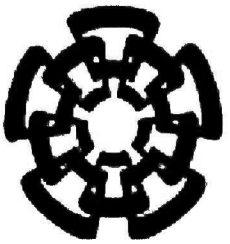
xx(86648.1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del IPN
Unidad Guadalajara

Optimización en sistemas de eventos discretos temporizados

Tesis que presenta
Juan Salvador Rodríguez Beltrán

Para obtener el grado de
Maestro en Ciencias

En la especialidad de
Ingeniería Eléctrica

Guadalajara, Jal., Agosto del 2000

Aspira hondamente la vida y piensa

como has de vivirla

Ama no lo que eres, sino lo que puedes

llegar a ser.

No corras tras los placeres,

pues puedes tener la desdicha de

alcanzarlos.

Mira siempre hacia delante

que en los nidos de antaño

no hay pájaros volando.

Se justo con todos los hombres,

se cortés con todas las mujeres.

Recuerda que no existe la maldad,

más que en los ojos del malvado,

reza siempre que puedas.

Mira siempre con el corazón

y no con los ojos,

vive siempre pendiente de tu ideal,

y de aquella que responde al nombre de

Dulcinea.

(Fragmento del Quijote)

Agradecimientos

A mi madre Patricia Beltrán Medina, por el apoyo incondicional que siempre me ha brindado a lo largo de mi vida.

A mi novia María de Lourdes Guechi García, por su compañía y palabras de aliento en los momentos difíciles.

A Juan Francisco Lizaola Gonzalez por brindarme su amistad, compañía y apoyo.

A mi asesor el Dr. Antonio Ramírez Treviño que dirigió y corrigió mi trabajo y esfuerzo durante este periodo de formación académica.

A mis compañeros Celia, Guillermo, Israel, Orlando, Enrique, Jaime, José Manuel, José Luis y Ramón por brindarme su amistad.

A los miembros del grupo de eventos discretos Alejandra, María y Luis Isidro porque su trabajo es la base de mi trabajo.

A los profesores del CINVESTAV-IPN Guadalajara por mostrarme que no sólo de ciencia vive el hombre.

A CONACyT.

A Lulu.

Índice General

Introducción

La teoría de scheduling trata con los problemas de optimización en sistemas de manufactura flexible. Por la naturaleza discreta de este tipo de sistemas, el problema de scheduling es un problema difícil de resolver, ya sea por la explosión combinatoria de estados o por la restricción a valores enteros de los parámetros del sistema. Como consecuencia, diferentes métodos se han desarrollado para resolver problemas específicos. Para dar solución a estos problemas, se emplean aproximaciones que utilizan técnicas de investigación de operaciones y técnicas computacionales como los sistemas basados en el conocimiento.

De entre las diferentes herramientas para el modelado de Sistemas de Eventos Discretos, las Redes de Petri Interpretadas Temporizadas (RPIT), que son una extensión de la Redes de Petri ordinarias, se seleccionan por que permiten añadir a los modelos el tiempo de duración de los eventos y un significado físico de los mismos, lo que resulta adecuado en el modelado de sistemas reales.

En este trabajo se tratan dos problemas de optimización: el problema del marcado inicial mínimo (MIM) en Grafos Marcados Temporizados y la ley de control de tiempo mínimo para un sistema de eventos discretos (DES).

En el primer problema, la red y el tiempo de ciclo son conocidos y el problema consiste en encontrar un marcado inicial mínimo M_0 tal que el tiempo de ciclo del GMT sea menor o igual al requerido. Se presentan dos algoritmos que resuelven el problema del MIM. El primer algoritmo "MIM_Solver" calcula un subconjunto de p-semiflujos y añade marcas a los lugares en dos pasos. Como el problema del MIM es NP, este algoritmo usa dos heurísticas basadas en los lugares pertenecientes al mayor y al menor número de p-semiflujos respectivamente en cada paso. Como se describe en este trabajo, este algoritmo presenta muchas mejoras sobre los algoritmos existentes. El segundo algoritmo "ALGEN" explora el espacio de los posibles marcados de un GMT hasta encontrar los que satisfagan las condiciones de tiempo de ciclo requerido π^d y un número de marcas dado. El algoritmo "MIM_Solver" es eficiente ya que requiere poco tiempo para obtener una solución. El GA siempre obtiene un resultado óptimo, aunque su tiempo de convergencia es infinito, pero puede ser de gran ayuda para estudiar el

comportamiento de otros algoritmos.

El segundo problema, el control de tiempo mínimo en DES, es también un problema NP y esto implica que se requiera un gran tiempo de cómputo para resolverlo. La forma propuesta en este trabajo para encontrar una ley de control de tiempo mínimo en tiempo razonable, es dividir el problema en pequeños subproblemas los cuales resultan más fáciles de resolver. En este trabajo se presenta una metodología que obtiene la ley de control de tiempo mínimo para un DES haciendo uso de un esquema de regulación, algoritmos heurísticos y de inteligencia artificial.

Capítulo 1

Panorama General

Resumen: La teoría de scheduling trata con los problemas de optimización en sistemas de manufactura flexible. Por la naturaleza discreta de este tipo de sistemas, el problema de scheduling es un problema difícil de resolver, ya sea por la explosión combinatoria de estados o por la restricción a valores enteros de los parámetros del sistema. Como consecuencia, diferentes métodos se han desarrollado para resolver problemas específicos, como lo son, la secuencia de operaciones más económica, la ruta de transporte más corta o el menor número de recursos. Para dar solución a estos problemas de optimización, se emplean aproximaciones que utilizan técnicas de investigación de operaciones y técnicas computacionales como los sistemas basados en el conocimiento, con los cuales en algunos casos, se obtiene una solución óptima o cercana a la óptima.

1.1 Introducción

Suponga que se desea iniciar un taller de maquilado. Después de un estudio de este tipo de industria se conoce: qué tipo de máquinas se utilizan en la manufactura de los diferentes productos que se van a trabajar; Las operaciones que deben hacerse para transformar la materia prima en productos terminados; Se cuenta con una cartera de clientes y un estimado de las posibles ventas que se tendrán. En este punto se llega a una de las etapas importantes en el desarrollo del proyecto: hacer la compra de la maquinaria, materia prima y la contratación de la mano de obra calificada, con las cuales se empezará a operar el taller. Si se adquiere poca maquinaria, material o se contrata un número reducido de personal, se tendrá que pagar horas extras y aun así, es posible que no cumpla con las entregas programadas lo que reducirá su margen de ganancia. Por otra parte si adquiere maquinaria o materia prima de manera excesiva o se contrata más personal del necesario para la producción estimada, es posible que mantenga máquinas con una baja utilización y personal sin trabajo aumentando los costos de producción, lo que a final de cuentas también reduce la ganancia obtenida. La pregunta que se plantea ahora es la siguiente: ¿Qué cantidad de mano de obra, maquinaria y materia prima se deben adquirir para satisfacer la demanda estimada?

Otro problema que puede presentarse son los cambios en los productos que se fabrican en un taller maquilador. En el taller se realizan distintas operaciones que transforman la materia prima hasta obtener un producto terminado. Para cada uno de los diferentes productos se requiere de un gran número de operaciones que varían dependiendo del producto final, ya que cada cliente tiene distintas especificaciones para productos semejantes. Hacer un cambio en las especificaciones de un producto requiere hacer cambios también en la forma de producirlo. Es necesario establecer las secuencias de operaciones que se deben hacer en el taller para obtener cada uno de los diferentes productos. Un cambio de producto o en la manera de producirlo requiere que en el taller se siga una nueva secuencia de operaciones. El objetivo de esta nueva secuencia es tener para cada producto, el menor tiempo ocioso en las máquinas y el personal. Por medio de un secuenciamiento óptimo se aumenta el volumen de producción, evitando que exista tiempo ocioso entre las operaciones. Conscientes de la importancia de las secuencias óptimas de operaciones para cada producto, nos planteamos la siguiente pregunta: ¿Cómo se determina para cada especificación la secuencia de operaciones con el menor tiempo ocioso?

Ya que uno de los principales objetivos de una empresa es aumentar el capital de los accionistas, los dos problemas anteriores, la asignación óptima de recursos y el secuenciamiento óptimo de las operaciones, son de gran importancia. Para estos dos problemas existe una gran cantidad de soluciones, pero sólo algunas de ellas son óptimas y de la correcta selección depende el incremento o decremento de las ganancias.

El problema de *Scheduling* consiste en seleccionar el orden de las tareas, la asignación de recursos y el calendario para su realización que optimiza un cierto criterio. Los problemas de scheduling se presentan en un tipo de sistemas llamados sistemas de eventos discretos. La asignación óptima de recursos y el secuenciamiento óptimo de las operaciones, son problemas de scheduling pues los sistemas de manufactura son sistemas de eventos discretos. Los sistemas en general pueden clasificarse dependiendo de las variables de estado que los describen[Ramírez_93d].

Un *sistema de eventos discretos*, es aquel cuyo espacio de estados es numerable, posiblemente infinito. Por ejemplo, en el modelado de la operación de un banco, el estado del sistema está en función de la cantidad de clientes en espera y los cajeros disponibles. El estado del sistema cambia sólo en el momento en que un cliente llega o se retira de la caja. En este trabajo se emplean las siglas en inglés DES (Discrete Event Systems) para nombrar estos sistemas.

Un *sistema continuo*, es en el que el estado del sistema cambia contínuamente en el tiempo. Por ejemplo, el simular el vuelo de un avión, el estado del sistema se describe mediante su posición, velocidad, aceleración, etc. Estas características cambian continuamente. Se emplearan las siglas CS (Continuous Systems) para nombrar este tipo de sistemas.

1.2 Planteamiento del Problema

Esta sección se enfocará a definir el problema de scheduling en el área de los sistemas de manufactura flexible, aunque el problema no se limita sólo a esta área.

1.2.1 Definiciones

Un *Sistema de Manufactura Flexible* (SMF), es un sistema que se modela como DES y está compuesto de máquinas-herramientas, sistemas automáticos de transporte, robots y otros dispositivos, todos controlados por un sistema de computo. El objetivo del sistema es fabricar

productos finales, obtenidos mediante trabajos realizados sobre las piezas que llegan al sistema. Cada pieza tiene asociada una serie de tareas (operaciones) llamada plan de proceso, que debe ser realizado por los dispositivos del SMF.

Las *Restricciones de un SMF* [Proth_96], son restricciones debidas a la tecnología usada, por el plan de proceso o por exclusión entre operaciones, pues dos o más de ellas no pueden hacerse al mismo tiempo.

Scheduling. El problema de scheduling consiste en la asignación de recursos a tareas y el establecimiento de los tiempos de inicio de cada tarea. Las tareas tienen cierta duración, que es el tiempo necesario para realizar dicha tarea; al terminar ésta, se dice que la tarea ha sido realizada y los recursos que le habían sido asignado son liberados. 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 la tarea con respecto a una referencia.

1.2.2 Clasificación de los problemas de scheduling

El problema de scheduling se puede dividir en tres clases de acuerdo al punto de vista de la investigación de operaciones [Martínez_87]:

1. El problema de balanceo de líneas de montaje.
   El objetivo de este problema consiste en decidir el número mínimo de dispositivos que se asignan a las distintas operaciones a realizar bajo algunas relaciones de precedencia, o bien en minimizar el tiempo de ciclo, esto es, el máximo de todos lo tiempos de proceso de todos los dispositivos.

2. El problema de scheduling en job-shops, también conocido como secuenciamiento.
   Consiste en determinar el orden y la hora de procesamiento de los productos en cada dispositivo para optimizar cierta función objetivo.

3. El problema de scheduling de proyectos.
   Consiste en la planificación de actividades que deben ser procesadas siguiendo relaciones de precedencia dadas con o sin restricciones de recursos.

En este trabajo son tratados los problemas de asignación óptima de recursos y el de secuenciamiento óptimo de las operaciones, los cuales son problemas del primer y segundo tipo

respectivamente. Para éstos problemas se proponen algunos métodos de solución.

1.2.3 Funciones de optimización

Las funciones de optimización permiten formular matemáticamente los objetivos de scheduling y evaluar la solución obtenida con respecto a un objetivo deseado.

Como criterios basados en los tiempos de terminación se tienen los siguientes:

- Minimizar el tiempo de flujo máximo. Se refiere al tiempo total consumido por cada producto j_i en el taller. El costo del schedule está relacionado directamente al tiempo que se invierte en procesar cada producto.

- Minimizar el tiempo de terminación máximo. Es el tiempo en que se termina la última operación hecha al producto j_i en el taller; Indica que el costo del schedule depende del tiempo requerido para procesar todos los productos.

Como criterios basados en las fechas debidas o fechas programadas para finalizar un producto se tienen los siguientes:

- Minimizar el retraso medio o máximo del producto. Es la diferencia algebráica entre el tiempo de terminación del producto j_i y su fecha debida. En este caso se recompensa que los productos sean terminados antes de la fecha programada para finalizarlo.

- Minimizar la tardanza media o máxima. Se refiere a la diferencia entre el tiempo de terminación y su fecha debida cuando el primero es mayor, esta función es aplicable cuando hay penalizaciones a los productos que no son terminados a tiempo.

- Minimizar el número de productos tardíos, es decir el número de productos terminados después de su fecha debida.

Como criterios basados en los costos de inventario y la utilización de los recursos se tienen los siguientes:

- Minimizar el número medio de productos esperando en las máquinas. Consiste en determinar el número mínimo de productos que se encuentran esperando entre los diferentes dispositivos en un determinado tiempo t.

- Minimizar el número medio de productos no terminados en un tiempo t.
- Maximizar el número medio de productos procesados en el tiempo t.
- Minimizar el tiempo medio y máximo de inactividad de las máquinas.

Algunos de las medidas de estos grupos son equivalentes, pero algunas de ellas son opuestas y no pueden ser empleadas como objetivo al mismo tiempo. Por ejemplo, minimizar el tiempo de terminación máximo es equivalente a minimizar el tiempo medio y máximo de inactividad de las máquinas, en cambio minimizar el tiempo de flujo máximo y minimizar el tiempo máximo de terminación son antagonicos cuando el tiempo de llegada de los productos a las máquinas es diferente de cero.

1.3 Solución del problema de scheduling

El problema de scheduling es un problema NP, es decir que el tiempo necesario para encontrar un schedule óptimo no puede ser representado por un polinomio en el número de datos de entrada y sí por una función exponencial [Russell_95]. A continuación se mencionan algunos métodos que se han empleado para dar solución a problemas de scheduling; Algunos de ellos se emplean de manera combinada.

1.3.1 Métodos de investigación de operaciones.

Durante la Segunda Guerra Mundial, la Fuerza Aérea Británica y las fuerzas armadas estadounidenses, formaron los primeros grupos dedicados a desarrollar métodos cuantitativos para resolver problemas operacionales, debido a la necesidad de administrar los escasos recursos, nació la investigación de operaciones. Después de la guerra, como resultado de las investigaciones realizadas por los grupos mencionados y otros más patrocinados por empresas privadas, surgieron métodos como la programación lineal, el método de ruta crítica CPM (Critical Path Method) y el PERT (Program Evaluation Review Technique), de los cuales a continuación se hace una breve descripción:

Programación Lineal [Luenberger_89]

En esta técnica, el problema es representado por un conjunto de ecuaciones lineales en la forma:

$$\min z(x) = c^T x$$
$$\text{sujeto a}$$
$$Ax = b$$
$$x \geq 0$$

Donde z es la función objetivo; x es el vector de variables de decisión del problema, de costos, tiempos de inicio, etc.; A y b representan restricciones existentes en el sistema. La programación lineal emplea el método llamado simplex para resolver en un número finito de pasos cualquier problema planteado como problema de programación lineal, donde todas las variables de decisión son reales.

Una extensión de la programación lineal, es la programación lineal entera, que es un modelo matemático que tiene una función objetivo y restricciones lineales, pero que requiere que algunas variables de decisión estén restringidas a tener valores enteros. Esta técnica también emplea el método simplex en combinación con la técnica de ramificación y acotamiento.

Estos modelos se aplican muy bien a los conocidos problemas de transporte, del agente viajero y de la mochila. Desafortunadamente, el problema de scheduling generalmente es de programación entera, es muy difícil de tratar y la aplicación de estas técnicas está limitada.

CPM-PERT [Mathur_96]

A mediados de los años cincuenta, Dupont Company creó la técnica llamada Método de Ruta Crítica (CPM), para administrar proyectos en los que el tiempo requerido para completar las tareas individuales se conocía con certeza. La Marina de los E.E.U.U. desarrolló la Técnica de Evaluación y Revisión de Proyecto (PERT) para administrar el proyecto de misiles Polaris, que implicaba alrededor de 500 tareas y el tiempo para completar muchas de ellas era incierto. Estas dos técnicas son similares excepto que CPM emplea tiempos determinados y PERT emplea tiempos probables, las dos utilizan como representación gráfica una red de proyecto, que consiste en una colección finita de nodos y arcos usados para representar las tareas y sus relaciones de precedencia en un proyecto. El método se realiza en tres pasos: calcula para cada actividad el tiempo de inicio más inmediato; el último tiempo de terminación y las tareas críticas. (con las

cuales se obtiene la ruta crítica).

1.3.2 Métodos basados en Inteligencia Artificial

La Inteligencia Artificial (AI) podría ser definida como la rama de la computación que se ocupa de la automatización del comportamiento inteligente, empleando para esto, estructuras de datos que representan el conocimiento, los algoritmos necesarios para aplicar el conocimiento y las técnicas de programación empleadas para su implementación.
En AI se tienen diferentes aproximaciones para resolver el problema de scheduling, dependiendo de las características del problema, algunas de éstas son mejores que las previamente citadas, a continuación se describen brevemente algunas de estas:

Métodos de búsqueda informada [Russell_95]

En los métodos de búsqueda "informada" se emplea una función heurística que guía la búsqueda de la solución, eliminando las rutas que no conducen a la solución óptima. La ventaja de la búsqueda informada es que obtiene más rápidamente una buena solución a diferencia de cualquier otro método que emplee una búsqueda ciega sobre todas las posibles soluciones. Algunos de estos algoritmos de búsqueda informada son los llamados *best-first* y A^*.

Sistemas Expertos [Luger_93]

Un sistema experto es un programa basado en conocimientos, que provee soluciones a problemas en un tema específico con la calidad de un experto. Generalmente, su conocimiento es extraído de personas expertas en dichos temas y el programa intenta emular su metodología y su desempeño. Como los humanos, los sistemas expertos tienden a ser especialistas, enfocándose en un conjunto limitado de problemas. El conocimiento de estos programas puede ser teórico y práctico, extraído de expertos y codificado en un lenguaje formal. A diferencia de las personas,

éstos sistemas no pueden aprender de sus propias experiencias.

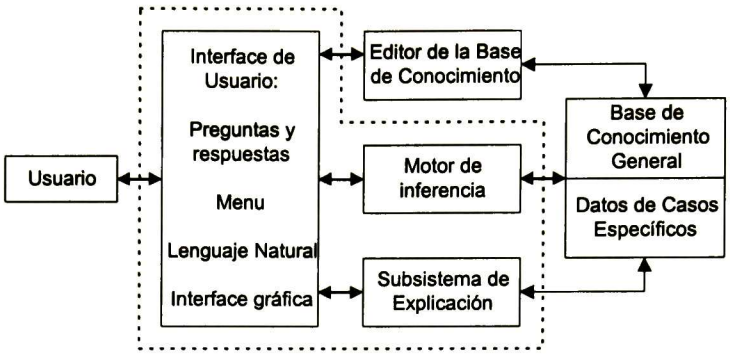


Figura 1.1. Arquitectura de un sistema experto típico.

En la figura 1.1 se muestran los módulos más importantes que conforman un sistema experto basado en reglas. El usuario tiene interacción con el sistema a través de una interfaz que puede ser de diferentes tipos. El programa debe poder guardar datos de casos específicos: hechos, conclusiones, y otra información de relevancia. El subsistema de explicación permite al programa mostrar el razonamiento que se siguío al usuario. Muchos sistemas incluyen un editor para la base de conocimiento. El programa central del sistema es la base de conocimientos, la cual contiene el conocimiento de problemas-soluciones del tema particular. El motor de inferencia aplica el conocimiento a la solución de los problemas.

Optimización Local y Temple Simulado

El método de Temple Simulado (SA: simulated annealing) es eficiente para resolver problemas de optimización combinatoria. Este método fue introducido por Kirkpatrick, Gelatt y Vecchi en 1983 y está basado en una analogía con la mecánica estadística.

El SA puede ser considerado como una mejora del método de optimización local (LO). El método LO comienza de una solución factible. En cada iteración, determina otra solución, empleando igualmente un procedimiento determinista o uno aleatorio. Si la nueva solución es mejor que la anterior, ésta permanece y el LO continua a partir de ésa solución. En otro caso, el método LO determina otra solución a partir de la solución previa. El método se detiene cuando la solución no puede ser mejorada en un número de iteraciones dado, o cuando se han hecho un número de iteraciones preestablecido.

