UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

OPTIMIZACIÓN DE SISTEMAS DISCRETOS A TRAVÉS DE SIMULACIÓN BASADA EN LA EXPLORACIÓN DEL ÁRBOL DE ALCANCE

TESIS

QUE PARA OBTENER EL GRADO DE:

DOCTOR EN INGENIERÍA

PRESENTA:

MIGUEL ANTONIO MÚJICA MOTA

ASESORES:

DR. MIGUEL ÁNGEL GUTIÉRREZ ANDRADE

DR. MIQUEL ÀNGEL PIERA EROLES

CIUDAD UNIVERSITARIA, 2011.

Universidad Nacional Autónoma de México
UNAM
Biblioteca Central
Dirección General de Bibliotecas de la UNAM

UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Agradecimientos

Primero que nada quiero agradecer a mi tutor y supervisor externo Dr. Miguel Ángel Gutiérrez y Miquel Angel Piera por haberme apoyado incondicionalmente durante estos años de estudio. Quiero agradecer también al resto de los sinodales, Dra. Idalia Flores, Dr. Javier Ramírez y Dr. Stanislaw Raczynski que gracias a sus sugerencias pudieron mejorarse las ideas presentadas en este trabajo de investigación. A todo el cuerpo administrativo del posgrado de ingeniería de la UNAM que me ayudaron siempre en todos los trámites y requerimientos que tuve que solventar para poder dar fin a este proyecto. Y al proyecto PAPIME PE 102709, en el cual he participado activamente.

Le agradezco a mi padre por darme el ejemplo siempre de esforzarme duro para obtener mis metas. A mi madre por demostrarme que se puede vencer la adversidad por muy dura que parezca.

Quiero dedicar esta tesis a mis padres y mis hermanos que siempre me han motivado a esforzarme día a día para lograr vencer el reto que significa ser mejor persona.

Miguel A. Mújica Mota
México D.F. 6 de junio 2011

Prefacio

Actualmente debido a la alta competitividad existente en el mercado la necesidad de contar con herramientas que permitan dar solución a los problemas industriales en tiempos adecuados ha adquirido gran importancia. La simulación digital es un enfoque reconocido que permite la experimentación con modelos de los sistemas industriales evitando incurrir en los costos económicos que implicaría experimentar con el sistema directamente. Esta experimentación se lleva a cabo desarrollando modelos del sistema original con el objetivo de mejorar los valores de indicadores clave del funcionamiento del sistema. La información obtenida de la experimentación y la metodología misma del enfoque permiten al analista entender las relaciones causa-efecto presentes en el sistema con lo cual es posible tomar decisiones que mejoren el funcionamiento del mismo.

En años recientes la comunidad científica ha realizado investigaciones para poder integrar las características de la simulación digital con técnicas de optimización. En particular se ha venido trabajando con el formalismo de redes de Petri coloreadas tanto temporales como atemporales ya que permiten el modelado adecuado de sistemas concurrentes y permiten entender claramente la causalidad existente en sistemas dinámicos que cambian de manera discreta en el tiempo. El árbol de alcance es una herramienta de análisis que ha sido tradicionalmente utilizada para evaluar las propiedades de los modelos y determinar de esa manera el comportamiento del sistema modelado. Debido a sus características es posible utilizarlo como el espacio de búsqueda de configuraciones que resultan de particular interés desde el punto de vista ingenieril y económico siempre tratando de mejorar medidas de funcionamiento como el tiempo total de proceso, retraso total, costos, etc. El uso del árbol de alcance en conjunto con el formalismo de redes de Petri coloreadas permite aprovechar las características descriptivas del formalismo y la capacidad de análisis del árbol de alcance. Si lo anterior se lleva a cabo adecuadamente, es posible desarrollar herramientas de apoyo a la toma de decisiones a nivel industrial que permitan una gestión más eficiente del sistema en estudio.

El trabajo realizado en esta tesis lleva a cabo avances en esta área desarrollando técnicas y algoritmos que permiten generar eficientemente el árbol de alcance de las redes de Petri coloreadas y a su vez analizan el árbol de manera inteligente buscando siempre obtener resultados en tiempos razonables.

Se desarrollaron algoritmos que utilizan dinámicamente las restricciones presentes en los modelos de redes de Petri coloreadas para generar las diferentes configuraciones que puede tener un sistema. Así también se desarrollaron técnicas de gestión de la información que permiten minimizar el espacio físico necesario para almacenar la información obtenida durante la generación del árbol de alcance donde el uso de las simetrías presentes en los modelos resulta clave para reducir el espacio total requerido. Con estas implementaciones se pudo desarrollar un entorno de prueba que permite la implementación de heurísticas para mejorar la exploración y análisis del árbol de alcance cuando se utilizan modelos en redes de Petri coloreadas para sistemas industriales. Todos los desarrollos se validaron a través de la resolución de problemas académicos e industriales interesantes debido a su complejidad.

Contenido

Lista de Figuras

Lista de Tablas

1 INTRODUCCIÓN

En este trabajo se presenta la investigación llevada a cabo para el diseño de algoritmos que servirán como base para el desarrollo de una herramienta de apoyo a la toma de decisiones en operaciones de logística y manufactura haciendo uso del formalismo de redes de Petri coloreadas en conjunto con el árbol de alcance para la generación y análisis del espacio de estados del sistema.

1.1 Simulación de Sistemas

Poder predecir el comportamiento de un *sistema* bajo nuevas condiciones resulta de gran interés en una gran diversidad de áreas. Particularmente en el sector industrial y productivo debido a que se tienen altos costos de operación, resulta impráctico experimentar con el sistema real.

Por motivos económicos, de seguridad y de fiabilidad, se recomienda desarrollar una copia del sistema que reproduzca las características y dinámicas que se consideren más importantes o representativas del mismo, es decir un *modelo* del sistema. El proceso llevado a cabo para obtener la descripción del sistema se denomina *modelado*.

Cuando la experimentación se lleva a cabo con el modelo en un ordenador se denomina *simulación digital*, la cual servirá para entender el comportamiento del sistema o para evaluar diferentes estrategias para la operación del mismo [101].

1.1.1 Tipos de Sistema

Para poder introducir el concepto de modelo de un sistema, se debe especificar que se entiende por sistema.

Definición 1.1 Sistema [45]

Un sistema se define como una colección de objetos o entidades que interactúan entre sí para alcanzar un cierto objetivo.

A su vez, para poder describir las características de interés del sistema en un instante de tiempo, es necesario hacer uso de las llamadas *variables de estado*.

Definición 1.2 Variables de Estado [45]

El conjunto mínimo de variables necesarias para caracterizar o describir todos aquellos aspectos de interés del sistema en un cierto instante de tiempo.

Considerando como finalidad el estudio del comportamiento de un sistema en el dominio temporal, los sistemas pueden clasificarse en: continuos, discretos, a eventos discretos y combinados,

atendiendo solo a la relación entre la evolución de las propiedades de interés y la variable independiente tiempo.

- Sistemas continuos: las variables de estado del sistema evolucionan de modo continuo a lo largo del tiempo.
- Sistemas discretos: se caracterizan en que las propiedades de interés del sistema cambian únicamente en un cierto instante o secuencia de instantes, y permanecen constantes el resto del tiempo. La secuencia de instantes en los cuales el estado del sistema puede presentar un cambio obedece a un patrón periódico.
- Sistemas a eventos discretos: las propiedades de interés del sistema cambian únicamente en una secuencia de instantes de tiempo, se puede considerar que permanecen constantes el resto del tiempo. La secuencia de instantes en los cuales el estado del sistema puede presentar un cambio obedece a un patrón aleatorio.
- Sistemas combinados: aquellos que combinan subsistemas cuyas dinámicas responden a características continuas y discretas.

1.1.2 Tipos de Modelos

Existen muchos tipos de modelos para representar las dinámicas de interés de los sistemas de estudio (modelos físicos, mentales, simbólicos, etc.), en el ámbito de la simulación digital se clasifican de la siguiente manera [45]:

- Modelos Estáticos: se utilizan para representar el sistema en un cierto instante de tiempo, y en su formulación no se considera el avance en el tiempo.
- Modelos dinámicos: este tipo de modelos permite deducir como las variables de estado evolucionan con respecto al tiempo.
- Modelos Deterministas: son aquellos modelos donde los nuevos estados del modelo se definen completamente a partir del estado previo y de sus entradas, por lo que para un conjunto de entradas particular ofrecerá un único conjunto de valores de salida.
- Modelos estocásticos: utilizan una o más variables aleatorias para formalizar las dinámicas de interés del sistema. Por lo que dado un conjunto de entradas particular, no se genera un único conjunto de valores de salida.
- Modelos continuos: se caracterizan por representar la evolución de las variables de interés de forma continua. En general se puede describir a través de ecuaciones diferenciales ordinarias si se considera la evolución de una sola propiedad respecto al tiempo, o ecuaciones en derivadas parciales si se considera la evolución respecto al espacio.
- Modelos discretos: representan la evolución de las variables de estado en instantes de tiempo aleatorios o periódicos.

Un modelo de simulación discreta no siempre es utilizado para modelar un sistema discreto, ni un modelo de simulación continuo para simular sistemas continuos, inclusive los modelos de simulación pueden ser mixtos tanto discretos como continuos. La elección de utilizar un modelo

de simulación continuo o discreto depende principalmente del objetivo del estudio y no de las características del sistema.

1.1.3 Modelos de Simulación a eventos discretos

Atendiendo a la clasificación presentada de los modelos, los modelos a eventos discretos son modelos dinámicos, estocásticos y discretos en los que las variables de estado cambian de valor aleatoriamente en el tiempo. El cambio de las variables de estado en el tiempo se corresponde con la ocurrencia de un evento. Por lo que en este trabajo se define:

Definición 1.3 Evento del modelo

Un evento es una acción instantánea que puede cambiar el estado del modelo.

A continuación se presentan los elementos más significativos de los modelos a eventos discretos [45]

- *Actividades*: son las tareas o acciones que tienen lugar en el sistema, ocurren entre dos eventos.
- *Entidades*: son el conjunto de objetos que constituyen o fluyen por el sistema. Pueden ser temporales o permanentes.
- *Entidades temporales*: son los objetos que se procesan en el sistema. Entidades diferentes pueden tener características diferentes que se denominan atributos.
- *Recursos o entidades permanentes*: son los medios por los cuales se pueden ejecutar las actividades. Los recursos pueden tener características como capacidad, velocidad, o estados en particular.

1.1.4 Modelado de Sistemas Industriales.

Los sistemas industriales son un campo muy fértil para explotar los beneficios del modelado y la simulación digital, en particular presentan muchos factores que hacen el modelado y la simulación aplicable [56]:

- Volumen importante de productos, materiales o información
- Variaciones
- Administración y flujo de recursos
- Necesidad de sincronía global de procesos concurrentes y paralelos
- Complejidad en la medida que los modelos matemáticos existentes son inefectivos o inapropiados
- Muchas decisiones que dependen de la dinámica del sistema y el estatus cambiante de muchas variables del sistema (sistemas a eventos discretos)

En el caso de los procesos de manufactura y logística, las dinámicas de interés para la mejora del rendimiento del sistema pueden modelarse como una secuencia de eventos que permiten conducir al sistema desde un estado inicial a un estado final deseado.

Muchas mejoras a los procesos industriales existentes comienzan con el modelado del proceso actual, el cual es validado contra las condiciones de operación actuales y entonces áreas de mejora potencial sobresalen para posteriormente desarrollar múltiples escenarios de prueba para llevar a cabo experimentación y análisis con el modelo de simulación sin afectar el sistema real. Por lo anterior, es necesario seleccionar adecuadamente el formalismo de modelado con el que se formalizará el modelo industrial de manera que se puedan desarrollar modelos sencillos pero que representen adecuadamente las características de interés al estudio.

Aunque existen otras herramientas para modelar, analizar y evaluar políticas de gestión, únicamente la simulación digital toma en cuenta los detalles complejos de los sistemas industriales aquí mencionados.

1.2 Las Redes de Petri

Las *redes de Petri* están siendo ampliamente utilizadas como formalismo de modelado para sistemas a eventos discretos. En particular resultan adecuados para su aplicación en sistemas industriales debido a que presentan características que modelan adecuadamente los fenómenos dinámicos que se presentan en los sistemas industriales (fenómenos de concurrencia, situaciones de conflicto y paralelismo).

1.2.1 Lugares, transiciones y arcos

Informalmente se puede decir que una Red de Petri (RdP) es una gráfica bipartita dirigida la cual tiene dos tipos de nodos llamados nodos lugar (que de ahora en adelante llamaremos solamente como lugares) y nodos transición (los cuales llamaremos transiciones) conectados entre ellos por arcos dirigidos. Los arcos dirigidos unen *lugares* con *transiciones*, o *transiciones* con *lugares*, pero un arco dirigido nunca une un lugar con un lugar o una transición con otra, ver Figura 1-1. El número de lugares es finito y no cero, la cantidad de transiciones es también finita.

Los nodos lugar se representan gráficamente por círculos y los nodos transición por rectángulos o barras. Los lugares se suelen utilizar para modelar recursos físicos del sistema real así como condiciones lógicas.

Los lugares contienen una o varias marcas representadas gráficamente por puntos (las cuales se denominan comúnmente *tokens*) los cuales se utilizan para representar entidades y sirven para modelar la dinámica del sistema [103].

Las transiciones generalmente están asociadas a eventos que ocasionan un cambio en el sistema real.

Los lugares que están conectados con una transición a través de arcos que parten de los lugares hacia la transición se denominan *lugares de entrada*, y los lugares que están conectados a una transición a través de arcos que parten de la transición se denominan *lugares de salida*.

1.2.2 Definición de Marcado

El llamado *marcado* de una RdP es la información referente al número y tipo de tokens presentes en los lugares del modelo, se puede representar como un vector cuyos componentes tienen valores positivos enteros. La dimensión de este vector es igual al número de nodos lugar, y su n-ésimo componente es igual al número de tokens presentes en el lugar enumerado ***n*** en la RdP. Obsérvese en la Figura 1-1 que cada lugar contiene un numero entero (positivo o cero) de tokens. El número de tokens contenido en un lugar P_i será llamado ya sea M (P_i) o m_i. Para el ejemplo de la Figura 1-1 se tiene $m_1=m_3=1$, $m_6=2$ y $m_2=m_4=m_5=m_7=0$. El marcado M se define como el vector de estas *marcas*[1], es decir M= [m_1,m_2,m_3,m_4,m_5,m_6,m_7]. Por lo que el marcado de la RdP correspondiente a este modelo es M= [1,0,1,0,0,2,0]. El estado del sistema en un instante particular queda definido por el marcado de la RdP y la evolución del estado del sistema corresponde con una evolución en el marcado ocasionado por el *disparo* de las transiciones.

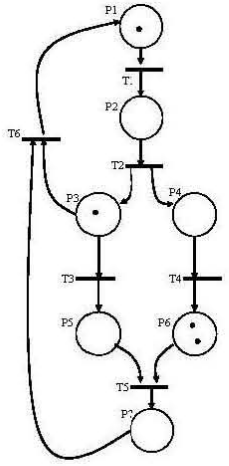


Figura 1-1: Marcado en las RdP

1.2.3 Disparo de las Transiciones

Cualquier arco de la red tiene asociado un peso, el cual es de valor entero y positivo. Cuando el peso no se especifica en los arcos se asume que es igual a uno.

[1] El número de tokens presentes en el lugar

Una transición solo puede ser *disparada* si cada uno de los lugares de entrada a la transición contiene al menos tantos tokens como peso del arco que lo conecta con la transición. En este caso se dice que la transición se encuentra *habilitada*. En las figuras 1.2a, 1.2b y 1.2c se observa que la transición T_1 se encuentra habilitada dado que los lugares de entrada tienen al menos tantos tokens como el peso del arco correspondiente (un token), lo cual no es el caso para la transición de la figura 1-2d.

El disparo de una transición T_j consiste en eliminar tantos tokens de los lugares de entrada como peso de los arcos correspondientes, y añadir tantos tokens a los lugares de salida de la transición como el peso que une cada uno de los arcos a cada uno de los nodos lugar de salida. La figura 1-2 ilustra posibles situaciones que ocurren en el disparo de una transición. En la figura 1-2b se puede observar que hay dos tokens en el lugar P_3 después del disparo ya que existía un token antes del mismo. En la figura 1-2c se puede observar que permanece un token en el lugar P_1 después del disparo.

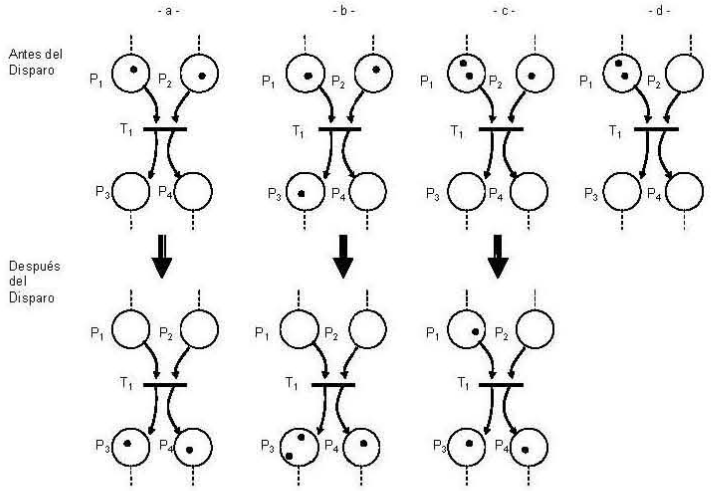


Figura 1-2: El disparo de una transición

1.2.4 Definición: Redes de Petri

Matemáticamente una Red de Petri (RdP) es una 5-tupla, RdP= (P,T,A,W,Mo) donde el significado es el siguiente [103]:

$P= \{P_1, P_2 \ldots P_n\}$ es un conjunto finito de lugares.

$T= \{T_1, T_2, \ldots, Tq\}$ es un conjunto finito de transiciones.

