Diligente y Algoritmo A* en Scheduling

Resumen: En este capítulo se menciona brevemente cómo se obtiene un schedule óptimo para una cierta clase de redes de Petri utilizando el grafo de alcanzabilidad, y mencionando, el por qué no es útil para otras clases de redes de Petri. Se define el grafo diligente (GD) y cómo construirlo, el cual utiliza el algoritmo A* para encontrar un schedule óptimo; este algoritmo es ilustrado con un ejemplo.

3.1 Introducción

Tal vez la forma más sencilla de obtener un schedule óptimo o bueno de un SED modelado con RP sea construyendo primeramente, su grafo de alcanzabilidad (GA) completo, y a éste aplicarle un método de búsqueda de los ya existentes, para encontrar la trayectoria óptima entre el nodo inicial y el nodo meta (al nodo que se desea llegar). Desafortunadamente, encontrar la solución de esta forma es muy lenta, ya que se tienen que analizar todos los posibles caminos que existen entre el nodo inicial y el nodo meta. Además existen varios aspectos importantes que limitan el uso de esta técnica. Por ejemplo, si la RP tiene muchos elementos, su GA tendrá muchos nodos, lo cual aumenta el grado de dificultad para encontrar la trayectoria óptima, ya que crece de manera no lineal la cantidad de trayectorias que tienen que ser analizadas (se produce una explosión combinatoria de las trayectorias), por esta razón se dice que el problema de scheduling es de naturaleza altamente combinatoria y de complejidad NP.

Otra limitante es que el GA describe los sistemas como si fueran secuenciales, no describe actividades que se realizan en paralelo, por lo cual si en el sistema se realizan actividades en paralelo, no se asegura encontrar el schedule óptimo. Por ejemplo, en la figura 3.1 se muestra una ME con su respectivo GA, se pueden modelar alternativas, pero si se tiene sólo una marca en la RP, no hay actividades en paralelo, ni sincronizaciones, y en este caso, la RP de la figura 3.1-a, y el grafo de alcanzabilidad de la figura 3.1-b son idénticos (esto si a la RP le quitamos los segmentos de línea que representan las transiciones). Para esta clase de RP el GA si es útil y permite obtener el schedule óptimo.

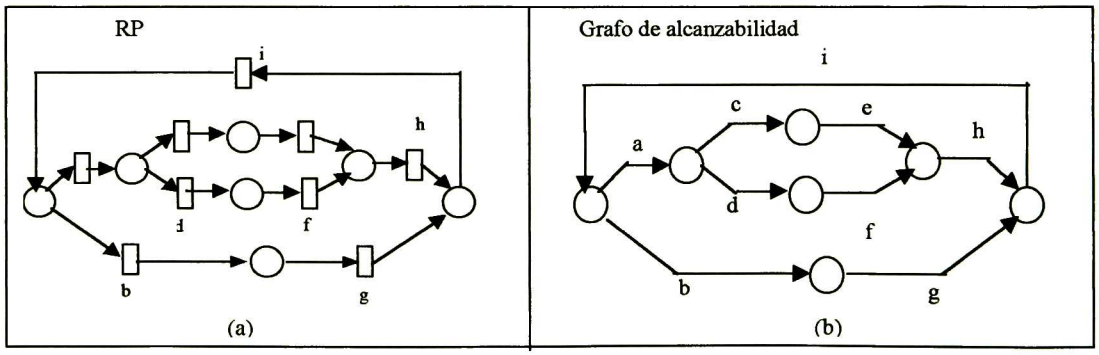


Figura 3.1: Ejemplo de ME con su respectivo GA.

Existen otros tipos de RP como GM y RPLE, donde existen sincronizaciones y actividades en paralelo, para las cuales el GA no es útil porque no permite que se realize el disparo de dos o más transiciones simultáneamente, y por lo tanto, no permite encontrar el schedule óptimo. En la figura 3.2-a se presenta una RP con sincronizaciones y paralelismo, y en la figura 3.2-b su correspondiente GA, donde se puede apreciar en sus arcos, que nunca se pueden disparar transiciones al mismo tiempo (siempre una a la vez). Por tal motivo en este trabajo no se utiliza el GA y se prefiere trabajar con el grafo diligente, ya que si permite representar transiciones que se disparan simultáneamente y con el algoritmo A* como método de búsqueda, se puede encontrar un schedule óptimo.

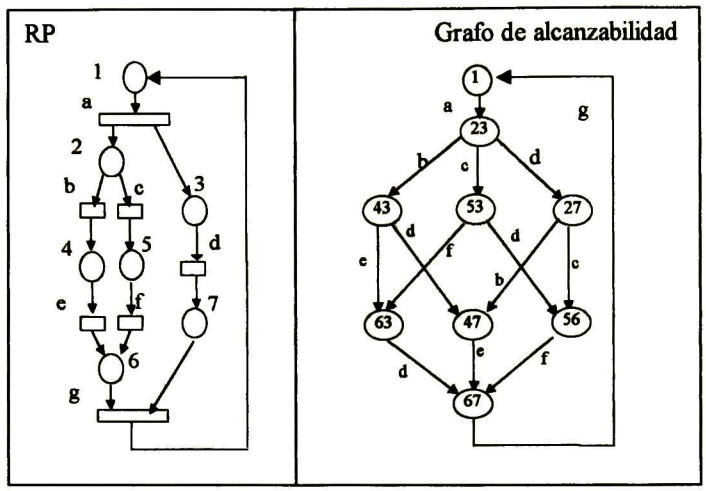


Figura 3.2: RP con paralelismo y su GA.

3.2 Grafo diligente

En esta sección se presenta una forma de obtener un schedule óptimo, pero primero se darán algunas definiciones y conceptos previos para comprender el Grafo Diligente (GD) de una RP [Carlier 84], que será utilizado en el algoritmo de optimización.

Una transición t_j está activa en el instante w si el tiempo de inicio de disparo de t_j, n_{t_j} cumple con:

$$n_{t_j} \leq w < n_{t_j} + d(t_j)$$

Definición 3.1 *El tiempo residual de una transición t_j en el instante w, se define como:*

$R_{t_j}(w) = 0$ *si t_j no está activa en el instante w*

$R_{t_j}(w) = n_{t_j} + d(t_j) - w$ *si t_j está activa en el instante w*

R(w) es el vector de los tiempos residuales al instante w.

Definición 3.2 *El grafo diligente de una RP se define como $GD = (V, A)$, donde V es un conjunto de vértices y A es un conjunto de arcos. El vértice $E^0 = (M_0, 0) \in V$. Si $E = (M, R) \in V$, entonces con cada subconjunto $\mathcal{T}$ de transiciones (incluso el conjunto vacío) satisfaciendo:*

$$M - \sum_{t \in \mathcal{T}} C^-(\cdot, t) \geq 0$$

se asocia un arco (E, E'), con peso δ, etiquetado por t; donde $E' = (M', R')$ y δ están definidas como sigue:

$$\delta = \min\{\ \min\ \{\ R_t | R_t > 0 \quad y \quad t \in T\ \},\ \min\{d(t) \mid t \in \mathcal{T}\ \}\};$$

$$sea\ T' = \{t|t \in T \quad y \quad R_t = \delta\} \cup \{t|t \in \mathcal{T} \quad y \quad d(t) = \delta\ \};$$

$$M' = M + \sum_{t \in T'} C^+(\cdot, t) - \sum_{t \in \mathcal{T}} C^-(\cdot, t);$$

$$\begin{aligned} R'_t &= 0 && si\ t \in T' \\ R'_t &= d(t) - \delta && si\ t \in \mathcal{T} \backslash T' \\ R'_t &= max\{0, R_t - \delta\} && si\ t \in T \backslash \mathcal{T} \backslash T' \end{aligned}$$

En la figura 3.4 se muestra el grafo diligente de la red de la figura 3.3. Los cuadros con esquinas redondeadas representan los nodos, cada nodo representa un posible estado de la red, en cada nodo se escriben los lugares que se encuentran marcados en un determinado instante, así como también el tiempo residual de las transiciones que están activas.

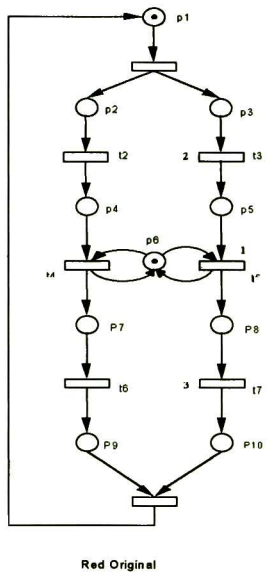


Figura 3.3: RP para el ejemplo del GD.

En la red original (figura 3.3) se puede ver que el estado inicial es cuando los lugares p1 y p6 tienen una marca. La única transición disparable es t1 con un tiempo de ejecución de 0 unidades de tiempo. Disparando t1 llegamos al estado donde m(p2)=m(p3)=m(p6)=1 con un tiempo residual de cero, los lugares restantes tienen cero marcas. En este nuevo estado se encuentran habilitadas t2 y t3, en el grafo diligente que se muestra en la figura 3.4, se puede ver que tenemos tres posibles alternativas, disparar t2, disparar t3 o ambas simultáneamente, cualesquiera de ellas nos lleva a estados diferentes. A diferencia del grafo de alcanzabilidad, aquí se puede representar el disparo de dos o más transiciones simultáneamente, y por tanto se puede representar el paralelismo de los sistemas.

Observación : En realidad, cuando en el grafo diligente (figura 3.4) estamos en el estado (p2,p3,p6 ; 0) existen muchas alternativas de disparo, por ejemplo, si se dispara t3, que

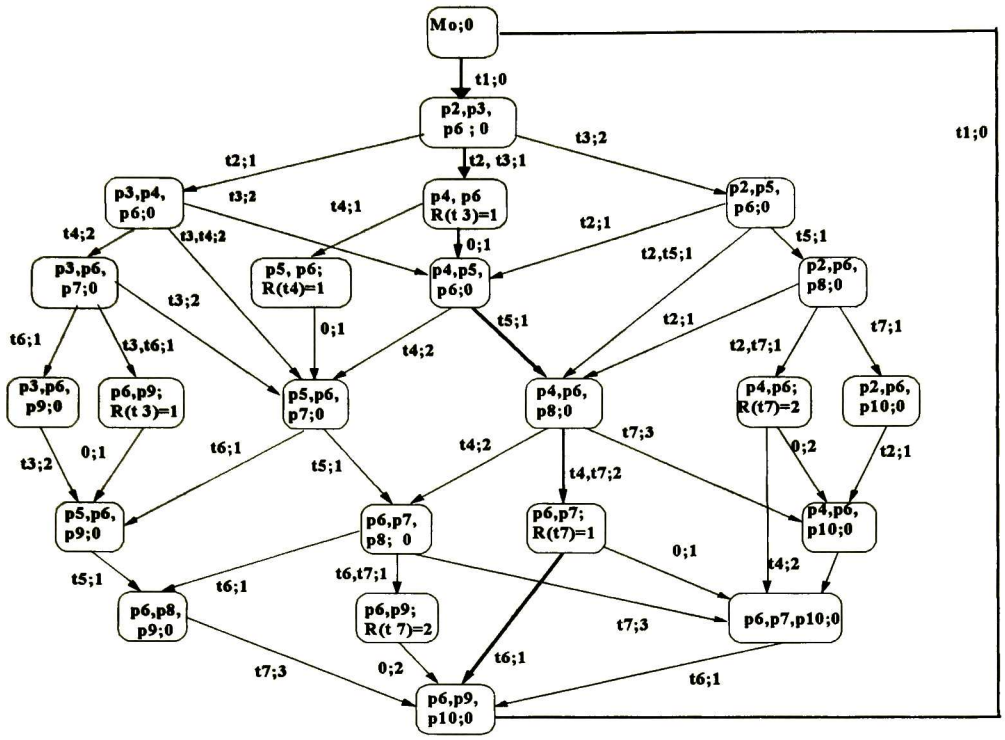


Figura 3.4: Ejemplo de un grafo diligente

consume 2 unidades de tiempo, el disparo de t2 se puede realizar a .3 unidades de tiempo después de disparar t3 o .5, .7, .967, 1, 1.2 o 1.5 unidades de tiempo por mencionar algunas, pero ninguna de ellas se encontrará en la trayectoria óptima. La razón es que si se puede disparar t2 en cualquiera de los tiempos mencionados, entonces también se podía haber disparado al mismo tiempo que t3, por lo que no tendría caso esperar algún tiempo, ya que, no nos llevaría al schedule óptimo, por esto el grafo diligente no las incluye.