El método SA se diferencía del LO en que el segundo puede aceptar una solución peor que

la previa, lo que resulta en un decremento de la probabilidad de llegar al óptimo. Se pierde también el incremento que se tenía en el valor del criterio a optimizar y se aumenta el número de iteraciones. El método de búsqueda SA tiene la ventaja de que evita el quedar atrapado en mínimos locales.

Algoritmos Genéticos [Goldberg_89]

Los Algoritmos Genéticos están basados en la teoría de la evolución de Darwing. Las soluciones a un problema dado son codificadas generalmente en palabras binarias que representan los cromosomas de un ser viviente. El algoritmo genético mantiene una población candidatos a soluciones de un problema. Como sucede en la naturaleza, la solución más apta no sólo sobrevive, sino que, a través de una analogía con la reproducción sexual, el algoritmo genético intercambia información con otros candidatos para formar nuevas soluciones. En el algoritmo genético se realiza una selección de los candidatos basándose en su desempeño, con el fin de producir nuevas soluciones hijo mediante una operación llamada cruza, que toma dos soluciones candidatas y las divide, intercambiando sus componentes para producir dos nuevos candidatos. Otra operación de estos algoritmos es la mutación, la cual cambia en un candidato elegido al azar uno o varios de los elementos de la palabra codificada.
El poder de los algoritmos genéticos es la naturaleza de la búsqueda ya que ésta se hace de manera paralela, manteniendo varias soluciones, eliminando las menos prometedoras y mejorando en cada generación las soluciones. Otra ventaja de esta técnica, es la exploración de diversas partes del espacio de soluciones, en la cual se evita caer en extremos locales gracias a las operaciones que realiza.

Todos estos métodos han sido aplicados para resolver el problema de scheduling para casos particulares, obteniéndose en cada caso schedules óptimos o buenos, sin embargo, dada la naturaleza combinatoria del problema, es difícil encontrar un algoritmo general, por lo tanto, se tienen que aplicar en cada caso heurísticas para recortar la búsqueda.

1.4 Objetivos

El objetivo de este trabajo es proponer tres algoritmos heurísticos que combinan algunas de los métodos previamente citados para resolver los problemas de asignación mínima de recursos y

el de secuenciamiento óptimo de operaciones en Sistemas de Manufactura Flexibles.

1.5 Organización del trabajo

En el capítulo dos se definen formalmente las Redes de Petri y algunas extensiones de ellas: las Redes de Petri Temporizadas, las Redes de Petri Interpretadas y se define una extensión de estas últimas, las Redes de Petri Interpretadas Temporizadas, que nos servirán como herramienta de modelado formal para describir los sistemas de eventos discretos. Se recuerdan también en este capítulo, algunos resultados para los Grafos Marcados, los cuales son una subclase de Red de Petri y serán tratados en capítulos posteriores.

En el tercer capítulo, se propone una metodología de modelado formal para sistemas de eventos discretos usando Redes de Petri Interpretadas Temporizadas. Para el modelado se supone que los sistemas están formados por tres clases de elementos: actuadores, sensores y la planta. En este capítulo se describen estos elementos, así como la forma de modelar cada uno de ellos y la manera de construir un modelo global para el sistema.

El capítulo cuatro presenta el problema del Marcado Inicial Mínimo (MIM) en Grafos Marcados Temporizados. En sistemas de manufactura flexible, este problema es equivalente a la asignación mínima de recursos y trabajo en proceso para obtener una cadencia de producción deseada. Para dar solución al problema del MIM en este capítulo se proponen dos algoritmos con diferentes características, uno heurístico y otro genético.

Finalmente, en el quinto capítulo se aborda el problema del secuenciamiento óptimo de operaciones. Se presenta un esquema de regulación de tiempo mínimo, el cual consiste en un DES regulado por otro llamado modelo de referencia. El modelo de referencia representa las especificaciones que se requiere que siga el DES. El esquema propuesto obtiene el secuenciamiento óptimo para una especificación dada.

1.6 Conclusiones

El problema de scheduling de sistemas de eventos discretos es un problema de optimización, pues se enfoca a la búsqueda entre las posibles soluciones de la que obtenga el mejor desempeño para un criterio dado. Para la solución del problema, se han estudiado varios de métodos que

se clasificaron en basados en investigación de operaciones y basados en inteligencia artificial.

Mediante los métodos de investigación de operaciones pueden resolverse algunos tipos específicos de problemas de scheduling, que se pueden representar como un conjunto de ecuaciones lineales o bien como un grafo. En la práctica, es difícil aplicar estos métodos en los sistemas de manufactura flexible, ya sea por la gran cantidad de variables de decisión o por la naturaleza entera de las variables.

Los métodos basados en inteligencia artificial hacen una representación de los sistemas describiendo su comportamiento y las relaciones existentes en él. Obtienen una solución buena a los problemas, considerando que satisface las restricciones establecidas y que se obtiene en un tiempo razonable, aunque es difícil en la mayoría de los casos probar si ésta es óptima.

Capítulo 2

Herramienta de Modelado de SED

Resumen: Existen diferentes herramientas para el modelado de Sistemas de Eventos Discretos, entre éstas las Redes de Petri (RP) se seleccionan en este trabajo por las ventajas que presentan, por ejemplo, capturan la concurrencia, sincronización, exclusión mutua y relaciones causales, además tienen una representación gráfica que las hace amenas y fáciles de entender y finalmente está su soporte matemático, que permite el análisis de propiedades.

Las Redes de Petri Temporizadas (RPT) y las Redes de Petri Interpretadas (RPI), permiten añadir a los modelos una base de tiempo y un significado físico, lo que resulta adecuado en el modelado de sistemas reales.

En este capítulo se presentan las RP, RPT, RPI y se definen las Redes de Petri Interpretadas Temporizadas (RPIT), su estructura y las reglas de su dinámica.

2.1 Introducción

A lo largo del tiempo se han empleado diferentes herramientas para hacer la representación de Sistemas de Eventos Discretos (DES), tales herramientas son las tablas de estado y los autómatas finitos, que describen los estados que pueden ser alcanzados por un sistema, o el PERT-CPM con el que se modela la ejecución de actividades de un sistema. Pero estas herramientas presentan algunas desventajas: los modelos de sistemas con autómatas finitos en muchos casos se vuelven incomprensibles debido a la gran cantidad de estados que utilizan en su representación, en el caso del PERT_CPM es imposible representar decisiones.

Por otro lado, las Redes de Petri (RP), son una herramienta adecuada para el modelado y análisis de DES, pues consiste en una representación gráfica compacta donde de manera clara pueden representarse los estados y cambios de un sistema. Además de que poseen un formalismo matemático que permite analizar propiedades de los sistemas. Otra ventaja de esta herramienta es que es capaz de representar decisiones, concurrencia, exclusión mutua, relaciones causales y de sincronía entre los eventos. Usando las RP pueden ser analizadas propiedades como comportamiento cíclico, vivacidad, ausencia de bloqueos, acotamiento, etc.

Existe una extensión hecha a las RP, la cual asigna un tiempo de retardo a las transiciones. Esta extensión permite capturar el tiempo de trabajo de un sistema, pues las marcas a través de la RP se detienen en cada transición y no se encuentran disponibles sino hasta que el tiempo de retardo haya transcurrido. Dichas extensiones son las Redes de Petri Temporizadas (con tiempos de retardo deterministas) y las Redes de Petri Estocásticas (con tiempos de retardo estocásticos), las cuales permiten analizar las prestaciones de un sistema y la búsqueda de schedules para el disparo de sus transiciones.

En las Redes de Petri Interpretadas (RPI) se logra asociar al modelo un significado físico de lo que ocurre en los sistemas, pues se agregan al modelo las funciones de entrada y salida que generan los lenguajes propios de un sistema, como señales de actuadores y sensores. La inclusión de estas funciones da la oportunidad de analizar el comportamiento de sistemas con eventos controlables e incontrolables y con estados no medibles.

En este trabajo se agrega a las RPI el tiempo de retardo de los eventos del sistema, definiendo así las Redes de Petri Interpretadas Temporizadas (RPIT); modelando los DES con RPIT se representa con detalle la estructura y la dinámica del sistema, sin alejarse del significado físico

y considerando el tiempo de trabajo del sistema.

2.2 Redes de Petri

En esta sección se introducen los conceptos básicos sobre Redes de Petri, su estructura, su dinámica y la inclusión del tiempo.

Definición 1 *Una red de Petri (RP) ordinaria es un grafo bipartido dirigido representado por una 4-tupla* $RP = (P, T, E, S)$, *donde:*

- $P = \{p_1, \ldots, p_n\}$ *es un conjunto de vértices llamados lugares los cuales se representan gráficamente con un círculo.*
- $T = \{t_1, \ldots, t_m\}$ *es un conjunto de vértices llamados transiciones los cuales se representan gráficamente con una barra o rectángulo.*
- $I : P \times T \rightarrow \{0, 1\}$ *es la función de entrada que representa los arcos que van de los lugares hacia las transiciones y se define de la siguiente forma:*

$$I(p_i, t_j) = \begin{cases} 1 & \textit{si el arco de } p_i \textit{ a } t_j \textit{ existe} \\ 0 & \textit{en caso contrario} \end{cases}$$

- $O : P \times T \rightarrow \{0, 1\}$ *es la función de salida que representa los arcos que van de las transiciones hacia los lugares y se define de la siguiente forma:*

$$O(p_i, t_j) = \begin{cases} 1 & \textit{si el arco de } t_j \textit{ a } p_i \textit{ existe} \\ 0 & \textit{en caso contrario} \end{cases}$$

$P \cap T = \emptyset; P \cup T \neq \emptyset.$

La función de marcado $M : P \rightarrow \mathbb{N}^+ \cup \{0\}$, asigna a cada lugar de RP un número dado de elementos llamados marcas. Las marcas se representan gráficamente por puntos en los lugares. El marcado inicial M_0, es una asignación arbitraria inicial de marcas.
Un sistema $\Sigma = (RP, M_0)$ o *red marcada* es una red de Petri con un marcado.

2.2.1 Estructura de las RP

La estructura de la *RP* se puede representar en forma matricial, donde:

- La función I de entrada o incidencia previa se representa por la matriz $C^- = \left[c_{i,j}^-\right]_{n\times m}$ donde $c_{i,j}^- = I(p_i, t_j)$.
- La función O de salida o incidencia posterior se representa por la matriz $C^+ = \left[c_{i,j}^+\right]_{n\times m}$ donde $c_{i,j}^+ = O(p_i, t_j)$.
- La matriz de incidencia $C = [c_{i,j}]_{n\times m}$, está definida por $c_{i,j} = c_{i,j}^+ - c_{i,j}^-$.

El conjunto $\{x^\bullet, x \in P \cup T\}$ es el conjunto de todos los sucesores inmediatos de x; el conjunto $\{^\bullet x, x \in P \cup T\}$ es el conjunto de todos los predecesores inmediatos de x.

2.2.2 Dinámica de las RP

La dinámica de una *RP* está sujeta a las siguientes reglas:

1. Regla de habilitación. Una transición $t_j \in T$ está habilitada si $\forall p_i \in P : M(p_i) \geq I(p_i, t_j)$. Si una transición t_j está habilitada, entonces se puede disparar.
2. Regla de disparo. Si una transición habilitada t_j es disparada en un marcado M_k entonces se alcanza el nuevo marcado M_{k+1}, eliminando $I(p_i, t_j)$ marcas de sus lugares de entrada p_i y añadiendo $O(p_i, t_j)$ marcas a sus lugares de salida p_i.

Definición 2 *Sea M_i un marcado de una RP Si $M_i \xrightarrow{t_i} M_{i+1} \xrightarrow{t_j} \cdots \xrightarrow{t_k} M_k$ entonces $\sigma = t_i t_j \ldots t_k$ es una* secuencia de disparos *que lleva del marcado M_i al marcado M_k y se escribe $M_i \xrightarrow{\sigma} M_k$ y se dice que M_k es un* marcado alcanzable *a partir de M_i.*

Definición 3 *Sea R una RP. El conjunto de alcanzabilidad de R, denotado como $\mathcal{R}(R, M_0)$ es el conjunto de todos los marcados alcanzables en R a partir de su marcado inicial M_0.*

Definición 4 *El vector característico o de Parikh de una secuencia σ, es un vector $\overrightarrow{v} \in (\mathbb{N}^+ \cup \{0\})^m$ cuyo i-ésimo elemento es el número de veces que la transición i se dispara en la secuencia σ.*

Al disparar una transición t_j habilitada en M_k se alcanza un nuevo marcado M_{k+1} el cual puede ser calculado mediante la *ecuación de estados* de una RP :

$$M_{k+1} = M_k + C\overrightarrow{v}$$

donde $\overrightarrow{v}$ es el vector de Parikh de la secuencia $\sigma = t_j$.

Definición 5 *El lenguaje generado por una RP es el conjunto de secuencias disparables a partir de* M_0.

$$L(RP, M_0) = \left\{\sigma | M_0 \xrightarrow{\sigma} M\right\}$$

Definición 6 *Los p-semiflujos son todos los vectores* Y *tal que* $Y^T C = 0, Y \geq 0$.

Definición 7 *Los t-semiflujos son todos los vectores* X *tal que* $CX = 0, X \geq 0$.

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

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

Un p-semiflujo (t-semiflujo) 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.

2.2.3 RP Temporizadas

Definición 8 *Una red de Petri temporizada (RPT) es una 5 tupla* $RPT = (R, d)$ *donde :*

- R *es una Red de Petri ordinaria y*
- $d : T \rightarrow \mathbb{R}^+$. $d(t_i) = d_i$ *es llamado el retardo de la transición* t_i *y es el tiempo necesario para terminar el disparo de la transición.*

Note que el tiempo se asigna a las transiciones y no a los lugares ya que éstas representan las actividades del sistema. Hay otras extensiones que agregan el tiempo a los lugares [Sifakis_78]. Una transición puede ser disparada si está habilitada. El disparo de una transición t_j elimina $I(p_i, t_j)$ marcas de sus lugares de entrada, pero éstas no se encuentran disponibles para el disparo de otra transición durante d_i unidades de tiempo, después de este lapso, son depositadas $O(p_j, t_j)$ marcas disponibles en los lugares p_j de salida de t_j. La distribución inicial M_0 de una RPT tiene sólo marcas disponibles.

2.2.4 Propiedades de las RP

Un lugar $p \in P$ de una RP R está *k-acotado* si y sólo si $\forall M \in \mathcal{R}(R, M_0)$ el marcado de p cumple con $M(p) \leq k < \infty$.

Una red marcada (R, M_0) es *k-acotada* si y solo si todos sus lugares son k-acotados.

Una RP R es *estructuralmente k-acotada* si y solo si $\forall M_0$ todas las redes marcadas (R, M_0) son k-acotadas.

Una transición es *viva* en (R, M_0) si y solo si $\forall M \in \mathcal{R}(R, M_0) : \exists M' \in \mathcal{R}(R, M_0)$ tal que M' habilita t.

Una red marcada (R, M_0) es *viva* si y solo si todas sus transiciones son vivas.

Una red R es *estructuralmente viva* si y solo si $\exists M_0$ tal que la red marcada (R, M_0) es viva.

La red marcada (R, M_0) es *libre de bloqueos* si y solo si $\forall M \in \mathcal{R}(R, M_0)$ $\exists t \in T$ tal que t esta habilitada por M.

Definición 9 *La* velocidad de disparos *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. El* tiempo de ciclo *de t_i es el inverso de su velocidad de disparo.*

Definición 10 *Un schedule factible S queda definido 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, τ_k es el tiempo de inicio de disparo de la transición t_{i_k} en σ.*

Definición 11 *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 se obtiene al aplicar dicho schedule es menor que la cota "c"*

Definición 12 *El schedule óptimo es un schedule factible cuya ejecución permite maximizar la velocidad de disparo de las transiciones del sistema.*

2.3 RP Interpretadas

Definición 13 *Una red de Petri Interpretada RPI es la 6-tupla* $Q = (N, \Sigma, \Phi, \lambda, D, \varphi)$ *donde:*

- N *es una red de Petri marcada i.e.* $N = (R, M_0)$.
- $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_r\}$ *es un conjunto finito de elementos* σ_i *llamados símbolos de entrada,* Σ *es llamado el alfabeto de entrada.*
- $\Phi = \{\phi_1, \phi_2, \ldots, \phi_s\}$ *es un conjunto finito de elementos* ϕ_i *llamado símbolos de salida,* Φ *es llamado el alfabeto de salida.*
- $\lambda : T \rightarrow \Sigma \cup \{\varepsilon\}$ *es una función de etiquetado de las transiciones. (*ε *es el símbolo nulo).*
- $D : T \rightarrow \top$ *es una función de prealimentación, donde* $\top = \tau \cup \{\varepsilon\}$ *y* τ *es el conjunto de símbolos de nombres de las transiciones. Cuando una transición* t_j *es disparada entonces su nombre* t_j *o el símbolo* ε *es dado como salida de la RPI.*
- $\varphi : \mathcal{R}(RP, M_0) \rightarrow \{\Phi \cup \{\varepsilon\}\}^q$ *es una función de salida. (*ε *es el símbolo nulo), donde* $\mathcal{R}(RP, M_0)$ *es el conjuto de alcanzabilidad definido para una RP.*

Observación 14 *Debido a que existe un marcado inicial en una RPI* $Q = (N, \Sigma, \Phi, \lambda, D, \varphi)$, *se utilizara* (Q, M_0) *para denotar una RPI de forma abreviada.*

Una transición $t_j \in T$ de una RPI está habilitada en el marcado M_k si $\forall p_i \in P, M_k(p_i) \geq E(p_i, t_j)$. Si $\lambda(t_j) = a_i \neq \varepsilon$ está presente y t_j está habilitada, entonces t_j debe dispararse. Si $\lambda(t_j) = \varepsilon$ y la transición t_j está habilitada entonces t_j puede ser disparada. Si una transición habilitada t_j es disparada en un marcado M_k, entonces un nuevo marcado M_{k+1} es alcanzado. M_{k+1} puede ser calculado usando la parte dinámica de la ecuación de estados $M_{k+1} = M_k + Cv_j$, con C y v_j definidas como en las RP ordinarias. Esto se representa también como: $M_k \xrightarrow{t_j} M_{k+1}$.

Definición 15 *Si* $\lambda(t_j) \neq \varepsilon$ *se dice que la transición* t_j *es manipulable, en otro caso se dice que es no manipulable. El símbolo* ε *representa cualquier evento interno del sistema. Un lugar*

$p_i \in P$ se dice que es medible si $\forall M_j(p_i) \neq M_k(p_i)$ se tiene que $\varphi M_j \neq \varphi M_k$, y es no medible en otro caso. Los lugares medibles son representados gráficamente por círculos sin sombrear y los lugares no medibles por círculos sombreados. La transición t_j se dice también que es medible si $D(t_j) = t_j$ y no medible en otro caso. Las transiciones medibles son representadas gráficamente con barras no sombreadas y las transiciones no medibles con barras sombreadas. El conjunto de lugares no medibles se denota como PNM y el conjunto de transiciones no medibles es denotado como TNM.

Este trabajo se enfoca al caso en el cual todos los eventos del sistema son manipulables.

2.3.1 RPI Temporizadas

Definición 16 *Una red de Petri Interpretada Temporizada RPIT es la 7-tupla $RPIT = (RPI, d)$ donde*

- *RPI es una red de Petri Interpretada y*
- *$d : T \rightarrow \mathbb{R}^+$ El valor $d(t_i) = d_i$ es llamado el retardo de la transición t_i y es el tiempo necesario para completar el disparo de la transición.*

Al igual que en las RPT, las $RPIT$ tienen los atributos de disponibilidad y no disponibilidad de las marcas. Al dispararse la transición t_i, las marcas de sus lugares de entrada estarán no disponibles para el disparo de otra transición. Después de que haya transcurrido el retardo $d(t_i)$ de la transición, se depositan marcas disponibles en los lugares de salida de t_i.

2.4 Grafos Marcados

Los Grafos Marcados son una subclase de RP. En esta sección se presentan la definición formal de un Grafo Marcado y algunos resultados obtenidos para grafos marcados fuertemente conexos.

Definición 17 *Un grafo marcado fuertemente conexo es una RP, RPT, RPI o RPIT que cumple con $|{}^{\cdot}p| = |p^{\cdot}| = 1, \forall p \in P$ y $\forall x, y \in P \cup T$ $\exists$ un camino (secuencia de vértices) de x a y $(x \rightarrow y)$ y un camino de y a x $(y \rightarrow x)$.*

2.4.1 Resultados en Grafos Marcados

Definición 18 *Si $S_i(k)$ representa el instante del k-ésimo disparo de la transición t_i, entonces el tiempo de ciclo π de la transición t_i es:*

$$\pi = \lim_{k \to \infty} \frac{S_i(k)}{k}$$

Teorema 19 (Ramamoorthy_80) *Para un Grafo marcado, el número de marcas en un p-semiflujo es el mismo después de cualquier secuencia de transiciones disparadas.*

Teorema 20 (Ramamoorthy_80) *En grafos marcados fuertemente conexos todas las transiciones tienen la misma velocidad de disparo.*

Del teorema anterior puede verse que en un grafo marcado todas las transiciones tienen el mismo tiempo de ciclo.