$A = \{A_1, A_2, \ldots, A_r\}$ es el conjunto de arcos de la RdP donde $A \subseteq (PxT) \cup (TxP)$ es un conjunto finito de arcos.
$W: A \rightarrow \{1,2,...\}$ es el peso de la función referenciada a los arcos,
$M_o: P \rightarrow \{0, 1,2....\}$ es la marca inicial.
Nótese que $P \cap T = 0$
Una red de Petri sin marca inicial se denota como N= (P, T, A, W) por lo que también puede definirse RdP = (N, Mo).
Una RdP donde todos los pesos de los arcos son iguales a 1 se dice que es una *Red de Petri Ordinaria.*

A título de ejemplo obsérvese la figura 1-3, donde se presenta una RdP que cuenta con 4 nodos lugar y 5 nodos transición, cuyo marcado es el siguiente vector: [1, 2, 1,1]. En este ejemplo las transiciones T_3 y T_5 no se encuentran habilitadas debido a que los lugares de entrada correspondientes no tienen el número de tokens equivalente al peso del arco que los une (3 y 2 respectivamente).

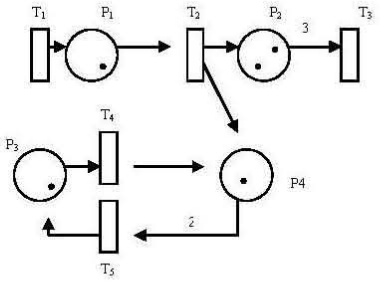


Figura 1-3: Red de Petri ordinaria

Las reglas para el comportamiento dinámico de una RdP son:

- Se dice que una transiciónT_i está *activada* si cada uno de los nodos lugar P_j de entrada al arco contienen al menos W (P_j, T_i) marcas. W (P_j, T_i) representa el peso del arco que une el nodo P_j con la transición T_i. Es decir:

$$M(P_j) \geq W(P_j, T_i).$$

En el caso de una RdP ordinaria, T_i está activada si cada uno de sus lugares de entrada contiene al menos un token.

- El disparo de una transición consiste en:
  Eliminar W (P_j, T_i) tokens de cada P_j que pertenece a los lugares de entrada a la transición.
  Añadir W (P_j, T_i) tokens a cada P_j que pertenece a los lugares de salida.

1.2.5 Ventajas del formalismo de modelado

Una de las características importante de las RdP es su capacidad para representar gráficamente ciertas situaciones que se presentan comúnmente en los sistemas industriales:

1.2.5.1 Paralelismo y Concurrencia

Se dice que dos transiciones son concurrentes si son causalmente independientes (es decir una transición puede ser disparada antes, después o paralelamente con la otra. Lo cual se representa en la Figura 1-4. Se observa que después que la transición T_1 ha sido disparada, dos procesos evolucionan independientemente.

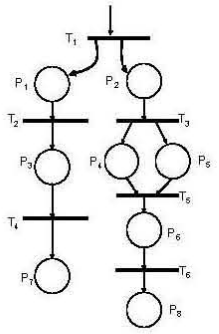


Figura 1-4: Paralelismo y concurrencia

1.2.5.2 Sincronía

La Figura 1-5 representa la sincronía recíproca de dos procesos. El lugar P_1 tiene una marca, pero antes que la evolución de este proceso continúe, se debe esperar hasta que el lugar P_3 también tenga una marca (sincronía), tan pronto como ambos lugares tengan un token, la transición T_1 puede ser disparada provocando el marcado de los lugares P_2 y P_4 para permitir la continuación de la evolución de la RdP.

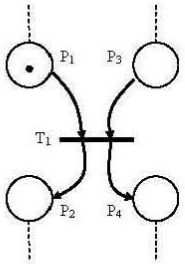


Figura 1-5: Sincronización

1.2.5.3 Compartición de Recursos

En la Figura 1-6 el lugar P_1 modela la disponibilidad de un recurso que puede ser utilizado por el proceso compuesto por las transiciones T_3 y T_5 a partir del disparo de la transición T_3 o por el proceso compuesto por las transiciones T_4, T_6 y T_7 cuando se realiza el disparo de la transición T_4 en condiciones similares, pero debido a que se comparte un recurso no es posible dispararlas simultáneamente. En un caso general, pueden existir muchos recursos idénticos (muchos tokens en P_9), y ambos procesos pueden utilizarlos simultáneamente.

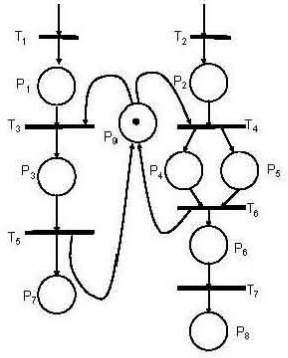


Figura 1-6: Compartición de recursos

Para el estudio de sistemas logísticos y de manufactura, estas propiedades resultan útiles para representar adecuadamente los fenómenos que se llevan a cabo en un entorno industrial.

Ejemplo 1.1
Una máquina que procesa material para dos procesos de manufactura distintos. El recurso compartido (máquina de procesamiento) se utiliza para llevar a cabo un proceso u otro. La figura

1-7 presenta un modelo en RdP donde es necesario decidir a que proceso darle mas prioridad ya que se comparte un recurso utilizado por dos procesos diferentes.

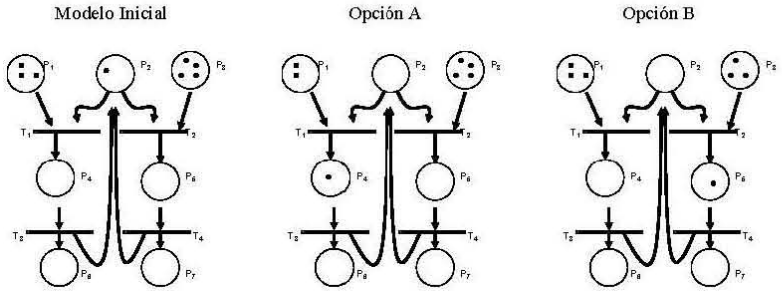


Figura 1-7: Modelo de RdP con recursos compartidos

Se puede observar que se dispone de 2 procesos en paralelo, el primer proceso esta modelado por las transiciones T_1 y T_3, y el segundo proceso por la transición T_2 y T_4, en este caso para que se active la transición T_1 y es necesario que el lugar P_1 y P_2 tengan cada uno al menos un token, y para que se active la transición T_2 es necesario que el lugar P_2 y P_3 tengan cada uno al menos un token, por lo que en este caso ambas transiciones se encuentran activadas, y solamente una de ellas se podrá activar a las vez. La evolución del sistema será disparando la transición T_1 (opción A) o disparando la transición T_2 (opción B).

Cuando se libera el recurso compartido (al disparar ya sea la transición T_3 o la transición T_4) se agregará un token al lugar P_6 o al P_7 y un token al lugar P_2, con lo cual el recurso estará nuevamente disponible para llevar a cabo alguno de los dos procesos.

1.3 Las Redes de Petri Coloreadas

Las *Redes de Petri Coloreadas* (RdPC) poseen características que las hacen muy útiles para llevar a cabo el modelado de sistemas con un nivel de abstracción tal que permite condensar mucha información del sistema modelado en un modelo relativamente sencillo. Presentan características añadidas que son muy útiles en el proceso de modelado:

- Se le pueden añadir atributos a los tokens, los que se denominan de aquí en adelante *colores* y son utilizados para modelar el flujo de información en el proceso.
- Es posible añadir expresiones a los arcos lo cual permite imponer restricciones sobre las características que deben satisfacer las entidades para activar el evento.
- Se pueden imponer restricciones asociadas a la transición, y que restringen el disparo dependiendo de las características del tipo de información asociada a los tokens. Estas restricciones se denominan *guardas*.

Para llevar a cabo el disparo de una transición, es necesario no solo el satisfacer los pesos asociados a los arcos, sino también que las expresiones de arco y las restricciones impuestas por los guardas sean satisfechas.

1.3.1 Definición formal de las RdPC

Matemáticamente una RdPC se define por medio de la siguiente tupla [53]:

$$CPN = (\Sigma, P, T, A, N, C, G, E, I) \tag{1.1}$$

Donde

$\sum = \{C1, C_2, \ldots, C_{nc}\}$ representa el conjunto finito no vacío de colores. Permiten la especificación de los atributos de cada entidad modelada.

$P = \{P_1, P_2, P_3, \ldots, P_{np}\}$ representa el conjunto finito de nodos lugar

$T = \{T_1, T_2, T_3, \ldots, T_{nt}\}$ representa el conjunto de nodos transición, los cuales generalmente están asociados a actividades que consumen tiempo.

$A = \{A_1, A_2, A_3, \ldots, A_{na}\}$ representa el conjunto de arcos dirigidos, los cuales establecen las relaciones entre transiciones y nodos lugar.

N= Es la función de nodos N (A_i), la cual está asociada a la entrada y salida de un arco. Cada uno de ellos debe ser diferente, si uno es un nodo lugar, entonces el otro debe ser un nodo transición y viceversa

C= Conjunto de funciones de color, C (P_i), los cuales especifican para cada nodo lugar el tipo de entidades que pueden ser almacenadas.

$$C(P_i) = C_j \qquad P_i \in P, C_j \in \Sigma$$

G= Función de guardas, asociada a los nodos transición G (T_i), la cual normalmente es utilizada para inhibir el evento asociado con la transición dependiendo del valor de los atributos de las entidades procesadas, por lo cual si los valores satisfacen las expresiones de arco pero no el guarda la transición no se disparará.

E = Expresiones asociadas a los arcos E (A_i). En el caso de los arcos de entrada especifican la cantidad y tipo de entidades que pueden ser seleccionadas de entre los tokens existentes en el lugar para habilitar la transición. Cuando se trata de los arcos de salida, sirven para especificar los nuevos valores de los tokens de salida.

I = Función de inicialización I (P_i), permite la especificación de valores para entidades almacenadas en los nodos lugar al comienzo de la simulación, es decir corresponde al estado inicial del sistema para un escenario en particular.

Una ventaja que presenta el formalismo de RdPC es que se pueden desarrollar modelos más compactos en comparación con las RdP, lo cual permite determinar más fácilmente las relaciones causa-efecto en el sistema modelado.

Ejemplo 1.2 Nivel de abstracción de las RdPC

El modelado con RdPC permite alcanzar un grado de detalle similar al de las RdP, pero con un modelo más compacto. Lo anterior se logra debido a que parte de la información que en un modelo con RdP se modelaría en nodos lugar y nodos transición, en el modelo en RdPC pasa a formar parte de los colores de los tokens.

En la Figura 1-8 se contrastan dos modelos para un mismo sistema, se puede observar que el modelo en RdPC (Figura 1-8b) presenta menos lugares y transiciones que el modelo desarrollado con RdP (Figura 1-8a).

El modelo con RdP requiere 11 nodos (7 lugar y 4 transición) mientras que el modelo desarrollado con RdPC solamente 6 nodos (4 lugar y 2 transición). En el modelo en RdPC se añade un color para representar la información que distingue el tipo de proceso. Se utilizan los lugares P_2' y P_3' junto con las transiciones correspondientes T_1' y T_2' para modelar los procesos paralelos.

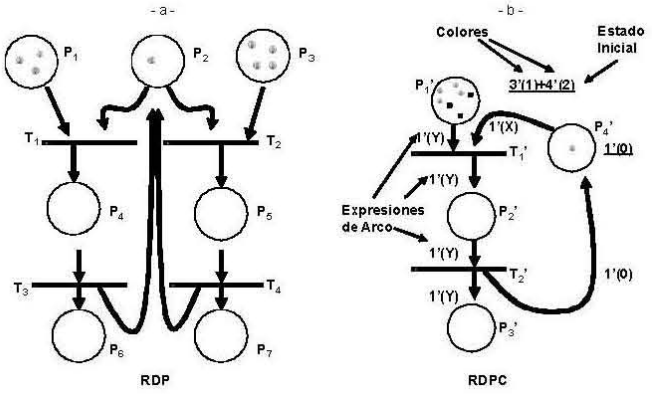


Figura 1-8: Modelado en RdP y RdPC

Las RdPC permiten construir modelos más compactos y paramétricos, lo que facilita considerablemente su mantenimiento y posterior codificación. Si se desarrollarán con el formalismo de RdP se requerirían de estructuras con un número elevado de componentes, por lo que considerando la importancia de la información en la toma de decisiones en los sistemas logísticos así como la ventaja de utilizar representaciones simplificadas se ha considerado el uso del formalismo de RdPC en este trabajo.

1.4 El árbol de alcance

El *árbol de alcance* es una herramienta de análisis que permite almacenar todos los estados que un modelo puede alcanzar a partir de un estado inicial específico. Típicamente ha sido utilizada por

diversos autores [19, 22, 35, 52, 61, 71, 110, 119, 128] para verificar propiedades del modelo en RdPC y de esta manera determinar el comportamiento de la misma. El árbol de alcance es una gráfica dirigida donde los nodos están relacionados por arcos dirigidos donde cada arco representa el disparo de una transición con una combinación de tokens en particular. Por lo que a partir del análisis del árbol es posible determinar la transición o transiciones que se deben disparar y la combinación de tokens para llegar a un estado en particular partiendo de un estado original. Es importante hacer notar que al generar los marcados a partir de un nodo (nodo padre), es posible en ocasiones disparar una misma transición, pero con diferentes combinaciones de tokens, lo que produce dos marcados diferentes (nodos hijos).
Partiendo de un estado en particular, y haciendo uso del árbol de alcance es posible analizar los estados alcanzables a través de los disparos de las diferentes transiciones, así como los tokens que ocasionan cada disparo.
El modelo en RdPC de la Figura 1-9a presenta diferentes estados a partir del disparo de las transiciones activadas. En la Figura 1-9b se ilustran los dos primeros niveles del árbol de alcance en el cual el nodo raíz corresponde al estado inicial y el resto de nodos representan los estados que se alcanzan disparando las transiciones especificadas en los arcos.

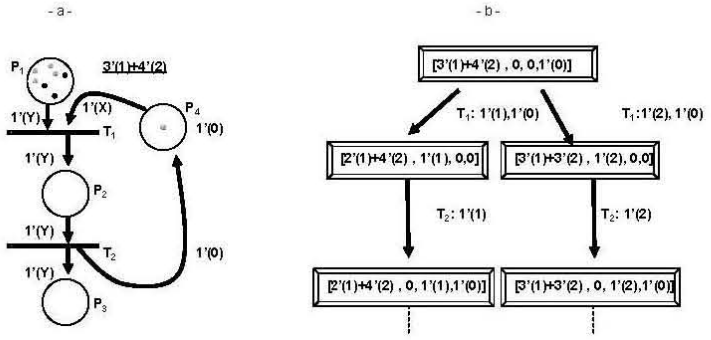


Figura 1-9: El árbol de alcance

La forma como se interpreta el árbol de alcance es que partiendo del nodo inicial [3'(1)+4'(2),0,0,1'(0)], podemos obtener dos nodos hijos. El primero se obtiene llevando a cabo un disparo de la transición T1 [T_1: 1'(1),1'(0)] haciendo uso de un token de color (1) y un token de color (0) de los correspondientes lugares de entrada a la transición. El otro nodo hijo se obtiene haciendo un disparo de la misma transición pero con tokens diferentes [T_1: 1'(2),1'(0)] .

Las reglas que se siguen para construir el árbol de alcance son las siguientes:

- La raíz del árbol representa el marcado del estado inicial del sistema

- Para cada nodo en el árbol se deben generar tantos hijos como combinaciones de tokens que habiliten el disparo de alguna de las transiciones del modelo. Los hijos generados representan el estado de la RdPC una vez que se han disparado las transiciones habilitadas.
- Los arcos que conectan los diferentes estados representan el cambio de estado del modelo debido al disparo de las transiciones con una combinación en particular de tokens.
- Cuando se genera un nuevo hijo cuyo estado ha sido previamente alcanzado (por el disparo de una transición habilitada correspondiente a otro estado distinto) se marca como nodo repetido (*Old*) y no es evaluado nuevamente.
- Cuando un nodo no presenta ninguna transición habilitada (y por lo tanto no puede generar nodos hijos) se denomina *nodo hoja* o marca muerta.
- Un nodo que no ha sido denominado como nodo hoja o nodo old es un nodo *Nuevo* (el cual representa un nuevo estado alcanzado por la RdPC)

1.5 Redes de Petri Coloreadas Temporales

El uso del tiempo en las RdPC permite describir un sistema cuyo funcionamiento es dependiente del tiempo, por lo que resultan adecuados para llevar a cabo el análisis y evaluación del desempeño dinámico de sistemas industriales (el ritmo de producción al que opera el sistema, tiempos de proceso, duración en colas de las entidades, etc.).

Para poder llevar a cabo el análisis del funcionamiento de un sistema es necesario extender el modelo en RdPC con un concepto temporal, utilizando un *reloj global*, y añadiendo *sellos de tiempo* a los tokens. Estas extensiones tienen las siguientes características [53]:

- El valor del reloj global representa el tiempo en que se encuentra el estado particular del sistema modelado (tiempo de simulación).
- El sello de tiempo de los tokens representa el tiempo mínimo global en que es posible utilizarlos para realizar la evaluación de la transición correspondiente y poder desencadenar algún proceso.

1.5.1 Marcados con Sellos de Tiempo

Cuando se modelan RdPC con extensión temporal, para que los tokens puedan habilitar una transición, es necesario que además de cumplir con las reglas de las RdPC atemporales satisfagan una regla asociada a la disponibilidad temporal del token:

- *Para poder utilizar un token en la evaluación de una transición, es necesario que el sello de tiempo asociado al token sea igual o menor que el tiempo global del modelo.*

Haciendo uso de esta regla, es necesario diferenciar el tipo de habilitación del token:

- *Habilitación por Color*. El token se encuentra habilitado debido a que cumple con las restricciones impuestas a los colores por la estructura de la red de Petri atemporal y los guardas de la transición correspondiente (ver sección 1.2.4) .
- *Disponibilidad por Tiempo*. El sello de tiempo asociado al token cumple con la regla de disparo temporal.

Para simular el desempeño temporal de los eventos del sistema, se asocia a las transiciones un tiempo Δt que representa el tiempo de simulación consumido por el evento representado por la transición.
En el caso de ocurrir el disparo de una transición, se asocia a los tokens de salida de la transición un sello de tiempo el cual depende del valor temporal de la transición y del momento del disparo; se calcula por la fórmula:

$$t_s = Gt + \Delta t \tag{1.2}$$

Donde
t_s : representa el sello de tiempo de salida de la transición.
Gt : el reloj global de la simulación.
Δt : es el tiempo asociado con la transición[2].