Si estamos en $(p2, p3, p6; 0)$, existen tres alternativas, disparar t_2, t_3 o ambas simultáneamente. Si se dispara t_2, después de una unidad de tiempo se llega al estado $(p3, p4, p6; 0)$ de donde existen otras tres alternativas de disparo como se muestra en la figura 3.4, si se dispara t_3, después de dos unidades de tiempo se llega al estado $(p2, p5, p6; 0)$ de donde se tienen tres alternativas de disparo como se muestra en la figura 3.4; pero, si se disparan t_2 y t_3 simultáneamente, entonces después de una unidad de tiempo se llega al estado $(p4, p6, \mathrm{R}(t_3)=1)$, esto es, t_2 quita una marca de $p2$ y la deposita en p4, p6 queda igual, t_3 quita una marca de p3, por lo cual, el lugar p3 ya no esta marcado, pero existe un tiempo residual de una unidad de tiempo, que es el tiempo que le falta a t_3 para consumirse. De este estado $(p4, p6, \mathrm{R}(t_3)=1)$, existen dos alternativas, el disparo de t4 y llegar al estado $(p5, p6, \mathrm{R}(t_4)=1)$, o no disparar ninguna transición y dejar pasar una unidad de tiempo para que termine el tiempo de ejecución de t3 y llegar al estado $(p4, p5, p6; 0)$. Siguiendo esta descripción se construye todo el grafo diligente que se muestra en la figura 3.4.

3.3 Algoritmo A*

Existen varios métodos para encontrar la distancia mínima de un nodo a otro en un grafo. Algunos algoritmos encuentran el mejor camino (el más corto), pero tienen que analizar todos los nodos del grafo, por lo que encontrar la solución en grafos grandes toma mucho tiempo. Para encontrar el camino de menor tiempo, se han desarrollado métodos de búsqueda, que encuentran la solución sin tener que analizar todo el grafo. Estos algoritmos utilizan una función de evaluación que les indica de alguna forma la dirección de la búsqueda. Por ejemplo, para encontrar el camino más corto, a partir del nodo actual, se analizan todos sus nodos vecinos y se determina cúal es el nodo más prometedor y, a través de él, se continua la búsqueda. A la función de evaluación también se le llama función guía, porque va indicando que nodos se deben analizar.

Para determinar si el algoritmo es bueno o no, podemos tomar como referencia ciertos criterios que se muestran en seguida:

- Completitud : Es la estrategia que garantiza encontrar una solución cuando ésta existe.
- Complejidad en tiempo: Cuánto tiempo tarda en encontrar la solución.
- Complejidad en espacio: Cuánta memoria se necesita para encontrar la solución.
- Optimalidad : Es la estrategia para encontrar la mejor solución cuando existen diferentes soluciones.

Por ejemplo, la búsqueda por medio de algoritmos voraces, utiliza una función $h(n)$, la cual determina el costo estimado del camino más corto para ir del nodo actual al nodo meta, con lo que se reduce el tiempo de búsqueda de la solución. Desafortunadamente esta búsqueda no cumple con la optimalidad y completitud.

El algoritmo que usa la búsqueda de costo uniforme utiliza una función que minimiza el costo del camino del nodo inicial al nodo actual $g(n)$, éste es óptimo y completo, pero puede ser ineficiente en tiempo.

Existe un algoritmo que combina las dos estrategias anteriores y toma las ventajas de ambos, simplemente sumando las dos funciones de evaluación que se mencionaron anteriormente.

$$f(n) = g(n) + h(n)$$

donde:

- $g(n)$ = es el costo del camino del nodo inicial a nodo actual n.
- $h(n)$ = es el costo estimado del camino más barato (más corto) del nodo actual n a un nodo objetivo.

- $f(n)$ = es el costo estimado de la solución más barata a través de n.

  Las pruebas de optimalidad y completitud de este algoritmo se presentan en [Norvig95]. La mejor solución se puede obtener aplicando un simple restricción a la función $h(n)$. Esta restricción es que $h(n)$ nunca sobre estime el costo para alcanzar el objetivo, si se cumple esto para $h(n)$, entonces la llamamos heurística admisible. Las heurísticas admisibles son por naturaleza optimistas, porque piensan que el costo de la solución del problema es menor que en la que están actualmente. Este optimismo es transferido a la función $f(n)$ ya que $h(n)$ es admisible y $f(n)$ nunca sobre estima el costo actual de la mejor solución a través de n. El algoritmo **el primero mejor**, usando $f(n)$ como función de evaluación y una función $h(n)$ admisible, es conocido como algoritmo de búsqueda A*

3.3.1 Aplicación del algoritmo A* en scheduling

Este algoritmo es un método de búsqueda guiada, si se utiliza una función guía minorante (admisible), entonces se puede garantizar que obtiene el scheduler óptimo, aunque en el peor de los casos puede expander todo el grafo de búsqueda, que en este caso es el grafo diligente de la RP. La función guía está basada en el tiempo de ciclo del p-semiflujo más lento de la red, es minorante porque el tiempo de ciclo de la red siempre es igual o mayor que el tiempo de ciclo del p-semiflujo más lento.

Como se dijo anteriormente la función guía esta formada de dos partes, $f(n) = g(n) + h(n)$. Calcular $g(n)$ es sencillo, porque cuenta el tiempo que cuesta ir del nodo inicial a nodo actual n. $h(n)$ cuenta el tiempo que cuesta ir del nodo actual a un nodo meta y es difícil de calcular, ya que como el problema es de naturaleza altamente combinatoria, pueden existir muchas formas de ir del nodo actual al nodo meta, y analizar todas las posibles trayectorias es difícil. Para resolver este inconveniente se propone utilizar las funciones $g(n)$ y $h(n)$ calculadas como sigue:

Sea una red de Petri $\mathcal{N} =< P, T, F >$ y M_o su marcado inicial, $\sigma = t_i, t_j, ...$ una secuencia disparable en $< \mathcal{N}, M_o >$ tal que $M_o \xrightarrow{\sigma} M_o$ y σ cumple con los ratios de visita impuestos por un vector v_k dado o calculado.

Sea M_x el marcado actual y $\sigma = \sigma_a, \sigma_b$ tal que $M_o \xrightarrow{\sigma_a} M_x$ y $M_x \xrightarrow{\sigma_b} M_o$; la función $g(n)$ se calcula como:

$$g(n) = \tau(\sigma_a)$$

Es decir, el tiempo transcurrido del estado inicial al estado donde se han disparado todas las transiciones de σ_a.

Ahora, para calcular $h(n)$ es más complicado porque no se conoce σ_b. En este caso se propone una función que se basa en la búsqueda del p-semiflujo más lento. Esta función es minorante, ya que el valor que obtiene como estimación es menor que el valor real y se obtiene con la siguiente ecuación:

$$\begin{array}{l} \tau(< N, M_o >) \geq \pi_k = \max \mathbf{Y}^T \cdot C^-[\, d(t_i) \cdot v_k(t_i)] \\ \text{sujeto a : } \mathbf{Y}^T \cdot C = \mathbf{0} \\ \mathbf{Y}^T \;\; M_o = 1 \\ \text{Donde : } D_k[i,i] = d(t_i) \cdot v_k(t_i), \text{ y } D_k[i,j] = 0, \forall i \neq j \end{array} \tag{3.1}$$

El uso de esta ecuación proporciona una cota inferior del tiempo de ejecución del sistema $< \mathcal{N}, Mo >$, ya que obtiene tiempos de ejecución de secuencias cíclicas (que regresan al marcado inicial). Por lo cual, se tiene que modificar para que sólo proporcione la estimación del tiempo de ejecución del estado actual al estado meta. Esta modificación consiste en agregar una transición a la red $\mathcal{N}$, de forma que se introduzca un camino alternativo σ_y tal que $M_o \stackrel{\sigma_y}{\rightarrow} M_x$ y $M_x \stackrel{\sigma_b}{\rightarrow} M_o$ con $d(t_i) = 0 \; \forall t_i \in \sigma_y$ y M_x como estado actual. σ_y es un camino creado artificialmente entre M_o y M_x a través de una única transición t_{m+1} de duración cero $(d(t_{m+1}) = 0)$ que se agrega a la red.

Los arcos de entrada de t_{m+1} se calculan de la siguiente forma :

$$Pre(p_i, t_{m+1}) = M_o(p_i), \forall p_i \in P$$

Los arcos de salida se calculan de la siguiente forma :

$$Post\ (p_i, t_{m+1}) = M_x(p_i), \forall p_i \in P$$

Es decir, el disparo de t_{m+1} pasa de M_0 a M_x en tiempo cero. Enseguida se presenta una propiedad que justifica el por qué la transición añadida no afecta la trayectoria óptima en el sistema.

Proposición 3.1 (Ramírez 93t) *Sea $\mathcal{N} =< P, T, F >$ una red de Petri y M_o su marcado inicial, $\sigma^* = \sigma_1 \sigma_2$ una ejecución de tiempo óptimo con $M_o \stackrel{\sigma_1}{\rightarrow} M_x \stackrel{\sigma_2}{\rightarrow} M_o$. Si al sistema se la añade una nueva transición tal que $M_o \stackrel{t_{m+1}}{\rightarrow} M_x$ y $d(t_{m+1}) = 0$ entonces $\sigma' = t_{m+1}, \sigma_2$ es una ejecución óptima en $\mathcal{N}' =< P, T \cup t_{m+1}, F' >$*

La prueba se hace aplicando el principio de programación dinámica. De esta forma se puede utilizar la ecuación 3.1 para estimar el tiempo de la secuencia que lleva a la red del marcado actual M_x al marcado inicial M_o, cambiando v_k por el vector característico de la nueva ejecución, la cual se forma con $\vec{\sigma}$ y un elemento más, igual a 1, para representar el hecho de que la transición añadida se dispara. Más adealnte se presenta un ejemplo para ilustrar este método de búsqueda del schedule óptimo, pero primero se presenta el algoritmo a seguir.

3.3.2 Algoritmo : Búsqueda en el GD del schedule óptimo

1. Teniendo la RP, construimos sólo el nodo inicial de su GD (que es el marcado inicial), se calcula $f(n)$, $g(n)$, $h(n)$ y el tiempo de ciclo para este nodo.

2. Para el nodo actual en el GD se expanden todos sus vecinos. Para cada nodo expandido se modifica la RP añadiendo una transición t_m+1 de duración =0 que permita alcanzar a dicho nodo expandido desde el marcado inicial Mo y se calcula (actualiza) $f(n)$.

3. Se elige el nodo de menor valor de $f(n)$ como nodo actual, o en el caso de haber varios, se elige el de mayor paralelismo o profundidad. Si después de esto hay empate se elige cualquiera de ellos.

4. Si ya estamos en el nodo meta, terminamos, de lo contrario regresamos al paso 2.

Algoritmo 3.1 *para encontrar el schedule:*

Entrada:

RP /* Red de Petri del sistema

Salida :

Schedule /* el schedule que se obtiene para el sistema

Inicialización:

$Mx \leftarrow \mathrm{E}^0 = (M_0, 0)$ /* Nodo actual= Nodo inicial del GD.

Schedule $\leftarrow$ { }

Tiempo $\leftarrow$ 0 /* para indicar el tiempo de M_0 a Mx.

TiempoCiclo $\leftarrow$ { } /* para indicar el tiempo de Mx a M_0.

$\mathrm{C}|_{[t_m+1]}= 0$ /* agregar a la matriz C la columna de ceros de $t_m + 1$

Realizar

Ms $\leftarrow$ conjunto de sucesores de Mx en el GD calculado para la RP.

Mientras Ms $\neq \emptyset$

tomar un elemento $m_{s_i} \in$ Ms

agregar a la RP $t_m + 1$ que alcance m_{s_i} desde M_0 en tiempo 0.

$$\pi = \max \mathrm{Y}^T[C^-[t_m + 1]\,] \quad \left[0(\vec{\sigma_b}) \cdot d(t_i)\right]$$

sujeto a : $\mathrm{Y}^T \; [C[t_m + 1]] = 0$

$\mathrm{Y}^T \; M_o = 1$

$\mathrm{Y} \geq 0$

TiempoCiclo $\leftarrow$ TiempoCiclo $\cup \pi_i$

Ms $\leftarrow$ Ms - m_{s_i}

Fin de Mientras

$\pi_j \leftarrow \min_{x_i \in Tiempo\,Ciclo}$ {xi+tiempo +peso del arco[Mi,m_{s_i}]}

schedule $\leftarrow$ schedule $\oplus(t_j, tiempo), M_i[t_j > m_{s_j}]$

tiempo $\leftarrow$tiempo +peso del arco$[Mi,m_{s_i}]$

$Mi \leftarrow m_{s_i}$

Fin realizar (hasta $M_i = M_0$)

Siguiendo el algoritmo anterior se mostrará como se aplica el algoritmo A* en scheduling, para ello, mostraremos el siguiente ejemplo:

Ejemplo 3.1 *Dada la RP de la figura 3.3 con su respectivo grafo diligente (figura 3.4), encotrar el schedule óptimo de esa red, es decir encontrar una secuencia de disparo de transiciones que permita llegar de nueva cuenta al marcado inicial y, que se ejecute en el menor tiempo posible.*

Para resolver el problema aplicamos el algoritmo A* al grafo diligente de la RP, cada nodo representa un posible estado de la red, los arcos representan las transiciones que se pueden disparar, a cada nodo del GD le corresponde una RP modificada (apartir de la red original) de tal forma que es posible calcular el tiempo que falta desde ese nodo actual al nodo meta, es decir, $h(n)$, que se calcula con la ecuación 3.1 sobre la RP del nodo actual.