Teorema 21 (Ramamoorthy_80) *El tiempo de ciclo de un grafo marcado es igual al tiempo de ciclo del circuito más lento.*

El cálculo del tiempo de ciclo se realiza en tiempo polinomial utilizando el método simplex, resolviendo el siguiente Problema de Programación Lineal [Magott_84]:

$$\begin{aligned} & \pi = \max Y^T \cdot C^- d \\ & \text{s.a.} \\ & Y^T \cdot C = 0 \\ & Y^T \ M_0 = 1 \\ & Y_i \geq 0 \end{aligned} \tag{2.1}$$

donde d es el vector de tiempos de retardo de las transiciones de una RPT o $RPIT$.

En un Grafo Marcado se obtiene un schedule óptimo aplicando la regla de "disparar las transiciones tan pronto estén habilitadas", ya que se obtiene la mayor cantidad de disparos de transiciones por unidad de tiempo [Carlier_88].

2.5 Conclusiones

En este capítulo, se establecieron las bases de la herramienta de modelado que será empleada en el estudio y análisis de los problemas de asignación de recursos y scheduling en sistemas de eventos discretos.

Se presentaron las Redes de Petri y la teoría básica sobre las mismas, haciendo énfasis en las ventajas que presenta sobre otros formalismos capaces de modelar DES, como son la representación de decisiones, concurrencia, exclusión mutua, relaciones causales y de sincronía entre los eventos, una representación gráfica amena, fácil de entender, y un soporte matemático que permite el análisis de diferentes propiedades de un sistema.

Se introdujeron las Redes de Petri Interpretadas Temporizadas, una extensión de las Redes de Petri Temporizadas y las Redes de Petri Interpretadas, las cuales permiten dar mayor detalle y significado al modelo de un sistema por la inclusión del tiempo y funciones de etiquetado de los eventos y estados del sistema.

Se presentaron también los Grafos Marcados Temporizados, una subclase de Redes de Petri, para las cuales se mencionaron algunos resultados que pueden encontrarse en la literatura sobre Redes de Petri y que serán empleados en el transcurso de este trabajo.

Capítulo 3

Modelado de los Sistemas

Resumen: En este capítulo se presenta una metodología para el modelado de DES usando Redes de Petri Interpretadas Temporizadas. En este trabajo se asume que los sistemas están compuestos por los siguientes tres grupos de elementos básicos: actuadores; planta y sensores. En esta metodología los diferentes elementos se modelan por separado y aplicando entre los actuadores y la planta las reglas de "sincronización" y "causalidad" se construye un modelo global.

3.1 Introducción

En este capítulo se presenta un método para el modelado de SED usando Redes de Petri Interpretadas Temporizadas, pues no existe una manera formal para el modelado de este tipo de sistemas usando esta herramienta. En trabajos previos, se han presentado ejemplos de cómo algunos tipos de SMF pueden ser modelados, pero ninguno establece un método formal para hacerlo, la aproximación que aquí se presenta es una extensión del trabajo presentado en [Cordova_98], donde los sistemas son divididos en tres categorías: a) Los actuadores; b) Los sensores y c) La planta.

Los actuadores pueden ser interruptores, relevadores, válvulas, etc., es decir, son elementos cuyo comportamiento se limita a producir cambios en el estado de la planta mediante señales de entrada, dadas por un usuario o un dispositivo de control automático. La planta puede ser una máquina; un conjunto de ellas o cualquier otro sistema como los sistemas continuos. Dependiendo del estado del actuador y un evento interno, el estado de la planta cambia. Los sensores son elementos encargados de medir el estado del sistema y/o actuador.

Cada elemento es modelado por separado y de acuerdo con dos reglas de relación: sincronización o causalidad, el modelo global es construido sin necesidad de ninguna etapa de ajuste posterior. En este capítulo, se explica la forma en que son clasificados y modelados los diferentes elementos del sistema, el tipo de relación que existe entre ellos y la forma de representarlos por medio de RPIT.

Existen buenos catálogos de actuadores o modelos de máquinas en términos de RPIT. Aquí se incluyen de manera breve algunos de estos modelos, pero el lector interesado puede consultar [Proth_96] [Silva_85] para una explicación más detallada.

3.2 Modelado de los actuadores.

Los actuadores o accionadores son dispositivos que producen la entrada a la planta de acuerdo con la señal de control, de modo que el estado del sistema corresponda a un estado de referencia deseado. Los actuadores son accionados por un usuario o por la señal de salida de un controlador automático.

Los actuadores producen entradas σ_i a la planta, en un modelo en RPIT están representadas

por transiciones. Donde cada transición t_i es etiquetada por la función λ con la señal σ_i si ésta es manipulable o ε en otro caso. Cada posición estable que el actuador alcanza antes y después de producirse la señal σ_i es representado por lugares p_i, p_j respectivamente, las transiciones y lugares se unen con arcos simples desde la posición p_i antes de la señal σ_i y después de esta señal a la nueva posición p_j. Las marcas en el modelo en RPIT de un actuador representan la posición inicial activa de éste.

Los actuadores son clasificados de acuerdo a su comportamiento como sigue: [Cordova_98]

- Secuenciales. Son los actuadores que presentan una evolución secuencial en sus estados estables. Como las válvulas, botones, interruptores, etc. La figura 3.1 muestra algunos de estos actuadores y su modelo en RPIT.

  - En el modelo de la válvula se asignan a las transiciones t_1 y t_2 por medio de la función λ, las señales c y a correspondientes a los eventos cerrar y abrir válvula, estos eventos son dados por un usuario o un sistema automático de control.
  - En el caso del botón normalmente cerrado, se asignan a las transiciones t_1 y t_2 las señales a y ε, pues el abrir el interruptor es un evento que es propiciado por un agente externo, pero el cerrar, es un evento interno del actuador, por eso se asigna la etiqueta ε de señal no manipulable.

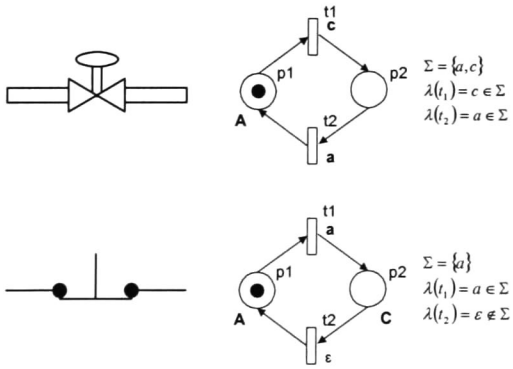
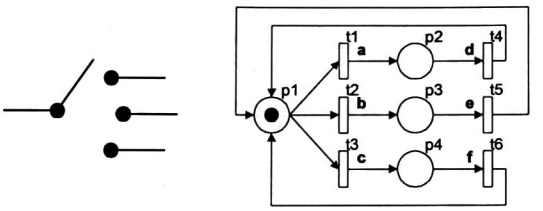


Figura 3.1. Actuadores Secuenciales.

- Elección. Son los actuadores en cuya evolución, es posible elegir el siguiente estado estable. Como ejemplo tenemos los selectores. La figura 3.2 muestra un ejemplo de este tipo de

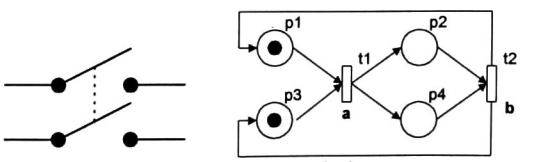
actuadores. Al igual que en el modelo anterior, por medio de la función λ, se asigna a cada transición una señal correspondiente al encendido o apagado de la terminal seleccionada. Como puede verse en el modelo, no es posible cambiar de una posición de elección, a otra posición de elección, por ejemplo de p_2 a p_3, sin llevar el selector primero a la posición de apagado p_1, de otra forma éste actuador estaría modelado como uno de secuencia.



$\Sigma = \{a, b, c, d, e, f\}$
$\lambda(t_1) = a \in \Sigma \quad \lambda(t_4) = d \in \Sigma$
$\lambda(t_2) = b \in \Sigma \quad \lambda(t_5) = e \in \Sigma$
$\lambda(t_3) = c \in \Sigma \quad \lambda(t_6) = f \in \Sigma$

Figura 3.2. Actuador de Elección.

- Paralelos. Son actuadores que presentan una evolución que se sincroniza en todas o en algunas de sus transiciones. Como en el caso de los interruptores dobles. La figura 3.3 muestra un ejemplo de este tipo de actuador y su modelo en RPIT. En éste modelo puede verse que se requieren sólo dos transiciones para representar el encendido y el apagado de un par de interruptores acoplados, los eventos a y b se realizan de manera sincronizada en cada uno de los interruptores, por lo que los dos interruptores estarán en la posición de encendido o en la de apagado, pero núnca uno en cada posición, pues la operación no sería la de un actuador paralelo.



$\Sigma = \{a, b\}$
$\lambda(t_1) = a \in \Sigma$
$\lambda(t_2) = b \in \Sigma$

Figura 3.3. Actuador Paralelo.

Los actuadores pueden ser modelados de forma simple como la mencionada previamente en el texto o de una manera más compleja considerando el comportamiento interno de los mismos, donde el actuador es modelado como una planta.

3.3 Modelado de la planta.

En nuestro caso, la planta es un arreglo, conjunto o colección de componentes relacionados entre sí cuyo objetivo es realizar una serie de operaciones determinadas, con el objetivo de fabricar bienes o cubrir necesidades de una sociedad. La planta puede ser del tipo continuo o discreto de acuerdo a las variables que representan el estado del sistema. Para plantas continuas, una forma propuesta para obtener un modelo discreto es hacer una partición del estado de la planta y denotar cada de clase de equivalencia de dicha partición como un estado discreto de la planta.

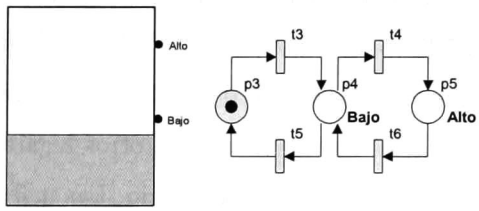


Figura 3.4. Modelo en RPIT de un tanque.

Como ejemplo de este método considere el tanque de la figura 3.4, donde el nivel de fluido fue dividido en tres posibles estados: vacío, bajo y alto en el modelo en RPIT.
Por otro lado, una planta discreta tiene un comportamiento interno que puede ser dividido en pequeños módulos para facilitar el modelado. Cada módulo tiene lugares de entrada y salida que son comunes a los módulos adyacentes. Los módulos pueden ser de los siguientes tipos:

- *Recursos*, estos son las máquinas y herramientas que se encuentran en la planta. Por medio de RPIT, las máquinas de producción, de ensamble, de inspección, máquinas en paralelo, puestas en marcha y descomposturas pueden ser modeladas empleando la técnica mostrada en [Proth_96]. Donde se construye una serie de lugares y transiciones que representan los estados y las operaciones por los cuales atraviesa un producto. Se agregan lugares que representan el estado del recurso y se unen con arcos a las transiciones que

representan las operaciones en que interviene dicho recurso. Finalmente se asigna a cada transición la duración del tiempo que tarda en realizarse la operación que está representando. La Figura 3.5, muestra el modelo en RPIT de una máquina de ensamble, donde un subproducto *C1* y otro *C2* son necesarios para obtener un producto *C3*. Las marcas en el lugar R representan el número y la disponibilidad de recursos, en este caso una máquina.

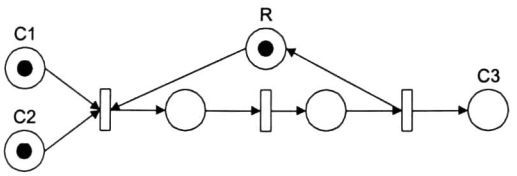


Figura 3.5. Modelo de una Máquina Ensambladora

- *Espacios y control*, se refiere a todo el espacio físico en el sistema. Los almacenes, colas, contenedores, etc. de capacidad limitada pueden ser representados mediante RPIT. En el módulo en RPIT de espacios y control, algunos lugares y transiciones están destinados a representar espacios físicos, en donde las marcas representan los productos almacenados o bien localidades disponibles, pero, otros lugares y transiciones están destinados a establecer la política de control del espacio. Las marcas en éstos lugares, pueden indicar el turno en un almacén tipo cola, o bien el permiso para retirar un producto de cierta localidad del almacén. Usando las técnicas propuestas en [Proth_96] y en [Lopez_97] se representan con lugares las posibles localidades del sistema que pueden retener productos. Con transiciones se representan las operaciones de almacenar, transferir o retirar del almacén un producto y el tiempo que se invierte en realizar dicha operación. Los arcos establecen la relación entre los eventos antes mencionados y las localidades de almacenamiento. El control en los espacios, se usa para retener o liberar productos con cierta política, dependiendo de las relaciones establecidas entre los lugares y transiciones que representan el espacio y los lugares y transiciones de control que se añaden al modelo. Dependiendo de la relación establecida, los almacenes puede ser de tipo LIFO, FIFO, etc. En la Figura 3.6, se presentan dos tipos de almacén, uno tipo LIFO y otro tipo circular, ambos con capacidad de n piezas. Los lugares claros, representan el espacio físico para retener productos, y los lugares sombrados el control de éstos espacios.

- En la Figura 3.6.a, se representa un almacén tipo LIFO. La marca en los lugares sombreados, indica el turno de la localidad que almacenará o proveerá el producto, según se requiera.
- En la Figura 3.6.b, se representa una modelo de un almacén circular. En éste, se tienen dos controles, el primero indica el turno de la siguiente localidad vacía y el segundo apunta a la localidad del siguiente producto que será extraído del almacén.

Como puede verse en los casos particulares, en el modelo se coloca un lugar para cada localidad destinada a almacenar productos y otro que indica el estado de la localidad (disponible o no disponible). Los lugares pertenecientes al control de los espacios físicos, habilitan o deshabilitan las transiciones de entrada y salida de los lugares que representan el espacio. Es decir, en base a la relación entre los lugares y transiciones que representan el espacio y el control, pueden construirse las diferentes políticas de retención de productos. Para el lector interesado, en [Proth_96], [Silva_85] y [Lopez_97] pueden encontrarse modelos con diferentes políticas de retención de productos.

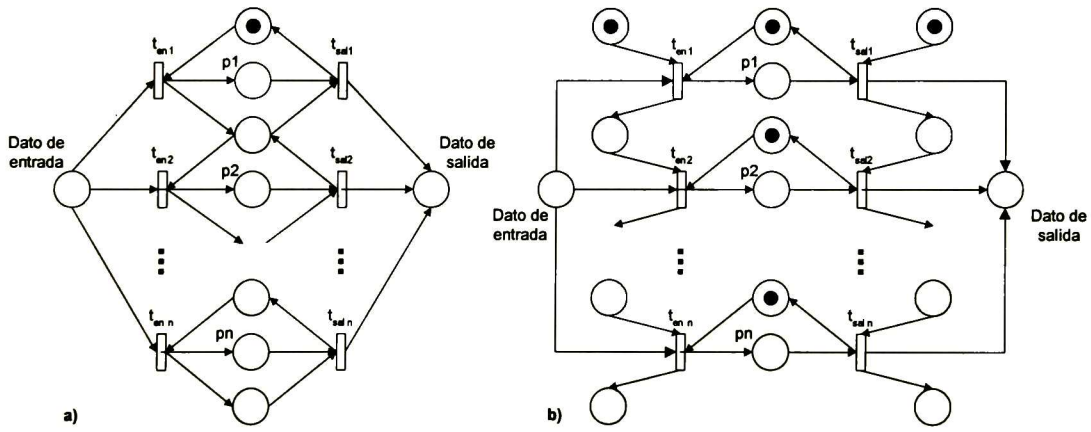


Figura 3.6. Modelos de Almacenes de tipo: a) LIFO y b) Circular.

- *Transportes*, éstos representan todas las formas en que los productos son llevados de un lugar a otro en la planta. Pueden ser bandas transportadoras, robots, vehículos autoguiados, grúas, etc. En la figura 3.7, se muestra como ejemplo el modelo en RPIT de una

banda transportadora.

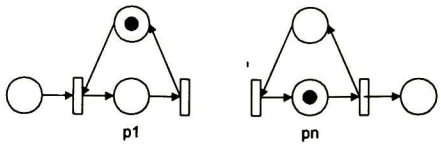


Figura 3.7. Modelo de una Banda Transportadora.

El modelo puede ser refinado para capturar más detalle del sistema de acuerdo a las reglas dadas en [Proth_96] [Silva_85] [Lopez_97].

Finalmente, las marcas son añadidas al modelo, éstas son representan los productos, recursos y espacios libres en los almacenes del sistema.

3.4 Modelado de los sensores.

Las señales de salida de los sensores de la planta están representados en el modelo de RPIT por etiquetas asociadas a ciertos marcados aplicando la función $\varphi_i\,(M_k)$.

3.5 Unión de los modelos aislados.

Para la unión de los modelos aislados se utilizará la tabla 3.1. Ésta captura relaciones de sincronización y causalidad entre los elementos de la planta y los actuadores de la siguiente forma:

Operación	Sincronización	Causalidad
t_i	t_k	
t_j		p_i
⋮	⋮	⋮
t_n	t_l	p_n

Tabla 3.1 Tabla de relaciones del modelo.

- La columna "*operación*" describe todas las operaciones de la planta (i.e. las transiciones del modelo en RPIT de la planta).

- La columna "*sincronización*" contiene sólo transiciones de actuadores. Si en la planta, la "operación" t_i es necesario que se realice al mismo tiempo en que la transición del actuador t_k es disparada, entonces en la columna "sincronización" del renglón de la "operación" t_i debe aparecer t_k. El significado es que en el modelo las transiciones t_i y t_k deben ser fusionadas en una sola.

- Finalmente la columna "*causalidad*" contiene sólo lugares de actuador. Si el lugar p_k del actuador debe estar marcado para que la transición t_i sea disparada. (i.e. mientras el actuador esté en la posición específica p_k la transición t_i puede ser disparada), entonces en la columna "causalidad" del renglón de la "operación" t_i debe aparecer p_k. El significado es que en el modelo el lugar p_k del actuador y la transición t_i de la planta deben ser unidos con un arco doble.

3.6 Ejemplo de Modelado.

Considere el Sistema del Auto Eléctrico de la figura 3.8.

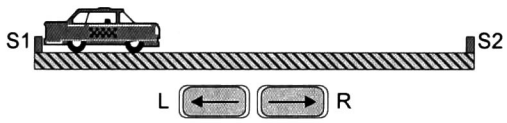


Figura 3.8. Diagrama del sistema del auto eléctrico.

Éste consiste en un auto eléctrico, dos sensores y un par de interruptores de presión. El carro se puede mover de izquierda a derecha o viseversa, activando el sensor *S2* cuando alcanza la posición extrema izquierda o activando el sensor *S1* cuando alcanza la posición extrema derecha. El interruptor L es usado para iniciar el movimiento del carro a la derecha y el interruptor R hace lo mismo para el movimiento a la derecha.

En este sistema los actuadores son los interruptores, los cuales son modelados como se mencionó con anterioridad, asignando un lugar a cada posición activa del interruptor y una transición a cada señal de entrada. El sistema tiene una planta continua que consiste en un riel donde el carro eléctrico se desplaza, éste es discretizado tomando como estados sólo las posiciones extremas y una posición intermedia. El modelo en RPIT de los actuadores y la planta es

mostrado en la figura 3.9.

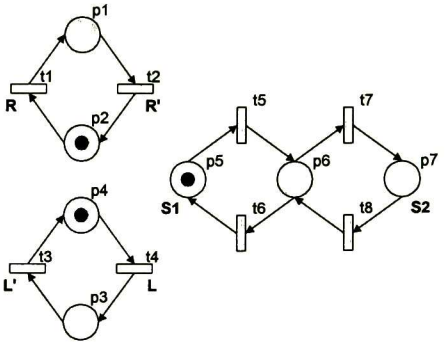


Figura 3.9. Modelos de los actuadores y la planta del sistema.

La unión de los modelos individuales de la planta y los actuadores se hace llenando la tabla descrita en la sección previa, donde las relaciones entre las transiciones de la planta, los estados y eventos de los actuadores son especificadas. La tabla 3.2 enumera las relaciones para el sistema del auto eléctrico.

Operación	Sincronización	Causalidad
t_5		p_1
t_6		p_1
t_7		p_3
t_8		p_3

Tabla 3.2. Tabla de relaciones del sistema.

El modelo completo del sistema del auto eléctrico se muestra en la figura 3.10, donde las transiciones de la planta han sido unidas a los lugares de los actuadores que habilitan su

operación por arcos dobles.

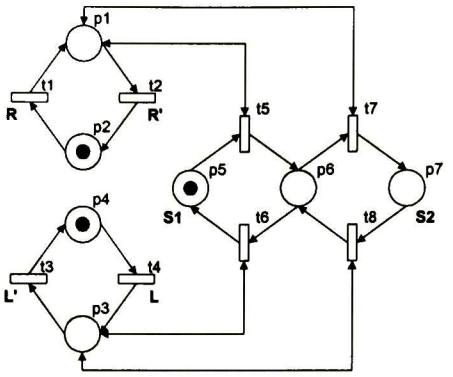


Figura 3.10. Modelo en RPIT del sistema.