La formula (1.2) implica lo siguiente:

- Los tokens de salida de la transición no estarán disponibles para su utilización en la evaluación de transiciones por un período Δt de tiempo, o hasta que el reloj global alcance el tiempo t_s.
- Al momento del disparo, los tokens de salida de la transición adquieren un sello de tiempo cuyo valor es calculado por la formula (1.2) sin importar los valores de los sellos que tenían los tokens causantes del disparo

1.5.2 Extensión temporal del árbol de alcance.

Como se mencionó en la sección 1.4 el árbol de alcance es una herramienta de análisis que permite almacenar los diferentes estados del sistema. Cuando se hace uso del árbol de alcance con RdPC temporales, es necesario asociar para cada nodo del árbol el tiempo global de simulación así como los sellos de tiempo de lo tokens que componen la marca. Entonces, con el uso de RdPC con extensión temporal, es importante especificar la dimensión que toman los conceptos de marcado, nodo repetido.
El *marcado temporal* consiste en la distribución de tokens en los lugares junto con sus sellos de tiempo.
El *nodo repetido temporal* es aquel donde el marcado temporal es el mismo para dos estados del sistema.

[2] En la literatura se emplea comúnmente el símbolo @ para indicar los sellos de tiempo de los tokens [53]

Para el objetivo del trabajo aquí presentado, al hacer referencia a nodo repetido o nodo *old*, se refiere a nodos donde la distribución de tokens coloreados es la misma sin hacer distinción de ellos por sus valores temporales (nodos simétricos). La razón de lo anterior se explica más a detalle en la sección 3.2.

Ejemplo 1.3 Árbol de alcance con extensión temporal

A continuación se presenta de manera gráfica un árbol de alcance con extensión temporal, donde se puede observar que a cada marcado le corresponde un reloj global y los tokens tienen además un sello de tiempo. En la Figura 1-10 se ilustra el caso de nodos como el 7 y 8, que pueden diferenciarse entre ellos por la cantidad y tipos de tokens que conforman el marcado, pero ambos generan un nodo repetido que en este caso es el nodo 12 por lo mismo ambos tendrán un arco que los conecta con el nodo repetido.

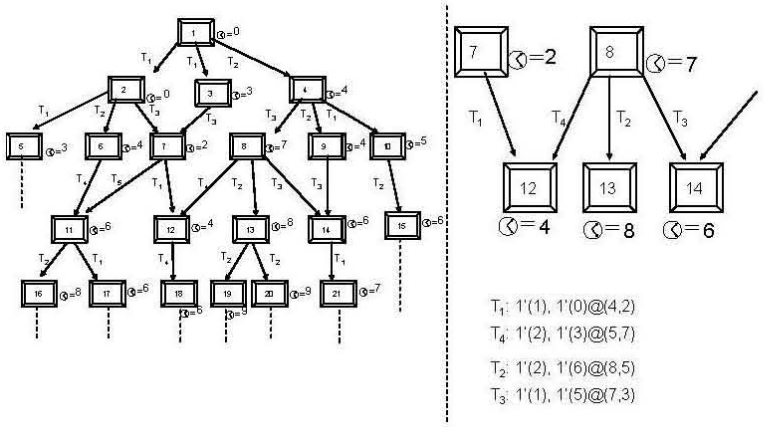


Figura 1-10: Árbol de alcance con extensión temporal

Es frecuente encontrar que tanto el reloj global como los sellos de tiempo de los tokens pertenecientes a los nodos padres sean distintos, y sin embargo generen un estado repetido con la misma distribución de tokens. Debido a esto y para evitar en lo posible el problema de la explosión de estados [120] es necesario implementar un análisis para decidir cuál de ellos genera mejores tiempos en el nodo hijo y de esta manera no repetir el almacenado de dos estados que difieren únicamente en sus valores temporales.

1.6 Evolución Histórica del Análisis del Árbol de Alcance para las redes de Petri

A continuación se presenta la evolución histórica del uso del árbol de alcance lo cual sirve para poner en contexto el estudio realizado en esta Tesis.
En 1967 Carl Adam Petri publica el artículo "*Fundamentals of the Description of Discrete Processes*" [95] en dicho artículo se establece la primera definición de las redes de Petri y las reglas básicas para modelar cambios discretos en procesos concurrentes. En 1969 Karp y Miller [57] presentan el artículo "*Parallel Program Schemata*" en el cual proponen como construir una gráfica que posteriormente se conocerá árbol de alcance de una RdP. En ese artículo se describe el concepto de *estados alcanzables*, se bosqueja además el concepto de reducción de estados a través de la fusión de estados que resultan idénticos. Es propiamente con estas publicaciones que el estudio del análisis de sistemas concurrentes a través de la exploración de estados da inicio.
En 1986 el artículo "*Reachability Trees for High-Level Petri Nets*" [50] define las reglas para la construcción del árbol de alcance para las RdPC.

En 1991 Valmari [119] presenta el artículo "*Stubborn Sets for Reduced State Space Generation*" donde detecta secuencias intermedias de estados en el árbol de alcance de procesos concurrentes, las cuales pueden ser omitidas debido a que llevan a un mismo estado, este tipo de caminos que son equivalentes los denomina *stubborn sets*, la idea principal es que se utilizan para controlar la explosión de estados dejando fuera del árbol de alcance estados y transiciones redundantes lo cual permite la generación de un árbol de alcance reducido (*Reduded Reachability graph*).
En 1993 Finkel [38] presenta el artículo "*A minimal Coverability Graph for Petri Nets*" en el cual propone una mejora del algoritmo de Karp y Miller determinando lo que denomina el árbol de alcance mínimo único. Este algoritmo utiliza propiedades que dependen del comportamiento monotónico de las redes para reducir y compactar la gráfica de Karp y Miller durante su generación.
En 1993 Van Der Aalst [122] en su artículo "*Interval Timed Coloured Petri Nets and their Analysis*" propone una reducción del árbol de alcance para redes de Petri coloreadas que además presentan características temporales. Plantea la primera idea de una agrupación de estados en un tipo de clases con características comunes pero que tienen un intervalo temporal de existencia.
En 1996 Jensen desarrolla la idea de la agrupación de estados semejantes y publica el artículo "Condensed State Spaces for Symmetrical Coloured Petri Nets" [52] en el cual define el concepto de estado simétricos en las RdPC, además explica cómo a partir de la explotación de la simetría se puede generar un árbol de alcance compactado. En este caso las simetrías inducen a un árbol de alcance donde cada estado representa una clase equivalente y cada arco representa una clase equivalente de cambios de estado, este enfoque lo utiliza principalmente para la verificación de propiedades de las RdPC.
En 1997 Chiola y otros autores [18] presentan "*A symbolic Reachability Graph for Coloured Petri Nets*" en el cual plantean la idea de generar el árbol de alcance de manera que los colores de las marcas (tokens) se represente de manera simbólica. La idea es explotar nuevamente las simetrías de los modelos de manera que se puedan utilizar símbolos en lugar de los valores particulares de los colores.

En el año 2001 Christensen y otros autores [19] aplican las ideas de compactación o agrupación de estados presentando el artículo "*Condensed State Spaces for Timed Petri Nets*", en el cual estudian la manera de reducir el árbol de alcance de las redes de Petri Coloreadas temporales; introducen la noción del tiempo relativa en lugar de absoluta para de esa manera poder definir clases equivalentes de estados.; nuevamente este tipo de compactación lo utilizan para verificar propiedades de las redes de Petri coloreadas pero en este caso temporales.
En 2001 los mismos autores presentan el artículo, "*A sweep line method for State Space Exploration*" [21] donde plantean una idea de exploración del árbol de alcance con el objetivo de verificar las propiedades de las redes de Petri coloreadas. El enfoque que plantean, establece una medida de progresión en los modelos y establece ir evaluando y generado los diferentes estados del árbol de alcance de manera que los estados analizados cuya medida de progresión es inferior al indicador actual pueden ser eliminados. Este enfoque permite por una parte verificar las propiedades de las RdPC temporales y atemporales, pero además (dependiendo de la naturaleza del modelo) es posible desechar estados visitados para de esta manera poder seguir explorando estados sin llegar a saturar la memoria local del ordenador.
En el año de 2007 Westeergard y otros [128] presentan el Comeback Method el cual tiene como objetivo la verificación de propiedades de las RdPC pero además convierte los estados a un código numérico para minimizar el espacio ocupado por los estados en la memoria del ordenador.
De la revisión presentada se resume que los desarrollos llevados a cabo tienen como objetivo principal la verificación de las propiedades de las RdPC y para llevar a cabo dicho estudio es necesario tener técnicas para poder analizar el árbol de alcance, los métodos desarrollados se pueden clasificar de tres tipos[60]:
a) Los métodos que aprovechan algún tipo de simetría para poder representar el árbol de alcance de forma compacta o condensada sin perder información relevante para el análisis.
b) El conjunto de métodos que no genera en su totalidad el árbol de alcance explotando la interdependencia existente entre los procesos para evitar representar todas las ejecuciones interpoladas del sistema.
c) Los métodos que generan un árbol de alcance parcial eliminando información del árbol de alcance cuando los estados ya han sido utilizados o explorados.
Desafortunadamente la mayoría de los enfoques estudiados tienen como objetivo el análisis de propiedades de las RdPC y no propiamente su uso como herramienta de optimización.

1.7 Evaluación de Transiciones: Aspectos Críticos

Uno de los principales elementos para la generación de estados en el árbol de alcance es la evaluación de transiciones, la cual consiste en realizar la evaluación de los lugares de entrada para cada una de las transiciones del modelo con el objetivo de determinar todas las posibles combinaciones de tokens que satisfacen las restricciones para poder habilitar una transición. Diferentes autores mencionan como una etapa clave para el buen rendimiento del algoritmo de simulación la etapa de evaluación de transiciones [39, 17, 79, 107].
La manera natural para realizar esta evaluación es la que se conoce como *Genera y Prueba* (G&P) la cual consiste en ir probando combinaciones de tokens aleatoriamente, y posteriormente probar si la combinación satisface las restricciones existentes, si las satisface se ha encontrado una solución, y es posible disparar la transición, en caso contrario se prueba una combinación distinta.

Este algoritmo de evaluación presenta una deficiencia muy importante y es que hace una evaluación de las restricciones de manera tardía, lo que provoca la evaluación de muchas combinaciones infértiles de tokens en el sentido que no permiten habilitar la transición.

Ejemplo 1.4 Evaluación del rendimiento de las transiciones con el algoritmo G&P

Evaluando una transición con el algoritmo G&P con dos lugares de entrada como la transición T_1 de la Figura 1-11 y suponiendo que el lugar P_1 tiene N número de tokens y el lugar P_4 tiene un número M de tokens. Pare llevar a cabo la evaluación de todos los tokens en los lugares de entrada es necesario realizar un numero de $N \times M$ combinaciones de tokens para determinar aquellos que habilitan la transición.

El número de combinaciones de tokens necesarias para evaluar cuales habilitan la transición se expresa matemáticamente con la siguiente fórmula:

$$\prod_{i=1}^{r} P_{ij} \tag{1.3}$$

Cuando se tienen r nodos lugar de entrada a la transición j, donde P_{ij} representa el número de tokens del i-ésimo lugar de entrada a la transición j.

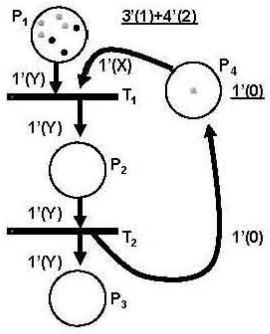


Figura 1-11: RdPC de un sistema con recursos compartidos

En algunas RdPC y debido al número de tokens presentes, la aproximación G&P pareciera la más natural, pero considerando que en los modelos de sistemas industriales el número de tokens que pueden estar presentes en los lugares de entrada puede ser elevado, el número de operaciones necesarias para seleccionar los tokens que habilitan la transición resulta restrictivo en un ambiente

de toma de decisión bajo límites de tiempo. Por lo anterior es necesario diseñar otras alternativas más eficientes que permitan determinar cuáles son los tokens que satisfacen todas las restricciones del modelo.

1.8 Importancia de la gestión de memoria durante la apertura del árbol de alcance.

El problema conocido como *explosión de estados* [120] se da cuando el número de estados generados por modelos incluso muy pequeños es tan grande que resulta imposible su almacenaje en la memoria de cualquier ordenador.
Tanto las RdPC como las RdP has sido utilizadas para modelar sistemas de manufactura debido a las características que presentan (ver sección 1.2.5). El llevar a cabo un análisis del funcionamiento así como la validación de sistemas industriales reales haciendo uso del árbol de alcance resulta muy adecuado siempre que se puedan explorar todos los estados del modelo. Sin embargo cuando se utiliza el árbol de alcance como herramienta de análisis con las RdPC el problema de la explosión de estados se presenta con frecuencia, lo cual ocasiona que no se puedan explorar todos los estados posibles. Para solventar esta dificultad, diferentes autores han desarrollado enfoques que evitan buscar a través de todo el espacio de estados o para llevar a cabo un almacenaje de información más eficiente [75, 88, 17, 19, 49, 51].
Sin embargo cuando el objetivo del modelado consiste en optimizar el funcionamiento de un sistema real se deben explorar todos los estados de un modelo, aunque generalmente el número máximo de estados explorados se verá limitado por la memoria física del ordenador. Por lo tanto, es necesario desarrollar enfoques eficientes para el almacenaje de la información del árbol de alcance para poder explorar un número mayor de estados evitando la rápida saturación de la memoria del ordenador.
La información de los estados de un modelo se puede codificar de diferentes maneras, dependiendo de la representación que se utilice ocuparan mayor o menor memoria física del ordenador, no implicando con ello que se reduzca la naturaleza del problema.

Ejemplo 1.5 Codificado de Estados

El marcado de una RdPC con 4 nodos lugar del tipo [2'(3,4)+4'(4,6) , 4'(8) , 2'(4,5) +1'(2,2), 3'(9)] que puede representar el estado de una RdPC puede ser codificada de diferentes maneras:

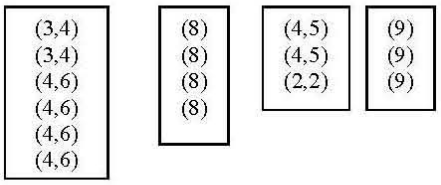
1. En bloques con una cadena de información del marcado:

2'(3,4)+4'(4,6) , 4'(8) , 2'(4,5) +1'(2,2), 3'(9)

2. Con bloques de información para cada marca de los nodos lugar:

2'(3,4)+4'(4,6)	4'(8)	2'(4,5) +1'(2,2)	3'(9)

3. Con bloques de información para cada token del marcado:



Tomando en cuenta únicamente el espacio ocupado por las cadenas de caracteres (Strings), las cuales ocupan una cantidad de 2 *bytes* por cada caracter[3], la representación numero uno ocupa una cantidad de memoria de 86 bytes, la representación dos ocupa 80 bytes, y la representación tres una cantidad de 66 bytes.

1.9 Motivación

La industria requiere de herramientas que ayuden en la toma de decisiones para la coordinación de actividades a diferentes niveles de operación. Entre dichas actividades se pueden mencionar la programación de producción, la política de flujo de materiales y productos, y en general decisiones logísticas donde es necesario que la toma de decisiones sea automática dado que las restricciones temporales impiden una etapa de análisis profundo que consumiría mucho tiempo.
Actualmente existen herramientas de ayuda a la toma de decisiones a diferentes niveles operativos, particularmente la s*imulación digital* permite modelar con un alto nivel de descripción un sistema, pero solo evalúa un número reducido de escenarios de las potenciales configuraciones bajo las que debería evaluarse el sistema modelado. Debido a esta característica no es posible garantizar optimalidad para las soluciones obtenidas a través de simulación.
Las RdPC resultan adecuadas para modelar fenómenos de concurrencia, paralelismo y situaciones de conflicto que se presentan comúnmente en sistemas industriales, por lo que surge la motivación de desarrollar una herramienta de apoyo a la toma de decisiones que utilice el formalismo de RdPC para generar automáticamente los estados alcanzables por el sistema haciendo uso del árbol de alcance para analizar y optimizar el modelo bajo una función de coste temporal.

[3] Microsoft Help and Support : http://support.microsoft.com/kb/137729

1.10 Objetivos

Para la realización de este trabajo se plantean los siguientes objetivos:

- Objetivo General. El desarrollo de una herramienta informática que permita el modelado de sistemas bajo el formalismo de RdPC temporales o atemporales, que sea posible analizar los estados del sistema utilizando el árbol de alcance para encontrar un óptimo bajo una función de coste temporal en tiempos adecuados.
- Objetivos Particulares:
    - Implementar un algoritmo de evaluación de transiciones de manera que se utilicen activamente todas las restricciones para encontrar de manera eficiente la combinación de tokens que habilita la transición.
    - Desarrollar un codificado adecuado para el almacenaje de la información del árbol de alcance de manera que se aprovechen las características de los modelos industriales para que sea posible almacenar la mayor información evitando en lo posible la saturación de memoria del ordenador.
    - Implementación de manera eficiente la generación del árbol de alcance que se pueda utilizar tanto para RdPC temporales o atemporales.
    - Una implementación que permita la optimización de una función de coste haciendo uso de la extensión temporal del modelo en RdPC y el árbol de alcance.
    - La obtención de resultados óptimos o cercanos al óptimo en tiempos de cómputo adecuados para que eso permita la implementación de herramientas de apoyo a la toma de decisiones basada en algoritmos que utilicen la exploración del árbol de alcance.

2 SIMULADOR DE REDES DE PETRI COLOREADAS

Para el desarrollo de un simulador que utilice las RdPC como formalismo de modelado y el árbol de alcance como herramienta de análisis y espacio de búsqueda, resulta importante determinar un algoritmo que permita la evaluación de las actividades en el modelo (transiciones) de forma rápida y eficiente, así como una codificación de estados del modelo que permita implementar técnicas eficientes de búsqueda en el árbol de alcance.

2.1 Simulación/Optimización con el árbol de alcance.

El árbol de alcance permite no solo la evaluación de los distintos estados que se pueden alcanzar por un modelo en RdPC sino que utilizando la extensión temporal de las RdPC (ver sección 1.5) permite el analizar el funcionamiento de modelos industriales de sistemas a eventos discretos (retrasos, tiempos de proceso, rendimiento de proceso, etc.), a través de la simulación del comportamiento del sistema a medida que ocurren cambios en el modelo (arcos del árbol).
Las ventajas que presenta este enfoque son las siguientes:

- La transformación de un problema de optimización [4] en uno de búsqueda generando todos los estados que se pueden alcanzar por un modelo para encontrar un estado en particular.
- Este enfoque permite encontrar la secuencia de operaciones que permite llevar al sistema desde un estado inicial (marca inicial) hasta un estado final encontrando soluciones óptimas o cuasi-óptimas analizando la componente temporal del modelo.