El primer paso se ilustra en la figura 3.5-a, nuestro subgrafo diligente expandido sólo contiene el nodo inicial, por lo cual su RP es la original (figura 3.3), y los valores de las funciones y de π (tiempo de ciclo) se muestran en la parte superior de la figura 3.5-a. En la figura 3.5-b se muestra el siguiente paso, como el nodo inicial sólo tiene un vecino, ese nodo es expandido, se genera su RP correspondiente y se calculan los valores de las funciones y de π. Como se puede ver, se añade una transición de tiempo cero, que permite calcular el tiempo de ciclo del resto de la red.

En la figura 3.5-c se muestra el siguiente paso, el nodo actual es $(P_2, P_3, P_6; 0)$, este tiene tres posibles alternativas (vecinos); disparar t_2; t_3; o t_2 y t_3 al mismo tiempo, por lo cual, se expanden esos tres nodos. En este caso, los nodos del GD a expander son: $(P_3, P_4, P_6;$ $0)$, $(P_4, P_6, R(t_3) = 1)$, $(P_2, P_3, P_6); 0)$. (En este ejemplo el nodo $(P_4, P_6, R(t_3) = 1)$ lo representamos como $(P_4, P_6, P_3'; 1)$ indicando que es un marcado intermedio, con tiempo residual igual a una unidad de tiempo). Para elegir el correcto, se genera para cada uno de ellos su RP modificada y se calcula para cada nodo $f(n)$, en este caso los nodos $(P_4, P_6, R(t_3) = 1)$ y $(P_2, P_5, P_6; 0)$ resultan con el mismo valor de $f(n)$, pero se elige $(P_4, P_6, R(t_3) = 1)$ por ser el de mayor paralelismo, en la figura 3.5-c se muestra su correspondiente RP modificada, en este caso llegamos a un marcado intermedio, ya que t_3 tiene una duración de 2 unidades de tiempo, pero en ese momento sólo lleva una unidad de tiempo y le falta una unidad de tiempo para terminar, esto se representa con la transición y lugar sombreados, en este marcado intermedio $(P_4, P_6, P_3'; 1)$ tenemos 2 alternativas, iniciar el disparo de t_2 o dejar que t_3 termine su ejecución, para elegir la correcta se expanden esos dos nodos, como se muestra en la figura 3.6-a, se generan sus RP correspondientes, y se calculan sus $f(h)$. En este caso se elige dejar que t_3 termine su ejecución y llegar al estado $(P_4, P_5, P_6; 0)$, su red modificada se muestra en la figura 3.6-d, haciendo lo descrito anteriormente, se elige $(P_4, P_6, P_8; 0)$, y su RP modificada se presenta en la figura 3.6-e, del estado $(P_4, P_6, P_8; 0)$ se expanden sus

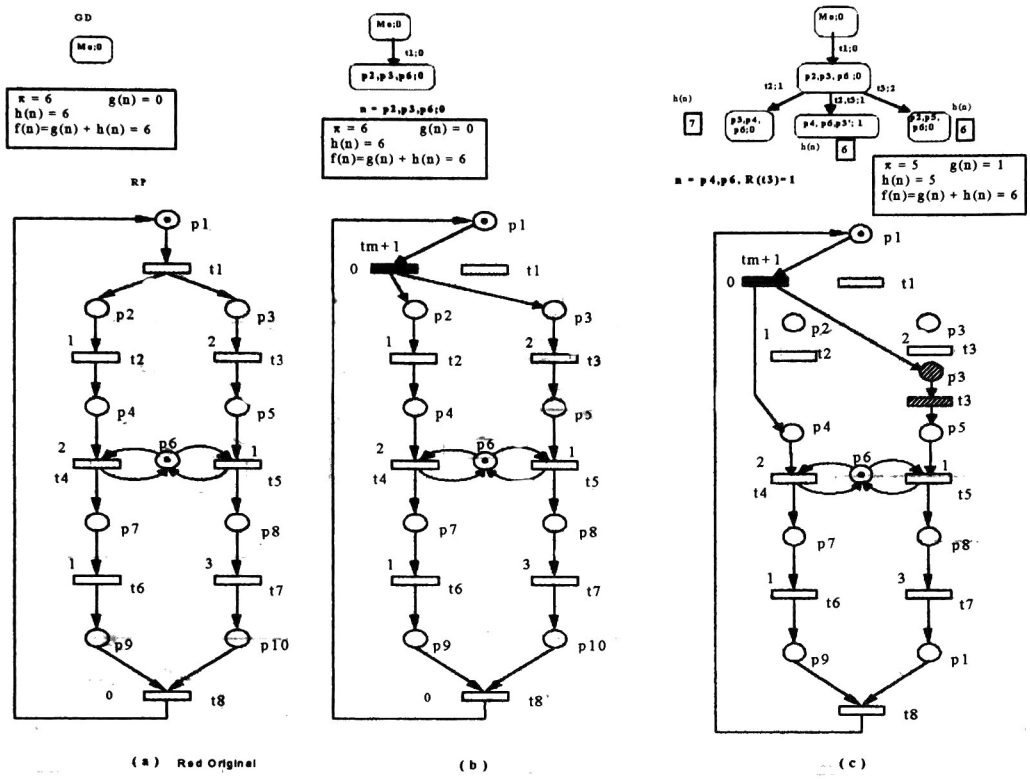


Figura 3.5: Ejemplo de como la RP (red original) se va modificando para obtener el schedule óptimo (parte 1).

vecinos, como se muestra en la figura 3.6-c, se elige $(P_6, P_7, R(t_7) = 1\ )$ y su RP modificada se presenta en la figura 3.6-f.

Si estamos en $(P_6, P_7, R(t_7) = 1\ )$, se elige disparar t_6 con una unidad de tiempo, que es también lo que le falta a t_7, y entonces llegamos al nodo meta $(P_6, P_9, P_{10}\ ; 0)$ como se muestra en la figura 3.7, terminando así la búsqueda y obteniendo la secuncia óptima $t_1, t_2 \| t_3, t_5, t_4 \| t_7, t_6, t_8$ sin necesidad de expander todo el GD. El schedule óptimo es el siguiente $S = (t_1, 0), (t_2, 0), (t_3, 0), (t_5, 2), (t_4, 3), (t_7, 3), (t_6, 5), (t_8, 6)$, este schedule nos permite alcanzar el estado meta en 6 unidades de tiempo.

Como se puede apreciar este método consiste en ir construyendo el GD conforme va evolucionando la RP. Además de acuerdo a la evolución, la RP se va modificando en cada paso, al final el GD obtenido, es un sub GD (porción del GD) que contiene todos los nodos que se encuentran en la trayectoria óptima y algunos otros que parecían prometedores del camino óptimo. A cada nodo se le asocia una modificación a la RP y para cada una se calculó el tiempo de ciclo óptimo.

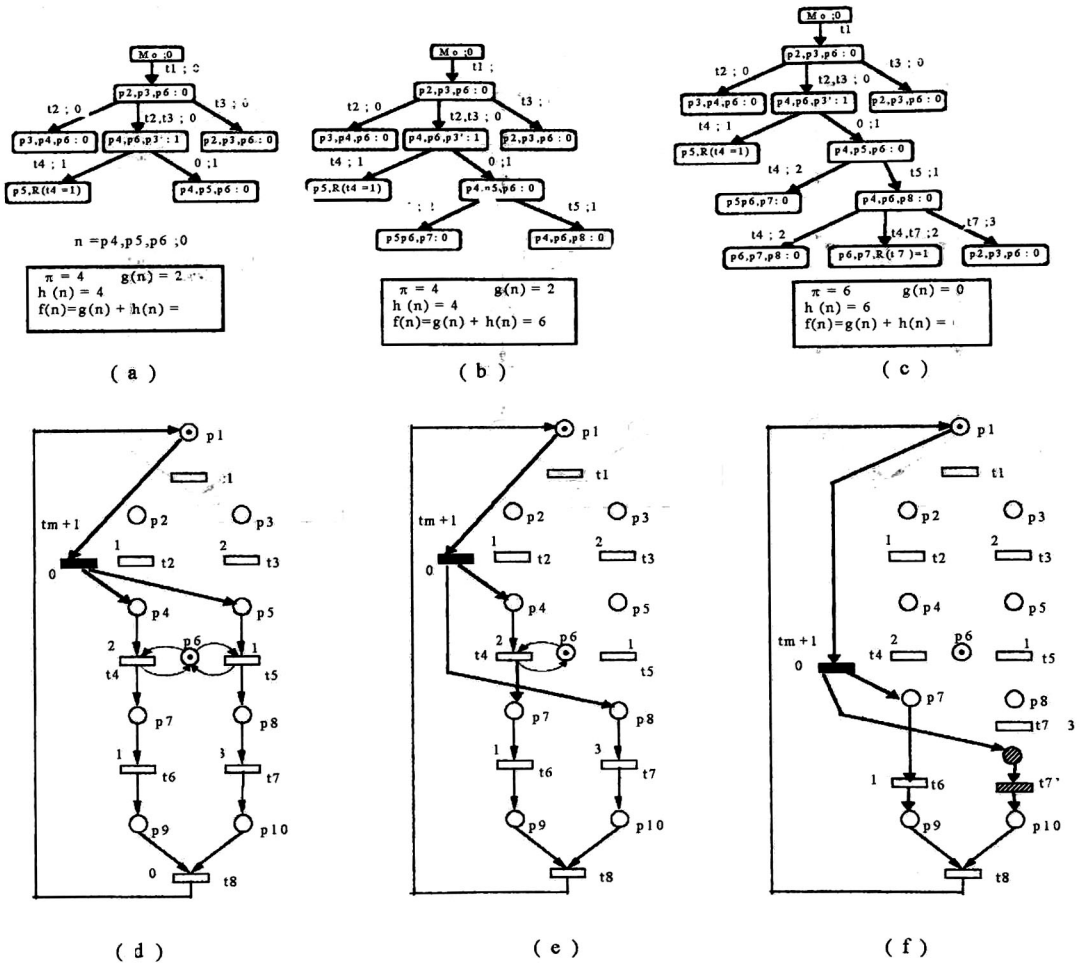


Figura 3.6: Continuación del ejemplo de búsqueda del schedule óptimo (parte 2).

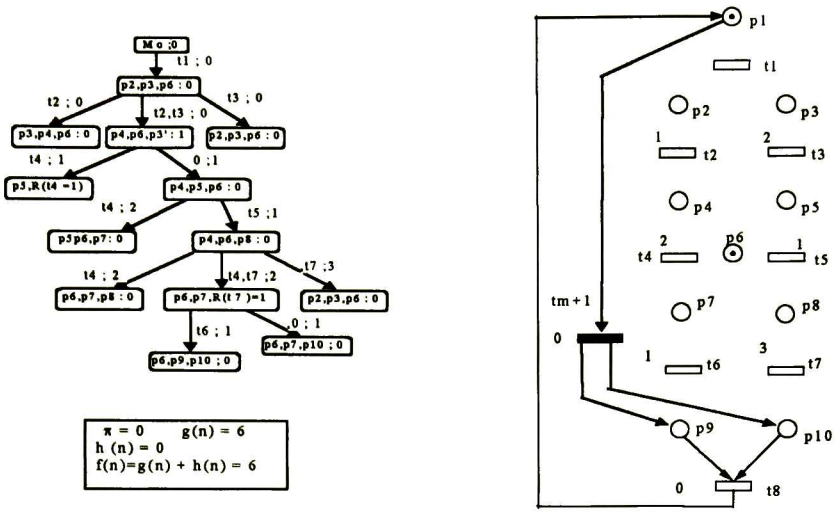


Figura 3.7: Ultima parte del ejemplo de búsqueda del schedule óptimo (parte 3).