Las funciones λ y φ son:

$$\begin{aligned}
&\lambda(t_1) = R && b_j = \varepsilon \quad 1 \leq j \leq 4 \\
&\lambda(t_1) = R' && b_5 = S1 \\
&\lambda(t_1) = L && b_6 = \varepsilon \\
&\lambda(t_1) = L' && b_7 = S2 \\
&\lambda(t_i) = \varepsilon \quad 5 \leq i \leq 8
\end{aligned}$$

Las transiciones manipulables y los lugares medibles son mostrados con color claro en la figura 3.10.

Note que si no existen sincronizaciones, al usar esta metodología se obtiene un modelo en RPIT cuya matriz de incidencia C tiene la forma

$$C = \begin{bmatrix} C_A & 0 \\ 0 & C_P \end{bmatrix}$$

Donde: C_A es la matriz diagonal compuesta por las matrices de incidencia de los modelos de los actuadores. Cuando todas las transiciones de los actuadores tienen una relación de sincronización con los eventos de la planta, el bloque C_A desaparece quedando el bloque C_P aumentado con las filas que describen los lugares de los actuadores. C_P es la matriz de incidencia

del modelo de la planta. La matriz a bloques obtenida, facilita la identificación de los diferentes componentes en su representación matricial, ventaja que será usada en los procedimientos matemáticos presentados en capítulos posteriores.

3.7 Conclusiones

En este capítulo se introdujo una metodología para el modelado de DES por medio de RPIT. En esta metodología se hace una diferenciación entre los lugares y transiciones pertenecientes a los actuadores y a la planta del sistema, con el proposito de obtener una representación matricial que permita la fácil identificación de los componentes del sistema y la simplificación de las operaciones matemáticas debido a la forma de bloques que la matriz de incidencia toma.
Con la forma de modelado propuesta, se facilita el modelado de los DES, pues el modelo global del sistema se construye con la unión de pequeños modelos de cada uno de los elementos del sistema. Evitando emplear etapas posteriores de ajuste en el modelo.
Se establecen también en esta metodología, la sincronización y la causalidad como los tipos de relación entre los actuadores y las operaciones de la planta, con esto la unión entre los modelos individuales se hace de una manera sencilla y preestablecida.

Capítulo 4

Marcado Inicial Mínimo en GMT

Resumen: Este capítulo trata el problema del marcado inicial mínimo (MIM) en Grafos Marcados Temporizados. En este problema, la red y el tiempo de ciclo son conocidos y el problema consiste en encontrar un marcado inicial mínimo M_0 tal que el tiempo de ciclo del GMT sea menor o igual al requerido.

Se presentan dos algoritmos que resuelven el problema del MIM. El primer algoritmo heurístico "MIM_Solver" calcula un subconjunto de p-semiflujos y añade marcas a los lugares en dos pasos. Este algoritmo usa dos heurísticas basadas en los lugares pertenecientes al mayor y al menor número de p-semiflujos respectivamente en cada paso. El segundo algoritmo "ALGEN" explora el espacio de los posibles marcados de un GMT hasta encontrar los que satisfagan las condiciones de tiempo de ciclo requerido π^d y un número de marcas dado. El algoritmo "MIM_Solver" es eficiente ya que requiere poco tiempo para obtener una solución. El GA siempre obtiene un resultado óptimo, aunque su tiempo de convergencia es infinito.

4.1 Introducción

La optimización de recursos y trabajo en proceso WIP (por sus siglas en ingles Work in Process) es uno de los principales problemas en los Sistemas de Manufactura Flexible (SMF) debido a que la solución óptima permite tener bajos costos de producción, poco inventario y también almacenes de proceso pequeños. El problema consiste en calcular el mínimo número de recursos y WIP necesarios para la fabricación de bienes en el sistema con una cierta cadencia de producción.

En términos de RPT, éste problema de optimización se conoce como el problema del Marcado Inicial Mínimo (MIM). Este problema fue tratado por [Onaga_87], quien demostró que éste es un problema NP Completo, desde entonces se han propuesto algoritmos heurísticos para resolver el problema. Por ejemplo, él propone los algoritmos AMIM, AMIMS y AMIM-MG para obtener un número mínimo de recursos para una secuencia específica de eventos.

También [Hillion_89] y [Proth_96] trataron este problema, en su trabajo, ellos proponen una metodología basada en circuitos de trabajo y de comandos para modelar SMF usando RPT. El modelo resultante cae en la subclase de RPT del tipo Grafos Marcados. [Hillion_89] y [Proth_96] proponen el algoritmo heurístico AHA para resolver el MIM en un grafo marcado donde cada lugar puede tener a lo más una marca en el marcado inicial.
[Proth_96] [Onaga_87] mencionan que para resolver el MIM se necesitan todos los p-semiflujos mínimos de la red. También ellos mencionaron que en un grafo marcado fuertemente conexo determinar todos los p-semiflujos mínimos es un problema muy complejo por la existencia de un número combinacional de ellos. Mas aún, si los p-semiflujos mínimos son conocidos, resolver el MIM sigue siendo un problema difícil, pues se debe hacer una minimización entera.
Otra aproximación para resolver este problema para cualquier RPT con un solo T-semiflujo es tratada en [Ramírez_98], la cual se basa en la conversión del comportamiento de la RPT en una función continua y la obtención entonces del valor mínimo para el marcado inicial, desafortunadamente esta aproximación obtiene resultados en los números reales.

En las secciones de este capítulo se define formalmente el problema del MIM y se explican los algoritmos propuestos "MIM_Solver" y "ALGEN" que dan solución al problema, presentando un ejemplo que ilustra su funcionamiento.
El resultado principal de este capítulo es el algoritmo heurístico "MIM_Solver" Éste calcula

un subconjunto de p-semiflujos y añade marcas a los lugares en dos pasos. El primero agrega el mínimo número de marcas necesarias para reducir la diferencia entre el tiempo de ciclo requerido π^d y el tiempo de ciclo π_i de cada p-semiflujo perteneciente al subconjunto calculado. La diferencia $\pi^d - \pi_i > 0$ debe ser mínima. Más tarde, el algoritmo añade el menor número de marcas en los p-semiflujos para satisfacer las restricciones del tiempo de ciclo. En estos pasos, dos heurísticas basadas en los lugares pertenecientes al mayor y al menor número de p-semiflujos respectivamente son usadas.

También en este capítulo se emplea la teoría de los Algoritmos Genéticos para desarrollar el algoritmo "ALGEN" como otro método para dar solución al problema del MIM. Los Algoritmos genéticos (GA) fueron inventados por John Holland y desarrollados por él, sus estudiantes y colegas. Holland publicó en 1975 su libro "Adaptación en Sistemas Naturales y Artificiales" En 1992 John Koza usó un algoritmo genético para desarrollar programas que realizaran ciertas tareas. El llamó a su método "programación genética" (GP). Las soluciones encontradas empleando algoritmos genéticos son consideradas con frecuencia como buenas soluciones, porque rara vez es posible demostrar que realmente es una solución óptima.

4.2 Definición del problema del MIM

El problema del marcado inicial mínimo en GMT se define formalmente como:

Definición 22 *El problema del Marcado Inicial Mínimo (MIM)*
Dados un GMT y un tiempo de ciclo π^d (tiempo de ciclo que se desea tenga el GMT) el problema del MIM se define como encontrar un marcado inicial M_0 (un marcado inicial con el mínimo número de marcas totales $\Sigma M(p), \forall\ p \in P$), tal que el GMT sea vivo y su tiempo de ciclo π satisfaga la condición $\pi \leq \pi^d$.

4.3 Algoritmo "MIM_Solver"

El algoritmo que a continuación se propone es mejor que los algoritmos previos porque puede ser usado en cualquier grafo marcado temporizado. El número inicial de marcas, en cualquier lugar de la red, puede ser mayor que uno y la distribución inicial de marcas es independiente

de la secuencia de disparo de transiciones de la red. La solución se completa en tres fases: a) Cálculo de un subconjunto de p-semiflujos mínimos que cubran toda la red; b) Adición a cada p-semiflujo de igual o menor número de marcas necesarias para satisfacer las restricciones debidas al tiempo de ciclo dado y c) Adición de marcas en los p-semiflujos cuándo éstos las necesiten, mientras el número de marcas en otros p-semiflujos se incrementa lo menos posible.

En un principio, el algoritmo obtiene un número reducido de p-semiflujos; el número de p-semiflujos es igual o menor que $n - rank\,(C)$, es decir, igual o menor al número de elementos en la base del ker $izq\,(C)$. Durante la ejecución del algoritmo, el número de p-semiflujos en el subconjunto se va incrementendo, podría pensarse que el algoritmo enumera todos los p-semifljos de la red, pero esto sólo sucede en el caso de redes pequeñas. El algoritmo propuesto explora sólo un subconjunto de p-semiflujos, evitando la enumeración de todos los p-semiflujos de la red. La adición de marcas es llevada a cabo mediante dos algoritmos heurísticos. El primero calcula el marcado mínimo por p-semiflujo y asigna a lo más esa cantidad de marcas a cada p-semiflujo, tratando de minimizar la diferencia entre las marcas necesarias y las asignadas por p-semiflujo. El segundo calcula el número de marcas que el algoritmo previo no pudo colocar y las asigna en los lugares donde el número de marcas excedentes permanezca lo más bajo posible. Si el tiempo de ciclo de la red es mayor al deseado después de realizar estos pasos, el p-semiflujo más lento es añadido al subconjunto para que sea considerado en la distribución de marcas posterior.

El primer paso del algoritmo propuesto está dedicado a calcular un subconjunto de p-semiflujos. Éste algoritmo está basado en la prueba del teorema 2.43 de [Esparza_95], el cual establece que: a) Todo p-semiflujo semi-positivo es la suma de p-semiflujos mínimos y b) Si una red tiene una p-semiflujo positivo, entonces cada p-semiflujo es una combinación lineal positiva de p-semiflujos mínimos. Las sentencias del teorema se demuestran por el hecho de que el conjunto de todos los p-semiflujos de una RP forman un espacio vectorial.
Elcodigo del algoritmo se muestra a continuación.

Algoritmo 23 ***P_SEM*** *(C, vecs)*

Entrada:

$C \leftarrow$ *Matriz de Incidencia*

Salida:

vecs ← *Conjunto de p-semiflujos*

Inicio

$vecs = \{\}$

$p_k = \min \sum_{i=1}^{n} Y_i$

sujeto a

$Y^T \cdot C = 0$

$Y_i \geq 1$

Mientras $\|p_k\| <> 0$

$z = \min \sum_{i=1}^{n} Y_i$

sujeto a

$Y^T \cdot C = 0$

$\sum_{i=1}^{n} Y_i \geq 0$

$Y_i \leq p_k$

$vecs = vecs \cup z$

$p_k = p_k - z$

fin ***Mientras***

fin ***P_SEM.***

El número de p-semiflujos encontrados por este algoritmo es igual o menor que $n - Rank\,(C)$ ya que un p-invariante que cubra toda la red puede ser formado sin necesidad de usar todos los p-semiflujos de una base del ker $izq\,(C)$, note que este algoritmo es de complejidad polinomial.

El siguiente algoritmo iterativo, añade marcas al lugar p_i que pertenece al mayor número de p-semiflujos en la red. Las marcas añadidas nunca exceden el mínimo número de marcas necesarias para satisfacer las restricciones impuestas por el tiempo de ciclo deseado π^d Los lugares marcados y los pertenecientes a los p-semiflujos que completaron las marcas requeridas no son tomados en cuenta en la siguiente iteración, porque el agregar marcas en esos lugares excedería las marcas requeridas en alguno de los p-semiflujos.

El codigo del algoritmo que añade a lo más el mínimo número de marcas necesario por p-semiflujo se muestra a continuación.

Algoritmo 24 ***ADD_TOKENS*** $(vecs, \pi^d, M_0')$

Entrada:

vecs $\leftarrow$ *Conjunto de p-semiflujos*

$\pi^d \leftarrow$ *tiempo de ciclo requerido*

Salida:

$M_0' \leftarrow$ *marcado parcial*

Inicio

para $i = 1$ *hasta* $|vecs|$ *hacer*

$\gamma_i = \lceil \frac{C^- \Pi_i \cdot D}{\pi^d} \rceil$; *el marcado mínimo del p-semiflujo* Π_i *para satisfacer las restricciones de tiempo*

Repetir

$p_i =$ *(el lugar desmarcado que pertenece al mayor número de p-semiflujos)*

$\Omega_i = \{x \mid x \in vecs \wedge p_i \in x\}$; *es el conjunto de p-semiflujos conteniendo al lugar* p_i

$\alpha_i = \min\{\gamma_k - \sum_{p_j \in \Pi_k} M(p_j)\}$

añadir α_i *marcas a* p_i

hasta que α_i *es cero para todos los lugares*

fin ***ADD_TOKENS***

Cuando el algoritmo previo termina, puede darse los casos:

a) El tiempo de ciclo de la red ya es igual o menor que el deseado. En este caso el marcado encontrado ya es el óptimo.

b) Algunos p-semiflujos del conjunto *vecs* pueden tener marcas, pero no las suficientes para satisfacer las restricciones de tiempo, o

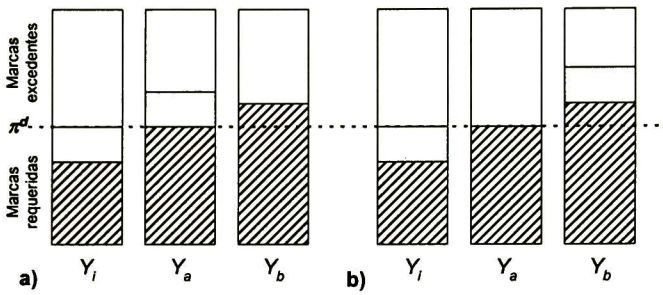
c) Algunos p-semiflujos del conjunto *vecs* permanecen desmarcados.

Los problemas en b) y c) se presentan porque el tiempo de ciclo no puede ser alcanzado con las marcas añadidas hasta el momento a la red.
Para abordar este problema, se establece la siguiente relación binaria en el conjunto vecs:

$$R = \{(x,y) |\ p_i \in x \wedge p_i \in y,\ \text{donde } p_i \in P \text{ y } x,y \in vects\}$$

Como ésta es una relación no transitiva, entonces cuando se añaden marcas al lugar p_i, los p-semiflujos conteniendo este lugar incrementan su número de marcas, incrementando también, el

número total de marcas de la red y, a su vez, aumentando el promedio de marcas excedentes de los p-semiflujos que contienen a p_i. Para mostrar este hecho asuma que un p-semiflujo $Y_i \in vecs$ necesita k marcas para satisfacer las restricciones del tiempo de ciclo. También asuma que las marcas pueden ser añadidas al lugar p_a o p_b del p-semiflujo Y_i y que p_a y p_b pertenecen también a los p-semiflujos Y_a y Y_b respectivamente. Entonces las k marcas deben ser añadidas a los lugares pertenecientes al menor número de p-semiflujos para mantener el marcado inicial tan bajo como sea posible.



4.1. Política de balanceo de carga.

En el caso de que dos o más lugares pertenezcan al mismo número de p-semiflujos, entonces una política de balanceo de carga es usada para añadir las marcas. La aproximación que se propone aquí está basada en la solución al problema de empacado en contenedores (bin packing). En la figura 4.1, puede verse cómo al agregar marcas en los lugares p_a y p_b los p-semiflujos Y_a y Y_b aumentan su número de marcas excedentes, pero al agregar las marcas en el lugar p_a la varianza entre los excedentes de Y_i, Y_a y Y_b es menor que la varianza obtenida al agregar las marcas en p_b. El siguiente algoritmo añade en p_i, las marcas necesarias para satisfacer las restricciones de tiempo del p-semiflujo Y_i con deficit de marcas; El lugar p_i debe pertenecer al menor número posible de p-semiflujos. En el caso de que dos lugares p_i y p_j pertenezcan al mismo número de p-semiflujos, entonces, la política de balanceo aplicada por el siguiente algoritmo, trata de mantener lo más bajo posible la varianza entre las marcas excedentes de cada p-semiflujo.

Algoritmo 25 ***FILL*** $(vecs, \pi^d, M_0', M_0)$

Entrada:

$vecs \leftarrow$ *Conjunto de p-semiflujos*

$\pi^d \leftarrow$ *tiempo de ciclo requerido*

$M'_0 \leftarrow$ *marcado parcial*

Salida:

$M_0 \leftarrow$ *marcado inicial mínimo*

Inicio

para $i = 1$ *hasta* $|vecs|$ *hacer*

$\lambda i = \Pi_i^T M'_0 - \frac{C^- Y_i \cdot D}{\pi^d}$; *son las marcas sobrantes en el p-semiflujo* Π_i

Repetir

$\Pi_j =$ *(un p-semiflujo con menor numero de marcas o desmarcado)*

calcular el conjunto $\Omega = \{p_k \mid p_k \in \Pi_j$ *y* p_k *pertenece al menor número de p-semiflujos*$\}$; *es el conjunto de lugares candidatos a ganar marcas porque incrementan el número de marcas en un subconjunto mínimo de p-semiflujos*

$\forall\, p_k \in \Omega$ *calcular* $\Theta_i = \{x \mid x \in vecs \wedge p_k \in x\}$; *note que este conjunto nunca es vacío, de otra manera el algoritmo* ***ADD_ TOKENS*** *añadiría las marcas suficientes a* Π_j *para satisfacer las restricciones de tiempo*

calcular el lugar p_k *con el minimo valor de* $\Pi_j \in \Theta_K \max\{\lambda_j\}$; *esto permite el balance de la carga en los p-semiflujos*

añadir una marca a p_k

hasta que *todos los p-semiflujos tengan la cantidad suficiente de marcas para satisfacer las restricciones de tiempo*

fin ***FILL***

Los algoritmos previos puede agruparse en uno solo para resolver el problema del MIM.

Algoritmo 26 ***"MIM_ Solver"*** (N, π^d, M_0)

Entrada:

$N, \pi^d \leftarrow$ *El GMT y el tiempo de ciclo requerido*

Salida:

$M_0 \leftarrow$ *el marcado inicial de la red*

Inicio

1. ***P_ SEM****(C, vecs); calcula un subconjunto de p-semiflujos y lo regresa en el conjunto vecs*

2. ***ADD_ TOKENS***$(vecs, \pi^d, M_0')$; *determina un marcado inicial optimista y lo regresa en* M_0'

3. ***FILL***$(vecs, \pi^d, M_0', M_0)$; *añade marcas en los p-semiflujos del conjunto vecs para satisfacer las restriciones de tiempo y regresa la nueva distribución* M_0

Calcular el p-semiflujo más lento Π

Si *el tiempo de ciclo de* Π *es mayor que el tiempo de ciclo requerido* π^d, *entonces* ***regresar a 2***

en otro caso regresar M_0 *como la solución del MIM*

fin ***"MIM_ Solver"***

Aunque el algoritmo previo es de complejidad NP completo porque podría calcular todos los p-semiflujos, en la práctica esto sólo ocurre cuando el número de p-semiflujos es muy pequeño y pueden ser calculados en un tiempo razonable. La figura 4.2 muestra el desempeño del algoritmo **"MIM_Solver"** en relación a los p-semiflujos expandidos para redes de diferentes tamaños. En el eje x se grafican diferentes GM, en esta ordenada sólo mostramos el número de p-semiflujos del GM en cuestión, y en el eje y aparece el número de p-semiflujos calculados por el algoritmo propuesto. Note que el número de p-semiflujos calculado crece muy lentamente con respecto al incremento en los p-semiflujos de la red. En la misma figura se incluyen dos gráficas para mostrar la variación de los p-semiflujos calculados para las mismas redes a diferentes tiempos de ciclo entre 1 y 24 unidades. Las gráfica representa el menor y mayor número de p-semiflujos

calculados para cada red en el intervalo de tiempos de ciclo mencionado.

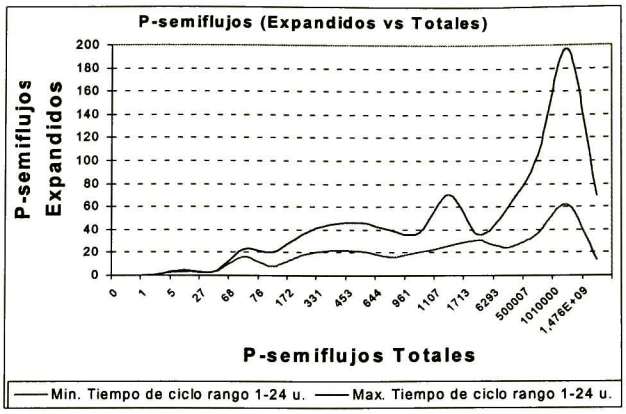


Figura 4.2. P-semiflujos de la red vs. los calculados por el algoritmo MIM_Solver

4.3.1 Ejemplo.del "MIM_Solver"

Ejemplo 27 *Considere la RP de la figura 4.3, donde el tiempo de ciclo requerido es* $\pi^d = 7$.