La conceptualización del simulador basado en el árbol de alcance se esquematiza en la figura 2-1.

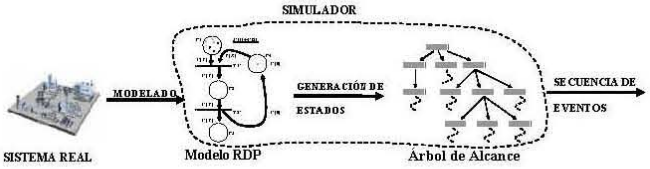


Figura 2-1: El simulador de RdPC

Al ingresar las entradas, el simulador realizará la generación del espacio de estados y búsqueda de los estados objetivos. Con este enfoque el proceso de búsqueda a través del árbol de alcance permanece invisible al usuario.

[4] Basada en la Simulación de una fracción de los posibles escenarios del sistema.

2.2 Programación con Restricciones

La programación con restricciones (CP[5]) es un paradigma adaptado a problemas de búsqueda tipo NP-duros [125]. Incorpora técnicas de diferentes campos del conocimiento como matemáticas, inteligencia artificial e investigación operativa, lo cual presenta grandes ventajas a la hora de incorporarlo en algoritmos entregando ventajas como el rápido desarrollo de programas, fácil mantenimiento y eficiente desempeño.

Las restricciones son consideradas el núcleo de la CP, y son consideradas como una relación lógica entre un conjunto de variables, las cuales toman valores dentro de un conjunto definido como un rango con cota superior e inferior, o una lista de números válidos.

La idea de CP es el resolver problemas a través de la declaración de restricciones (requerimientos) acerca del área del problema, y consecuentemente encontrar una solución o soluciones que satisfagan todas las restricciones.

2.2.1 Propagación de Restricciones

La propagación de restricciones es una manera de generar las consecuencias de una decisión. En muchas aplicaciones, la propagación de restricciones se utiliza junto con herramientas informáticas, tomando el resultado de estas como entrada, realizar al propagación, y generando las consecuencias.

La propagación de restricciones ha sido el algoritmo principal utilizado en resolver un gran conjunto de problemas denominados *problemas de satisfacción de restricciones* (CSP[6]). Existen dos tipos de CSP dependiendo de los dominios sobre los cuales son definidos: *satisfacción de restricciones* o *resolución de restricciones*.

Satisfacción de restricción es una tecnología de resolución la cual resuelve problemas sobre dominios finitos, mientras que la resolución de restricciones es mayormente definida sobre dominios infinitos o más complejos.

Definición 2.1 Etiqueta [117]

Una etiqueta es un par variable-valor que presenta la asignación del valor a una variable. Es utilizado $\langle x, v \rangle$ para denotar la etiqueta de asignar el valor v a la variable x.

Definición 2.2 Etiqueta Compuesta [117]

Una etiqueta compuesta es la asignación simultánea de valores a un conjunto de variables. Se utiliza ($\langle x_1, v_1 \rangle, \langle x_2, v_2 \rangle, \ldots, \langle x_n, v_n \rangle$) para denotar la etiqueta compuesta de asignaciones a las variables $v_1, v_2, \ldots, v_n$ con $x_1, x_2, \ldots, x_n$ respectivamente.

Definición 2.3 Problema de Satisfacción de Restricciones (CSP) [117]

Un CSP es una tripleta

(Z, D, C)

---

[5] CP de su denominación en inglés *Constraint Programming*

[6] CSP de su denominación en inglés *Constraint Satisfaction Problems*

Donde
Z= es un conjunto finito de variables $x_1, x_2, \ldots, x_n$;
D= una función que mapea cada variable de Z a un conjunto de objetos de cualquier tipo:

D: Z $\rightarrow$conjunto finito de objetos (de cualquier tipo)
Se puede tomar D_x como el conjunto de objetos mapeados de x_i por D. Llamaremos a estos objetos los posibles valores de x_i y el conjunto D_{x_i} el *dominio* de x_i;

C= un conjunto finito de restricciones sobre un subconjunto arbitrario de variables en Z. En otras palabras, C es un conjunto de conjuntos de etiquetas compuestas.
Una solución de un CSP es la completa asignación de valores a las variables del problema de manera que todas las restricciones son satisfechas a la vez. Se puede requerir encontrar:

- Solo una solución, sin preferencia sobre alguna en particular.
- Todas las soluciones
- Un óptimo, o al menos una buena solución, dada una función objetivo definida en términos de algunas o todas las variables.

Ejemplo 2.1
Dada una restricción:

$$X = Y + 1$$

Donde X e Y son variables en un programa, cualquier asignación a la variable Y (Y=5) causa una asignación a la variable X (X=6). En general, cuando una variable perteneciente a una restricción es etiquetada, ese valor se propaga al resto de las variables relacionadas con esa restricción.

2.2.2 Satisfacción de Restricciones

La satisfacción de restricciones se relaciona con problemas definidos sobre dominios finitos. Soluciones al problema CSP se pueden encontrar por medio de búsqueda (sistemática) a través de las posibles asignaciones a las variables, o sea generando todo el árbol de búsqueda[7] sobre las variables. Los métodos de búsqueda se pueden dividir en dos grandes clases: aquellas que recorren el espacio de soluciones parciales al CSP (asignaciones parciales de valores), y aquellas que exploran el espacio completo de asignaciones de valor (para todas las variables) de manera estocástica.

2.2.2.1 Salto hacia Atrás.

El *salto hacia atrás* (*backtracking*) [90] es un método utilizado para resolver CSP a través de ir extendiendo progresivamente una solución parcial que especifica valores consistentes para algunas de las variables. Esta actividad se lleva a cabo seleccionando repetidamente un valor para otra variable que sea consistente con los valores de la solución parcial, hasta alcanzar una solución completa. Las variables son asignadas secuencialmente y tan pronto como las variables relevantes a la restricción son instanciadas, la validez de la restricción es verificada. Si una solución parcial no cumple algunas de las restricciones, se lleva a cabo un retroceso hacia la última variable

[7] No confundir con el árbol de alcance

instanciada que aun tiene alternativas disponibles. Claramente cuando una instanciación parcial viola una restricción, el BT puede eliminar un subespacio del producto cartesiano del dominio de todas las variables.

Existen tres principales desventajas del BT estándar:

- *El golpeado*: Es un fallo repetido (y consecuentemente un BT) debido al mismo motivo en una evaluación, lo cual sucede debido a que no se almacena información cuando ocurre un fallo. Por lo tanto si ocurriera una situación similar en el futuro la búsqueda fallaría y realizaría el BT.
- *Trabajo redundante*: Los valores en conflicto de las variables no son memorizados, por lo que ocurre que la búsqueda falla de la misma manera en diferentes etapas de la búsqueda.
- *Detección tardía de conflictos*: Los conflictos no se detectan hasta que realmente ocurren.

2.3 Evaluación de Transiciones

Uno de los puntos clave para el buen funcionamiento de cualquier simulador basado en RdPC es la etapa de la evaluación de las transiciones [39, 18, 79]. En este trabajo se presenta un algoritmo que utiliza algunos conceptos de la CP para resolver el problema de la satisfacción de restricciones impuesta por los guardas y las expresiones de arco. Se han utilizado conceptos como la reducción de escenarios para crear los subconjuntos iniciales que se generan para la evaluación de las expresiones de arco y la propagación de restricciones con el objetivo de satisfacer todas las restricciones existentes. Este enfoque permite utilizar de manera activa las restricciones declaradas en los guardas para evitar evaluar combinaciones infértiles de los tokens presentes en los lugares de entrada.

A continuación se explican los principios en los cuales están basados el algoritmo desarrollado así como una tabla comparativa del algoritmo basado en este enfoque respecto al enfoque G&P.

2.3.1 Evaluación de transiciones como un problema de satisfacción de restricciones.

Considerando que la evaluación de una transición debe cumplir con las reglas de disparo de las RdPC (peso de arcos, expresiones de arcos, etc.), la etapa de la evaluación de transiciones se puede formular como un problema de CSP.

A nivel local (para cada una de las transiciones) se pueden definir los elementos del CSP de la siguiente manera:

Z: El conjunto de las variables que intervienen en las expresiones de arco en los lugares de entrada.
D: La función o funciones que mapean los valores de las variables en su dominio es realizada por la función identidad para las diferentes variables representadas en las expresiones de arco, haciendo uso de los colores de los tokens pertenecientes a los lugares de entrada.
C: Es el conjunto de restricciones que debe satisfacer el conjunto de variables Z, este conjunto de restricciones corresponde a los guardas que presenta la transición en cuestión.

Ejemplo 2.2 Identificación local de los elementos del CSP para una transición en RdPC

Se tiene una transición de una RdPC con dos lugares de entrada como la que se observa en la Figura 2-2. Se observa que el marcado para esta transición es:

[3'(3,2,5)+ 1'(2,2,5)+ 2'(3,3,5) , 2'(3,3) + 1'(4,2) + 1'(4,4)]

Si se plantea la evaluación de transiciones como un CSP se tiene para esta transición en particular:

Z= {X, Y, J}
D= se utiliza la función identidad para obtener los dominios de las respectivas variables.
En este caso utilizando los colores de los tokens de entrada se tiene:
X:{2,3}
Y:{2,3}
J:{2,3,4}

C= { J=X+2 , Y>=2 }

- Se etiqueta la variable X con el primer elemento de su dominio <X, 2>, correspondiente al token 1'(2,2,5)
- Se propaga su valor a la restricción correspondiente (J=X+2), y se obtiene el valor que debe adquirir la variable J para que no se viole la restricción
- J adquiere el valor 4, el cual se encuentra dentro del dominio de J y el color pertenece al token 1'(4,4) del lugar P2, por lo que se etiqueta <J, 4>.
- Debido a que los colores van asociados a un token en particular, la variable restante Y adquiere el valor del color correspondiente de la solución parcial de tokens en este caso su valor es 2 correspondiente al token 1'(2,2,5).

En este caso se ha encontrado una solución al problema de encontrar los tokens que satisfacen las restricciones del CSP:
<X, 2>
<Y, 2>
<J, 2>
La cual corresponde a los tokens:
P1: 1'(2,2,5)
P2: 1'(4,4)

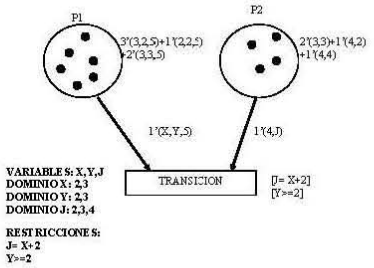


Figura 2-2: Elementos del CSP

Este enfoque presenta ciertos aspectos que deben de ser tomados en cuenta para desarrollar un algoritmo de evaluación de transiciones.

- Cuando se realiza una asignación de valor de una variable, y existe una segunda o tercera variables asociadas a otro color perteneciente al mismo token, quedan también determinados el valor de la segunda o tercera variables (como el caso de las variables X e Y de la Figura 2-2).
- Se puede observar que la sola satisfacción del CSP por sí misma no resuelve del todo el problema de la evaluación de transiciones, también es necesario resolver los valores constantes de las expresiones de arco.

Todos estos aspectos se deben de tomar en cuenta en el desarrollo de un algoritmo que utilice algunos de los principios de CP para la evaluación de transiciones en un entorno para modelos de RdPC.

2.3.2 Algoritmo de Evaluacion de transiciones

Es posible resolver el problema de la evaluación de transiciones de una manera más eficientemente respecto del enfoque clásico de *G&P* si la satisfacción de expresiones de los guardas se aborda como un problema de satisfacción de restricciones. El algoritmo implementado evalúa de manera secuencial cada una de las transiciones del modelo. Se conforma como primera etapa un subconjunto de tokens para los lugares de entrada que satisfacen las expresiones de arco. La segunda etapa de evaluación consiste en un planteamiento de CSP para satisfacer las restricciones planteadas por los guardas. Con la implementación de este algoritmo es posible encontrar las combinaciones de tokens que satisfacen todas las restricciones que intervienen en una transición (expresiones constantes en los arcos, guardas, pesos específicos de arco) y permite disparar o no la transición. Las etapas principales del algoritmo se detallan a continuación.

2.3.2.1 Satisfacción de expresiones de arcos

Para comenzar con la selección de los tokens que habilitan una transición, se realiza una pre-evaluación de los tokens para seleccionar aquellos que cumplen con los valores constantes de las expresiones de arco. Al comparar los colores de los tokens en los lugares de entrada contra los valores constantes de las expresiones de arco, se forma un subconjunto con aquellos tokens cuyos colores satisfacen los valores constantes de las expresiones de arco.
En el caso que este conjunto resulte vacío se detiene el proceso y se continúa con la evaluación de otra transición del modelo.

Ejemplo 2.3 Primer paso en la evaluación de transiciones.

En la figura 2-3 se representa lugar de entrada (lugar en evaluación) el cual contiene 4 tokens:

$$2'(3,3)\ ,1'(4,2),\ 1'(4,4)$$

El valor constante en la expresión de arco específica que el primer color de los tokens debe de tener el valor 4 por lo cual se seleccionan dos tokens: 1'(4,2) y 1'(4,4).

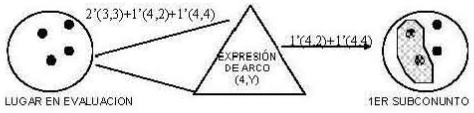


Figura 2-3: Subconjunto generado por expresión de arco

Esta etapa es el primer paso del algoritmo de evaluación de transiciones, y este procedimiento se realiza con todos los lugares de entrada a la transición.
Al realizar esta selección inicial, el dominio de las variables se actualiza con el nuevo subconjunto formado para que la siguiente etapa donde se implementa el CSP resulte más eficiente.

2.3.2.2 Satisfacción de Guardas

El siguiente paso en el algoritmo de evaluación de transiciones consiste en satisfacer las expresiones indicadas por las expresiones de los guardas.

Al realizar la satisfacción de expresiones de guardas se utilizan los token contenidos en los subconjuntos creados con las expresiones de arco en los lugares de entrada.
Al trabajar con los subconjuntos, se plantea la satisfacción de restricciones como un CSP tomando en cuenta también la dependencia que existe entre variables cuando dos o más variables tienen su dominio sobre colores de un mismo lugar de entrada, (ver ejemplo 2.2).

La satisfacción de restricciones se lleva a cabo analizando secuencialmente los lugares de entrada, asignando valores a las variables que se van encontrando durante la evaluación satisfaciendo las restricciones encontradas. Dependiendo de la habilidad del modelador, es posible mejorar la eficiencia de la evaluación evaluando primeramente las restricciones que reducen mayormente los dominios de las variables y por tanto el número de tokens a evaluar con la siguiente restricción. De esta manera, a medida que se evalúan los lugares, se reducen paulatinamente los conjuntos de tokens que satisfacen las restricciones evitando de esa manera la evaluación de tokens infértiles y cálculo innecesario.
Cuando se termina de evaluar el último lugar de entrada, se tiene como resultado un subconjunto en cada uno de los lugares de entrada (de menor tamaño que el subconjunto inicial) con los tokens que satisfacen tanto los valores constantes de las expresiones de arco, como las restricciones expresadas por la expresión guarda (ver Figura 2-4).

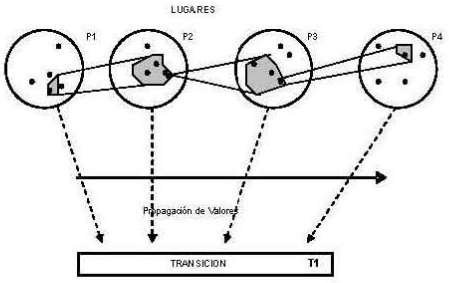


Figura 2-4: Tokens que habilitan la transición

La generación de estados se lleva a cabo realizando los disparos de la transición con las distintas combinaciones de tokens que se pueden obtener del conjunto solución.
Después de cada disparo, el proceso continúa seleccionando un nuevo token del lugar anterior al último evaluado y llevando a cabo nuevamente el procedimiento descrito hasta haber evaluado todos los tokens de los subconjuntos iniciales.

Ejemplo 2.4 Algoritmo de evaluación de transiciones.

En la Figura 2-5 se observa una transición a evaluar para encontrar las combinaciones de tokens que satisfacen las restricciones y habilitan la transición.