En general, obtener la solución de esta forma es más rápida que la mencionada anteriormente, pero en el peor de los casos también puede llegar a expandir todo el GD lo que lo haría ineficiente para RP grandes.

En el siguiente capítulo se combina el uso de esta técnica con los conceptos de optimalidad local para encontrar el schedule óptimo en menor tiempo.

Capítulo 4

Optimalidad Local y Pares Inevitables

Resumen: En este capítulo se presentan los principales resultados de la optimalidad local, y algunos criterios para determinar si una red de Petri posee la propiedad de optimalidad local. Se obtiene un procedimiento para encontrar un schedule óptimo para las redes que exhiben la propiedad de optimalidad local y se mencionan las ventajas que se obtienen al optimizar los sistemas utilizando esta propiedad. Además se introducen otros conceptos como el de pares inevitables, y algunas proposiciones que utilizan este concepto para encontrar un schedule óptimo para las redes con optimalidad local.

4.1 Introducción

Existen problemas de cómputo de complejidad mayor que la polinomial, en los cuales se puede aplicar la filosofía "divide y vencerás" porque se pueden resolver en mucho menor tiempo dividiendo el problema en partes que tratar de resolverlo en forma global (todo completo). Esto se debe a que al dividir el sistema en subsistemas, estos últimos pueden ser optimizados en poco tiempo. Por ejemplo, si $\tau(X)$ es el tiempo en que se puede resolver un problema de un sistema en forma global, y $\tau(x_i)$ es el tiempo en que se puede resolver el problema del subsistema x_i; con $\cup_{i=0}^{n} x_i = X$ y $x_i \cap x_j = 0 \quad \forall x_i \neq x_j$.

Entonces

$$\tau(X) \geq \sum \tau(x_i)$$

Es decir

$$k^{|X|} \geq \sum_{i=0}^{n} k^{|x_i|}$$

ahorrándose una considerable cantidad de tiempo de cómputo.

Desafortunadamente, en scheduling, no todos los sistemas pueden ser optimizados de esta forma, en las siguientes secciones se explican los criterios que se deben tomar en cuenta para aplicar esta idea a scheduling.

4.2 Conceptos de optimalidad local y sus principales resultados

Todos los sistemas modelados en RP se pueden dividir en subsistemas, además pueden existir varias formas de dividir el sistema, pero no cualquier partición (subdivisión), ni cualquier sistema cumple con las características para ser optimizado aplicando esta idea. Enseguida se presentan algunas definiciones y conceptos que son importantes para determinar si se puede realizar la optimización de un sistema dividiéndolo en subsistemas.

Definición 4.1 (Ramírez 93t) *Si un sistema puede ser dividido en subsistemas y el desempeño óptimo de los subsistemas garantiza el desempeño óptimo de todo el sistema, entonces el sistema posee la propiedad de* ***Optimalidad Local*** *(OL) con respecto a esa división.*

Definición 4.2 *Subsistema Lugar-Lugar (SLL) [Ramírez 93t]: Sea* $\sum = (\mathcal{N}, M_0)$ *un sistema. Una subred* $\mathcal{N}' = (P', T', PRE', POST')$ *de* $\mathcal{N}$ *(es decir,* $P' \subseteq P$, $T' \subseteq T$, $PRE' = PRE|_{P' \times T'}$, $POST' = POST|_{P' \times T'}$*) con un* $M_0'|_{P'}$ *(la proyección del marcado inicial de* $\mathcal{N}$ *sobre* $\mathcal{N}'$*) es un SLL si y sólo si:*

- ${}^\bullet t$ y $t^\bullet \in P'; \forall t \subset T'$
- $\exists! p_{entrada} \in P'$ tal que ${}^\bullet p_{entrada} \subseteq T \setminus T'$ y ${}^\bullet p_{entrada} \neq \emptyset$.

- $\exists! p_{salida} \in P'$ tal que $p^{\bullet}_{salida} \subseteq T \setminus T'$ y $p^{\bullet}_{salida} \neq \emptyset$.
- Si $x_i \in P' \cup T'$ entonces existe un camino en $PRE' \cup POST'$ dirigido de $p_{entrada}$ a x_i y otro camino de x_i a p_{salida}.
- $\forall t \in T', \exists p \in {}^{\bullet}t,\ Y \geq 0$ tal que $Y^T C = 0, Y(p) = Y(p_{entrada}) = Y(p_{salida}) > 0$.

En la figura 4.1-a se muestra un esquema general de una RPT que contiene un SLL.

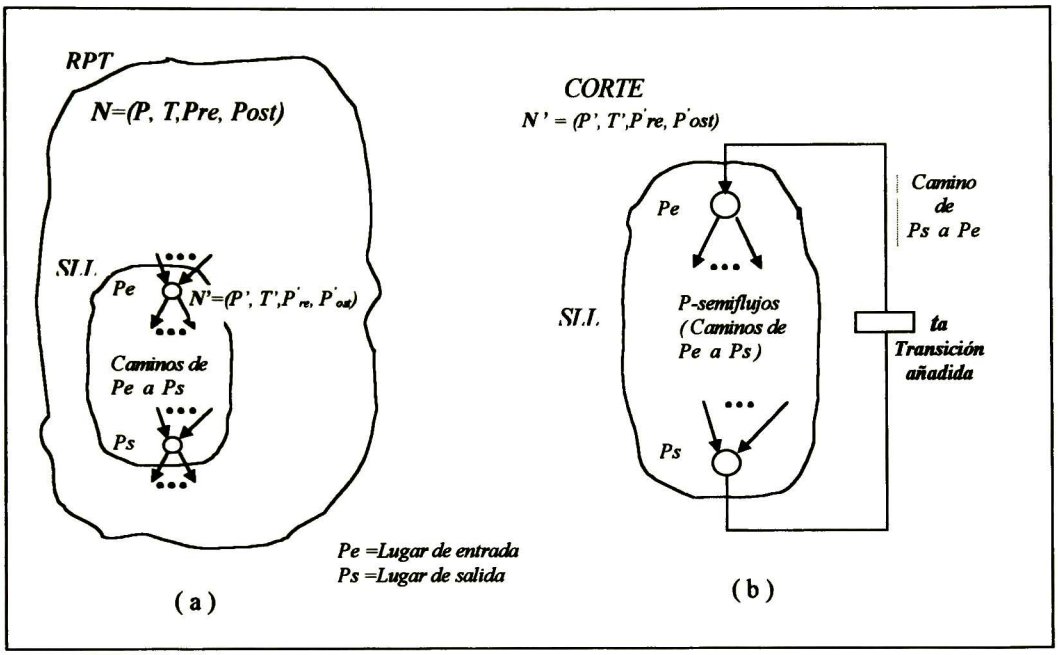


Figura 4.1: Esquema general de una RPT que contiene un SLL.

En general los subsistemas son denotados SX_iX_j, donde X_i es el elemento de entrada al subsistema y X_j es el elemento de salida. Cuando X_i y/o X_j son transiciones, entonces se pueden expander a una transición-lugar-transición como se muestra en la figura 4.2.

4.2.1 Análisis de Redes con la propiedad de optimalidad local

Para analizar los subsistemas, a estos se agrega una transición que crea un camino de p_{salida} a $p_{entrada}$, y una vez que han sido removidas las marcas del subsistema (solo para el análisis), se deja sólo una marca en el lugar de entrada, al hacer esto, el subsistema se convierte en una subred fuertemente conexa y entonces se pueden realizar ciertos tipos de cálculos, entre ellos una cota de la velocidad de ejecución del subsistema y el schedule (sub) óptimo del subsistema. Además si el subsistema resulta ser una subclase de RP de la que ya se conoce cómo obtener su schedule óptimo, como GM, ME o cierta subclase de redes de libre elección, entonces se podrá obtener un schedule óptimo para dicho subsistema reduciendo en cierta

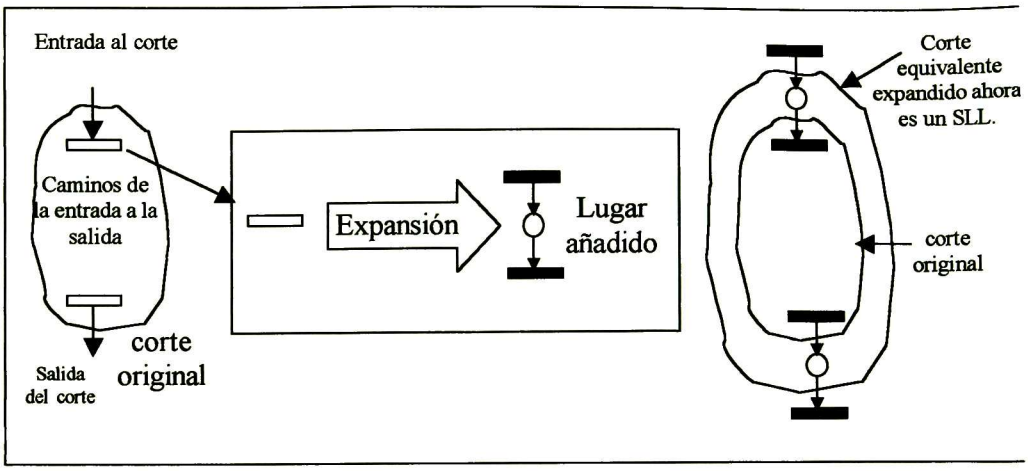


Figura 4.2: Conversión de un sistema con transición de entrada y salida a un SLL.

forma de tiempo para obtener el schedule óptimo de la red global, de lo contrario se puede utilizar el algoritmo del capítulo anterior a cualquier tipo de corte. En la figura 4.1-b se muestra un esquema general de un SLL al que se le agrega una transicíon t_a para formar el camino del lugar de salida al lugar de entrada.

Por el tipo de estructura de los subsistemas se garantiza que si se marca el lugar de entrada del subsistema, entonces inevitablemente se marcará su lugar de salida. Es decir, en el grafo de alcanzabilidad todos los posibles caminos que llegan al estado cuando el lugar de salida del subsistema está marcado son los que parten del estado cuando el lugar de entrada está marcado. Esto quiere decir que se puede optimizar el camino entre estos dos estados localmente, sin analizar el resto de los estados.

Cuando una red marcada $\sum =< \mathcal{N}, M >$ tiene la propiedad de Optimalidad Local, un schedule óptimo puede calcularse fácilmente con el siguiente procedimiento[Ramírez 93t]:

1. Calcular un schedule óptimo de cada subsistema C_i.

   $s_j^{i*} = s_{j_1}^{i*}\ t_a^{i*}\ \ s_{j_2}^{i*}\ t_a^{i*}\ \ldots\ldots\ s_{j_k}^{i*}\ t_a^{i*}$

   El elemento t_a^{i*} es la transición añadida al subsistema y s_j^{i*}, es una secuencia cíclica del corte.

2. Calcular el vector de ratios de visita para la red global y el subsistema. Además, calcular un schedule óptimo de la red reducida, el cual es :

   $s_r = t_a, t_b, \ldots\ldots t_{k_1}^1, \ldots\ldots t_q, \ldots\ldots, t_{r_1}^i, \ldots., t_a, t_b, \ldots\ldots t_{k_2}^1, \ldots\ldots t_q, \ldots\ldots, t_{r_2}^i, \ldots\ldots$

   donde el superíndice de las transiciones indica que son las transiciones de entrada al subsistema indicado por el propio superíndice.

3. Formar una nueva secuencia s_g a partir de la secuencia s_r, simplemente insertando después de $t_{y_k}^i$ el schedule óptimo $s_{j_k}^{i*}$ correspondiente al subsistema C_i.

   $s_g = t_a, t_b, \ldots\ldots t_{k_1}^1, s_{j_1}^{1*}, \ldots\ldots t_q, \ldots, t_{r_1}^i, s_{j_1}^{i*}\ldots., t_a, t_b, \ldots\ldots t_{k_2}^1, s_{j_2}^{1*}, \ldots\ldots t_q, \ldots, t_{r_2}^i, s_{j_2}^{i*}, \ldots\ldots$

Si se obtiene una máquina de estados cuando cada subsistema es reducido a un lugar entonces, las ratios de visita de los subsistemas, de la red reducida y de la red global están relacionados de la siguiente forma [Ramírez 93t]:

La red reducida $\mathcal{N}_r$ es una máquina de estados y su vector de ratios de visita es V_k^r

En un subsistema C_i se tiene un vector de ratios de visita V_k^i, en este vector el elemento correspondiente a la transición añadida t_a, es q_i. La ratios de visita de la red global se pueden calcular a partir de estos datos, como se explica a continuación.