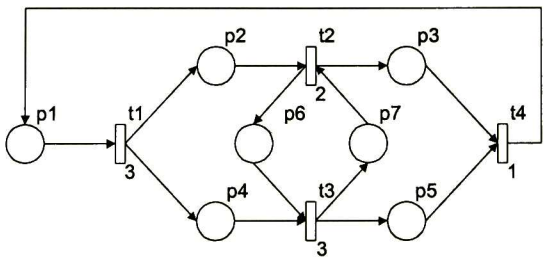


Figura 4.3. Grafo Marcado Temporizado

$$C = \begin{bmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & 0 & -1 \\ 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 1 & -1 & 0 \\ 0 & -1 & 1 & 0 \end{bmatrix} ; \; d = \begin{bmatrix} 3 \\ 2 \\ 3 \\ 1 \end{bmatrix}$$

Usando el algoritmo ***P_SEM****, se obtiene el siguiente conjunto de p-semiflujos:*

$$\begin{aligned}
\Sigma p_i &= \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \\
\Pi_1 &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \\
\Pi_2 &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \\
\Pi_3 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}
\end{aligned}$$

Aplicando el algoritmo ***ADD_TOKENS*** *se obtienen en γ las marcas requeridas por p-semiflujo para satisfacer las resticciones de tiempo y la distribución parcial M_0':*

$$\begin{aligned}
\Sigma p_i &= \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \gamma \\
\Pi_1 &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad 1 \\
\Pi_2 &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad 1 \\
\Pi_3 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad 1 \\
M_0' &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}
\end{aligned}$$

Más adelante, el algoritmo ***FILL*** *es ejecutado, en λ se obtienen las marcas sobrantes por p-semiflujo, si todo $\lambda_i \geq 0$ un marcado M_0 es obtenido sin aplicar la política de balanceo:*

$$\begin{aligned}
\Sigma p_i &= \begin{bmatrix} 2 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \gamma \quad \lambda \\
\Pi_1 &= \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \quad 1 \quad 0.143 \\
\Pi_2 &= \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix} \quad 1 \quad 0 \\
\Pi_3 &= \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \quad 1 \quad 0.286 \\
M_0 &= \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}
\end{aligned}$$

Con la distribución obtenida el GMT tiene un tiempo de ciclo $\pi = 9$, esta distribución no satisface el tiempo de ciclo π^d, entonces el método propuesto continua obteniendo el p-semiflujo mas lento, i.e. el que su tiempo de ciclo es igual a 9 y lo agrega al subconjunto de p-semiflujos. Entonces los algoritmos ***ADD_TOKENS*** *y* ***FILL*** *son ejecutados obteniendo la siguiente*

solución:

$$
\begin{array}{lcccccccc}
\Sigma p_i = & \left[\begin{array}{ccccccc} 3 & 1 & 2 & 2 & 1 & 1 & 2 \end{array}\right] & \gamma & \lambda \\
\Pi_1 = & \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right] & 1 & 0.143 \\
\Pi_2 = & \left[\begin{array}{ccccccc} 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right] & 1 & 0 \\
\Pi_3 = & \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] & 1 & 0.286 \\
\Pi_4 = & \left[\begin{array}{ccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] & 2 & 0.714 \\
M_0' = & \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] & & \\
M_0 = & \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] & &
\end{array}
$$

Nuevamente el tiempo de ciclo del GMT es $\pi = 9$*, entonces un nuevo p-semiflujo es añadido y el mismo procedimiento es ejecutado obteniendo la siguiente solución:*

$$
\begin{array}{lcccccccc}
\Sigma p_i = & \left[\begin{array}{ccccccc} 4 & 2 & 2 & 2 & 2 & 2 & 2 \end{array}\right] & \gamma & \lambda \\
\Pi_1 = & \left[\begin{array}{ccccccc} 1 & 1 & 1 & 0 & 0 & 0 & 0 \end{array}\right] & 1 & 0.143 \\
\Pi_2 = & \left[\begin{array}{ccccccc} 1 & 0 & 0 & 1 & 1 & 0 & 0 \end{array}\right] & 1 & 0 \\
\Pi_3 = & \left[\begin{array}{ccccccc} 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right] & 1 & 0.286 \\
\Pi_4 = & \left[\begin{array}{ccccccc} 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right] & 2 & -0.286 \\
\Pi_5 = & \left[\begin{array}{ccccccc} 1 & 1 & 0 & 0 & 1 & 1 & 0 \end{array}\right] & 2 & 0.714 \\
M_0' = & \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right] & & \\
M_0 = & \left[\begin{array}{ccccccc} 1 & 0 & 0 & 1 & 0 & 1 & 0 \end{array}\right] & &
\end{array}
$$

Con la distribución de marcas M_0 *obtenida el tiempo de ciclo del GMT es* $\pi = 6$*, que es menor que* $\pi^d = 7$*, el tiempo de ciclo requerido.*

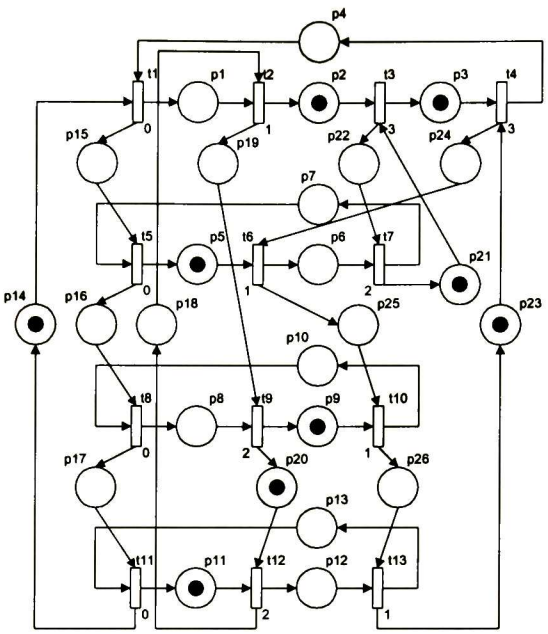


Figura 4.4. Ejemplo tomado de [Hillion_98]

El algoritmo ha sido probado usando varios GMT, por ejemplo es usado en [Hillion_89] (ver figura 4.4), donde el mismo MIM reportado

$M_0 = \begin{bmatrix} 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$

es obtenido calculando menos de 20 p-semiflujos. Este mismo ejemplo pero a tiempos de ciclo menores a 4 unidades de tiempo no pueden ser resueltos por el algoritmo propuesto en [Hillion_89] ya que se requere de mas de una marca en algunos lugares de la Red. Este problema que no se presenta en el algoritmo "MIM_Solver" propuesto.

4.4 Algoritmo "ALGEN"

Para comprobar la optimalidad de las soluciones obtenidas por el algoritmo "MIM_Solver" se desarrolla un algoritmo que resuelva el mismo problema y que su resultado sea óptimo. El algoritmo "ALGEN" se propone como una manera alternativa de solución al problema del MIM. Éste es un algoritmo del tipo genético, el cual siempre converge a una solución óptima, aunque

en un tiempo que puede llegar a ser infinito debido a la naturaleza de este tipo de algoritmos.

Los algoritmos genéticos están inspirados en la teoría de la evolución de Darwin. La cual dice que los individuos más aptos al medio sobreviven, reproduciéndose para crear nuevos individuos cada vez más aptos y los individuos menos aptos desaparecen.

Un Algoritmo Genético (GA Genetic Algorithm) comienza con un conjunto de soluciones (representadas por cromosomas) llamadas población. Algunas soluciones de la población son seleccionadas para formar una nueva población. Esto con la esperanza de que la nueva población sea mejor que la anterior. Las soluciones seleccionadas (padres) para formar las nuevas soluciones (hijos) son elegidas de acuerdo a su conveniencia como solución del problema, la mejor de ellas tendrá más oportunidades para reproducirse.

Esto se repite mientras alguna condición (por ejemplo, un número de generaciones o la conveniencia de la mejor solución) no sea alcanzada.

A continuación se presenta el funcionamiento del Algoritmo Genético Básico:

Algoritmo 28 ***GA_Básico***

1. *[**Inicio**] Generar una población aleatoria de n cromosomas (posibles soluciones al problema).*

2. *[**Aptitud**] Evaluar la aptitud $f(x)$ de cada cromosoma x de la población, si la condición se satisface, parar, y regresar la mejor solución en la población actual.*

3. *[**Nueva población**] Crear una nueva población de posibles soluciones repitiendo los siguientes pasos mientras el l tamaño n de la nueva población es alcanzado.*

   (a) *[**Selección**] Seleccionar dos cromosomas (padres) de la población de acuerdo con su conveniencia (el mejor adaptado de acuerdo al valor obtenido al evaluar $f(x)$, es el que tiene mayor oportunidad de ser seleccionado).*

   (b) *[**Cruza**] Con una probabilidad de cruza dada, se efectua la operación de cruza los padres para formar un nuevo cromosoma hijo (ésta operación se explica más adelante en el capítulo). Si no se hace una cruza, los hijos son una copia exacta de los padres.*

(c) ***[Mutación]*** *Con una probabilidad de mutación dada, mutar los genes de los hijos en cada posición del cromosoma (la operación de mutación se tratará en detalle más adelante en el capítulo).*

(d) ***[Aceptación]*** *Colocar los nuevos cromosomas hijos como miembros en la nueva población.*

4. ***[Remplazo]*** *Usar la nueva población generada con cromosomas hijos para continuar ejecutando el algoritmo.*

5. *Regresar al paso 2.*

El algoritmo genético **"ALGEN"** que aquí se presenta para la solución del MIM, está basado en el algoritmo **GA_básico**, pero en éste se incluyen además otros factores llamados elitismo y una adaptación.
En las siguientes secciones se explican brevemente las partes que conforman el algoritmo genético propuesto.

4.4.1 Codificación del Cromosoma

La codificación, es el primer problema que se presenta al diseñar e implementar un GA, pues de ésta depende en gran medida el buen desempeño del algoritmo en la velocidad de búsqueda de soluciones.
Los cromosomas deben contener información sobre las soluciones a las cuales están representando. La forma más usada para codificar una solución es como una palabra binaria, donde cada bit en esta palabra representa alguna característica de la solución o toda la palabra puede representar un número.
Existen muchas otras formas de codificación. Éstas dependen principalmente del problema que se va a resolver, por ejemplo, se puede codificar directamente números enteros o reales, esto es adecuado para codificar problemas donde intervienen permutaciones.
En el algoritmo propuesto, la codificación binaria es elegida, se asignan para cada lugar del GMT w bits dependiendo de la cota máximo de marcas que se desee para los lugares. Para el GMT de la figura 4.3, con una cota máxima de 3 marcas por lugar, los cromosomas estarán

formados por 14 genes (bits) y se tendrán de la siguiente forma:

$$\begin{array}{lll} \text{Cromosoma 1:} & [01000000000100] & M(x_1) = \left[\begin{array}{ccccccc} 1 & 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right]^T \\ \text{Cromosoma 2:} & [10000000000101] & M(x_2) = \left[\begin{array}{ccccccc} 2 & 0 & 0 & 0 & 0 & 1 & 1 \end{array}\right]^T \end{array}$$

donde 2 bits representan el marcado de un lugar p_i.

4.4.2 Aptitud de un Cromosoma

Una vez que se ha establecido la forma de codificar las soluciones en el GA, la evaluación de la aptitud de cada cromosoma es establecida. En general, la forma de evaluar la aptitud de una solución x es probando una función $f(x)$ que captura los parámetros que se desean optimizar. En el caso del algoritmo **"ALGEN"**, ésta es una doble función de evaluación:

$$\begin{aligned} f_1(x) &= abs\left(\pi^d - \pi_x\right) \\ f_2(x) &= \sum M(p_i), \forall p_i \in P \\ f(x) &= [f_1(x), f_2(x)] \end{aligned}$$

donde:

π^d es el tiempo de ciclo requerido y

π_x es el tiempo de ciclo del GMT con la solución x, si $\pi_x = \infty$ entonces $\pi_x = \sum \delta_i, \delta_i \in d$..

El valor de π_x se obtiene resolviendo el Problema de Programación Lineal [Magott_84]

$$\begin{aligned} &\pi_x = \max_Y Y^T C^- d \\ &\text{sujeto a} \\ &Y^T C = 0 \\ &Y^T M(x) = 1 \\ &Y \geq 0 \end{aligned}$$

que obtiene el tiempo de ciclo del GMT para el marcado $M(x)$ correspondiente a cada cromosoma x.

Cada vez que es obtenida una solución π_x menor o igual a π^d, el algoritmo guarda esta solución

que es remplazada por una más cercana a π^d o por una que tenga el mismo valor π_x pero con un valor menor de $f_2(x)$.

4.4.3 Selección de Cromosomas

Los cromosomas son seleccionados de la población para ser padres en la operación cruza. De acuerdo con la teoría de la evolución de Darwing, los mejores deberán sobrevivir y reproducirse. Existen muchos métodos de selección de los mejores cromosomas, como:

a) La selección de rango, donde son eliminados todos los cromosomas que tengan una aptitud menor a un rango dado del valor deseado;

b) La selección de estado estable donde la mayoria de los cromosomas con aptitud superior permanecen en la población y los hijos de estos sustituyen a los menos aptos.

c) La ruleta de selección, donde a los mejores cromosomas se les asigna una oportunidad proporcional a su aptitud para ser seleccionados.

Ésta última es la más usada en GA y es empleada también en el algoritmo propuesto. Imagine una ruleta donde se encuentran colocados todos los cromosomas de la población, a cada uno se le asigna una superficie mayor o menor de acuerdo a su grado de adaptación. Como se muestra en la figura 4.5, se hace girar la ruleta y una marca selecciona el cromosoma. El cromosoma con mayor aptitud será seleccionado mas veces.

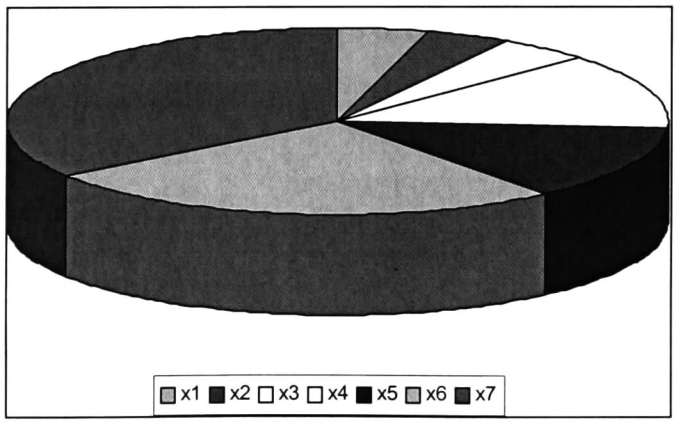


Figura 4.5. Ruleta de selección

El algoritmo propuesto implementa la ruleta de selección de la siguiente manera:

a) Las diferencias $s_i = \pi^d - \pi_x$ son sumadas para obtener un total S.

b) Se asigna a cada cromosoma el intervalo correspondiente en $(0,1)$ de su valor, a x_1 le correspondera el intervalo $\left[0, 1 - \frac{s_1}{S}\right)$, a x_2 el intervalo $\left[1 - \frac{s_1}{S}, 1 - \frac{s_1+s_2}{S}\right)$, y así sucesivamente.

c) Se genera un número aleatorio α entre 0 y 1, y es seleccionado el cromosoma que tenga asignado el rango al que pertenece α.

En el algoritmo genético desarrollado para resolver el problema del MIM, la idea de elitismo también se ha introducido. Elitismo se le dice al método por medio del cual el mejor cromosoma (o los mejores cromosomas) son incluidos directamente en la nueva población. El resto de los cromosomas de la nueva población se genera de la manera clásica, es decir por cruzas o mutaciones. El elitismo puede incrementar rápidamente el desempeño de un GA porque previene la pérdida de la mejor solución encontrada.

4.4.4 Cruza y Mutación de Cromosomas

La *cruza* es una de las operaciones básicas de un GA, ésta toma un par de cromosomas de la selección (los cuales se convierten en cromosomas padres) y crea nuevos cromosomas hijos. La forma más simple de hacer una operación de cruza es:

a) Generar de manera aleatoria un número k en el rango $(0, l)$, donde l es la longitud del cromosoma y k es llamdo *punto de cruza*;

b) Los primeros k genes son copiados del primer padre, el resto de ellos son una copia del segundo padre a partir del punto de cruza.

En la figura 4.6, se muestra de manera esquemática la operación de cruza de cromosomas en un GA.

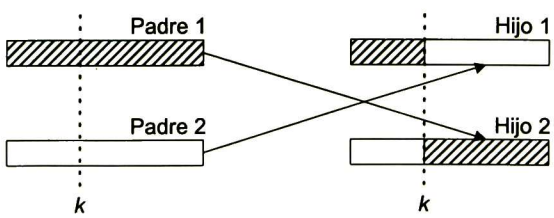


Figura 4.6. Operación de cruza de dos cromosomas.

Existen otras formas de realizar cruzas, por ejemplo se pueden seleccionar más puntos de cruza. Formas específicas de cruza hechas para un cierto problema pueden mejorar el desempeño del algoritmo genético. En el algoritmo propuesto se implementan 3 clases de cruza:

- Con un punto de cruza, que es de la manera más simple de cruza como se explicó con anterioridad, esta operación en los cromosomas se ve de la manera siguiente (el punto de cruza se representa |):

Padre 1:	[010010|00100001]
Padre 2:	[100000|00000101]
Hijo 1:	[010010|00000101]
Hijo 2:	[100000|00100001]

- Con dos puntos de cruza, donde se eligen en forma aleatoria dos puntos de cruza, esto es:

Padre 1:	[010|0100010|0001]
Padre 2:	[100|0000000|0101]
Hijo 1:	[010|0000000|0001]
Hijo 2:	[100|0100010|0101]

- Con puntos de cruza multiples, de igual forma que en las cruza anteriores aquí son elegidos varios puntos de cruza, un ejemplo de este tipo de cruza:

Padre 1:	[010|010|0010|00|01]
Padre 2:	[100|000|0000|01|01]
Hijo 1:	[010|000|0010|01|01]
Hijo 2:	[100|010|0000|00|01]

Estos tipos de cruzas se realizan de manera aleatoria, teniendo todas ellas la misma posibilidad de utilización en el algoritmo.

La *mutación* es la otra operación básica de los GA, ésta previene que todas las soluciones de una población queden atrapadas en un óptimo local de la solución al problema. La mutación

cambia aleatoriamente los genes de un nuevo hijo como se muestra en la figura 4.7.

Figura 4.7. Operación de mutación de un cromosoma.

Para codificación binaria se pueden cambiar aleatoriamente bits escogidos de 1 a 0 o de 0 a 1. La mutación depende de la codificación así como de la cruza, por ejemplo, cuando se codifican permutaciones, las mutaciones pueden ser el intercambio de dos genes. En el algoritmo **"ALGEN"** se implementaron 11 tipos diferentes de mutaciones de cromosomas para aumentar la exploración en el espacio de búsqueda de soluciones. Algunos tipos de mutación se muestran a continuación ($\underline{x_i}$ gen que sufre una mutación):

Hijo 1:	[0100**0**000100101]	Hijo 2:	[**01000000100101**]
Hijo 1':	[0100**1**000100101]	Hijo 2':	[**10111111011010**]
Hijo 3:	[**0100**0000100101]	Hijo 4:	[0**1**0**0**0**0**0**1**0**1**]
Hijo 3':	[**1111**0000100101]	Hijo 4':	[0**0**0**1**0**1**0**1**1**1**0**0**0**0**]

Al igual que las cruzas, todos estos tipos de mutación están uniformemente distribuidos para su aplicación en el algoritmo.

4.4.5 Parámetros de los GA

Enseguida se presentan los principales parámetros de un GA, incluyendo el algoritmo aquí propuesto:

La *probabilidad de cruza* es un parámetro del GA que establece la frecuencia con la que una cruza será hecha. Si no se hace la cruza, los hijos serán una copia exacta de los padres. Si la cruza se lleva a cabo, los hijos son hechos con partes de los cromosomas padres. Si la probabilidad de cruza es 100% entonces todos los hijos están hechos por cruzas. Si ésta es 0% toda la nueva generación está hecha de copias exactas de los cromosomas de la población anterior (pero esto significa que la nueva generación es la misma). Generalmente la probabilidad

de cruza debe ser grande, sobre 85% - 95%. (Sin embargo en algunos casos se ha demostrado que una probabilidad de cruza del 60% es el mejor [Obitko_98])

La *probabilidad de mutación* es otro parámetro del GA. Determina la frecuencia con la cual los cromosomas tendrán una mutación. Si la mutación no se lleva a cabo, los cromosomas son tomados después de la cruza sin ningún cambio. Si se realiza la mutación, parte del cromosoma es cambiado. Si la probabilidad de mutación es del 100% entonces todos los cromosomas sufrirán una mutación, si ésta es del 0% entonces ningún cromosoma sufre una mutación. La mutación está hecha para prevenir la caída del GA en un extremo local, pero esto no debe ocurrir con mucha frecuencia, porque entonces el GA se convertirá en una búsqueda aleatoria. La probabilidad de mutación debe ser pequeña. Los mejores porcentajes reportados son del 0.5% - 1%.