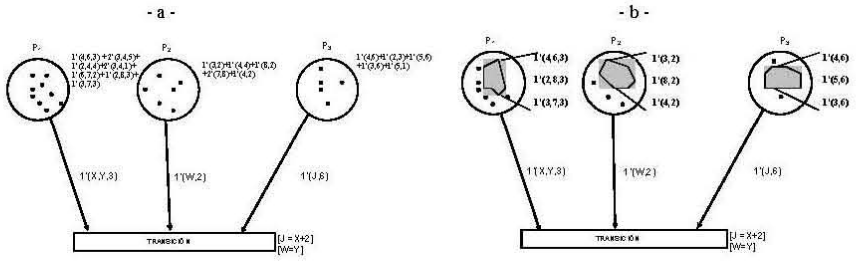


Figura 2-5: Selección inicial a) Estado inicial b) Filtrado con arcos

El marcado inicial del modelo es el siguiente (Figura 2-5a):

[1'(4,6,3) + 2'(3,4,5)+1'(2,4,4)+2'(3,4,1)+1'(6,7,2)+ 1'(2,8,3)+1'(3,7,3) , 1'(3,2)+1'(4,4)+1'(8,2)+2'(7,8)+1'(4,2) , 1'(4,6)+1'(2,3)+1'(5,6)+1'(3,6)+1'(5,1)**]**

El dominio inicial de las variables es:
X: {2,3,4,6}
Y: {4,6,7,8}
W:{3,4,7,8}
J:{2,3,4,5}

- Paso 1) Satisfacción de expresiones de arcos. Se inicia comparando los valores constantes en las expresiones de arco con todos los tokens en cada uno de los lugares de entrada correspondientes, con lo cual se generan los subconjuntos iniciales (áreas sombradas en lugares de la figura 2-5b), los tokens que conforman los subconjuntos iniciales son:
  - Lugar P1: [1'(4,6,3), 1'(2,8,3), 1'(3,7,3)]
  - Lugar P2:[1'(3,2), 1'(8,2), 1'(4,2)]
  - Lugar P3: [1'(4,6), 1'(5,6), 1'(3,6)]

  Utilizando estos subconjuntos, los dominios de las variables se actualizan:

  X: {2,3,4}
  Y: {6,7,8}
  W:{3,4,8}
  J:{3,4,5}

- Paso 2) Satisfacción de guardas. Al inicio de esta etapa se utilizan los tokens de los subconjuntos iniciales, y se plantea el siguiente problema a resolver:

Z= { X, Y, W,J}
X: {4,2,3}
Y: {6,8,7}
W:{3,8,4}
J:{4,5,3}

C= { J=X+2 , W=Y }

La secuencia de acciones para encontrar los tokens que habilitan la transición es la siguiente:

- Se selecciona el primer token del lugar P_1 : 1'(4,6,3)
- Se realiza el etiquetado <X, 4>
- Se propaga el valor a través de la restricción J=X+2, y se obtiene J =6 para no violar la restricción.
- Con el etiquetado <J, 6> se detecta que no existe ningún token en el lugar P3 que tenga ese valor, por lo que se realiza un BT a la ultima variable instanciada.
- Se selecciona un nuevo token del lugar P_1: 1'(2,8,3)
- Se etiqueta <X, 2>
- Se propaga el valor y se tiene <J, 4>, al buscar en su correspondiente lugar P_3, se encuentra un token que satisface la asignación: 1'(4,6).
- Resuelta la primera restricción se procede con la siguiente variable del lugar P_1, cuyo valor se encuentra determinado ya que el dominio se encuentra en los colores de los tokens del mismo lugar que la variable X. Por lo tanto: <Y, 8>
- Se propaga el valor a través de la restricción : W=Y
- Como resultado de la propagación de valor, se tiene el etiquetado <W, 8>
- Se realiza la búsqueda de tokens en el lugar correspondiente P_2, y se encuentra el token correspondiente: 1'(8,2)
- Al haber terminado con las variables del primer lugar, se realiza el mismo procedimiento con el lugar P_2. Debido a que en el lugar P_2 se asignaron los valores de las variables (y correspondientes tokens seleccionados) se procede a evaluar el lugar P_3.
- El análisis en el lugar P_3 no hay variables por asignar y se tiene seleccionado el subconjunto que satisface la restricción W=Y.
- Al haber evaluado los lugares de entrada se termina el procedimiento produciendo el resultado :

Variables: <X, 2> , <J, 4>,<Y, 8>,<W, 8>
Tokens lugar P1: 1'(2,8,3)
Tokens lugar P2: 1'(8,2)
Tokens lugar P3: 1'(4,6).

2.3.3 Eficiencia de Cálculo del Algoritmo de Evaluación

Debido a que el algoritmo de evaluación de transiciones primero satisface los valores constantes presentes en las expresiones de arco, y posteriormente las restricciones de los guardas, el número de evaluaciones a realizar para satisfacer los arcos serán tantas como tokens se tenga en los lugares de entrada. En la etapa de satisfacción de restricciones, en el peor de los casos serán tantas operaciones como combinaciones distintas se pueden tener con los tokens de los subconjuntos iniciales. Lo anterior se expresa matemáticamente con la siguiente formula cuando se tienen **r** nodos lugar a la entrada de la transición:

$$\sum_{i=1}^{r} T_{ij} + \prod_{i=1}^{r} S_{ij} \qquad (2.1)$$

Donde T_{ij} y S_{ij} representan respectivamente el número de tokens del í-ésimo lugar de entrada a la transición j al comienzo del proceso de evaluación y los que se encuentran en el subconjunto formado por los tokens que satisfacen los valores constantes de las expresiones de arco de los lugares de entrada.

Ejemplo 2.5 Comparación de algoritmos.

Se evalúa una transición con 50 tokens en cada uno de los tres lugares de entrada, como la mostrada en la figura 2-6.

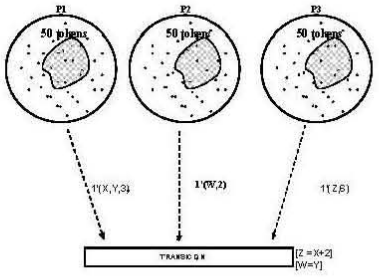


Figura 2-6: Evaluación de una transición

Si la evaluación se realiza con el enfoque más conservador (G&P), se tienen que evaluar todas las combinaciones diferentes que se pueden obtener con los tokens de los lugares de entrada (ver sección 1.7), y en el peor de los casos para encontrar las combinaciones de tokens se tienen que realizar la siguiente cantidad de operaciones, expresadas por la formula (1.3):

En este ejemplo j=1, r=3

$$\prod_{i=1}^{r} P_{ij} = 50 \text{ x } 50 \text{ x } 50 = 125{,}000 \text{ operaciones}$$

Utilizando el algoritmo desarrollado en este trabajo, el número de operaciones se reduce considerablemente:

- Suponiendo que del primer subconjunto que se forme para cada uno de los lugares se obtienen subconjuntos de 20 elementos, el número de operaciones a realizar en el peor de los casos se obtendría al aplicar la formula (2.1) :

  En este ejemplo j=1, r = 3

  $\sum_{i=1}^{r} T_{ij} = 50 + 50 + 50 = 150$, son las operaciones para formar los conjuntos iniciales

  $\prod_{i=1}^{r} S_{ij} = 20 \text{ x } 20 \text{ x } 20 = 8000$, la satisfacción de restricciones bajo el peor escenario

  Operaciones Totales: 150 + 8000 = 8150
  Reducción = (125000-8150/125000)*100 = 93%

2.3.4 Eficiencia Práctica de la Evaluación de Transiciones

El Taller de trabajo (*job shop*) 6x6 [29]es un problema de referencia (*benchmark*) muy estudiado en la literatura, y de antemano se sabe que el número de estados diferentes que tiene un modelo de este sistema asciende a más de 117,000, dicho modelo se analiza más a profundidad en la sección 4.3. Con el fin de poder establecer la mejora que representa el utilizar el algoritmo propuesto en este trabajo comparado con un esquema clásico de evaluación de transiciones, se plantea la siguiente prueba:

- Analizar una cantidad limitada de nodos en el árbol de alcance, para contrastar el tiempo de evaluación contra el esquema G&P :

Tabla 2-1: Comparativa de algoritmos de evaluación

EVALUACIÓN	NO. EVALUACIONES EN EL PEOR ESCENARIO	NÓDOS EXPLORADOS	TIEMPO DE EXPLORACIÓN (REAL)	REDUCCIÓN OBTENIDA
Genera y Evalúa (G&P)	1296 eval.	200	50.4 seg.	--
Algoritmo CLP	300 eval.	200	11.6 seg.	77%

En la Tabla 2-1 se puede observar que existe una reducción considerable en el tiempo de evaluación al implementar el algoritmo descrito en este trabajo.

Es importante hacer notar que el tiempo que se presenta en esta tabla, implica toda la operación que requiere el simulador para llevar a cabo el desempeño (búsqueda de nodos, generación de nuevos estados en el árbol, y toda la secuenciación de la lectura del árbol).
Se puede concluir que una reducción de 77% es una reducción global muy importante para lograr el objetivo particular de implementar un mecanismo de evaluación de transiciones eficiente para poder desarrollar un simulador que genere soluciones factibles en tiempos de cómputo razonables.

2.4 Administración de la Información del Árbol de Alcance

El principal problema a resolver cuando se desarrollan herramientas basadas en el árbol de alcance, es la explosión de estados (ver sección 1.8). Dada las ventajas que confiere la exploración de estados a la toma de decisiones en diversas áreas y en particular dentro del modelado industrial, diferentes autores han abordado la simulación de sistemas aplicando diferentes enfoques para el tratamiento del tamaño de estados:

- Almacenando sub espacios del árbol de alcance que a partir de una medida de avance del modelo resultan útiles para el análisis posterior, y desechando todos los restantes [51].
- Detección eficiente de estados utilizando principios de Hashing [49].
- Compactando estados tomando en cuenta las características dinámicas del sistema modelado [19].

Estos enfoques presentan limitaciones dependiendo de las características de modelado y de los objetivos para los cuales fueron diseñados.
Como se mencionó en la sección 1.8, existen diferentes maneras de implementar el codificado de la información por lo que el tamaño de almacenaje varía dependiendo de la forma como se realiza el codificado. Con el objetivo de poder desarrollar una herramienta que permita simular el comportamiento de un sistema así como la optimización de los mismos, en este trabajo se aborda el problema de la explosión de estados con un enfoque que almacena todos los estados optimizando el tamaño ocupado para definir un marcado del árbol. En este trabajo se propone una descomposición de la información de los estados en dos niveles de agrupación, la cual se lleva a cabo tomando en cuenta la naturaleza de los modelos y permite por un lado minimizar el espacio físico del árbol de alcance y por otro mejorar el desempeño en etapas de búsqueda de estados.
En el primer nivel de agrupación, se condensa la información referente a la combinación de colores que aparecen en los lugares del modelo de la RdPC. El segundo nivel contiene la información necesaria para definir el marcado del modelo de la RdPC así como la relación de descendencia entre marcados. Cuando se tratan de redes con extensión temporal se almacenan los atributos temporales de los tokens referente al marcado (reloj global y sellos de tiempo) en este mismo nivel.

2.4.1 Nivel Color de Información: agrupación por colores

El *principio de vecindad* en RdPC implica que el disparo de una transición modificará únicamente la información de los lugares adyacentes a la transición que se dispara (lugares de entrada y salida),

por lo que aprovechando esta propiedad, se agrupan las marcas que aparecen en los lugares a medida que evoluciona la exploración.
Cuando se van generando los diferentes estados del árbol de alcance, la diferencia entre estados viene determinada por el número y tipo de tokens que se genera o se elimina de los lugares afectados, por lo que en los lugares restantes del modelo la información permanece inalterada.

Ejemplo 2.6 Variado en el marcado de una RdPC

En la figura 2-7 se presenta un modelo en RdPC junto con parte del estado de alcance del modelo.

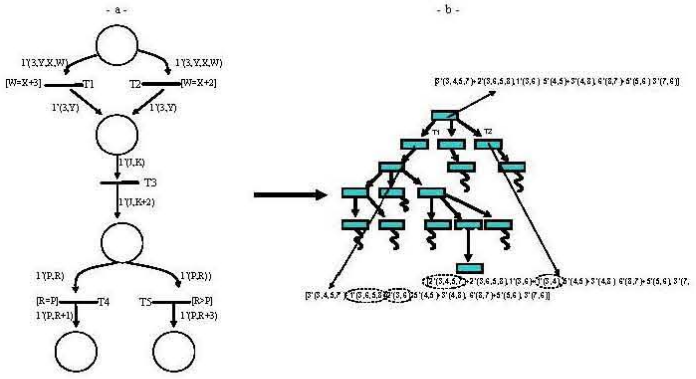


Figura 2-7: Información de un árbol de alcance típico

Para este modelo se tiene un estado inicial:

[3'(3,4,5,7)+2'(3,6,5,8),1'(3,6), 5'(4,5)+3'(4,8), 6'(8,7)+5'(5,6), 3'(7,6)]

El cual presenta diferentes transiciones habilitadas: T_1, T_2, T_3, T_5
Atendiendo al árbol de alcance figura 2-7b, al disparar la transición T_1 utilizando el token 1'(3,6,5,8), se genera un nuevo estado cuyo marcado es:

[3'(3,4,5,7)+1'(3,6,5,8),2'(3,6), 5'(4,5)+3'(4,8), 6'(8,7)+5'(5,6), 3'(7,6)]

Si de la misma manera, se dispara la transición T_2 utilizando el token 1'(3,4,5,7), se genera el nuevo estado:

[2'(3,4,5,7)+2'(3,6,5,8),1'(3,6)+1'(3,4), 5'(4,5)+3'(4,8), 6'(8,7)+5'(5,6), 3'(7,6)]

Dado que la misma situación ocurre con las transiciones T_3 y T_5, se observa que la mayor parte de la información del marcado permanece inalterada (ver zona punteada Figura 2-7b) y únicamente el cambio se da cuando se genera un nuevo o nuevos tokens cuya combinación de colores anteriormente no existía o cuando se genera uno o varios tokens con alguna combinación de colores que ya anteriormente había aparecido.
Aprovechando estas características, se estructura la información que aparece en los lugares del modelo en dos bloques de información, lo que se denominará de aquí en adelante *nivel color* de información:

- Cardinalidades. La cantidad de tokens (representada en números enteros) que aparecen con una misma combinación de colores se almacena en este bloque de información.
- Colores. Es un bloque de información para cada tipo de color que aparece en un lugar, el cual tiene relación con el bloque de cardinalidades.

Mediante esta descomposición se pretende un ahorro de espacio de memoria al almacenar en una sola ocasión la combinación de colores que aparece en un lugar, utilizando una sola estructura de datos para cada combinación de colores y otra independiente para las cardinalidades que aparecen conforme se van explorando los estados del árbol. La información almacenada para cada uno de los lugares es independiente entre sí.
Implementando este arreglo de datos para cada uno de los lugares pertenecientes al modelo, la reducción en el consumo de memoria es considerablemente menor si se compara a un arreglo donde se almacene la información de cada estado independiente. La Figura 2-8 muestra la idea de la agrupación por colores realizada.

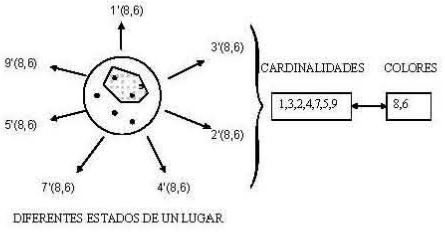


Figura 2-8: El agrupamiento de diferentes estados de un nodo lugar

Se observa que para un mismo lugar del modelo, es posible tener diferentes cantidades de tokens con una combinación de colores única, en este ejemplo para una combinación de colores (8,6).
En simulación de sistemas industriales se modelan comúnmente stocks, buffers, colas de materiales, etc. [103]. Dado que son recursos con capacidad limitada, el estado de los mismos es común modelarlo con entidades que se diferencian únicamente por los atributos de la entidad y por la cantidad de entidades presentes, por lo que esta estructura de información resulta adecuada para optimizar el espacio de memoria utilizado por modelos de sistemas que presentan características de este tipo.

2.4.2 Nivel Relación de Información: marcado del modelo

El *nivel relación* es otra estructura de información que se utiliza para especificar el estado del modelo a través del establecimiento de relaciones entre los nodos lugar, para lo cual se almacena en este nivel la información siguiente para cada marcado:

- Relación entre marcas de cada lugar. La información almacenada para los lugares en el nivel color es independiente entre ellos, por lo que se define la relación entre las marcas correspondientes de cada lugar a través de la posición de cada marcado (combinación de colores) en la estructura de almacenaje del nivel color para cada uno de los lugares.
- Cardinalidad utilizada. Es necesario especificar también la cantidad de tokens presentes en cada lugar, por lo que debido a que las cardinalidades se encuentran agrupadas en bloques de información se especifica que cardinalidad se utiliza para la combinación de colores de un marcado en particular.
- Reloj Global y Sellos de tiempo[8]. Es necesario definir el instante de tiempo en el cual el modelo del sistema presenta un marcado en particular, así como los sellos de tiempo de los tokens que definen al marcado.

Figura 2-9: La relaciones entre los niveles de información

En la Figura 2-9 se puede observar la relación que se establece entre los dos niveles. El nivel relación hace referencia a la información contenida en el nivel color (el que almacena las marcas que aparecen en los lugares del modelo). Por lo que la especificación de los estados del modelo se lleva a cabo en el nivel relación.

Ejemplo 2.7 Implementación de la relación entre niveles.

Se tiene un modelo en RdPC como el mostrado en la figura 2-7a.
El estado inicial del modelo corresponde a:

Estado 1: [3'(3,4,5,7)+2'(3,6,5,8),1'(3,6), 5'(4,5)+3'(4,8), 6'(8,7)+5'(5,6), 3'(7,6)]

La Figura 2-10a ilustra cómo se almacena la información del marcado para los lugares del modelo. Se puede observar (ver Figura 2-7a) que las transiciones T_1, T_2, T_3 y T_5 se encuentran habilitadas, por lo que llevando a cabo el disparo:

[8] Solamente cuando se utilicen RdPC temporales

T1: 1'(3,6,5,8)

Se genera un nuevo estado del modelo definido por el marcado:

Estado 2: [3'(3,4,5,7)+1'(3,6,5,8),2'(3,6), 5'(4,5)+3'(4,8), 6'(8,7)+5'(5,6), 3'(7,6)]

Este marcado, se descompone en los siguientes estados para los lugares del modelo[9]:

P_1 :	Cardinalidad= 3,1	Colores=3,4,5,7&3,6,5,8
P_2 :	Cardinalidad– 2	Colores–3,6
P_3 :	Cardinalidad=5,3	Colores=4,5&4,8
P_4 :	Cardinalidad= 6,5	Colores=8,7&5,6
P_5 :	Cardinalidad= 3	Colores=7,6

La Figura 2-10b muestra como se almacena la información generada por el disparo teniendo en cuenta que ya existía información previa del modelo. Se puede observar que para este caso solamente se modifican los campos que almacenan las cardinalidades del lugar P_1 y P_2.
A partir de este nuevo estado, es posible disparar la transición T_1 nuevamente:

T1: 1'(3,6,5,8)

Generando el estado:

Estado 3: [3'(3,4,5,7) , 3'(3,6) , 5'(4,5) + 3'(4,8) , 6'(8,7) + 5'(5,6) , 3'(7,6)]

El cual se descompone en los siguientes elementos:

P_1 :	Cardinalidad= 3	Colores=3,4,5,7
P_2 :	Cardinalidad= 3	Colores=3,6
P_3 :	Cardinalidad=5,3	Colores=4,5&4,8
P_4 :	Cardinalidad= 6,5	Colores=8,7&5,6
P_5 :	Cardinalidad= 3	Colores=7,6

En este caso se puede observar (Figura 2-10c) que con la generación de este estado se modifican las cardinalidades del lugar P_2 y se añade una nueva combinación de colores que antes no existía para el lugar P_1.