Se define $k = i\prod q_i$, entonces la ratio de visitas para una transición t_g que pertenece a la red reducida se calcula como :

$$t_g = k \cdot V_r(g)$$

y la ratio de visita para una transición t_h^i en el subsistema C_i se calcula como :

$$V(t_h^i) = k(V_r(i)/q_i)$$

donde $V_r(i)$ es la ratio de visita de la transición de entrada al lugar que representa al subsistema C_i en la red reducida. Este vector se multiplica por un número para obtener un vector de enteros de mínima norma.

En la matriz de incidencia se puede observar que los subsistemas son bloques aislados de información. De no ser así, el subsistema tendría más de un lugar de entrada o de salida. La única comunicación con el resto de la red es por medio de dos columnas, con valor de 1 o -1 o la red reducida no sería una máquina de estados y por lo tanto los ratios de visita de las transiciones del subsistema son exactamente escalables por el valor de la ratio de visita de la transición agregada. Enseguida se presentan algunas propiedades que nos ayudan a determinar si la RP posee la propiedad de optimalidad local y a calcular el schedule óptimo.

Proposición 4.1 (Ramírez 93t) *Si se obtiene una máquina de estados sana* $\sum =< \mathcal{N}, Mo >$ *cuando cada subsistema es reducido a un lugar entonces* $\sum$ *tiene la propiedad de Optimalidad Local.*

Si la red reducida no es una ME entonces, en general, el tiempo de ejecución no puede ser calculado como la suma de tiempos individuales, ya que existirían operaciones de máximo en el cálculo del tiempo; si la red no es sana entonces existen subsistemas trabajando en paralelo y el tiempo de ejecución tampoco podría ser calculado como la suma de los tiempos de los subsistemas.

El hecho de que los lugares de entrada y de salida están incluidos en todos los p-semiflujos del subsistema, permite asegurar que la actividad del subsistema empieza cuando se marca el lugar de entrada y termina cuando se marca el lugar de salida; es decir no hay actividad remanente en el subsistema que pueda realizarse en paralelo con otro subsistema.

Otro tipo de redes que cumplen con la propiedad de optimalidad local son los siguientes.

Proposición 4.2 (Ramírez 93t) *Si una red marcada viva y sana* $\sum =< \mathcal{N}, M >$ *tiene solamente subsistemas que tienen un único t-semiflujo con todos sus elementos iguales a uno*

y se cumple que, cuando reducidos los subsistemas de la red resulta en un grafo marcado, entonces la red marcada $\sum =< \mathcal{N}, M >$ *posee la propiedad de Optimalidad Local.*

Si no se cumple esta propiedad, es decir, si existe algún subsistema con más de un t-semiflujo, o en el cual no todos los elementos del vector del t-semiflujo son unos, entonces el subsistema no puede ser reducido de la forma anterior, ya que el tiempo asociado a la transición que representa al corte sería variable; Es decir, podría variar, dependiendo del t-semiflujo que se está ejecutando.

4.3 Optimalidad local, GD, A* y Pares inevitables

Se puede reducir considerablemente el tiempo necesario para obtener el schedule óptimo de una RP combinando el método del capítulo anterior con el uso de los conceptos de optimalidad local y pares inevitables, pero primero es necesario introducir algunas definiciones y conceptos.

Definición 4.3 *El reverso del grafo diligente* (GD^R) *se define como* $RGD = (V^R, A^R)$ *donde* $V^R = V$ *el conjunto de vértices del grafo diligente y* $A^R = \{ a_{ji} \mid a_{ij} \in A$ *(el conjunto de arcos del grafo diligente)*$\}$.

En la figura 4.3 se muestra un grafo diligente con sólo cuatro estados, y a su derecha se muestra su reverso. Como se puede apreciar la única diferencia entre ellos es el sentido de los arcos.

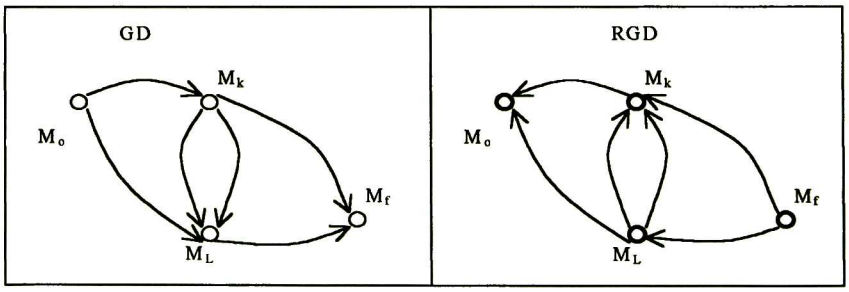


Figura 4.3: Esquema general de un GD y su reverso.

Definición 4.4 *Sea* v_i, v_j *vértices del grafo diligente* $GD(V, A)$*, entonces el lenguaje de* v_i, a v_j (L_{v_i,v_j}) *se define como* $L_{v_i,v_j} = \{\sigma \mid v_i \xrightarrow{\sigma} v_j, v_i, v_j \in V, \sigma = [a_i, a_l]...[a_q, a_r]...[a_k, a_j]; [a_q, a_r] \in A\}$ *en el reverso del grafo diligente el lenguaje* $L^R_{v_iv_j}$ *se define similarmente* $L^R_{v_iv_j} = \{\sigma | v_j \xrightarrow{\sigma} v_i, v_i, v_j \in V^R, \sigma = [a_j, a_l]...[a_q, a_r]...[a_k, a_i]; [a_q, a_r] \in A^R\}$*, pero en este caso se debe ir de* v_j *a* v_i.

Definición 4.5 *Un alfabeto* $\sum$ *es un conjunto finito de elementos llamados símbolos.*

Los símbolos pueden ser letras, números o cualquier etiqueta que permita distinguir entre sí los elementos del alfabeto. En este trabajo los símbolos del alfabeto están relacionados con el nombre de las transiciones.

Definición 4.6 *Sea $GD(N, M_0)$ el grafo diligente de RP= $< \mathcal{N}, M_0 >$, y RGD el reverso de su grafo. $[M_K, M_L]$ forman un par inevitable (PI) en GD si:*

1. $(\overline{L_{M_k,M_L}} - L_{M_k,M_L}) \sum \cap \overline{L_{M_k,M_f}} \subseteq \overline{L_{M_k,M_L}}$, *donde M_f es un marcado final y*
2. $(\overline{L^R_{M_L,M_k}} - L^R_{M_L,M_k}) \sum \cap \overline{L^R_{M_L,M_0}} \subseteq \overline{L^R_{M_L,M_k}}$, *donde M_0 es el marcado inicial.*

Para ejemplificar la forma en que se determina si dos marcados forman un par inevitable, tomemos como referencia la figura 4.4 donde se muestra una RP y correspondiente grafo diligente. En dicho GD se encierran con lineas punteadas dos PI, [P_{10},P_{13}] y [P_2,P_8]. De la RP podemos obtener:

$\sum = \{a, b, c, d, e, f, g, h, i, j, k, l, n, m\}$

Para este ejemplo tomaremos como referencia el PI formado por [P_2,P_8] de donde podemos obtener:

Lp_2p_8 = {bcdedg, bcdeg, bdceg} Lp_2p_9 = {bcdedgh, bcdegh, bdcegh}

$\overline{Lp_2p_8}$ = {∈, b, bc,bd, bce,bcd,bdc, bced, bcde, bdce, bcedg, bcedg, bcdeg, bdceg}

$\overline{Lp_2p_9}$ = { $\overline{Lp_2p_8}$∪ bcedgh, bcdegh, bdcegh}

$L^Rp_8p_2$ = {gdecb, gedcb, gecdb}

$\overline{L^Rp_8p_2}$ = {∈, g, gd, ge, gde, ged, gec, gdec, gedc, gecd, gdecb, gedcb, gecdb}

$L^Rp_8p_1$ = {gdecba, gedcba, gecdba}

$\overline{L^Rp_8p_1}$ = {$\overline{L^Rp_8p_2}$∪ gdecba, gedcba, gecdba}

Obteniendo estos datos se puede verificar la condición 1.

Condición 1 : $(\overline{Lp_2,p_8} - Lp_2, p_8) \sum \cap \overline{Lp_2, p_9} \subseteq \overline{Lp_2,p_8}$

{ b j, bcd h, bdc l, ...} $\cap \overline{Lp_2, p_9} = \in$ y $\in \subseteq \overline{Lp_2,p_8}$ si cumple condición 1.

La condición 2 se verifica de forma similar a la 1, por lo que también sólo presentaremos 3 casos (los restantes también cumplen la condición 2).

$(\overline{L^Rp_8, p_2} - L^Rp_8,p_2) \sum \cap \overline{L^Rp_8, p_1} \subseteq \overline{L^Rp_8, p_2}$

{ gd i, gedc k, gecd a, ... } $\cap \overline{L^Rp_8, p_2} = \in$ y $\in \subseteq \overline{L^Rp_8, p_2}$ si cumple condición 2.

Como los marcados [P_2,P_8] cumplen las dos condiciones, entonces forman un PI.

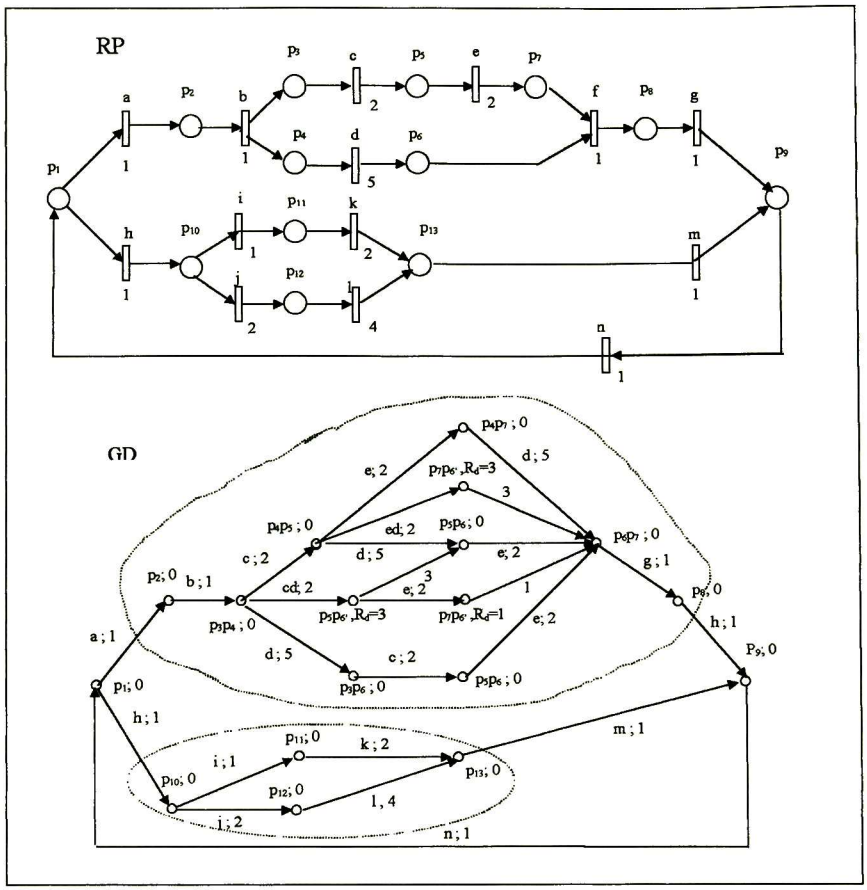


Figura 4.4: RP y su GD con dos Pares Inevitables

Definición 4.7 *Sea RP= (N, M_0) una red de Petri, y $GD(N, M_0)$ el grafo diligente de RP, para construir un grafo diligente reducido (GD_R) a partir de GD se tienen que realizar los siguientes pasos :*

1. Para cada par inevitable $P_i = [M_K^i, M_L^i]$ sustituir todas las trayectorias entre M_K^i y M_L^i por una única trayectoria, que sea de menor tiempo (σ_i^*)
2. Eliminar los nodos internos de P_i que no están contenidos en σ_i^*. Para encontrar la trayectoria óptima (σ_i^*) se emplea el algoritmo A* que se explico en el capítulo anterior.
3. El resto de las trayectorias de GD deben aparecer en GD_R

En la parte izquierda de la figura 4.5 se puede apreciar un esquema general de un GD, el cual contiene cuatro cortes (pares inevitables). En el centro de la figura 4.5 se puede observar que cada PI se debe sustituir por otro PI que sólo tenga una secuencia (la secuencia óptima de ese corte). Al hacer esta sustitución para cada PI, se obtiene otro GD que denominamos

GD_R. Este GD_R tiene menos secuencias que el GD original, por lo cual es más fácil encontrar la trayectoria óptima. Enseguida se presentan los elementos necesarios para demostrar la validez de esta metodología.