El *tamaño de la población* dice cuántos cromosomas hay en la población (en cada generación). Si hay muy pocos cromosomas, el GA tiene pocas posibilidades de hacer cruzas y sólo una pequeña parte del espacio de búsqueda será explorado. De otra forma, si hay muchos cromosomas, el GA se hace lento. Investigaciones muestran que después de algún limite (el cual depende principalmente de la codificación del problema) no es bueno incrementar el tamaño de la población, porque esto hace lento al algoritmo tardando en encontrar la solución al problema. Un buen tamaño de la población es entre 20 y 30, aunque existen heurísticas que dicen que debe ser igual al número de genes del cromosoma y hasta dos veces este número [Obitko_98].

4.4.6 "ALGEN"

A continuación se presenta el algoritmo propuesto

Algoritmo 29 ***"ALGEN"***$(\pi^d, pops, bt, el, cr, mt, nger, token)$

Entrada:

$\pi^d \leftarrow$ *Tiempo de ciclo requerido.*

$pops, bt \leftarrow$ *Tamaño de la población y bits por lugar.*

$el \leftarrow$ *1 si se aplica el elitismo 0 otro caso.*

$cr, mt \leftarrow$ *Porcentajes de cruza y mutación.*

$nger \leftarrow$ *Número máximo de generaciones.*

$token \leftarrow$ *Número máximo de marcas en el GMT.*

Salida:

$M_0 \leftarrow$ *Marcado inicial mínimo.*

Inicio

$xf = \{\}$; *Cromosoma con la mejor solución.*

Se genera una población aleatoria con pops elementos de longitud $|L| \cdot bt$.

Repetir

Evaluar $f_1(x) \quad f_2(x)$.

$xf =$ *mejor solución entre* $\{xf, x_1, x_2, \ldots, x_{pops}\}$.

Mientras $|npops| < pops$

Si $el = 1$, *entonces copiar* xf *a npops; nueva población*

Seleccionar 2 *cromosomas.*

Realizar cruza con probabilidad cr.

Realizar mutación con probabilidad mt.

fin ***Mientras***

$nger = nger - 1$

cambiar la pobalación actual por la nueva población.

hasta que $\left((nger > 0) \vee \left(\pi^d \neq \pi_x \wedge f_2(xf) > token\right)\right)$

fin ***"ALGEN"***

Ejemplo 30 *Aplicando el algoritmo* ***"ALGEN"*** *al GMT de la figura 4.3 en* 3 *ocasiones, para obtener un tiempo de ciclo* $\pi^d = 7$, *se encontraron los siguientes marcados:*

Marcado	π_x	*Generación*
$\begin{bmatrix} 1 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}^T$	7	5
$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}^T$	7	1
$\begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}^T$	7	1

con los parámetros siguientes:

Tamaño de la población	=	20
Bits por lugar	=	1
Elitismo (0 sin, 1 con)	=	1
Probabilidad de cruza	=	80%
Probabilidad de mutación	=	1%
Número máximo de generaciones	=	30
Número máximo de marcas	=	3

En la figura 4.8 puede verse una gráfica del número mínimo y máximo de generaciones empleadas para encontrar el marcado óptimo en diferentes redes

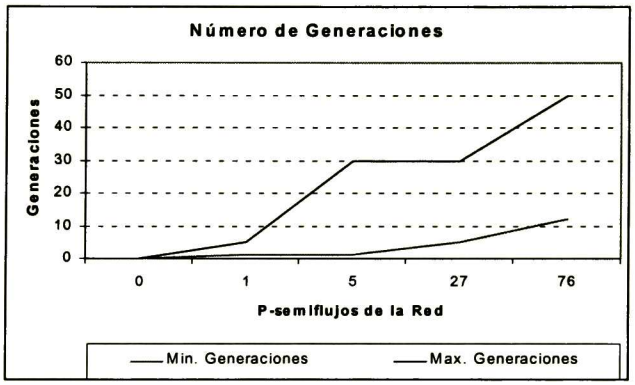


Fig.4.8. Número de Generaciones del ALGEN

4.5 Conclusiones

El problema del MIM consiste en encontrar el mínimo número de marcas en un Grafo Marcado Temporizado necesarias para alcanzar un tiempo de ciclo dado. La solución de este problema es importante porque provee bajos costos de operación en Sistemas de Manufactura Flexible. Se presentaron dos algoritmos que resuelven el problema del MIM. El algoritmo "MIM_Solver" el cual está basado en la enumeración de un subconjunto de p-semiflujos y un algoritmo heurístico de asignación de marcas en los lugares. Aunque el algoritmo "MIM_Solver" pudiera enumerar todos los p-semiflujos, en la práctica esto sólo ocurre para sistemas pequeños como lo muestra la

gráfica mostrada en la figura 4.2. Ejecutando el algoritmo "MIM_Solver" con varios ejemplos (por ejemplo el presentado en [Hillion_89]), el número de p-semiflujos calculados fue razonable y el marcado inicial calculado es el obtenido por otros autores. Este algoritmo es mejor que los algoritmos previos porque puede ser usado en cualquier GMT.

El otro algoritmo presentado está basado en la teoría evolutiva de Darwin. La programación genética a desarrollado algoritmos capaces de encontrar soluciones a problemas NP Completos.

Los Algoritmos Genéticos hacen la exploración de espacios de búsqueda muy complejos en busca de una solución óptima, ya que evitan caer en extremos locales y en base a su fundamento matemático, convergen a una solución óptima, aunque esto sea en el infinito.

Los algoritmos propuestos fueron presentados como un método para optimizar sistemas de manufactura flexible, pero pueden ser usados para optimizar cualquier sistema de eventos discretos modelados por grafos de eventos temporizados.

Capítulo 5

Control de Tiempo Mínimo y Esquema de Regulación

Resumen: El control de tiempo mínimo en DES es un problema NP y esto implica que se requiera un gran tiempo de cómputo para resolverlo.
Una forma de encontrar una ley de control de tiempo mínimo en tiempo razonable es dividir el problema en pequeños subproblemas los cuales resultan más fáciles de resolver.
Este capítulo muestra que si el problema satisface las condiciones para la aplicación de un esquema de regulación, el problema puede ser dividido y entonces el control óptimo resulta fácil de resolver.

5.1 Introducción

La técnica de control por regulación, usa un controlador para restringir el comportamiento del sistema al de otro sistema llamado modelo de referencia o especificación. Se asume que el modelo de referencia es dado como una secuencia de estados que el sistema debe seguir, los cuales son un subconjunto de los estados alcanzables por el sistema principal.

Para un sistema modelado con Redes de Petri Interpretadas y Temporizadas, la solución al problema de regulación particiona el conjunto de transiciones de la red que modela al sistema. El control óptimo trabaja en cada partición de transiciones y obtiene la secuencia óptima que usa sólo transiciones pertenecientes a dicha partición y que alcanza el marcado deseado. Gracias a las particiones que se obtienen de la aplicación del control regulador, se realiza una búsqueda más corta y veloz que la hecha solamente por un control óptimo.

En este capítulo se define el problema de control por regulación y son citadas las condiciones para la existencia del controlador. Se ejemplifica a través de las secciones de este capítulo la técnica de control por regulación y también se propone un método para encontrar el schedule óptimo para el disparo de cada partición de transiciones, el cual consiste en la búsqueda de la secuencia óptima de eventos que deben ocurrir en el sistema para seguir los cambios en el estado de la especificación. Para lograr este objetivo se usa la técnica de control por regulación y el algoritmo de inteligencia artificial A^*.

5.2 Control de tiempo mínimo

Un control de tiempo mínimo tiene como objetivo llevar y estabilizar la salida de un sistema a un estado deseado en el menor tiempo posible. En DES la complejidad de éste problema es NP. Para dar solución al problema del control de tiempo mínimo proponemos dividirlo en pequeños subproblemas más fáciles y rápidos de resolver. Para hacer la división del problema, es necesario conocer qué estados del sistema se desean alcanzar y establecer la secuencia en que estos deberán ser visitados. Es decir, se construye un modelo en el cual se especifica cómo deberá evolucionar el sistema. Como el modelo construido contiene sólo los estados que se desean alcanzar, los eventos de dicho modelo deben corresponder a una secuencia de eventos en el sistema. Para ejemplificar esta idea observe la figura 5.1, donde se muestra cómo un evento

del modelo corresponde a un conjunto de eventos del sistema a través de una cierta función.

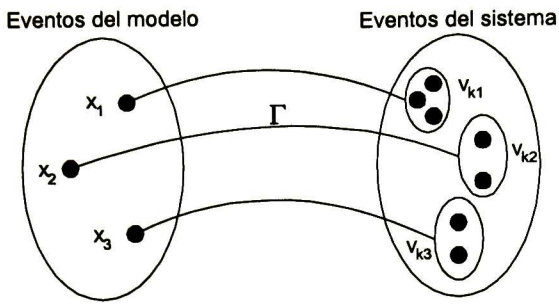


Fig. 5.1. División del problema.

Se deben establecer por medio de una función Γ clases de equivalencia entre los eventos del modelo y los del sistema. Los elementos de cada clase son aquellos eventos que pudieran formar parte de una secuencia que lleve el sistema de un estado deseado a otro. Si se consideran los eventos de una clase de equivalencia a la vez en la búsqueda de la secuencia más corta, el tiempo que se debe emplear en explorar los posibles estados alcanzables se reduce. Es decir, el control de tiempo mínimo para cada conjunto de eventos del sistema puede ser considerado como un subproblema con menor cantidad de eventos. Esto reduce la búsqueda global de la secuencia de eventos más corta a la búsqueda de las secuencias mínimas de cada partición o subproblema.

En forma concreta, los pasos que se proponen para obtener un control de tiempo mínimo en un DES son los siguientes:

1. Establecer un esquema de regulación, se debe tener un modelo de referencia que es el que dicta cómo se comportará el sistema.

2. Obtener la función Γ que relaciona los eventos del modelo y del sistema así como los elementos de cada una de las clases de equivalencia para poder dividir el problema. Como se muestra en el problema de regulación, la función Γ depende de otra función Π que relaciona estados del modelo y del sistema (ver [Ramírez_2000]). En este trabajo se proponen dos algoritmos basados en problemas de Programación lineal para obtener la función Π y los eventos pertenecientes a cada clase de equivalencia.

3. Realizar la búsqueda de la secuencia con menor costo para cada subproblema.

   Para realizar la búsqueda de la secuencia más corta en cada subproblema, se propone en

este trabajo el uso del algoritmo A^* empleando una heurística basada en el p-semiflujo más lento de la red y explorando el grafo de estados diligente [Carlier_88] de la RPIT.

Para formalizar la solución del control de tiempo mínimo, en las siguientes secciones se tratan de una manera más detallada los diferentes pasos mencionados.

5.3 Control por regulación

El control por regulación [Ramírez_2000] es una técnica que se usa para el control de DES. En esta técnica, el lenguaje de salida del sistema debe ser restringido al lenguaje descrito por una determinada RPI llamada modelo de referencia. Para lo cual es necesario definir que eventos del sistema pueden ser controlables:

Definición 31 $\forall\, t_i, t_j$ *tal que* $I(p_k, t_i) = 1$ *e* $I(p_k, t_j) = 1$ *entonces si* $\lambda(t_i) \subset \Sigma$. $\lambda(t_i) \neq \lambda(t_j)$. *Si* $\lambda(t_i) = \varepsilon$ *entonces la transición* t_i *no puede ser habilitada o inhabilitada. En este caso, se dice que* t_i *es una transición* incontrolable, *en otro caso es llamada* controlable.

El problema de control por regulación se define formalmente como sigue:

Definición 32 *Problema de control por regulación.*
Dado

$$S_f = \left(N_{S_f}, M_0^{S_f}, \Sigma_{S_f}, \Phi_{S_f}, \lambda_{S_f}, \varphi_{S_f}, d_{S_f}\right)$$

el sistema representado por una RPIT ordinaria, viva y de etiquetado distinto,

$$R_m = \left(N_{R_m}, M_0^{R_m}, \Sigma_{R_m}, \Phi_{R_m}, \lambda_{R_m}, \varphi_{R_m}, d_{R_m}\right)$$

el modelo de referencia. El problema del seguimiento de modelo consiste en encontrar un controlador H:

$$\begin{aligned} H &: R\left(S_f, M_0^{S_f}\right) \times R\left(R_m, M_0^{R_m}\right) \rightarrow \Sigma_{S_f} \\ u_l^n &: = H\left(M_l^{S_f}, M_n^{R_m}\right) \end{aligned}$$

que alimenta la ley de control apropiada a $(S_f, M_0^{S_f})$ *para forzarlo a alcanzar un marcado* $M_0^{'S_f} \in R(S_f, M_0^{S_f})$ *tal que:*

$$\forall w \in \mathcal{L}_{\text{salida}}\left(R_m, M_0^{R_m}\right) \quad w \text{ es controlable en } (S_f, M_0^{'S_f})$$

El problema de seguimiento de modelo consiste en encontrar un controlador, para hacer que el sistema siga las trayectorias del lenguaje de salida del modelo de referencia.

Aun cuando el controlador está definido sobre los lenguajes de salida, el controlador puede ser diseñado usando principalmente la información estructural de S_f y R_m. Note que cuando existe el controlador H, entonces para toda salida del modelo de referencia existe una salida igual en el sistema, i.e. $\exists\ M_k^{S_f}$ tal que $\varphi_{R_m}\left(M_i^{R_m}\right) = \varphi_{S_f}\left(M_k^{S_f}\right)$.

Teorema 33 *El controlador H. [Ramírez_2000]*
Dados

$$\begin{aligned} S_f &= \left(N_{S_f}, M_0^{S_f}, \Sigma_{S_f}, \Phi_{S_f}, \lambda_{S_f}, \varphi_{S_f}, d_{S_f}\right) \\ R_m &= \left(N_{R_m}, M_0^{R_m}, \Sigma_{R_m}, \Phi_{R_m}, \lambda_{R_m}, \varphi_{R_m}, d_{R_m}\right) \end{aligned}$$

dos RPIT, donde S_f *y* R_m *son las representaciones del sistema y el modelo de referencia respectivamente. Donde la ocurrencia de eventos en* S_f *es mayor que la ocurrencia de eventos en* R_m *y las ecuaciones de estado de* S_f *y* R_m *son:*

$$\begin{aligned} S_f &= \begin{cases} M_k^{S_f} = M_0^{S_f} + C \cdot \overrightarrow{v_{k-1}} \\ b_{k_i} = \varphi_{S_f}\left(M_k^{S_f}\right) \end{cases} \\ R_m &= \begin{cases} M_i^{R_m} = M_0^{R_m} + Q \cdot \overrightarrow{z_{i-1}} \\ r_i = \varphi_{R_m}\left(M_i^{R_m}\right) \end{cases} \end{aligned}$$

y la imagen de φ_{S_f} *es igual a la imagen de* φ_{R_m}
Existe el controlador H que resuelve el problema de control por regulación si y solo si existe la función Π, $M_0^{'S_f} \in R(S_f, M_0^{S_f})$ *donde* $(S_f, M_0^{S_f})$ *es controlable con respecto a* $M_0^{'S_f}$ *y* $\forall\ M_k^{S_f} \in R(S_f, M_0^{S_f})\ \exists\ v_k$ *generando una palabra de salida manipulable* $\varphi_{S_f}\left(M_k^{S_f}\right) \ldots \varphi_{S_f}\left(M_q^{S_f}\right)$ *tal*

que:

$$
\begin{aligned}
&a).\ \ \Pi\ M_0^{R_m} = M_s^{S_f} \\
&b).\ \ \forall\, t_i^{R_m} \in T_{R_m}\ \textit{donde}\ M_i^{R_m} \overset{t_i^{R_m}}{\rightarrow} M_{i+1}^{R_m}, \\
&\qquad \Pi\left(Q \cdot t_i^{R_m}\right) = C \cdot v_k,\ d\left(t_i^{R_m}\right) \geq d(v_k)\ \ y \\
&\qquad \varphi_{S_f}\left(M_k^{S_f}\right) \ldots \varphi_{S_f}\left(M_q^{S_f}\right) = \varphi_{R_m}\left(M_i^{R_m}\right) \varphi_{R_m}\left(M_{i+1}^{R_m}\right) \\
&c).\ \ \forall M_i^{R_m} \in R\left(R_m, M_0^{R_m}\right), \varphi_{S_f} \circ \Pi\ \ M_i^{R_m} = \varphi_{R_m}\left(M_i^{R_m}\right)
\end{aligned}
$$

El teorema anterior garantiza la existencia del controlador H si es posible encontrar:

- Una función Π con la cual se puedan crear las clases de equivalencia entre los eventos del modelo de referencia y del sistema.
- Un vector de Parikh v_k para cada clase de equivalencia representando una secuencia de eventos manipulable en el sistema. Con esta secuencia, el sistema debe poder alcanzar, a partir de un marcado dado el siguiente marcado deseado en el sistema.

La prueba de este teorema puede ser encontrada en [Ramirez_2000].

Para poder implementar este esquema de regulación para un DES modelado con RPIT, es necesario tener un controlador H. Por lo tanto, las siguiente sección se enfocará en encontrar la función Π y los elementos para cada clase de equivalencia que forman los vectores v_k que llevan al sistema del estado actual $E_0 = M_k^{S_f}$ al estado deseado $E_F = M_q^{S_f}$ i.e. $E_0 \xrightarrow{\sigma_i} E_F, \overrightarrow{\sigma_i} = v_k$. Se incluye además un algoritmo que complementa cada conjunto de transiciones, pues la secuencia de eventos σ_i puede tener transiciones de los actuadores. Cuando éstos tienen una relación causal con los eventos de la planta, sus transiciones no aparecen en el vector v_k.

5.4 Obtención de la función Π y los vectores v_k

A continuación se propone un método para encontrar la matriz Π y los vectores v_k, que son dos de las condiciones para hacer control usando la técnica de regulación. La Π y los vectores v_k que se obtienen, satisfacen las condiciones mencionadas en el teorema de anterior. Este método está basado en un Problema de Programación Lineal (PPL) que utiliza como variables de decisión los elementos de la matriz Π y de los vectores v_k. En las restricciones del PPL se utilizan como

datos: el marcado inicial, las matrices de incidencia y la función de salida del sistema y del modelo de referencia. Por lo que el PPL es fácilmente programable. El PPL se plantea de la siguiente forma:

$$\begin{aligned} &\min \sum_{i,j}^{|L_{S_f}|,|L_{R_m}|} \pi_{i,j} + v_k d,\ \pi_{i,j} \in \Pi \\ &\text{sujeto a} \\ &\Pi\ \ M_0^{R_m} = M_0^{'S_f} \\ &\Pi \cdot Q \cdot t_i^{R_m} = C \cdot v_k \\ &\varphi_{S_f} \circ \Pi \cdot M_i^{R_m} = \varphi_{R_m} \cdot Q \cdot t_i^{R_m} \end{aligned} \tag{5.1}$$

donde la función objetivo es minimizar la suma de los elementos de la matriz Π y de los retardos de las transiciones que es necesario disparar en v_k. Las restricciones son las condiciones que deben cumplir Π y v_k en el teorema anterior. De esta forma, si el PPL tiene solución para toda $t_i^{R_m}$, sólo resta buscar la secuencia de disparos σ_i para cada conjunto v_k, con lo cual se garantiza la existencia del controlador H. El PPL (5.1) puede no tener solución, siendo en este caso imposible encontrar el controlador H.

Resolviendo el PPL (5.1) para toda $t_i^{R_m}$, se obtiene el vector v_k para cada $t_i^{R_m}$. Cada vector v_k representa una parte del conjunto de transiciones de la planta que deben ser disparadas para alcanzar el marcado equivalente al obtenido en el modelo de referencia. El método anterior obtiene una diferencia de marcado entre dos estados del sistema, cuándo las relaciones entre la planta y los actuadores de un sistema son causales, el problema anterior no obtiene las transiciones de los actuadores que deben ser disparadas. Esto se debe a que los actuadores ya están en la posición que se desea tengan después de cambiar es estado de la planta. Como los conjuntos v_k obtenidos pueden ser sólo una parte de las transiciones de la secuencia σ_i, es necesario complementar cada v_k con las transiciones de dichos actuadores.

La tabla 5.1 ayuda en la obtención del control de tiempo mínimo. Ya que ésta se usa para registrar los resultados parciales obtenidos en la búsqueda de las secuencias mínimas de cada subproblema. En la tabla se registran las aproximaciones que se hacen del conjunto de

transiciones que forman la secuencia σ_i.

Especificación	Paso 1	Paso 2	Secuencia
x1	$t_f, \ldots, t_h$	$t_f, \ldots, t_h$	σ_1
⋮	⋮	⋮	⋮
xn	$t_k, \ldots, t$	$t_k, \ldots, t_h$	σ_n

Tabla 5.1. Elementos de la secuencia de disparo.