[9] El símbolo & se utiliza en este trabajo como separador de elementos

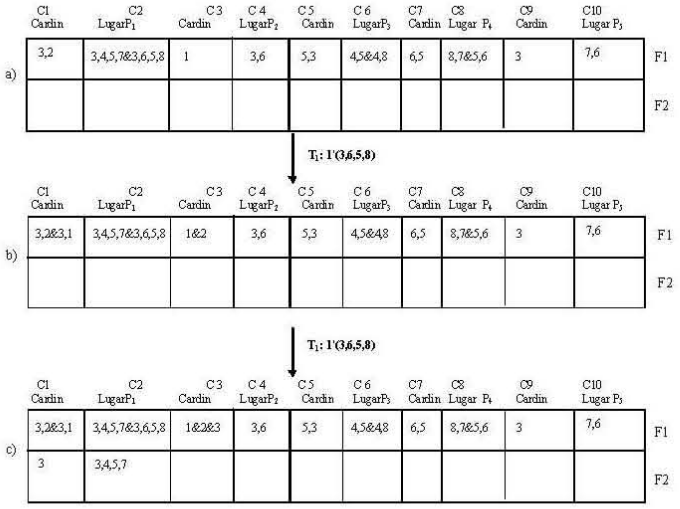


Figura 2-10: Evolución de marcado en los lugares del modelo

Se puede observar que al realizarse los disparos, la información de los lugares P_3 a P_5 resulta inalterada, por lo que no es necesario añadir información a la previamente almacenada en estos lugares.

Para la definición de un estado del modelo se especifica la relación entre lugares, así como las cardinalidades utilizadas por las marcas de los lugares del modelo.
Para los ejemplos mostrados, la relación se especifica por la posición de las celdas que contienen la combinación de colores del marcado correspondiente[10]:

Estado 1: F1C2-F1C4-F1C6-F1C8-F1C10
Estado 2: F1C2-F1C4-F1C6-F1C8-F1C10
Estado 3: F2C2-F1C4-F1C6-F1C8-F1C10

La cardinalidad correspondiente de estos estados se especifica para poder diferenciar estados que tienen la misma combinación de colores pero diferente cardinalidad (estado 1 y 2):

Cardinalidad 1: 3,2&1&5,3&6,5&3
Cardinalidad 2: 3,1&2&5,3&6,5&3
Cardinalidad 3: 3&3&5,3&6,5&3

[10] La posición especificada como filas y columnas.

Cualquier marcado en el árbol de alcance se puede definir utilizando la cardinalidad con su correspondiente relación de colores. Este mismo proceso es llevado a cabo cada vez que se genera un nuevo estado del modelo.

Debido a la forma de agrupación de la información resulta práctico colocar la información en forma de matrices bidimensionales. Una matriz bidimensional representará la información del nivel color y la segunda matriz para el nivel relación en forma de coordenadas (fila y columna) referidas a la matriz del 1er nivel (Figura 2-11).

Fila1	1,1,1,1,1,1	11,2,1-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3	1,1,1,1,1,1,1,1,1,1,1	11,0,3-12,1,6-13,3,7-14,5,3-15,4,6-16,8,1-21,2,5-22,4,10-23,5,10-24,0,10	1,1,1,1,1,1	0-1-2-3-4-5
Fila2	1,1,1,1,1,1	12,0,3-21,1,8-31,2,5-41,1,5-51,2,9-61,1,3				
Fila3	1,1,1,1,1,1	11,2,1-22,2,5-31,2,5-41,1,5-51,2,9-61,1,3				
Fila4	1,1,1,1,1,1	11,2,1-21,1,8-32,3,4-41,1,5-51,2,9-61,1,3				
Fila5	1,1,1,1,1,1	11,2,1-21,1,8-31,2,5-42,0,5-51,2,9-61,1,3				

Nivel Color- Matriz 1

Col1 (cardinalidad)	Col2 relación de lugares	Col3 Origen	Col4 Hijos	Col5	Col6
1,1,1,1,1,1&1,1,1,1,1,1,1,1,1,1&1,1,1,1,1,1	C2F1-C4F1-C6F1	Null	C2F2,C2F3		
1,1,1,1,1,1&1,1,1,1,1,1,1,1,1,1&1,1,1,1,1,1	C2F3-C4F1-C6F1	C2F1			
1,1,1,1,1,1&1,1,1,1,1,1,1,1,1,1&1,1,1,1,1,1	C2F4-C4F1-C6F1	C2F1			

Nivel Relación – Matriz 2

Figura 2-11: Las matrices de almacenaje de información

Este enfoque para la gestión de información permitirá tener el árbol de alcance en una estructura de listas que será más eficiente para la detección y análisis de estados repetidos.

2.4.3 Evaluación Experimental de la Eficiencia de Almacenado

Para poder lleva a cabo la evaluación de la eficiencia de almacenado del simulador se implementó un modelo en RdPC temporales como el mostrado en la Figura 2-12.

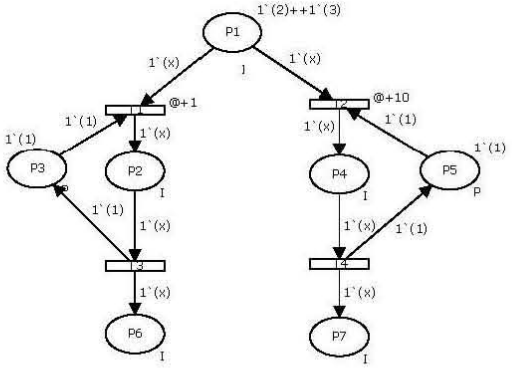


Figura 2-12: Modelo en RdPC temporales de un sistema de manufactura

Los tokens presentes en el nodo lugar P1 representan las entidades o materias primas que deben ser procesadas. Los tokens presentes en los nodos lugar P3 y P5 representan la disponibilidad de las máquinas de procesamiento. En este caso las materias primas pueden ser procesadas a través de dos procesos. Los procesos se distinguen esencialmente que uno es más lento que otro (situación común en la industria). El proceso #1 se inicia con el disparo de la transición T1; se extrae un token del nodo lugar P1 y P3 y se añade un token al nodo lugar P2; el tiempo consumido por esta actividad es de 1 unidad de tiempo, la cual se encuentra especificada en el modelo (@+1 en T1). Al llevarse a cabo este disparo, la máquina que se modela con el token presente en el nodo lugar P3 pierde disponibilidad hasta que es liberada, situación que se modela con el disparo de la transición T3. El proceso #2 es similar al proceso #1 con la excepción que en este caso el tiempo consumido por la actividad es de 10 unidades de tiempo (@+10 en T2). En el caso de las transiciones T3 y T4 no se les asoció ningún tiempo consumido debido a que en este ejemplo todo el tiempo asociado al proceso de manufactura se asocia a las transiciones que requieren las máquinas.

Diferentes espacios de estados fueron generados con este modelo variando únicamente la carga de trabajo inicial. La información del espacio de estados se almacenó en una hoja de cálculo para poder evaluar la diferencia en memoria utilizada para el almacenado de la información.

La Tabla 2-2 resume la información obtenida al generar los diferentes espacios de estados para este modelo. Por un lado se calculó la cantidad de memoria utilizada por el espacio de estados sin utilizar ningún tipo de mecanismos de compactación de información. Posteriormente se generaron los mismos espacios de estados pero utilizando la administración de información propuesta en la sección anterior.

Tabla 2-2: Comparación del almacenado de la información

No. Productos	Tokens Iniciales(P1)	Número estimado de nodos en el espacio de estados	Kbytes Utilizados	Número de nodos en el espacio de estados temporal.	Kbytes utilizados	Reducción obtenida
8	4'(2)+4'(3)	1,820	253 kb	1,205	244 kb	3.5%
10	5'(2)+5'(3)	6,660	931 kb	2,571	519 kb	44.25%
14	7'(2)+7'(3)	130,700	16,550 kb	8,408	1,708 kb	89.67%
20	10'(2)+10'(3)	1,400,000	177,300 kb	30,866	6,444 kb	96.36%

Se puede apreciar que haciendo uso de las técnicas de almacenado presentadas la cantidad de recursos utilizados para el almacenado (memoria de CPU) se ven reducidas considerablemente. Lo anterior se puede apreciar más adecuadamente con una representación gráfica del número de productos vs la cantidad de memoria utilizada para el almacenado. La Figura 2-13 es una representación gráfica de tipo semi-log para poder apreciar que con la implementación el crecimiento exponencial de estados se ve reducido considerablemente.

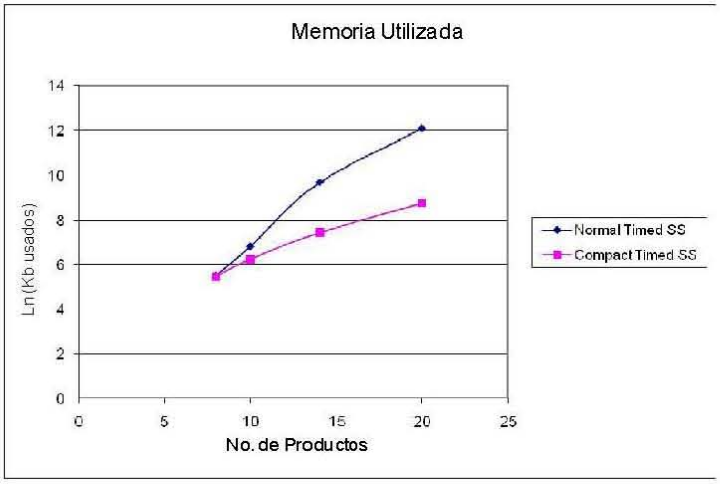


Figura 2-13: Uso de memoria para el modelo de manufactura en RdPC

Desafortunadamente el crecimiento exponencial no puede ser mitigado, pero con el uso de este tipo de administración de información es posible llevar a cabo el análisis de espacios de estados más grandes (en número de nodos) que los que se podrían analizar con un enfoque más tradicional.

2.5 Análisis y Búsqueda de los Nodos Repetidos

Debido a la explosión de estados, en el modelado industrial con RdPC no es recomendable el almacenar la información de un estado repetido cada vez que aparece, ya que la memoria física del ordenador se saturaría rápidamente. Para no repetir información, es necesario detectar de una manera eficiente el almacenado previo de un estado, por lo que esta actividad cobra gran importancia para lograr el desempeño eficiente de un simulador basado en la exploración del árbol de alcance.
La eficiencia de detección depende principalmente de dos cosas, el tamaño de la información a analizar y el mecanismo utilizado para llevar a cabo el análisis de la misma.
Para resolver la problemática planteada, es necesario realizar una búsqueda eficiente que detecte cuando un nodo recién generado ha aparecido previamente, almacenando al mismo tiempo toda la información que aparece en el árbol de alcance y accediendo de manera rápida a la información almacenada en el árbol.

2.5.1 Detección de Estados Nuevos

Atendiendo a la estructura de información planteada en el apartado 2.4 para el almacenado del espacio de estados, cada vez que se dispara una transición, se genera una serie de tokens con una combinación de colores para los lugares de salida y de entrada. Para cada marcado que se genera a partir del disparo, se realiza una búsqueda, utilizando la información del nivel color, sobre los lugares de entrada y los de salida para determinar si la marca del lugar ya ha sido generada previamente, lo cual presenta ventajas como desventajas, las cuales se enumeran a continuación:

Ventajas:

- Debido a la estructura de la información no se realizaran búsquedas sobre todos los lugares del modelo, solamente sobre los lugares conectados a la transición (ver Figura 2-14).
- Dado que el estado del sistema se encuentra definido por las relaciones de marcas existentes en los lugares, al analizar la información de lugares a *nivel color*, se determina que un estado es nuevo si una marca generada para un lugar no se encuentra previamente almacenada en este nivel. Evitando de esta manera una búsqueda en el *nivel relación* de información.

Desventajas:

- Debido a que se realiza una búsqueda para cada lugar conectado con la transición, el tiempo de búsqueda tiene una relación de proporcionalidad directa con la cantidad de lugares implicados en el disparo.

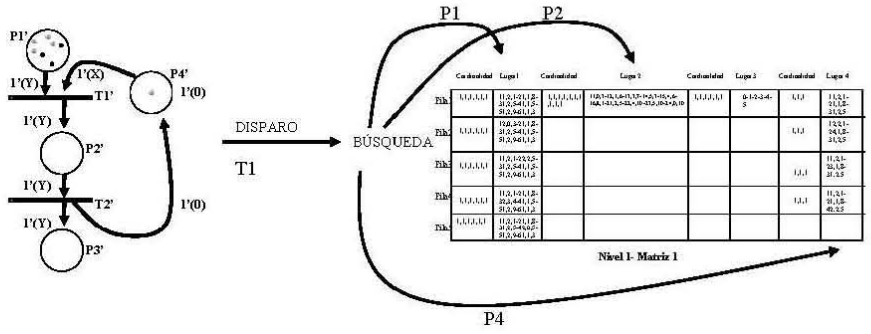


Figura 2-14: Búsqueda de estados en los lugares

En el caso que a nivel color todas las marcas generadas por el disparo se encuentren previamente almacenadas, no es posible asegurar de antemano que sea un estado repetido, por lo que para establecer si es un nuevo estado o uno repetido se realiza una búsqueda a *nivel relación* de información.

2.5.2 Implementación de la Búsqueda.

Una medida para evaluar la *complejidad computacional* es por un lado el tiempo de ejecución, sin embargo, dado que es una medida que depende del tipo de computadora, lenguaje de programación, la habilidad del programador y otras características, resulta difícil el realizar la comparación de algoritmos solamente en base al tiempo de ejecución.

Una medida apropiada para evaluar el funcionamiento es contar el número de pasos que el algoritmo debe llevar a cabo como función del tamaño del problema que se define. Se dice que un algoritmo tarda o tiene tiempo *de orden f(n)*, donde *f* es una función dada, si existe una constante positiva *c* [11]y una implementación del algoritmo capaz de resolver cualquier instancia del problema en un tiempo acotado por *cf(n)*, donde *n* es el tamaño de la instancia. Se denota como *O(f(n)) y* se denomina *complejidad temporal* del algoritmo [36].

La actividad de realizar búsquedas, ya sea a nivel color o relación, es llevada a cabo de manera exhaustiva por el simulador como se puede observar en la Tabla 2-3.

[11] La constante c esta basada en el principio de invarianza algorítmica [36]

Tabla 2-3: Evaluación de Búsquedas en el árbol de alcance.

MODELO	NO. ESTADOS EXPLORADOS	NO. DE ESTADOS REPETIDOS EVALUADOS	NO DE BÚSQUEDAS REALIZADAS
3x3	693	680	10,204
5x5	7,776	7,750	121,825
6x6	117,650	117,612	2,302,579

Se reportan en la literatura diferentes métodos de búsqueda los cuales presentan diferentes complejidades [73]:

Búsqueda secuencial: O(N)
Búsqueda Binaria: O (Log N)
Búsqueda Interpolada: O(Log Log N)
Métodos Hashing: Orden variable

La búsqueda binaria esta reportada como un algoritmo óptimo cuando restringimos las operaciones únicamente a comparaciones entre valores, además presenta las características siguientes [43]:

- Mantiene siempre un orden logarítmico si se utiliza solo para comparación de valores.
- Tiene la ventaja que funciona de manera eficiente para búsqueda de rangos de datos como para la búsqueda de un dato único.
- Es un algoritmo muy estable: el rango del tiempo de búsqueda se mantiene muy cercano al tiempo promedio de búsqueda, y la varianza de los tiempos de búsqueda es O(1).
- Una desventaja es que requiere que los elementos se encuentren ordenados.

Para la realización de búsquedas de datos durante la generación y análisis de estados, el algoritmo cuenta con búsquedas binarias cuyas características se presentan a continuación:

- Copias ordenadas del nivel color de información y del nivel relación.
- La información estructural del árbol se mantiene en las copias originales de los dos niveles de información, y se utilizan copias ordenadas de la información para realizar las búsquedas.
- Las copias ordenadas son dinámicas en la medida que se modifican a medida que se añaden elementos nuevos.
- A medida que se añade información a la copia ordenada, se mantiene una referencia a la posición de la información en la copia estática (que contiene la estructura del árbol de alcance).
- Las copias ordenadas se utilizan para encontrar los elementos que han aparecido previamente durante la exploración de los estados del modelo.

La Figura 2-15 presenta un ejemplo de la implementación de la búsqueda binaria para el nivel color de información.

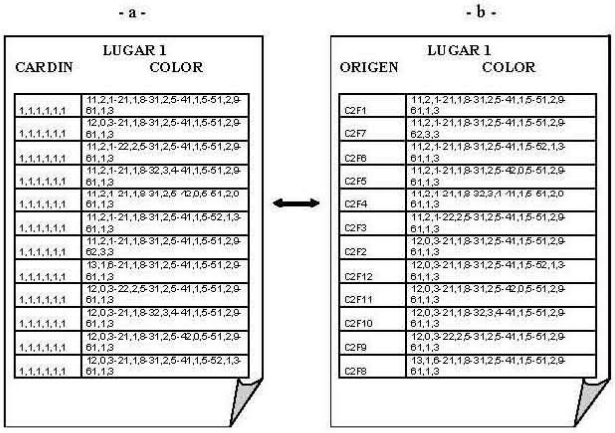


Figura 2-15: El manejo de la información en listas

En la Figura 2-15a se pueden observar las columnas 1 y 2 de la estructura estática donde se almacena la información del lugar P_1, esta información se encuentra almacenada de manera secuencial a medida que aparece durante la generación del árbol de alcance. Por otra parte la tabla de la Figura 2-15b contiene la misma información pero en este caso se encuentra ordenada en base a un orden lexicográfico de la cadena de caracteres con la información de los colores del lugar P_1 de un modelo cualquiera.

2.5.3 Rendimiento de la búsqueda.

La implementación de búsquedas binarias incrementa por un lado el uso de memoria (debido a las copias dinámicas), pero por otro lado disminuye drásticamente el tiempo de búsqueda del simulador debido a la dependencia logarítmica con los elementos de las listas, lo que contrasta con una búsqueda puramente secuencial. En la Tabla 2-4 se muestran los tiempos de exploración para un taller de trabajo (T-T) 6x6 con el fin de comparar el desempeño del simulador con una búsqueda secuencial exhaustiva contra las binarias implementadas.

Tabla 2-4: Comparación entre los tipos de búsquedas implementadas

TIPO DE BÚSQUEDA UTILIZADA	NO. NODOS EXPLORADOS	TEMPO CONSUMIDO	REDUCCIÓN
Búsqueda Secuencial	117,600	5.5 hrs.	--
Búsqueda Binaria	117,600	1.5 hrs.	72.7%

La reducción de tiempos reportada se refiere al tiempo consumido para explorar todos los estados del modelo desarrollado en RdPC.