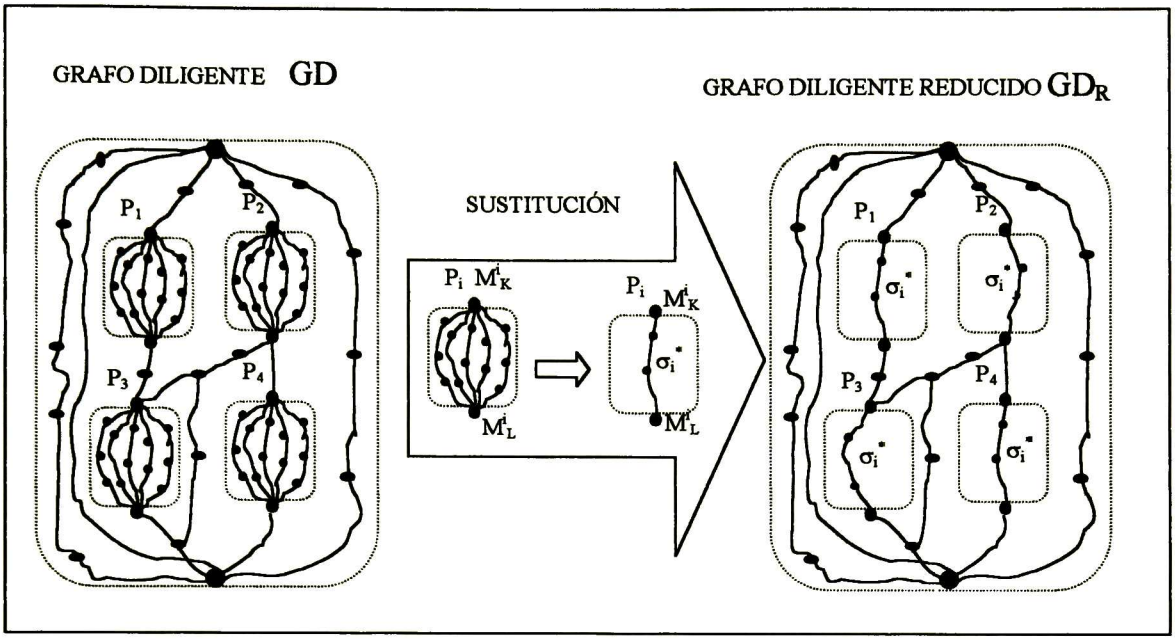


Figura 4.5: Esquema general de reducción del GD al GD reducido

Lema 4.1 *El tiempo de una trayectoria óptima en GD es igual a la de una trayectoria óptima en* GD_R

Demostración. Sea $\sigma = M_o t_i M_i t_j M_j \ldots\ldots t_f M_f$ una trayectoria óptima en GD entonces se pueden dar dos casos :

1) σ no contiene ninguna trayectoria de ningún PI en GD_R, afirmamos que σ es óptima en GD_R. Por contradicción, si σ no es óptima, entonces $\exists \sigma_{GD_R} \in GD_R \ \nexists \ \tau(\sigma_{GD_R}) < \tau(\sigma)$, pero por construcción de GD_R, $\sigma_{GD_R} \in GD$ y por tanto $\sigma \in GD$ no es óptima, lo cual está en contra de la hipótesis. Por lo que σ también es óptima en GD_R.

2) σ contiene al menos una trayectoria de un PI en GD_R, sea (M_K, M_L) uno de esos pares inevitables. Si $\sigma \in GD_R$ por un razonamiento similar al anterior se tiene que también es óptima en GD_R. Si $\sigma \notin GD_R$ entonces afirmamos que $\exists$ en GD_R una $\sigma_{GD_R} = M_o t_{GD_{Ri}} M_i \ldots\ldots M_K \ldots M_L \ldots t_{GD_{Rf}} M_f$ donde el tiempo de σ y σ_{GD_R} son iguales. Se procede a la demostración por contradicción para lo cual se tienen dos casos:

a) $\tau(\sigma_{GD_R}) < \tau(\sigma)$ entonces como $\forall \sigma_{GD_R} \in GD_R \rightarrow \sigma_{GD_R} \in GD$ por la construcción de GD_R se tiene que $\sigma \in GD$ no es óptima en GD, lo cual está en contra de la hipótesis. Por lo tanto $\tau(\sigma_{GD_R})$ no puede ser menor que (σ)

b) $\tau(\sigma_{GD_R}) > \tau(\sigma)$. Esto no es posible, ya que entre pares inevitables no entran ni salen secuencias y la secuencia que quedan entre pares inevitables en GD_R es la óptima, por tanto no puede ser que $\tau(\sigma_{GD_R}) > \tau(\sigma)$

Como en todos los casos la única posibilidad es $\tau(\sigma_{GD_R}) = \tau(\sigma)$ el lema se cumple. ■

Teorema 4 *Sea* $RP =< \mathcal{N}, M_0 >$ *una red de Petri,* $C = \{C_1, C_2, \ldots C_k\}$ *un conjunto de cortes tal que cada corte* C_i *forma un par inevitable (PI) en GD de RP, entonces RP tiene la propiedad de Optimalidad Local con respecto a ese conjunto de cortes C.*

Demostración. Como cada corte Ci genera pares inevitables, entonces se puede construir el GD_R, es decir se puede optimizar cada corte por separado (lema anterior), entonces por definición la RP posee la propiedad de Optimalidad Local con respecto a esos cortes. ■

Proposición 4.3 *Sea* $\mathcal{N} =< L, T, E, S >$ *una RdP y* $C = \{C_1, C_2, ..., C_q\}$ *un conjunto de cortes de* $\mathcal{N}$ *Si al reducir los cortes (SLL) de C, de* $\mathcal{N}$ *se obtiene una red libre elección A*, entonces* $\mathcal{N}$ *tiene la propiedad de optimalidad local.*

La prueba se realiza en las siguientes fases.

1. Si los cortes son mono t-semiflujo y al reducir la red se obtiene un grafo marcado, entonces la red tiene la propiedad de optimalidad local. La prueba de esta parte se encuentra en [Ramírez 92, Ramírez 93].

2. Una red libre elección puede ser cubierta (descompuesta) en t-componentes mínimas [Thiagaragan 84]. Cada t-componente es un grafo marcado.

3. Si al reducir los cortes (cada C_i en dos lugares y una transición t_{C_i}), queda una FCA*, entonces se puede descomponer esta última en las t-componentes $\tau_1, \tau_2, \cdots, \tau_q$. Donde en algunas t-componentes aparecerán las t_{C_i}. La unión de dichas $\tau_1, \tau_2, \cdots, \tau_q$ genera la FCA*

4. Si las t_{C_i} son expandidas en las diferentes $\tau_1, \tau_2, \cdots, \tau_q$, entonces se forman las "t-componentes" expandidas $\tau'_1, \tau'_2, \cdots, \tau'_q$. La unión de estas nuevas subredes genera todo $\mathcal{N}$. Todos los lugares y todas las transiciones están en $\tau'_1, \tau'_2, \cdots \tau'_q$, por lo que pueden faltar son arcos. Un arco que está en N y no en $\tau'_1, \tau'_2, \cdots, \tau'_q$ no puede haber sido eliminado de los cortes (por la definición de corte y por la manera en que se conecta con el resto de la red), tampoco puede ser del resto de la red, ya que el resto aparece en las $\tau_1, \tau_2, \cdots \tau_q$, por tanto la unión $\tau'_1, \tau'_2, \cdots, \tau'_q$ de genera toda $\mathcal{N}$

5. Entonces al reducir cada corte de las $\tau'_1, \tau'_2, \cdots, \tau'_q$ se forman los grafos marcados $\tau_1, \tau_2, \cdots, \tau_q$ y por tanto, cada t-componente tiene la propiedad de optimalidad local (punto 1 anterior).

6. Como el óptimo de la red reducida (FCA*) es la suma ponderada de los tiempos de ejecución de las t-componentes $\tau_1, \tau_2, \cdots, \tau_q$, entonces se mantiene que la suma de óptimos es óptima y por tanto es posible optimizar cada t-componente por separado y dentro de cada t-componente optimizar los cortes por separado, para obtener un óptimo global. Por tanto la FCA* posee la propiedad de optimalidad local.

7. Por último, una secuencia óptima de la FCA* induce una secuencia óptima en $\mathcal{N}$. Para probar esto, suponer que $\alpha_i\alpha_j \cdots \alpha_q$ es una secuencia óptima en la FCA*, esta secuencia induce una secuencia γ en $\mathcal{N}$, simplemente expandiendo cada transición t_{C_i} por la ejecución óptima del corte C_i. γ también es óptima en $\mathcal{N}$.

   Si no es así implicaría que hay otra secuencia de menor tiempo en $\mathcal{N}$, la cual no es la que induce $\alpha_i\alpha_j \cdots \alpha_q$ o implicaría que los cortes no han sido optimizados por separado y esto último no puede ser, porque se supone que los cortes han sido optimizados. Si γ no es la palabra que induce $\alpha_i\alpha_j \cdots \alpha_q$, entonces pueden ocurrir dos casos: a)que sea inducida por otra palabra β de la FCA*, lo cual implica que β es óptima y por tanto $\alpha_i\alpha_j \cdots \alpha_q$ no es óptima, lo cual no es cierto, así que β no podría ser inducida por otra palabra de la FCA*; b) que β no sea inducida por la reducción de los cortes de la FCA*, en este caso implicaría que β es una secuencia de la FCA* y hubiese sido la óptima, como y por tanto $\alpha_i\alpha_j \cdots \alpha_q$ es la óptima, entonces tampoco β puede ser la óptima.

   En conclusión si $\alpha_i\alpha_j \cdots \alpha_q$ es óptima en la FCA*, entonces induce una palabra que es óptima en $\mathcal{N}$.

Capítulo 5

Ejemplo : Obtención del schedule

Resumen: En este capítulo se presenta un ejemplo, donde se parte de una RPT que modela un SED del cual se quiere obtener el schedule óptimo. Para obtener dicho schedule se utilizan los conceptos de optimalidad local, grafo diligente, pares inevitables y la aplicación del algoritmo A* para optimizar los pares inevitables.

5.1 Introducción

En este capítulo se presenta un ejemplo, donde se parte de una RPT que modela un SED, del cual se quiere obtener el schedule óptimo. Para obtener dicho schedule se utilizan los conceptos de optimalidad local, grafo diligente, pares inevitables y la aplicación del algoritmo A* para optimizar los pares inevitables.

5.2 Ejemplo:

En la figura 5.1-a se muestra una RPT que contiene dos cortes que se encuentran encerrados con lineas punteadas. Cada uno de estos cortes se puede reducir a dos lugares y una transición como se muestra en la figura 5.1-b. Los cortes reducidos, sólo contienen una transición de tiempo igual al de la secuencia óptima que lleva del lugar de entrada al lugar de salida del corte. Además suponemos que la ratio de visita para cada transición en conflicto es 1. En la figura 5.2 se muestra el GD completo de la RPT de la figura 5.1-a. Se puede comprobar (como se explicó en el capítulo anterior) que los marcados [$p_2p_3p_8p_{19}$, $p_2p_8p_{13}p_{19}$] y [$p_2p_8p_{14}p_{19}$, $p_2p_8p_{19}p_{24}$] forman cada uno un par inevitable. En la figura 5.2 cada PI se encuentra encerrado entre lineas punteadas. Como se mencionó anteriormente, cada corte se puede analizar en forma independiente ya que la actividad dentro de estos cortes no afecta la actividad en el resto de la RPT.

En la figura 5.3 se muestra la sub RPT generada por el corte 1 y su GD. En dicho GD se muestra con una linea más gruesa la trayectoria óptima para este corte. Esta trayectoria consume 6 unidades de tiempo. Como mencionamos anteriormente este corte se puede reducir a dos lugares y una transición cuya duración sea de 6 unidades de tiempo.

El schedule óptimo para el corte 1 es:

$$Sc_1 = c\ d\ e\ g\ f\ i\ h\ j$$

viéndolo con tiempos de inicio tenemos:

$$Sc_1 = (c,0),(d.0),(e,0),(g,2),(f,3),(i,3),(h,5),(j,6).$$

En la figura 5.4 se muestra la sub RPT generada por el corte 2 y su GD. En dicho GD se muestra con una linea más gruesa la trayectoria óptima para este corte. Esta trayectoria consume 9.5 unidades de tiempo. Como mencionamos anteriormente este corte se puede reducir a dos lugares y una transición cuya duración sea de 9.5 unidades de tiempo.