Donde:

- La columna "*Especificación*" contiene todas las transiciones del modelo de referencia.
- La columna "*Paso 1*" es llenada con el primer conjunto de transiciones con coeficiente diferente de 0 de cada v_k obtenido al resolver el PPL (5.1).
- La columna "*Paso 2*" es llenada con el conjunto total de transiciones de σ_i, obtenido por el algoritmo "Recorte" que se menciona enseguida.
- Finalmente en la columna "*Secuencia*" se escribe la secuencia mínima obtenida por el método de búsqueda del scheduling óptimo, propuesto en la siguiente sección. La secuencia de eventos global del control de tiempo mínimo es la concatenación de todas las secuencias σ_i ordenadas en forma ascendente.

A continuación se propone un algoritmo que obtiene las transiciones faltantes de la secuencia σ_i, añadiendo las transiciones de los actuadores mediante relación causal, y eliminando de la búsqueda las transiciones que no son necesarias disparar i.e. las que no aparecen en la columna "*Paso 1*" de la tabla 5.1.

5.4.1 Algoritmo de Recorte

Si son conocidas las transiciones que componen la secuencia óptima σ_i, el número de nodos (o estados representados en un grafo) expandidos se reduce. El objetivo de utilizar un algoritmo de recorte de nodos es el de reducir la búsqueda y aumentar la velocidad de ésta. Como ejemplo de lo anterior considere seis interruptores, los cuales pueden tener la posición de encendido y apagado. También suponga que para abrir una puerta de seguridad sólo los dos primeros

interruptores deben estar en la posición de encendido y el resto en la posición de apagado y para cerrar la misma puerta todos los interruptores deben estar en la posición de apagado. Si esto no es conocido, la cantidad de combinaciones que se tienen entre las posiciones de los interruptores igual a $2^6 = 64$. Pero si se sabe que sólo los dos primeros interruptores actuan sobre la puerta, las búsqueda se limita a $2^2 = 4$ combinaciones. Claramente se puede apreciar que eliminar las transiciones que no intervienen en el logro del objetivo reducen y agilizan el tiempo de búsqueda.

El siguiente algoritmo usa como datos, los subconjuntos de transiciones de la planta v_k obtenidos al resolver el PPL (5.1), a partir de éstos completa el conjunto de transiciones de la secuencia σ_i, pues como se dijo con anterioridad, las transiciones de los subconjuntos v_k pertenecen básicamente a la planta, y es posible que algunas de estas transiciones requieran el disparo previo de otra transición de los actuadores para quedar habilitadas.

Este algoritmo es llamado de recorte de nodos porque una vez obtenidos todos los elementos de σ_i, el resto de las transiciones son ignoradas en la búsqueda del schedule óptimo, el algoritmo de recorte funciona de la manera siguiente:

1. Busca las transiciones del actuador que preceden a las transiciones de la planta obtenidas en v_k en la columna "*Paso 1*" de la tabla 5.1. Como primer paso, es necesario separar de v_k las transiciones que pertenecen sólo a la planta mediante la ecuación:

$$t_{planta} = D_{TP} \cdot v_k$$

donde:
$D_{TP} = \begin{bmatrix} 0 & 0 \\ 0 & I_{|T_{CP}|} \end{bmatrix}$ es una matriz cuadrada de tamaño $|L|$ e $I_{|T_{CP}|}$ (la matriz identidad de tamaño $|T_{CP}|$)

Una vez que las transiciones de la planta que forman parte de la secuencia σ_i son conocidas, se obtienen los lugares que preceden a estas transiciones, para ésto empleamos la matriz de incidencia previa C^- en la ecuación:

$$p_{pred} = \left(D_{LA} \cdot C^- \cdot t_{planta}\right)^T$$

donde:

$D_{LA} = \begin{bmatrix} I_{|L_{CA}|} & 0 \\ 0 & 0 \end{bmatrix}$ es una matriz cuadrada de tamaño $|L|$ e $I_{|L_{CA}|}$ (la matriz identidad de tamaño $|L_{CA}|$)

Las transiciones de los actuadores que se están buscando son aquellas que al dispararse depositan marcas en los lugares que preceden a las transiciones de la planta, es posible encontrar dicho conjunto de transiciones resolviendo:

$$t_{act} = \begin{pmatrix} p_{pred} \cdot C^{+} & D_{TA} \end{pmatrix}^{T}$$

donde:

$D_{TA} = \begin{bmatrix} I_{|T_{CA}|} & 0 \\ 0 & 0 \end{bmatrix}$ es una matriz cuadrada de tamaño $|T|$ e $I_{|T_{CA}|}$ (la matriz identidad de tamaño $|T_{CA}|$)

Los tres pasos anteriores pueden reducirse a la solución de la ecuación

$$u_k = \left(\begin{pmatrix} D_{LA} \cdot C^{-} & D_{TP} \cdot v_k \end{pmatrix}^{T} \cdot C^{+} \; D_{TA} \right)^{T} + D_{TP} \cdot v_k \tag{5.2}$$

en esta ecuación, se obtiene en u_k las transiciones de la planta contenidas en v_k y de los actuadores cuyo disparo marca los lugares que habilitan el disparo de las transiciones mencionadas de la planta, aprovechando la forma a bloques de la matriz de incidencia del modelo.

2. Complementa los elementos de la secuencia. Como ya se obtuvieron las transiciones que activan los actuadores en la secuencia σ_i, es necesario encontrar qué transiciones desactivan dichos actuadores. Para obtener estas transiciones se añaden como restricciones a un PPL las transiciones del conjunto u_k. El siguiente PPL obtiene un nuevo conjunto

v_k que contiene todas las transiciones de la secuencia σ_i. El PPL tiene la siguiente forma:

$$\begin{aligned} &\min v_k^T \cdot d \\ &\text{sujeto a} \\ &M_0^{S_f} + C \cdot v_k = \Pi \cdot M_0^{R_m} + Q \cdot t_i^{R_m} \\ &x_i = y_i \ \{x, y \mid x \in v_k, y \in u_k \neq 0\} \end{aligned} \tag{5.3}$$

donde el nuevo v_k obtenido contiene las transiciones necesarias para habilitar los eventos de la planta y también las que deben ser disparadas para deshabilitarlos nuevamente. v_k es el conjunto total de transiciones que deben ser disparadas para alcanzar el estado deseado E_F.

3. Se eliminan del modelo del sistema las transiciones que no pertenecen a v_k (ya que ellas no tomarán parte de la secuencia de disparo) y se añade una transición *bypass* de E_F a E_0 para formar un circuito y poder aplicar los algoritmos de tiempo mínimo de [Ramamoorthy_80]. Esto se logra concatenando una columna t_b a la matriz C, el vector t_b es calculado como

$$t_b = \Pi \ \ M_0^{R_m} + Q \cdot t_i^{R_m} - M_0^{S_f} \tag{5.4}$$

y la matriz de incidencia modificada tiene la forma

$$C_n = C \oplus t_b = \left[\begin{array}{cc} C & t_b \end{array} \right]$$

Todo los pasos mencionados previamente se traducen como el algoritmo **RECORTE**:

Algoritmo 34 ***RECORTE*** (v_k, v_k, C_n)

Entrada:

$v_k \leftarrow$ *Vector de transiciones.*

Salida:

$v_k \leftarrow$ *Vector de transiciones.*

$C_n \leftarrow$ *Matriz de Incidencia Modificada.*

Inicio

$$u_k = \left((D_{LA} \cdot C^- \cdot D_{TP} \cdot v_k)^T \cdot C^+ \cdot D_{TA} \right)^T + D_{TP} \cdot v_k$$

$\min v_k^T \cdot d$

s.a.

$M_0^{S_f} + C \cdot v_k = \Pi \cdot M_0^{R_m} + Q \cdot t_i^{R_m}$

$x_i = y_i \ \{x, y \mid x \in v_k, y \in u_k \neq 0\}$

$t_b = \Pi \ M_0^{R_m} + Q \cdot t_i^{R_m} - M_0^{S_f}$

$C_n = C \oplus t_b$

fin ***RECORTE***

5.4.2 Ejemplo de obtención de la función Π y los vectores v_k

Considere el ejemplo del auto eléctrico del capítulo 3 donde la especificación o modelo de referencia es presentado en la figura 5.2. En este caso, la ecuación del modelo de referencia es:

$$M_i^{R_m} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix} \cdot t_i^{R_m}$$

$$r_i = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} M_i^{R_m}$$

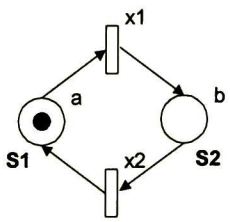


Figura 5.2. Modelo en RPIT de la especificación.

Por otra parte, el marcado inicial del sistema del auto eléctrico es $M_0^{S_f} = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}^T$ para este ejemplo M_0 del sistema tiene una salida equivalente a la salida del marcado inicial de

la especificación $M_0^{R_m} = \begin{bmatrix} 1 & 0 \end{bmatrix}^T$ Con esta equivalencia y las funciones de salida

$$\varphi_{R_m} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \quad \varphi_{S_f} = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

es posible iniciar la solución del problema de seguimiento de modelo, obteniendo la matriz Π y el primer conjunto de transiciones de la secuencia. En este caso el PPL es el siguiente:

$$\begin{aligned} &\min \textstyle\sum_{i,j}^{|L_{S_f}|,|L_{R_m}|} \pi_{i,j} + v_k d,\ \pi_{i,j} \in \Pi \\ &\text{sujeto a} \\ &\Pi \begin{bmatrix} 1 & 0 \end{bmatrix}^T = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}^T \\ &\Pi \cdot Q \cdot t_i^{R_m} = C \cdot v_k \\ &\varphi_{S_f} \circ \Pi \cdot M_i^{R_m} = \varphi_{R_m} \cdot Q \cdot t_i^{R_m} \end{aligned}$$

Donde la matriz Π obtenida es:

$$\Pi = \begin{bmatrix} 0 & 0 \\ 1 & 1 \\ 0 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix}$$

y los vectores v_k obtenidos son:

$$\begin{aligned} x_1 \rightarrow &\begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T \\ x_2 \rightarrow &\begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}^T \end{aligned}$$

la tabla 5.2 de secuencias, es construida para el sistema del auto eléctrico y su especificación

Especificación	Paso 1	Paso 2	Secuencia
x1	t_5, t_7		
x2	t_6, t_8		

Tabla 5.2. Elementos de la secuencia para el sistema del auto eléctrico.

En el sistema del auto eléctrico y la especificación dada, los marcados iniciales tienen una salida equivalente. Entonces $x1$ se dispara en la especificación, cambiando el marcado del modelo de referencia a $M_1^{R_m} = \begin{bmatrix} 0 & 1 \end{bmatrix}^T$ donde la salida de la especificación y del sistema ya no son equivalentes.

Se lleva a cabo entonces el algoritmo **RECORTE**, resolviendo la ecuación (5.2), se obtiene

$$u_k = \begin{bmatrix} 2 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$$

lo que significa que para disparar t_5 y t_7 es necesario disparar t_1 dos veces (como máximo) en el sistema. Complementando la secuencia, se obtiene mediante la ecuación (5.3),

$$v_k = \begin{bmatrix} 2 & 2 & 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}^T$$

Esto es, si t_1 es disparada 2 veces, entonces t_2 deberá hacerlo también con la misma frecuencia. La columna "*Paso 2*" de la tabla de secuencias es llenada con todas las transiciones que tienen un coeficiente diferente de cero en v_k, como se muestra en la tabla 5.3.

Especificación	Paso 1	Paso 2	Secuencia
x1	t_5, t_7	t_1, t_2, t_5, t_7	
x2	t_6, t_8	t_3, t_4, t_6, t_8	

Tabla 5.3. Conjuntos completos de transiciones de la secuencia óptima.

la transición *by pass* t_b añadida a la matriz C obtenida por la ecuación (5.4) es representada por el vector

$$t_b = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{bmatrix}^T$$

Después de la aplicación del algoritmo **RECORTE**, se obtiene la red mostrada en la figura 5.3.

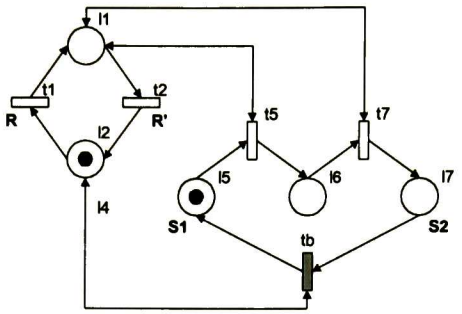


Figura 5.3. Modelo después de usar el algoritmo de recorte.

donde, se han eliminado del modelo las transiciones que no pertenecen a la secuencia σ_i

Como se encontraron para el sistema del auto eléctrico una matriz Π y un subconjuto de transiciones v_k, de acuerdo al teorema de existencia del controlador H, si encontramos las secuencias manipulables que generan los vectores v_k y que hacen que el sistema siga a la especificación, entonces el controlador H existe.

5.5 Schedule óptimo

Esta sección está dedicada a encontrar las secuencias disparables σ_i de los vectores v_k encontrados en la sección previa. Hace uso de un algoritmo que busca entre las posibles secuencias la secuencia de tiempo mínimo.

En la figura 5.4, se muestra el grafo de alcanzabilidad del sistema. Los estados del sistema y la especificación que tienen la misma salida están sombreados. El control por regulación funciona de la forma siguiente:

- Cuando el estado inicial del sistema es diferente al de la especificación, se habilitan los eventos del sistema y éste evoluciona hasta que su señal de salida iguale a la salida de la especificación.

- Cuando las dos señales son iguales, todos los eventos en el sistema son deshabilitados.

Explicamos el algoritmo a través del ejemplo del auto eléctrico, donde el tiempo asignado a las transiciones está dado por

$$d = \begin{bmatrix} 1 & 1 & 1 & 1 & 2 & 3 & 2 \end{bmatrix}^T$$

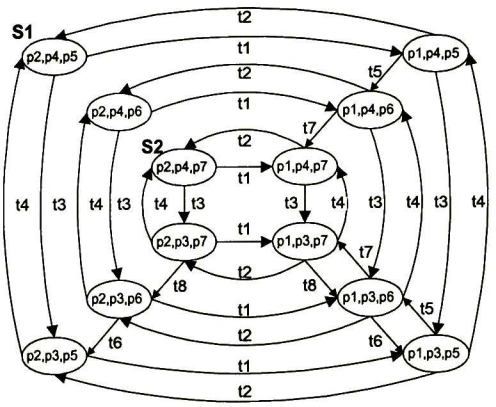


Figura 5.4 Grafo de alcanzabilidad del sistema del auto eléctrico.

En la figura 5.4, es posible ver como secuencias de diferente tiempo de duración alcanzan el nodo $S2$ desde $S1$. Las secuencias presentadas en la tabla 5.4 alcanzan desde $S1$ el estado objetivo $S2$.

Secuencia	Longitud
$t_1, t_3, t_4, t_5, t_2, t_3, t_1, t_7, t_2, t_4$	13
$t_1, t_2, t_3, t_1, t_5, t_7, t_4, t_2$	11
t_1, t_5, t_7, t_2	7
$t_3, t_1, (t_4, t_3)n, t_5, (t_1, t_2)m, t_1, t_7, t_4, t_2$	5 + 2n + 2m

Tabla 5.4. Conjunto de secuencias que alcanzan S2 desde S1.

Se observa que al menos una de ellas es de menor tiempo. Como el problema de control por regulación y el teorema del controlador H no presentan un método para obtener la secuencia de eventos que es necesario disparar en la planta para cada cambio en el estado de la especificación, este trabajo propone un método de búsqueda del schedule óptimo para un DES.

La búsqueda de la secuencia de eventos óptima que deben ser disparados en la planta para obtener la misma salida que la especificación se basa en un algoritmo A^* [Rusell_95]. Éste algoritmo obtiene la ruta óptima entre dos nodos en un grafo. Al realizar el recorte de eventos al modelo del sistema antes de la expansión del grafo (utilizando el algoritmo de recorte de nodos) se mejora la búsqueda del algoritmo A^*.

5.5.1 Búsqueda de la secuencia mínima

Para un nodo E_X dado, el algoritmo A^* hace la expansión de los sucesores inmediatos, seleccionando de entre éstos y los expandidos previamente que no fueron seleccionados, el que tenga el menor valor al evaluar en cada nodo una función guía $f(E)$. $f(E)$ consta de dos partes: una llamada $g(E)$ que contabiliza el tiempo trascurrido a partir del nodo inicial E_0 hasta el nodo E_X y otra llamada $h(E)$, que estima por medio de una función heurística el tiempo que hace falta para llegar al nodo meta E_{F1}. En este trabajo se utiliza "*el grafo de estados diligente*" [Carlier_88] para la búsqueda del schedule óptimo por su característica de mostrar los schedules que no permiten ociosidad en una Red de Petri. La figura 5.5 presenta una representación esquemática de la búsqueda de la ruta óptima.

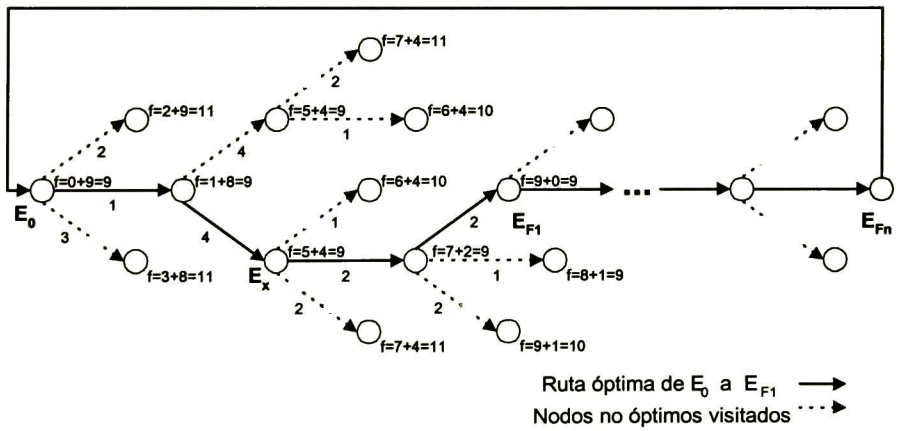


Figura 5.5. Representación esquemática de la búsqueda.

5.5.2 Grafo de Estados Diligente.

El grafo de estados diligente está basado en el grafo de alcanzabilidad de una Red de Petri, esta modificación al grafo de alcanzabilidad, sólo presenta caminos que no permiten ociosidad en el sistema, conocidos como schedules más tempranos y que son caminos con schedules candidatos a ser óptimos [Carlier_88]. El grafo de estados diligente para una red marcada $\Sigma = (N, M_0)$ se define como sigue:

Definición 35 *Grafo de estados diligente.*
El nodo $E_0 = (M_0, 0)$ *es un nodo del grafo de estados diligente.*
Si $E = (M, R)$ *es un nodo del grafo mencionado, entonces con cada subconjunto* τ *de transiciones (incluyendo el conjunto vacío) las cuales satisfacen:*

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

Se asociará un arco (E, E')*, con valor* δ*, etiquetado con* t*; donde* $E' = (M', R'_t)$ *y* δ *están definidas como sigue:*

$$\delta = \min\left\{\min\left\{R_t \mid R_t > 0 \ y \ t \in \tau\right\}, \min\left\{d(t) \mid t \in \tau\right\}\right\}$$

Siendo T' *el conjunto de transiciones definidas como*

$$T' = \{t \mid t \in T \ y \ R_t = \delta\} \cup \{t \mid t \in T \ y \ d(t) = \delta\}$$

Entonces tenemos

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

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

El algoritmo $A*$ emplea el grafo de estados diligente para realizar la búsqueda del schedule óptimo, una ventaja dada por el algoritmo de recorte de nodos, es la reducción del conjunto τ del grafo de estados diligente, porque, para el nodo $E = (M, R)$ el nuevo subconjunto τ está compuesto sólo por las transiciones que satisfacen

$$M - \sum_{t \in \tau} C(\cdot, t) \geq 0 \cap v_k \tag{5.5}$$

evitando expandir nodos de transiciones fuera de la secuencia buscada, reduciendo substancialmente el número de nodos que son explorados por el algoritmo $A*$.

5.5.3 La Función Guía

La función guía $f(E) = g(E) + h(E)$ evalúa el tiempo de la ruta que va desde E_0 hasta E_{F1} a través de E_X, donde $g(E)$ es la suma de los retardos de las transiciones disparadas desde E_0 hasta E_X y $h(E)$ es una función heurística que estima el tiempo restante para alcanzar el nodo objetivo.

La función guía propuesta $f(E)$ está basada en la búsqueda del p-semiflujo con tiempo de ejecución más lento de la red y la adición de una transición *"bypass"* al modelo del sistema. Esta función $f(E)$ es minorante, ya que el tiempo de real de ejecución de la red siempre es igual o mayor que el tiempo de ejecución del valor obtenido por $f(E)$.