2.6 El uso del Tiempo en la Simulación.

La mayoría de las aplicaciones de las RdPC son utilizadas para verificar el funcionamiento lógico y estructural de un sistema analizando las propiedades dinámicas y funcionalidades del mismo. Las RdPC pueden ser utilizadas también para la mejora del rendimiento de sistemas, como puede ser el tiempo total de un proceso de fabricación de un producto en un sistema de producción (*makespan*), el tiempo de espera de elementos en alguna cola de producción, etc. Para realizar este análisis es importante extender el uso del tiempo a las redes de Petri coloreadas utilizando las redes de Petri coloreadas temporales.

2.6.1 Reglas de Disparo con Sellos de Tiempo

Al hacer uso de la extensión temporal, es importante definir las reglas para decidir el avance del reloj global de la simulación así como las reglas de evaluación de los sellos de tiempo de los tokens que intervienen en el marcado en el momento de llevar a cabo la habilitación de las marcas. Como se mencionó en la sección 1.5.2 cada marcado en el árbol de alcance tiene asociado un reloj global, y cada que se genera un nuevo nodo del árbol a partir de un estado se asigna el reloj global que le corresponde, el cual se calcula a partir de los sellos de tiempo que habilitan por color la transición. Debido a esto el reloj global del modelo dependerá del evento disparado.[12]

Las reglas utilizadas en este trabajo son las siguientes:

[12] *Event Driven Simulation*

- Para disparar una marca, se evalúan cada una de las transiciones para determinar cual o cuales se encuentran activas en base a las reglas de las RdPC sin tiempo. Las transiciones activas se dispararan todas de manera independiente.
- Al momento de llevar a cabo el disparo de una transición, se analiza el sello de tiempo de la combinación de tokens que habilita la transición, y se selecciona aquel que tenga el sello de tiempo más pequeño de entre ellos.
- Si el sello de tiempo del token seleccionado es mayor que el reloj global de la simulación, entonces se adelanta el reloj global hasta igualarlo con el valor del sello de tiempo y este tiempo es el que se asocia a la nueva marca.

Haciendo uso de estas reglas, el reloj global avanza lo menos posible en cada disparo.

Estas reglas se expresan de manera formal con la siguiente fórmula:

$$Avance\_del\_\mathrm{Re}loj = Max\left\{\left|\mathrm{Re}loj\_Global - Sello\_Tiempo_i\right| \ni i = 1..n\right\} \qquad (2.2)$$

Donde:
Reloj_Global: Es el valor del reloj global del marcado en evaluación.
Sello_Tiempo$_i$: Es el valor del sello de tiempo del token *i* del total de *n* tokens que habilitan por color una transición.
Dado que la diferencia cuando un sello de tiempo es mayor que el reloj global siempre será negativa, el grado de avance del reloj global para el estado que se genera se debe calcular obteniendo el valor absoluto de la diferencia calculada.
Utilizando esta regla, cada marcado del árbol de alcance tendrá un reloj asociado el cual representa el tiempo mínimo en que es posible utilizar todos los tokens que habilitan por color el marcado correspondiente.
Para ilustrar lo anterior observemos la Figura 2-16.

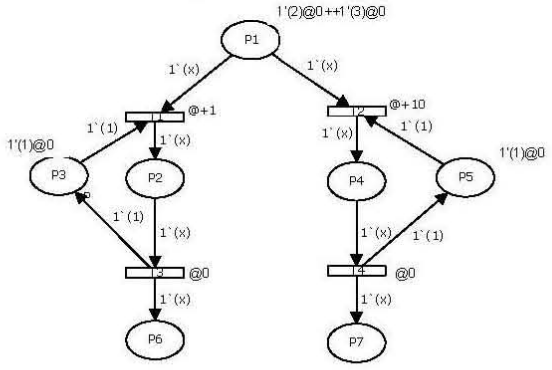


Figura 2-16: Modelo en RdPCT con transiciones en conflicto

Este modelo representa el sistema mencionado anteriormente donde se tienen en conflicto dos transiciones que se encuentran habilitadas al mismo tiempo y cada una puede ocurrir al instante 0 de tiempo. Este puede ser el caso de productos que se necesitan procesar y que pueden llevarse a cabo a través de dos procesos distintos donde la diferencia entre los procesos es debida al tiempo de procesamiento (dicha situación suele ocurrir cuando se adquieren máquinas nuevas en un sistema de manufactura). El proceso más rápido se encuentra modelado por las transiciones T1 y T3, y el proceso más lento por las transiciones T2 y T4. Dado que las transiciones T1 y T2 se encuentran activadas, cualquiera de las dos se puede disparar modificando de esta manera el estado del modelo; los disparos pueden llevarse a cabo con cualquiera de los tokens presentes en el nodo P1, ya sea un token 1'(2) o uno 1'(3) debido a que en este caso no existe restricción asociada al color.

Si se dispara la transición T2 a tiempo 0 haciendo uso de un token de color 3 proveniente del nodo lugar P1 con el token del nodo lugar P5 (T2@0: 1'(3)@0, 1'(1)@0) el modelo se ve modificado obteniéndose el modelo representado por la Figura 2-17 y el sello de tiempo del token de salida se calcula haciendo uso de la fórmula (1.2). En este caso el valor del reloj global del nuevo estado se calcula con la fórmula (2.2) el cual corresponde a 0 unidades de tiempo,

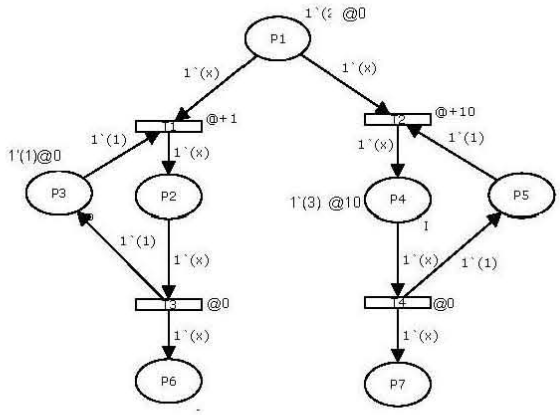


Figura 2-17: Modelo resultante del disparo de la transición T2

En este caso el nuevo token generado en el nodo lugar P4, 1'(3)@10, quedará disponible nuevamente cuando el reloj de la simulación alcance el valor de 10 unidades de tiempo. Si se disparará la transición T4 a partir de este nuevo estado el reloj global correspondiente sería de 10 unidades de tiempo. Calculando de esta manera el espacio de estados para este modelo quedaría como el representado por la Figura 2-18

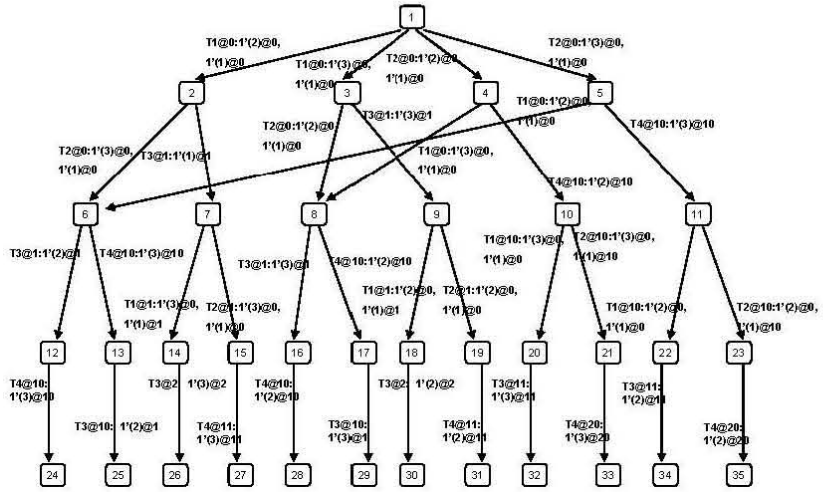


Figura 2-18: Espacio de estados del modelo en RDPCT

En esta figura el nodo 5 representa el estado del modelo de la Figura 2-17. Las inscripciones presentes en los arcos del árbol de alcance representan el tiempo global de disparo y la combinación de tokens utilizada para llevar a cabo el mismo p. ej. el nodo 5 se genera por disparar a tiempo 0 la transición T2: T2@0; con un token de color 3 con sello de tiempo 0 y un token de color 1 con sello de tiempo 0: 1'(3)@0, 1'(1)@0.
De manera similar se obtienen los diferentes estados que puede alcanzar el modelo donde todos los posibles estados finales del modelo están representados por los nodos 24 a 35.

2.6.2 Optimización Temporal de Modelos de RdPC

Dado que en los sistemas industriales el desempeño temporal adquiere gran valor (tiempo de proceso, tiempo en colas, etc.), es importante el análisis temporal del mismo con el objetivo de entender, y en muchos casos optimizar su funcionamiento. La evolución temporal del sistema es posible analizarla al descender a través del árbol de alcance partir del estado inicial del mismo hasta alcanzar un nodo particular del mismo donde cada camino que llega hasta un nodo objetivo correspondería a una réplica de simulación del sistema.
Con la comparación de los valores temporales del nodo objetivo es posible seleccionar el camino que genera los mejores valores para el estado objetivo de interés; en este trabajo se utiliza el

análisis del árbol de alcance con el objetivo de llevar a cabo una optimización temporal de las redes de Petri coloreadas temporales.

Para la optimización del rendimiento de sistemas industriales, un factor muy importante que relaciona tiempos, costos de proceso y eficiencia en general es común utilizar el tiempo total de procesamiento o *makespan*. En este trabajo se utiliza el valor del tiempo total de proceso como función de coste para llevar a cabo la optimización de los modelos industriales. La idea principal es utilizar el árbol de alcance como espacio de búsqueda para desarrollar algoritmos que optimicen dicha función de coste.

3 ALGORITMO DE SIMULACIÓN/OPTIMIZACIÓN

Haciendo uso de los algoritmos desarrollados para la evaluación eficiente de transiciones así como las búsquedas eficientes ha sido desarrollado un algoritmo que permite la generación automática de estados de un modelo en RdPC tanto temporales como atemporales, así como la optimización temporal analizando los nodos repetidos generados durante la exploración del árbol. En esta sección se presentan los algoritmos para la implementación del algoritmo desarrollado para la optimización del tiempo total de proceso de sistemas industriales.

En la primera etapa del algoritmo la cual se denomina etapa de *generación de estados* se generan todos los posibles estados del sistema, almacenando en una lista independiente todos los nodos repetidos que aparecen.

La segunda etapa del algoritmo denominada *etapa de optimización* hace uso de la lista con los nodos repetidos analizando las características temporales (sellos de tiempo y reloj global) y las compara contra la información almacenada en el árbol para decidir cuales tiempos son mejores en base a la función temporal de coste.

La implementación del algoritmo se implementó codificando un entorno para pruebas en Visual Basic. net

3.1 Etapa de Generación de Estados

Durante esta etapa, se genera (siempre que la memoria física del ordenador lo permita) todo el árbol de alcance utilizando las reglas de las RdPC previamente mencionadas. En los casos que la memoria física del sistema impida la generación de toda la información del árbol de alcance se lleva a cabo una generación parcial y se analizan los estados repetidos del modelo.

El algoritmo utilizado para ir generando el árbol de alcance es el conocido como búsqueda en profundidad (*depth-first search)*. Al utilizar la búsqueda en profundidad, se comienza con el nodo raíz, que en el caso de las RdPC será el estado inicial del sistema, y se evaluarán los hijos que se generan en cada nivel, siempre descendiendo en una rama tanto como sea posible antes de retroceder y evaluar otra rama.

La búsqueda en profundidad es una búsqueda que progresivamente desciende en los niveles del árbol de alcance analizando en cada ocasión un estado hijo del nodo evaluado en el nivel previo. El análisis de estados va descendiendo a través de los niveles hasta que un nodo objetivo es encontrado o hasta que el estado en evaluación no genera más estados hijos. Al llegar a este punto, la búsqueda retrocede hasta el nodo mas reciente que contenga algún nodo hijo no explorado.

La etapa de generación contiene los procedimientos que permiten generar automáticamente todos los posibles estados del sistema:

- Construcción de la estructura de la red de Petri en memoria
- Lectura del estado inicial del sistema
- Evaluación de Transiciones
- Generación y almacenado de estados hijos.
- Visualización de Resultados
- Análisis de Resultados de la Simulación

3.1.1 Estructuras de Datos

La definición de la estructura de la red de Petri coloreada se codifica utilizando hojas de cálculo como interfase de usuario a través de una sintaxis específica y permite desarrollar las siguientes estructuras que son utilizadas en los procesos básicos que componen el simulador.

TOKENS.
Esta estructura de datos se utiliza para representar los tokens de la red de Petri coloreada, cuenta con tres campos:

- CARDINALIDAD. Este campo utiliza números enteros, y representa el número de tokens presentes que comparten una misma combinación de colores.
- COL. Es un arreglo de enteros donde se definen los valores de los colores.
- T_STAMP. Es también un arreglo de enteros el cual tiene la dimensión de la cardinalidad y se utiliza para asignar el sello de tiempo de los tokens cuando se modelan redes temporales.

LUGAR.
Con esta estructura se representa el marcado de la RdPC, cuenta con el siguiente campo

- SUBLISTA. Es un arreglo dinámico de la estructura *tokens*. El marcado de cada lugar se modela con un arreglo de este tipo.

LUGARES.
Esta estructura es un arreglo de la estructura *lugar* y presenta tantos elementos como lugares tiene el modelo de RdPC

ARCOS.
Representa los arcos que unen un nodo lugar con nodo transición. Está diseñado con cuatro campos que se utilizan para las siguientes funcionalidades:

- APUNTADOR1. Es un entero cuyo valor es la posición en la lista de tokens disponibles en el lugar. El valor hacer referencia al token que habilita el disparo.
- APUNTADOR2. Es un valor entero el cual hace referencia al que token de la lista que se encuentra en evaluación.
- TOKENS ().Un arreglo de enteros que se utiliza para construir el subconjunto que satisface los colores constantes de las expresiones de arco (ver sección 2.3.2.1).
- TOKS_DYN().Un arreglo de enteros con las posiciones en la lista de los tokens que satisfacen las restricciones de guarda, este arreglo es dinámico debido que se va reduciendo su dimensión a medida que se satisfacen las diferentes restricciones.
- COMPARADOR_ENT. Es un campo con un elemento tipo *estructura_token*, el cual es utilizado para definir las expresiones de arco de entrada
- COMPARADOR_SAL. Es un campo con un elemento tipo *estructura_token*, el cual es utilizado para definir las expresiones de arco de salida.

Para acceder de manera rápida a cada arco, se utiliza un arreglo bidimensional en el que cada elemento es una estructura *ARCOS* y se accede con el número de transición y de lugar.

TRANSICIONES.
Esta estructura se utiliza para mantener información particular de cada transición a través del uso de cinco campos:

- TIME. Es un entero y representa el tiempo que consume la transición cuando se trata de redes temporales, en caso de redes de Petri atemporales este campo no se utiliza.
- ARC_SALIDA. Es un arreglo de enteros que almacena las posiciones de los lugares que salen de la transición.
- ARC_ENTRADA. Es un arreglo de enteros que almacena las posiciones de los lugares de entrada a la transición.
- EXPRESIÓN. Es una cadena de caracteres donde se encuentra codificada la expresión de guarda asociada a la transición. Puede ser una expresión simple o un arreglo de expresiones.
- GUARD1. Es una expresión *boleana* que se utiliza como *bandera* para indicar que la transición tiene guardas asociados.

VARIABLES.
Se utiliza para definir las variables que aparecen en el modelo y contiene diferentes campos que son utilizados para la implementación de los algoritmos de propagación de restricciones así como para los valores de los colores en los tokens de salida:

- NOMBRE. Es una cadena de caracteres, y se utiliza para identificar la variable
- VALOR. En un campo donde se asigna el valor que adquiere la variable.
- POS_COL. En un valor entero utilizado para definir la posición en el arreglo de colores que corresponde a la variable
- LUGAR_ORIG. Un valor entero que indica el número del lugar en el que aparece la variable.
- RELACION. Arreglo de los enteros identificadores de las variables con que se relaciona a través de la expresión de guarda.
- EXPRESION. Es una cadena de caracteres, con la cual se codifica la expresión del guarda asociada a la variable.
- VAL_MIN. Un número entero que expresa el valor mínimo del dominio de la variable.
- VAL_MAX. Un número entero que expresa el valor máximo del dominio de la variable.
- OPERADOR. Es un carácter, el cual indica la operación algebraica que se lleva a cabo en la expresión de guarda (=, <>, <,>).

La información de los campos que especifican la relación entre variables y sus correspondientes posiciones dentro de la estructura del modelo se especifican utilizando una tabla que contiene esta información. Un ejemplo de cómo se especifica la relación entre variables se muestra en la tabla 3-1 la cual define la información de los diferentes campos de las variables a partir de la estructura de la RdPC y los guardas presentes.