El schedule óptimo para el corte 2 es:

Sc_2= lnmporqs viéndolo con tiempos de inicio tenemos:

Sc_2= (l,0),(n,0),(m,3),(p,3),(0,6),(r,6),(q,8),(s,9.5);

El schedule óptimo para la RPT reducida es:

$$S_{RPT_R} = at_{c_2}tbt_{c_1}kt_{c_2}t;$$

Sustituyendo t_{c_2} por Sc_2 y t_{c_1} por Sc_1 tenemos:

$$S_{RPT_R} = \text{a}, \underbrace{l,n,m,p,o,r,q,s}_{tc_2}\ \text{t}\ ,\text{b},\ \underbrace{c,d,e,g,f,i,h,j,}_{tc_1}\ \text{k},\ \underbrace{l,n,m,p,o,r,q,s}_{tc_2}\ ,\text{t}\ ;$$

Nota: A las transiciones de t_{c_1} *y* t_{c_2} se les tiene que sumar el tiempo transcurrido desde el marcado inicial de la RPT, hasta el marcado del lugar de entrada a su respectivo corte. En este caso a las transiciones de tc_2,la primera vez se les suma una unidad de tiempo. Y la segunda vez 19.5 unidades de tiempo. A las transiciones de t_{c_1} se les sumara 11.5 unidades de tiempo.

Tomando en cuenta lo anterior el schedule con los tiempos de de inicio queda de la siguiente manera:

S_{RPT_R} = (a,0,),$\underbrace{(l,1),(n,1),(m,4),(p,4),(o,7),(r,7),(q,9),(s,10.5)}$ (t,10.5), (b,10.5),

$\underbrace{(c,11.5),(d,11.5),(e,11.5),(g,13.5),(f,14.5),(i14.5),(h,16.5),(,j,17.5),}$ (k,17.5),

$\underbrace{(l,19.5),(n,19.5),(m,22.5),(p,22.5),(o,25.5),(r,25.5),(q,27.5),(s,28.5),}$ (t,28.5).

Reduciendo el corte 1 y el corte 2 como mencionamos, la RPT de la figura 5.1-a se puede tranformar en la RPT reducida mostrada en la figura 5.1-b.

En la siguiente sección, se presentan las conclusiones y el trabajo futuro.

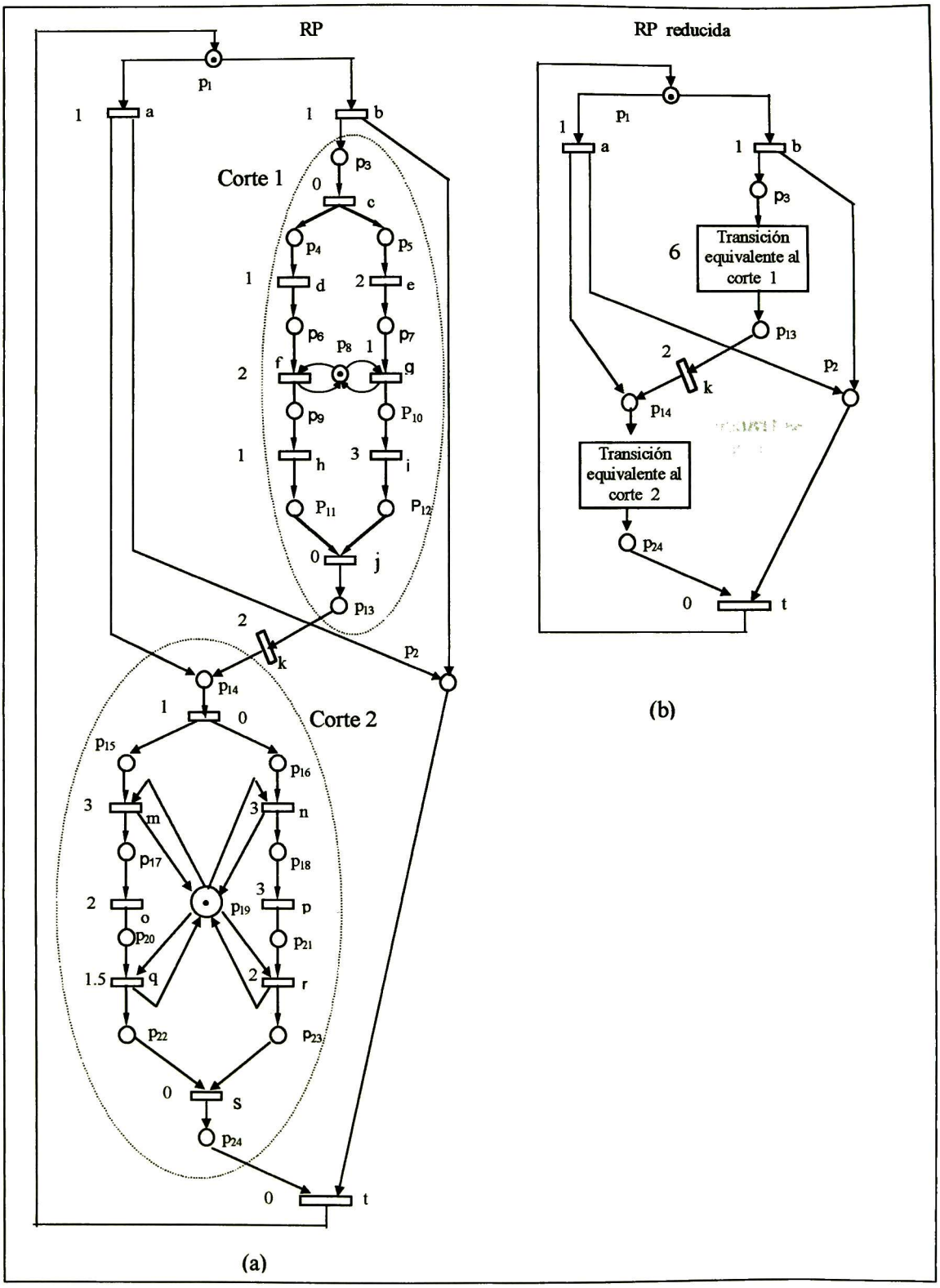
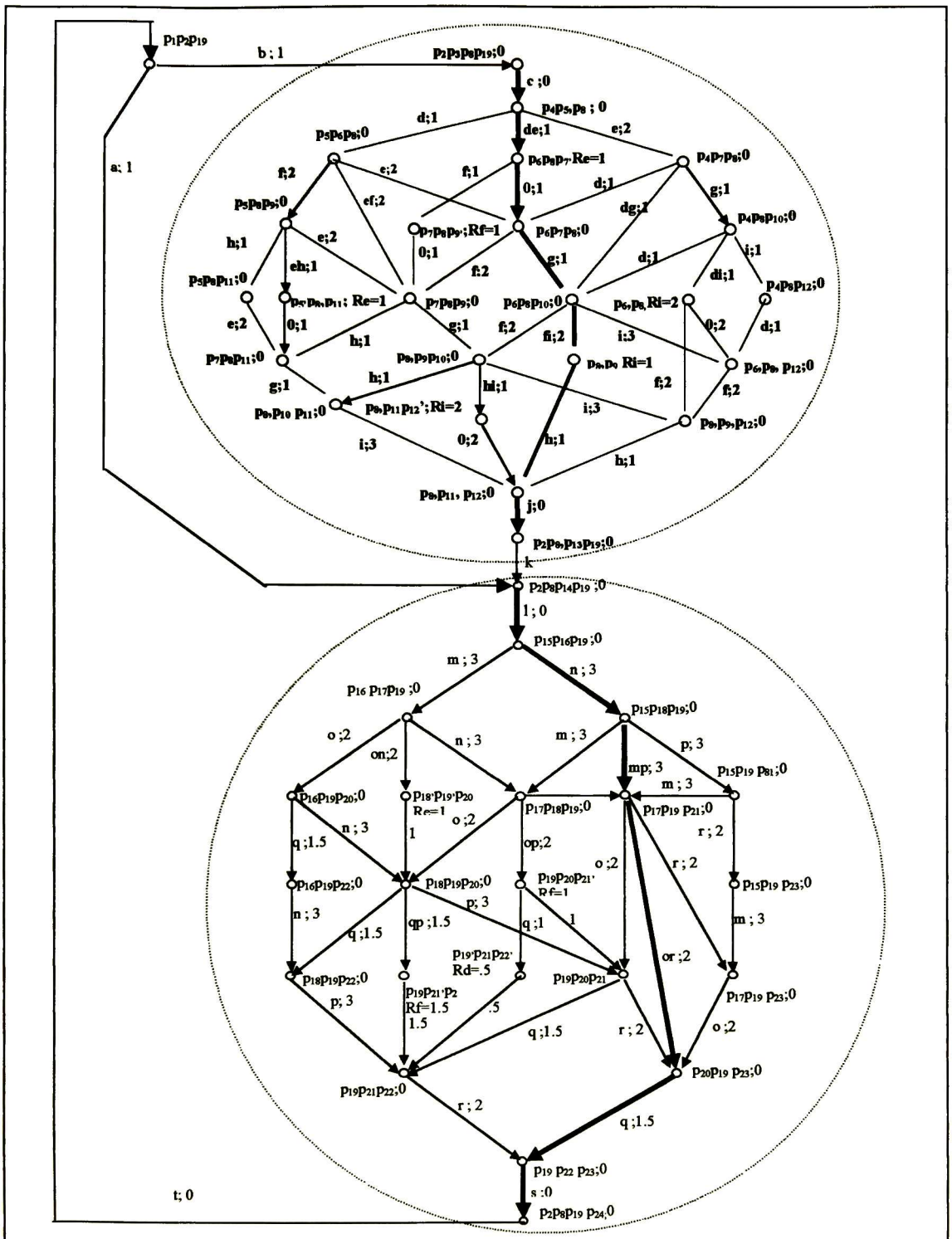


Figura 5.1: Ejemplo de RPT con dos PI.



En todos los nodos del corte 1, se encuentran marcados los lugares p_2 y p_{19} y en todos los nodos del corte 2, se encuentran marcados los lugares p_2 y p_8 aunque en la figura no se muestran para ahorrar espacio.

Figura 5.2: Grafo diligente completo de la RPT de ejemplo

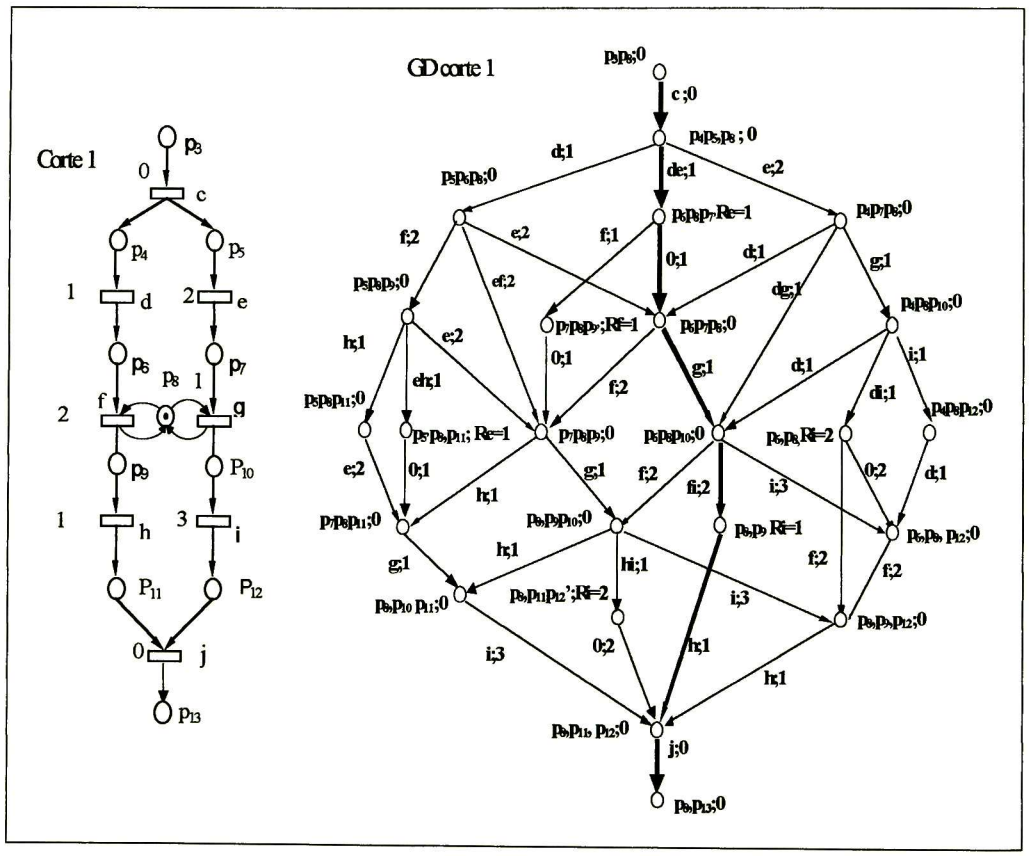


Figura 5.3: Corte 1 de la RPT de ejemplo y su GD correspondiente.