La función heurística $h(E)$ hace la búsqueda del P-semiflujo más lento de la red. El valor de la función $h(E)$ se obtiene resolviendo el siguiente PPL:

$$\begin{aligned} & h(E) = \max Y^T \begin{bmatrix} C & t_b & t_c \end{bmatrix} \begin{bmatrix} v_k \cdot d \\ 0 \\ 0 \end{bmatrix} \\ & \text{s.a.} \\ & Y^T \begin{bmatrix} C & t_b & t_c \end{bmatrix} = 0 \\ & Y^T \ M_0^{S_f} = 1 \\ & Y_i \geq 0 \end{aligned} \tag{5.6}$$

Donde t_c es otra transición *"bypass"* de tiempo 0 añadida a la matriz C, t_c va desde el nodo E_0 hasta el nodo E_X y se obtiene de la siguiente manera:

- Los arcos de entrada $Pre\,(p_i, t_c) = M_0^{S_f}\,(p_i)\ \forall p_i \in P$
- Los arcos de salida $Post\,(p_i, t_c) = M_x^{S_f}\,(p_i)\ \forall p_i \in P$

La Figura 5.6 representa de manera esquemática cómo trabaja este método. Puede verse cómo son añadidos caminos de tiempo 0 t_b y t_c para eliminar nodos y reducir el ciclo del modelo y cómo $h\,(E)$ estima el tiempo de ciclo de la red desde E_X hasta E_0.

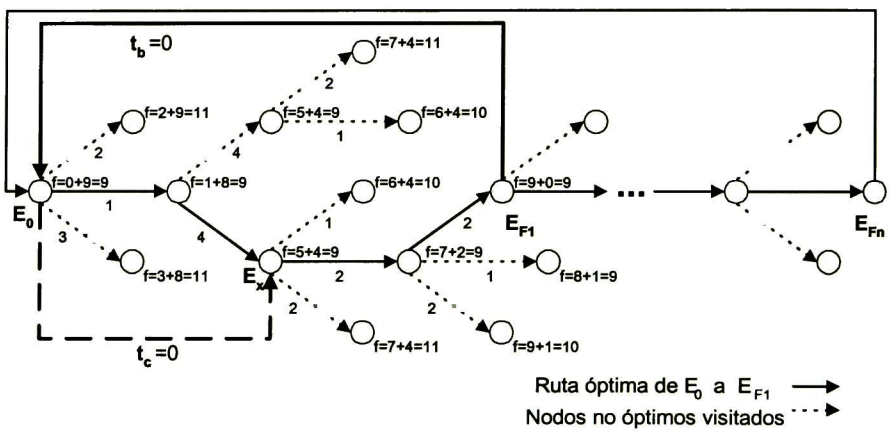


Figura 5.6. Representación de la función guía en el método de búsqueda.

Una vez que el nodo E_{F1} es alcanzado, las transiciones ordenadas de la ruta óptima son escritas en la tabla de secuencias en la columna *"secuencia"*.

El método de búsqueda del control de tiempo óptimo completo, es presentado en el siguiente algoritmo

Algoritmo 36 ***CTMínimo*** *()*

Entrada:

$S_f \longleftarrow$ *Sistema a controlar.*

$R_m \longleftarrow$ *Modelo de refencia.*

Salida:

schedule $\longleftarrow$ *schedule óptimo del sistema.*

Inicio

$E_x \longleftarrow E_0$; *nodo inicial del grafo de estados diligente.*

$tiempo \longleftarrow 0, schedule \longleftarrow \emptyset$

obtener Π *y los vectores* v_k.

Realizar $\forall\ t_i^{R_m}$

CORTE; *Algoritmo para obtener las transiciones de* σ_i *y eliminar las restantes de la búsqueda.*

$t_c = 0$

$E_s \longleftarrow$ *conjunto de sucesores de* E_x *en el grafo de estados diligente.*

Mientras $E_s \neq \emptyset$

tomar un elemento de $e_{si} \in E_s$

agregar a la RP t_c *que alcance* E_X *desde* E_0 *en tiempo* 0.

$$\pi = \max Y^T \begin{bmatrix} C & t_b & t_c \end{bmatrix} \begin{bmatrix} v_k \cdot d \\ 0 \\ 0 \end{bmatrix}$$

s.a. *Obtener el tiempo de ciclo* π_i *de esta red*

$Y^T \begin{bmatrix} C & t_b & t_c \end{bmatrix} = 0$

$Y^T \ M_0^{S_f} = 1$

$Y_i \geq 0$

$\Omega \longleftarrow \Omega \cup \pi_i$

$E_s \longleftarrow E_s - e_{si}$

fin ***Mientras***

$\pi_j \longleftarrow x_i \in \Omega \min \{x_i + tiempo + \ peso\ del\ arco\ [E_X, e_{sj}]\}$

$schedule \longleftarrow schedule \oplus (t_j, tiempo), E_0 [t_j > e_{sj}]$

$tiempo = tiempo + \ peso\ del\ arco\ [E_X, e_{sj}]$

$E_X \longleftarrow e_{sj}$

fin ***Realizar***

fin ***CTMínimo***

5.5.4 Ejemplo de Búsqueda del Schedule Óptimo

Continuando con la obtención del control por regulación de las secciones previas. $E_0 = ([p_2, p_4, p_5], 0)$ es considerado como el nodo inicial del grafo de estados diligente, éste nodo corresponde a los lugares marcados inicialmente en la red y al tiempo residual del disparo de transiciones del sistema. La expansión del grafo de estados diligente comienza evaluando $f(E_0) = 7$, donde $g(E_0) = 0$ y resolviendo el PPL (5.6) con t_c igual al vector de elementos iguales a cero, $h(E_0) = 7$.

Como puede verse en la figura 5.5, a partir de E_0 la única transición que está habilitada de acuerdo con la ecuación (5.5) es t_1. Ésta es disparada y el nuevo nodo $E_X = ([p_1, p_4, p_5], 0)$ es alcanzado, éste tiene un valor $f(E_X) = 7$. La figura 5.7 muestra cómo el método de búsqueda modifica el modelo para obtener el valor de la función guía.

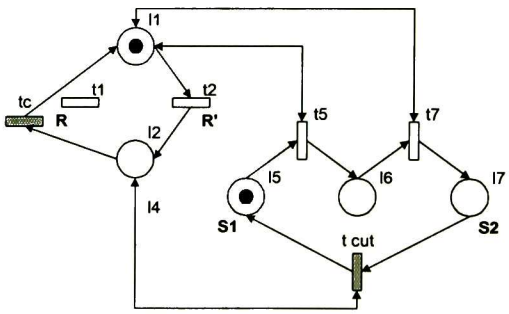


Figura 5.7. Modelo después del disparo de t_1.

A partir de este nuevo marcado E_X, t_2 y t_5 están habilitados por la ecuación (5.5) y el método de búsqueda hará la expansión de dos nodos. Disparando t_2 el nodo alcanzado es $Ex = ([p_2, p_4, p_5], 0)$ el cual tiene un valor $f(E_X) = \infty$ (la razón de esto es claramente visible en la

figura 5.8, donde puede verse que la RPIT del modelo ha perdido su vivacidad).

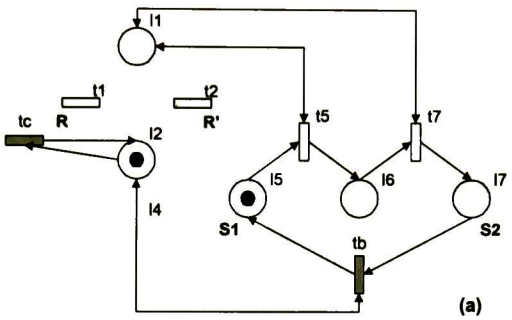


Figura 5.8. Modelo después del disparo de t_2.

Si se dispara t_5 el nodo alcanzado es $E_X = ([p_1, p_4, p_6], 0)$ y el valor de la función guía $f(E_X) = 7$ (en la figura 5.9 se representa el modelo modificado sobre el cual el método de búsqueda evalua la función guía de este nodo).

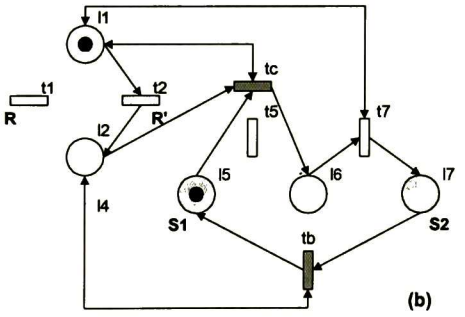


Figura 5.9. Modelo después del disparo de t_5.

De esta forma el grafo es expandido hasta que el nodo E_F es alcanzado, la expansión del grafo hecha por el algoritmo $A*$ para el modelo del auto eléctrico, se muestra en la figura 5.10.

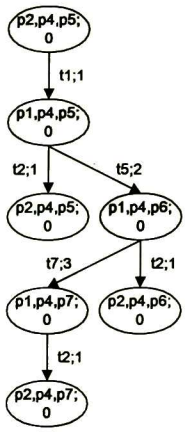


Figura 5.10. Grafo de estados diligente expandido por el algoritmo A*

Como resultado del disparo de $x1$ y $x2$ en la especificación y tras aplicar el método propuesto para hacer coincidir las salidas del sistema y de la especificación, las secuencias óptimas que deben ser disparadas y fueron obtenidas se muestran en la tabla 5.5. Cada una de estas tienen una longitud de 7 unidades de tiempo para el ejemplo del auto eléctrico.

Especificación	Paso 1	Paso 2	Secuencia
x1	t_5, t_7	t_1, t_2, t_5, t_7	t_1, t_5, t_7, t_2
x2	t_6, t_8	t_3, t_4, t_6, t_8	t_4, t_8, t_6, t_3

Tabla 5.5. Secuencias óptimas en el sistema para el disparo de t_i en la especificación.

5.6 Conclusiones

En este capítulo se propone un método para encontrar una ley de control de tiempo mínimo. El método propuesto emplea un esquema de regulación que divide el problema y facilita la búsqueda del schedule óptimo de los eventos de un sistema. El esquema de control por regulación utilizado, es la técnica de seguimiento de modelo [Santoyo_99].

En éste trabajo, se extiende la teoría de seguimiento de modelo al introducir los tiempos de retardo a las transiciones del sistema controlado con esta técnica y al proponer un método que obtiene un schedule óptimo para los eventos que deben ser disparados en el sistema cuando

ocurre un cambio en el estado de la especificación.

Se propone un método que utiliza el algoritmo de inteligencia artificial A^*, el cual expande el grafo de estados diligente para una RPIT usando una función heurística basada en un PPL. La función heurística opera como una función de estimación del tiempo restante en la ruta al objetivo. Con el algoritmo A^*, se asegura la convergencia al schedule óptimo.

Al aplicar el algoritmo de recorte de nodos al sistema, se reduce el número de nodos que son explorados en su grafo de estados diligente por el algoritmo A^*, lo que propicia un menor uso de los recursos de memoria y una reducción en el tiempo de cómputo para resolver el problema.

El método de control de tiempo óptimo presentado en este capítulo no elimina la NP-completitud del problema de scheduling, pero al dividir el problema en una serie de pequeñas búsquedas en un conjunto restringido de nodos, se reduce considerablemente la complejidad del problema.

Conclusiones

Este trabajo presenta a los Sistemas de Eventos Discretos (SED) cuando son modelados utilizando Redes de Petri Interpretadas. En dichos sistemas se abordaron los problemas de balanceo de líneas de montaje y secuenciamiento óptimo de operaciones, ambos pertenecientes al campo del scheduling en DES.

El problema de balanceo de líneas consiste en encontrar el mínimo número de marcas en un Grafo Marcado Temporizado tal que cada una de las líneas tenga una cadencia de producción deseada y fija.

Para este problema se propuso un algoritmo (MIM-Solver), el cual tiene ventajas sobre los ya existentes. Por ejemplo no necesita la enumeración de todos los p-semiflujos de la red, la secuencia de disparo de transiciones no necesita estar fija y además se pueden resolver casos donde el marcado inicial en cada lugar puede ser mayor que uno.

Más aún, este algoritmo presentó un buen desempeño, ya que en los casos peores necesitó tres minutos para encontrar una solución, mientras que los algoritmos ya existentes necesitaron más de quince minutos, aun suponiendo que ya se conocían los p-semiflujos.

Otro algoritmo desarrollado para resolver el mismo problema es ALGEN. Este algoritmo está basado en los algoritmos genéticos, y si bien siempre encuentra una solución óptima, el tiempo de cómputo puede ser demasiado grande para ser utilizado en la práctica. Sin embargo, en este trabajo le dimos la aplicación de validar los resultados encontrados por MIM_Solver.

El problema de secuenciamiento óptimo de las operaciones de un SED se abordó como una combinación de un problema de control por regulación y búsqueda de una secuencia óptima. Este es un esquema es común en sistemas de manufactura, ya que en la gran mayoría de los casos se requiere encontrar la secuencia óptima de entre las que realmente son producitvas.

En este caso el control por regulación nos permitió seleccionar sólo las secuencias productivas y, además, particionar cada secuencia productiva en pequeñas subsecuencias. Esta característica fue explotada por los algoritmos de optimización, ya que la búsqueda de la secuencia óptima se realiza rápidamente por trabajar en conjuntos de secuencias de cardinalidad pequeña.

En un futuro se abordará el problema de balanceo de líneas para sistemas cuyo modelo en

redes de Petri no sea un grafo marcado. Este problema se complica mucho porque las piezas pueden cambiarse de línea de producción y por tanto el WIP en cada línea resulta variable en el tiempo.

También se deben buscar nuevas heurísticas para acelerar la búsqueda en los algoritmos propuestos. Por último se debe realizar un esfuerzo en demostrar que los algoritmos propuestos convergen al óptimo.

Bibliografía

[Basse_88] Sara Baase. "Computer Algorithms Introduction to Design and Analysis". 2nd. Edition. Ed. Addison-Wesley. July, 1988.

[Carlier_88] J.Carlier and Ph Chretienne. "Timed Petri net schedules". G. Rozenberg, editor, Advances in Petri Nets 1988, volumen 340 Lecture Notes in Computer Sciences, páginas 62-84. Springer-Verlag, Berlin, Germany, 1988.

[Córdova_98] Rosa M. Córdova Canela. "Sistema de Pruebas Para Modelado Y Control de Sistemas de Eventos Discretos" CINVESTAV-IPN. Guadalajara, Jalisco, México. Noviembre, 1998.

[Esparza_95] Jörg Desel, Javier Esparza. "Free Choice Petri Nets". Ed. Cambridge University Press, 1995.

[Fanni_98] Alessandra Fanni and Alessandro Guia. "Discrete Event Representation of Qualitative Models Using Petri Nets" *IEEE Transactions on systems, man and cybernetics.* Vol. 28 pp. 770-780. December 1998.

[Goldberg_89] Goldberg, David E. "Genetic Algorithms in Search, Optimization and Machine Learning" Ed. Addison-Wesley Publishing Company, Inc. January, 1989.

[Hanzálek_98] Zdenek Hanzálek. "Petri Net Based Cyclic Scheduling". *IEEE, WODES*, Trnka Laboratory for Automatic Control, Czech Technical University in Prague. 1998.

[Hillion_89] Hervé P. Hillion and Jean-Marie Proth. "Performance Evaluation of Job-Shop Systems Using Timed Event-Graphs" *IEEE Transactions on Automatic Control.* Vol. 34, pp. 3-9. January, 1989.

[López_97] Ernesto López Mellado. "Introducción a Las Redes de Petri, Facultad de Ciencias Físico Matemáticas Universidad Autónoma de Nuevo León". Octubre, 1997.

[Luenberger_89] David G. Luenberger. "Programación Lineal Y No Lineal" 2nd. Edición. Ed. Addison-Wesley. 1989.

[Luger_93] George F. Luger. "Artificial Intelligence: structures and strategies for complex problem solving" 2nd. Edición. Ed. The Benjamin/Cummings. 1993.

[Magott_84] J. Magott. "Performance Evaluation of Concurrent Systems Using Petri Nets". Information Processing Letters. Vol. 18, pp. 7-13. January, 1984.

[Martínez_87] Pedro R. Muro Medrano, Javier Martínez Rodríguez. "Panorámica de Soluciones al Problema de Secuenciamiento o Scheduling en Job-Sops" Informe Tècnico Interno ETSIIZ-GSI-ITI-87-3. Grupo de Ingeniería de Sistemas e Informatica, Departamento de Ingeniería Eléctrica e Informática, UNIVERSIDAD DE ZARAGOZA. Julio 1987.

[Mathur_96] Kamlesh Mathur, Daniel Solow. "Management Science, The art of decision making". Ed. Prentice Hall. 1996.

[Muro_93] P. Muro, A. Ramírez, J. Pujol, and J.L. Villarroel. "A Frame-Rule Based Approach for Petri Nets Integration in F.M.S. Control Models" *Proceedings of the 1993 IMMACS International Workshop on Qualitative Reasoning and Decision Technologies.* Pp. 737-744. June, 1993.

[Obitko_98] Marek Obitko. "Intoduction to Genetic Algorithms" Czech Technical University. http://cs.felk.cvut.cz/~xobitko/ga/ September 1998.

[Onaga_87] Kenji Onaga, Qi-Wei Ge, and Norihiko Ono. "On Optimizing the Initial Token Distribution for a Periodic Petri Net Firing Sequence with Prescribed Firing Numbers" *Transactions of the Society of Instrument and Control Engineers.* Vol. 23-10, pp. 1076-1083. October, 1987.

[Proth_96] Jean-Marie Proth and Xiaolan Xie, "Petri Nets, A Tool for Desing and Management of Manufacturing Systems" Ed. John Wiley & Sons. 1996.

[Ramamoorthy_80] C. Ramamoorthy and G. Ho. "Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets" *IEEE Transactions on Software Engineering.* Vol. 6-5, pp. 440-449. September, 1980.

[Ramchandani_74] Chander Ramchandani. "Analysis of Asynchronous Concurrent Systems by Petri Nets" (Tesis Doctoral). Massachusetts Institute of Technology. February, 1974.

[Ramírez_2000] G. Ramírez, A. Santoyo, A Ramírez and O. Begovich. "A regulation control schema for Discrete Event Systems" *To appear in the proceedings of the IEEE Conference on Systems man and cybernetics*, Nashville, USA. 2000.

[Ramírez_90] A. Ramírez Treviño. "Modelado de Tareas Y Especificación de Controladores En Células Robotizadas" Departamento de Ingeniería Eléctrica, Sección de Control Automático, CINVESTAV. México, D.F., Agosto, 1990.

[Ramírez_92] A. Ramírez, J.L. Villarroel, M. Silva, and P.R. Muro. "Scheduling En Celdas Autónomas de Ensamblaje Basado En un Algoritmo de Tiempo Mínimo En RdP" *Revista de Robótica y Automatizacao.* Pp. 55-60. November, 1992.

[Ramírez_93a] A. Ramírez. "Optimal and Suboptimal Schedules in DEDS Modeled Using Petri Nets". *En Proceedings of the 1993 IEEE International Conference on Systems, Man and Cybernetics.* Pp. 289-294. October, 1993.

[Ramírez_93b] A. Ramírez, J. Campos and M. Silva. "On Optimal Scheduling in DEDS" *En Proceedings of the 1993 IEEE International Conference on Robotics and Automation.* Pp. 821-826. May 1993.

[Ramírez_93c] A. Ramírez. P. Muro and M. Silva. "Modular Composition of Intelligent Control Policies for F.M.S. Models" *En Proceedings of the 1993 IEEE International Conference on Intelligent Control.* Pp. 405-409. August, 1993.

[Ramírez_93d] A. Ramírez Treviño. "Scheduling En Redes de Petri" Tesis Doctoral. María de Luna 3 E-50015, Zaragoza. Universidad de Zaragoza. 1993.

[Ramírez_95] A. Malo-Tamayo, A. Ramírez and E. López-Mellado. "A Petri Based Approach to the F.M.S. Design Problem" IASTED. Pp. 372-375. June, 1995.

[Ramírez_98] A. Ramírez, A. Malo and A. Sanchez. "Optimal Desing of Flowshop Systems Using Petri Nets" *IEE, WODES*, Dept. of Elec. Eng. And Compt. Sci. CINVESTAV-Gdl, 1998.

[Rivera_99] J.I. Rivera, J.S. Rodríguez, G. Ramírez, A. Santoyo. "Aplicación Del Seguimiento de Modelo En Sistemas de Eventos Discretos" CIEE 1999.

[Russell_95] Stuart Russell, Peter Norvig. "Artificial Inteligence: A Modern Approach". Ed. Prentice-Hall, Inc. 1995.

[Santoyo_99] A. Santoyo. "Seguimiento de Modelo en SED usando RPI" Tesis de Maestría, CINVESTAV GDL. Guadalajara, Jalisco México. Julio, 1999.

[Silva_85] Silva, M. "Las Redes de Petri: En la Automática Y En la Informática". Editorial AC. Madrid 1985.

[VanDyke_91] H. Van Dyke Parunak. "Characterizing the Manufacturing Scheduling Problem" *Journal of Manufacturing Systems.* Vol. 10-3 Pp. 241-259. 1991.

[Watanabe_89] Toshimasa Watanabe, Yutaka Mizobata, Kenji Onaga. "Minimum Initial Marking Problems of Petri Nets" *The Transactions of the IEICE.* Vol. 72, pp. 1390-1399. December, 1989.

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: "Optimización en sistemas de eventos discretos temporizados" el día 31 de Agosto de 2000.

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

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

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

Dr. José Javier Ruiz León
Investigador Cinvestav 2 A
CINVESTAV DEL IPN
Guadalajara