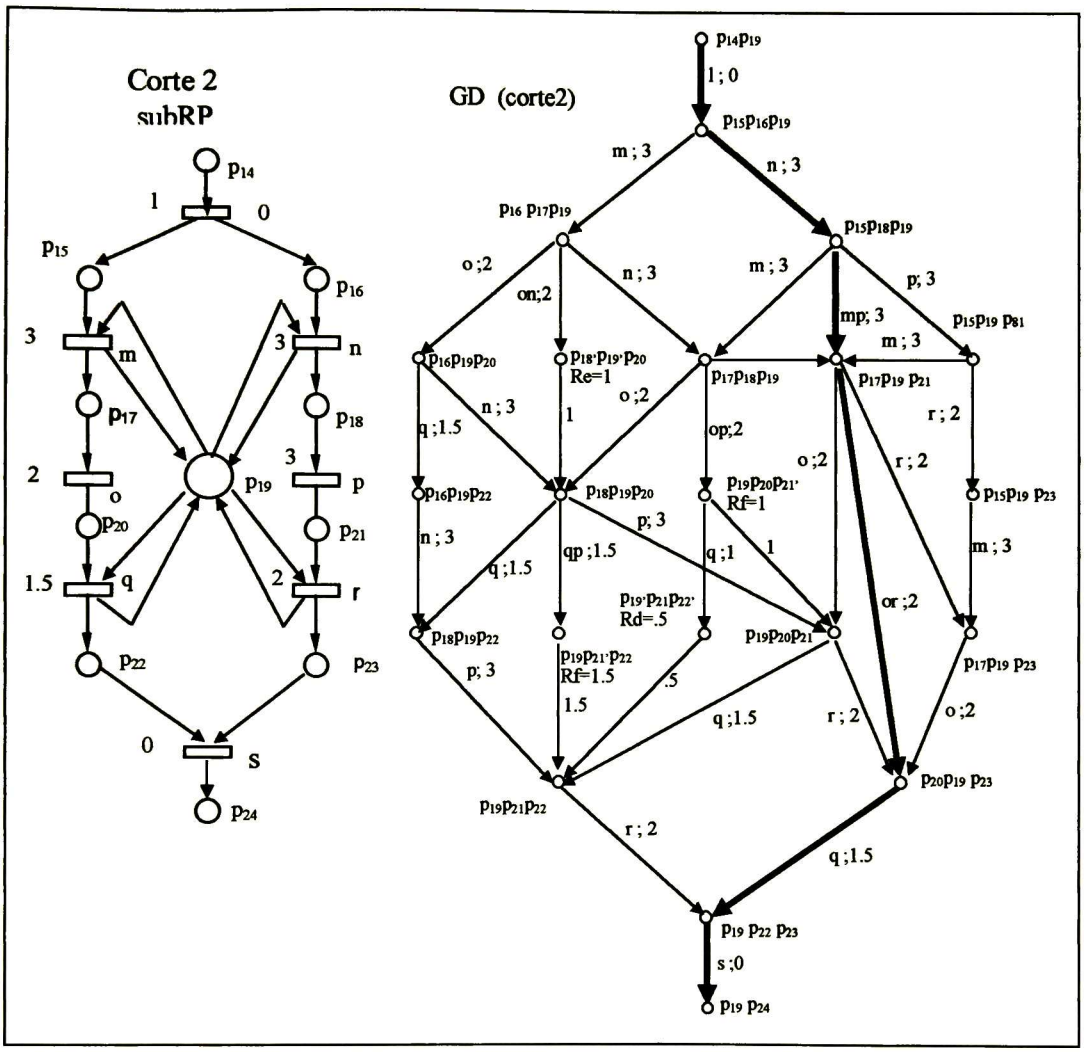


Figura 5.4: Corte 2 de la RPT de ejemplo y su GD correspondiente.

Conclusiones

En este trabajo se estudió el problema de scheduling para sistemas dinámicos de eventos discretos modelados con redes de Petri temporizadas. El problema de scheduling es un problema de optimización, donde se debe obtener el orden adecuado (secuencia óptima) en el que se tienen que realizar ciertas tareas, basándose en un criterio de optimización. En este trabajo dicho criterio fue realizar las tareas en el menor tiempo posible, y como el sistema esta modelado en redes de Petri, se tomó como criterio de optimización el tiempo de ejecución cíclica de la red.

Para abordar este problema se utilizaron algunas propiedades de las redes de Petri (principalmente estructurales), también se utilizaron los conceptos de corte (SLL) y la propiedad de optimalidad local para obtener el schedule óptimo en menor tiempo cuando es posible.

Se estudió la relación que existe entre los marcados de la red de Petri del sistema, analizando su grafo diligente, de donde resultó una nueva definición de pares inevitables, las características que debe cumplir un corte para generar un par inevitable y como realizar su optimización.

Se presentó otro criterio para determinar si una red de Petri posee la propiedad de optimalidad local basándose en la estructura de las FCA*

Se propusó un método para encontrar el schedule óptimo de las redes que poseen cortes, construyendo el grafo diligente, buscando los pares inevitables, optimizando localmente dichos pares inevitables utilizando el algoritmo A* y construyendo un grafo diligente reducido, que generalmente es de menor cardinalidad que el original. La ventaja que proporciona este método es de que si la red de Petri posee cortes, dichos cortes y el grafo diligente reducido pueden ser optimizados en menor tiempo porque las cardinalidades de los conjuntos a analizar son menores o porque resulten ser de alguna clase de RP de las que ya se sabe cómo obtener el schedule óptimo, entonces se puede reducir notablemente el tiempo de cálculo.

Desafortunadamente no todos los sistemas se pueden optimizar siguiendo la metodología propuesta, ya que en general el problema de scheduling es un problema NP, además que existen una gran variedad de sistemas dinámicos de eventos discretos que pueden ser modelados con redes de Petri, y los algoritmos (cuando estos existen) para analizar una RP arbitraria (cualquiera) resultan ser de una alta complejidad. Esta limitante se puede eliminar si se restringe el análisis a ciertas clases de RP. En este trabajo nos restringimos a las RP que poseen cortes, para los cuales se presentaron las reglas que deben cumplir y la metodología para el cálculo del schedule óptimo.

5.2.1 Trabajo futuro

Como trabajo futuro se tienen los siguientes puntos:

- Implementar el algoritmo presentado en este trabajo.
- Elaborar un algoritmo más general para obtener un schedule óptimo que abarque cualquier tipo de red de Petri.
- Caracterizar los cortes que generan pares inevitables.
- Buscar otros tipos de redes de Petri que posean la propiedad de optimalidad local con respecto a cortes.
- Estudiar el caso en que se ejecutan dos o más cortes en paralelo.

Bibliografía

[**Baker 91**] A Baker. Manofacturing Control with a Market-Driven Contract Net. Tesis Doctoral, Rensselaer Polytechnic Institute, Troy, New York, USA, 1991.

[**Bellman 82**] R. Bellman, A. Esogbue, y 1. Nabeshima. Mathematical Aspect of scheduling & Applications, volumen 4 de Modern Applied Mathematics and Computer Science. Pergamom Press, Great Britain,1982.

[**Campos 90**] J. Campos. Performance Bounds for Synchronized Queueing Networks. Tesis Doctoral, Universidad de Zaragoza., María de Luna 3 E-50015 Zaragoza, España., 1990.

[**Carlier 84**] J. Carlier, Ph. Chretienneand y C. Cirault. Modelling sheduling with Timed Petri net. En G. Rozenberg, H.Genrich, y G Roucairol, editors, Advances in Petri Nets 1984, volome.188 of Lecture Notes in computer Sciences, Springer-Verlag, Berlin . Germany, 1988.

[**Carlier 88**] J. Carlier y Ph. Chretienne. Timed Petri net schedules. En G. Rozenberg, editor, Advances in Petri Nets, vol. 340 LNCS. Springer-Verlag, Berlin, Germany, 1988.KH

[**Chretienne 83**] Ph. Chretienne. Les Reseaux de Petri Temporises. Tesis Doctoral, Universite Pierre et Marie Curie., Paris, France., 1983.

[**Dyke 91**] H.van Dyke. Characterizing the manofacturing scheduling ploblem. Journal of Manufacturing System. May 1991.

[**Ho 91**] Yu-Chi Ho y Hi-Ren Cao. Pertubation Analysis of Discrete Event Dynamic System. Engineering and computer science. Kluwer Academic Publishers, USA, 1991.

[**Kligerman 86**] E. Kligerman y A. Stoyenko. Real-time euclid: A lenguaje for reliable real-time system. IEEE Transactions on Software Engineering, (12(9):941-949, September 1986.

[**Lee 92**] D. Y. Lee y F. DiCesare. FMS scheduling using Petri nets and heuristic search. Proceedings of the International Conference on Robotics and Automation, pp 1057-1062, Nice France, May 1992.

[**Magott 84**] J. Magott. Performance evaluation of concurrent systems using Petri Nets. Information Processing Letters. Vol 18, pp 7-13, January 1984.

[**Magott 87**] J. Magott. New NP complete problems in performance evaluation of concurrent systems using Petri Nets. IEEE Transactions on Software Engineering. Vol (5): pp 587-581, May 1987.

[**Norvig 95**] Stuart Russell & Peter Norvig. Artificial Intelligence A Modern Approach. Pretice Hall. 1995.

[**Onaga 91**] K. Onaga, M.Silva, y T. Watanabe. On periodic schedules for deterministically timed Petri net systems. En Proceedings of the fourth International Workshop on Petri Nets and Performance Models. Melbourne, Australia, December 1991.

[**Ramamoorthy 80**] Ramamoorthy C.V., Ho G.S. Performance Evaluation of Asynchronus Concurrent Systems Using Petri Nets. IEEE Transactions on Software Engineering, Vol. SE-6, No. 5, September 1980.

[**Ramchandani 74**] Ramchandani C. Analysis of Asynchronous Concurrent Systems by Petri Nets. Ph.D. Thesis, L.C.S. M.I.T., Cambridge, MA, February 1974.

[**Ramírez 93a**] On Optimal Scheduling in DEDS. A. Ramírez, J. Campos & M. Silva Proc. of the IEEE International Conference on Robotics & Automation pp 821-826, May 1993, Atlanta, USA

[**Ramírez 93**] Scheduling in Free Choice Nets. A. Ramírez Proceedings of the IEEE International Conference on Systems, Man & Cybernetics pp. 289-294, October 1993, Le Touquet, France.

[**Ramírez 93t**] Scheduling in Petri Nets. A. Ramírez PhD. Thesis. U. Zaragoza [In Spanish] December 1993, Zaragoza, Spain.

[**Sifakis 78**] J. Sifakis. Use of Petri Nets for Performance Evaluation. Acta Cybernetica, Vol. (2) pp 185-202, 1987.

[**Silva 85**] M. Silva. Las Redes de Petri: en la Automática y en la Informática. Ed. AC Madrid, España, 1985.

[**Thiagarajan 84**] P. S. Thiagarajan and K. Voss. A fresh look at the Free Choice Nets. Information and Control Vol 2. Pp 85-113, May 1984.

CENTRO DE INVESTIGACION Y DE ESTUDIOS AVANZADOS DEL IPN
UNIDAD GUADALAJARA

El Jurado designado por el Departamento de Ingeniería Eléctrica y Ciencias de la Computación del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional, aprobó la tesis: "Scheduling en Sistemas de Eventos Discretos Modelados con Redes de Petri" el día 28 de Enero del 2000.

Dr. Luis Ernesto López Mellado
Investigador Cinvestav 3 A
CINVESTAV DEL IPN
Guadalajara

Dra. Ofelia Begóvich Mendoza
Investigador Cinvestav 3 A
CINVESTAV DEL IPN
Guadalajara

Dr. Antonio Ramírez Treviño
Investigador Cinvestav 2 A
CINVESTAV DEL IPN
Guadalajara