Tabla 3-1: Tabla de especificación de variables

VARIABLES A UTLIZAR						
Número de Variable	Identificador Variable	Expresión	Relación	Lugar de Origen	Posición Color	Operador
1	X	E	7	1	1	=
2	Y	Z	6	1	2	=
3	J	X+**1**	1	-1	1	=
4	W	W	4	2	2	=
5	P	P	5	2	3	=
6	Z	Y	2	3	1	=
7	E	X	1	2	1	=
8	R	R	8	1	3	=

La relación que se implementa en la tabla es para los siguientes guardas:

- X=E
- Y=Z

Se puede observar que la variable J, tiene como lugar de origen '-1', lo cual significa que la variable J pertenece al lugar de salida 1 (ver sintaxis en el anexo 8), cuya expresión es:

- X+1

3.1.2 Evaluación de Transiciones

Mediante las estructuras de datos *Arcos, Lugar, Variable, Transiciones* se ha codificado el método presentado en la sección 2.3. En la Figura 3-2 se presenta en pseudo código el algoritmo implementado para llevar a cabo la evaluación de transiciones.

```
INICIO
        ciclo1 ⟵ FALSO
        trans_time ⟵ 0
        cuenta_lug ⟵ 1
        CREAR Subconjuntos Iniciales ( SI no se pueden crear entonces SALIR)
        ASIGNAR de Maximos y Minimos Globales
        lugar_actual ⟵ Primer lugar de entrada
        ultimo_lugar ⟵ ultimo lugar de entrada
DO
        ciclo1 ⟵ FALSO
            WHILE Apuntador1(lugar_actual) <> ultmo token de Arcos(lugar_actual).toks_dyn
                mueve apuntador1
                BUSCAR VARIABLES e INICIALIZA VALORES en
                arcos(trans_num,lugar_actual).comparador_entrada
                SI cuenta_lug = 1 ENTONCES
                        INICAILIZAR TODAS LAS VARIABLES
                        INICIALIZAR SUBCONJUNTOS DINAMICOS
                FIN
            TOKEN_EVALUADO ⟵ EL token que apunta el APUNTADOR1 del SUBSET DINAMICO del LUGAR
            ACTUAL
           SI EXISTEN Variables ENTONCES
            INICIO
                REPETIR para todas las Variables encontradas:
                    INICIO
                        VAR_Origin ⟵ variable en evaluación
                        REPETIR para TODAS LAS VARS que se RELACIONAN con Var_Origin
                      INICIO
                                RESETEA VALOR de (Var_Origin)
                                VAR_propaga ⟵ Variable relacionada con origin a traves del GUARDA
                                Col_EValuado ⟵ posición de color de variable en evaluacion
                                DETECTAR si Var_Origin tiene Valor
                                        SI Var_Origin sin Valor ENTONCES
                                                VERIFICAR Sentido Propagacion
                                                SI Propagacion es Retroceso ENTONCES
                                                INICIO
                                                Verificar Restricciones
                                                FIN
                                                CREA SUBSET Dinamico
                                                SI no CREO Subset ENTONCES
                                                INICIO
                                                cambiar de token
                                                FIN
                                        ELSE
```

Figura 3-1: Pseudo código para la evaluación de transiciones

```
                                                    Verificar que no se VIOLAN restricciones
                                                FIN
                                            FIN
                                        FIN
                                                ELSE
                                                  Verificar comparador de entrada contra Subconjunto original
                                                FIN
                                    Ciclo1 ← VERDADERO
                                SI Ciclo1 = VERDADERO Y lugar_actual < ultimo lugar ENTONCES
                                  INICIO
                                  cuenta_lug ← cuenta_lug+1
                                   Ciclo1 ← FALSO
                                  FIN
                                 SI cuenta_lug> Numero de lugares de entrada ENTONCES Cuenta_lug=cuenta_lug – 1
                                 Lugar_actual  ← Lugar de entrada nc. (cuenta_lug)
                                 SI lugar_actual  = ult_lugar AND ciclo1=VERDADERO ENTONCES
                                    INICIO
                                       SI TRANSICION(trans_num). guard1 = VERDADERO ENTONCES
                                              INICIO
                                          VERIFICAR Guarda Secundario
                                           SI VERIFICAR=FALSO ENTONCES
                                           INICIO
                                            cambiar de TOKEN
                                           FIN
                                              FIN
                                      Trans_time ← tiempo de transicion
                                            DISPARA TRANSICION
                                            Ciclo1 ← FALSE
                                    FIN
                            FIN DEL WHILE
                        SI lugar_actual <> primer lugar de entrada ENTONCES INICIALIZAR APUNTADORES del lugar _actual
                    Cuenta_lug  ← cuenta_lug – 1
                Lugar_actual  ← Transiciones .arc_entrada(cuenta_lug)
LOOP HASTA apuntador1 = utlimo token de subset dinamico AND cuenta_lug=0

FIN
```

Figura 3-2: Continuación pseudo código para la evaluación de transiciones

3.1.3 Disparo de Transiciones

Los subconjuntos encontrados por el algoritmo contienen los tokens que satisfacen las restricciones y habilitan la transición. Los estados generados se obtienen a partir de disparar la transición con todas las combinaciones distintas posibles de tokens. La Figura 3-3 presenta la situación que se tiene al momento de iniciar el disparo de la transición. La región sombreada representa los subconjuntos que satisfacen los valores constantes de las inscripciones de arco y las restricciones impuestas por los guardas de la transición representada en la figura. Las flechas representan a los apuntadores a los tokens que generarán las diferentes combinaciones factibles.

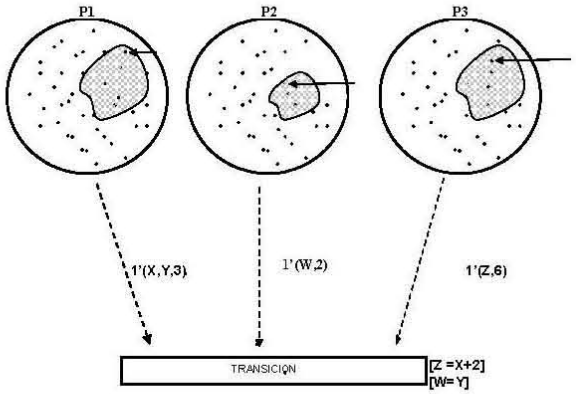


Figura 3-3: Transición lista para disparo

En la Figura 3-4 se presenta el pseudo código implementado para el disparo de las transiciones.

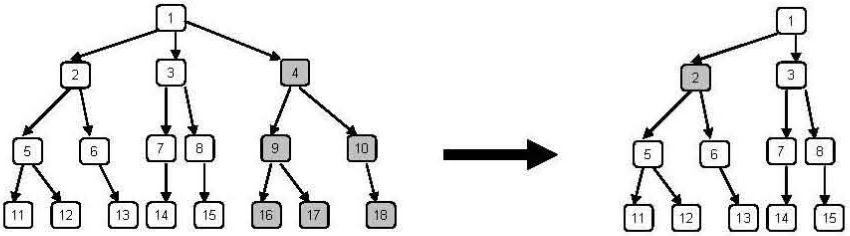
```
PROCEDIMIENTO DISPARO(TRANSICION, ROW1, COLUMN1, TIME_TR)
INICIO
NUEVA_MARCA ← FALSO
INICIO_TIEMPO ← CALCULO_ADELANTO _RELOJ
TIEMPO_FIN ← INICIO_TIEMPO + TIME_TR
LONGIT1 = NUMERO TOTAL LUGARES
LONGIT2 = NUMERO  DE LUGARES A LA ENTRADA
LONGIT3=NUMERO DE LUGARES A LA SALIDA
FOR  PLACES = 1 TO LONGIT1
          LUGAR1 ← LUGAR_ACTUAL(PLACES)
                    FOR  LUGARES2 = 1 TO LONGIT2
                              LUGAR_ENTRADA ← TRANSICIONES .LUGAR _ENT(LUGARES2)
                              IF PLACES = LUGAR_ENTRADA THEN
                               LUGAR1 ← RESTA_TOKEN (LUGAR_ENTRADA)
                               EXIT FOR
                              END IF
                    NEXT
                    FOR LUGARES 3= 1 TO LONGIT3
                              LUGAR_SALIDA ← TRANSICIONES.LUGAR_SAL(LUGARES3)
                              IF PLACES = LUGAR_SALIDA THEN
                              LUGAR1 ← SUMA_TOKEN(LUGAR_ENTRADA)
                              EXIT FOR
                              END IF
                    NEXT
  CALL ORDENAR_TOKENS(LUGAR1)
   MARCA_2ND_NIVEL ← ADD_MARCA(PLACES,LUGAR1, CADENA2)
   IF MARCA = TRUE THEN
     NUEVA_MARCA = TRUE
   END IF
   NEXT PLACES
   IF NUEVA_MARCA = TRUE THEN
          AGREGA_RELACION (CELDA2, CARDIN_GLOBAL,INICIO TIEMPO, TIEMPO_FIN)
   ELSE
        BUSCA_MARCA(CELDA2,CARDIN_GLOBAL,INICIO_TIEMPO,TIMEPO_FIN)
   END IF

   FIN
```

Figura 3-4: Algoritmo de disparo de Transiciones

3.2 Etapa de optimización de Estados

La segunda etapa del algoritmo, denominada *etapa de optimización*, hace uso de la lista de estados repetidos analizando la extensión temporal para comparar los estados repetidos en la lista contra el correspondiente estado almacenado en el árbol de alcance.
En particular para obtener una mejor administración de memoria fue necesario utilizar una definición de nodo repetido particular que servirá para la optimización del tiempo total de proceso. Esta definición se basa en la simetría presente en ciertos estados; en este sentido se definieron los *nodos simétricos repetidos* los cuales son marcados del modelo que tienen la misma distribución de tokens pero difieren de los sellos de tiempo. Con el uso de los nodos simétricos es posible reducir la cantidad de información a almacenar en el árbol de alcance ya que cuando se presentan este tipo de marcados no es necesario almacenar toda la información del nuevo marcado sino únicamente la información relativa a los sellos de tiempo y el reloj global. La Figura 3-5 ilustra la idea de la utilización de los nodos simétricos repetidos.



2: [2'(3,4)@3,5 , 1'(1)@3 + 1'(2)@5 , 0 , 0]

4: [2'(3,4)@1,3 , 1'(1)@1 + 1'(2)@3 , 0 , 0]

Figura 3-5: La explotación de la simetría en los marcados.

La Figura 3-5a representa la situación cuando se generan dos nodos que presentan este tipo de simetría. El nodo 2 presenta la misma distribución de tokens (en cantidad y colores) que el nodo 4; pero se puede observar que los sellos de tiempo del nodo 4 varían respecto del nodo 2. Basándonos en las reglas de disparo es claro que todos los nodos que se generen a partir del nodo 4 tendrán la misma distribución de tokens que los nodos que se generen a partir del nodo 2, o lo que es lo mismo que todos los nodos que cuelgan del nodo 4 tendrán su correspondiente *nodo simétrico repetido* entre los que cuelgan del nodo 2. Debido a esto la idea del los nodos simétricos repetidos es almacenar la información temporal del primer nodo simétrico repetido (nodo 4 en el ejemplo) para evitar el almacenamiento de información repetida (Figura 3-5b) asociándola al nodo simétrico repetido. La información temporal almacenada servirá para que en una etapa posterior se analice si alguno de los nodos que cuelgan del nodo 4 mejora la función de coste del nodo objetivo.

Los nodos simétricos repetidos se definen formalmente de la siguiente manera.
Sea $\mathbf{M}^T$ el conjunto de marcados temporales del árbol de alcance. Sean M_i^T y M_k^T marcados temporales con sus correspondientes marcados atemporales M_i^U y M_k^U.
Un marcado M_i^T es un nodo simétrico repetido a otro M_k^T cuando la siguiente condición ocurre:

$$M_i^T, M_k^T \in \mathbf{M}^T \wedge M_i^U = M_k^U$$

Es importante señalar que los nodos simétricos repetidos representan en sistemas logísticos o de manufactura un mismo estado logístico pero alcanzado a tiempos distintos (los valores temporales de los nodos simétricos repetidos).

3.2.1 Análisis de estados simétricos repetidos

Como se mencionó anteriormente la lista de estados repetidos o nodos simétricos repetidos se analiza para evaluar si es que alguno de ellos mejora el valor del nodo objetivo.
La comparación entre los valores temporales de los nodos repetidos se lleva a cabo para decidir cuál de los nodos origina el estado repetido con mejores valores de tiempo. Si dos estados son iguales en su marcado, las ramas que se generan a partir de este nodo son iguales en ambas situaciones [88]. En el caso de redes temporales, aunque las ramas son las mismas, los sellos de tiempo y relojes globales de los diferentes estados pueden ser distintos.
Para seleccionar el mejor nodo entre el utilizado en la generación del árbol de alcance y los nodos almacenados en la lista de nodos repetidos, se hace una comparación de los sellos de tiempos de los tokens del marcado, siguiendo la evaluación de sellos de tiempo planteados por [89]. Esta situación se ilustra en la Figura 3-6 donde se tienen los marcados M_i y M_j que representan el mismo estado del sistema pero alcanzado de dos diferentes maneras y con diferentes valores temporales.

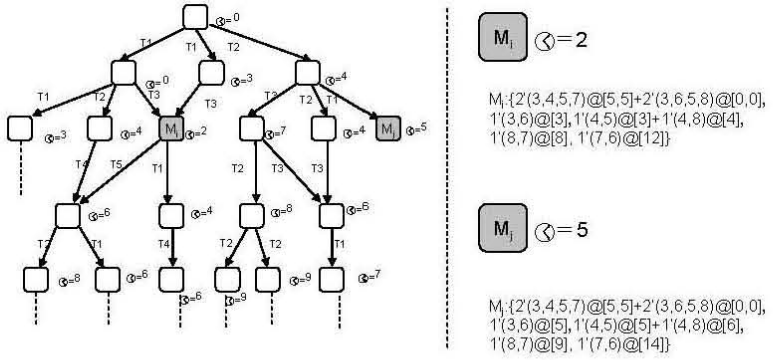


Figura 3-6: El análisis temporal de los nodos M_i y Mj.

El nodo M_i representa el estado alcanzado por el sistema durante la generación de estados, y forma parte del árbol de alcance cuyos sellos de tiempo son resultado de la evaluación del nodo padre. El nodo M_j representa un estado con los mismos valores de los tokens que conforman el marcado M_i pero los sellos de tiempo y reloj global son distintos debido a que es generado a partir de una rama distinta a la de M_i. Al realizar la comparación de sellos de tiempo, se presentan tres posibles situaciones:

a) Sellos de tiempo del estado M_j menores que los sellos de tiempo del estado M_i: en este caso se puede asegurar que todos los estados que se generen utilizando los valores de los sellos de tiempo del nodo M_j tendrán tiempos menores que los que tienen los nodos actuales almacenados en el árbol. Por lo que el nodo M_j reemplaza al nodo M_i y se actualiza la rama que se considere la mejor.
b) Sellos de tiempo del marcado M_j mayores que los del marcado M_i: en este caso se puede asegurar que todos los estados que se generen a partir del marcado M_j tendrán sellos de tiempo mayores que cualquier estado almacenado en el árbol de alcance y que se generen a partir del marcado M_i. Este caso no se lleva a cabo una sustitución de nodos y se mantienen las relaciones y los tiempos del nodo que se encontraban almacenados originalmente en el árbol de alcance puesto que los valores del nodo objetivo empeorarían.
c) Numero parcial de sellos de tiempo del marcado M_j menores en valor que los correspondientes sellos de tiempo del marcado M_i y el resto mayores o iguales: en este caso es imposible decidir cual marcado producirá ramas con mejores tiempos, por lo cual se explora la mejor rama hasta llegar al nodo final utilizando los tiempos del nodo M_j. Si al evaluar que el nodo final obtenido con los tiempos del nodo M_j son mejores que los tiempos obtenidos del nodo original M_i se remplaza el marcado M_i y se actualizan los tiempos de la rama explorada, en caso contrario no se realiza ningún remplazo.

Haciendo uso de estas reglas para el análisis de los nodos repetidos se lleva a cabo una optimización progresiva de los tiempos de simulación debido a que la ruta *principal* que va desde el nodo raíz hasta el nodo objetivo se va actualizando a medida que se analizan los estados repetidos. El procedimiento anterior se esquematiza en la Figura 3-7.

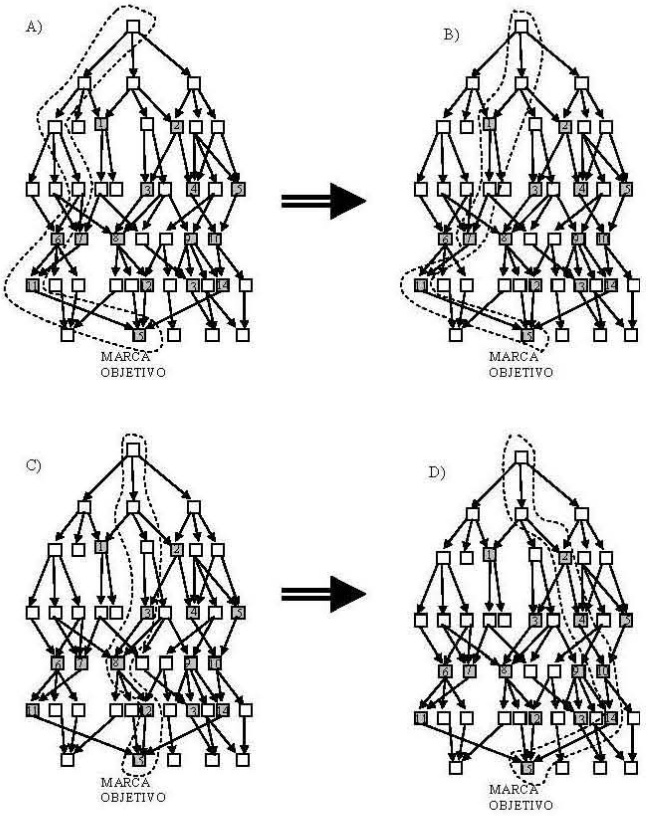


Figura 3-7: Optimización de ruta analizando nodos repetidos

Esta figura ilustra cómo se mejora progresivamente la secuencia de estados que parte desde el nodo raíz hasta el marcado objetivo de manera que al término de la etapa de optimización se obtiene una ruta óptima.

En el paso A la línea punteada representa la primera ruta factible encontrada al explorar por primera vez el espacio de estados, los nodos grises representan los estados repetidos almacenados en la lista de nodos repetidos, los cuales son analizados en esta etapa para llevar a cabo la mejora. En el paso B se analiza el estado repetido número 1 y al encontrar que mejora los tiempos para la

duración total del proceso la ruta factible es actualizada conforme a las reglas mencionadas, obteniendo la ruta factible marcada con línea punteada (la cual es mejor que la anterior en lo referente a tiempo total). En el caso C, se analiza el nodo repetido número 3 el cual produce una ruta factible distinta pero mejor que la anterior. En el caso D se analiza el estado repetido número 4 el cual mejora nuevamente los tiempos de la ruta total. Este proceso mejora progresivamente el tiempo total a medida que se analizan los estados repetidos, y termina en el momento que todos los estados repetidos han sido analizados.

3.2.2 Rendimiento Práctico del Algoritmo de Optimización

Los nodos repetidos juegan un papel muy importante en la optimización temporal del modelo haciendo uso del árbol de alcance. Al llevar a cabo el análisis de los sellos de tiempo y reloj global de las marcas repetidas se seleccionan los estados o nodos padres en base a una función de coste temporal mejorando los tiempos progresivamente con la evaluación de los estados repetidos.
La optimización de un sistema de manufactura se lleva a cabo a través de dos etapas:

1) Generación de todos los estados del árbol de alcance. A medida que se va realizando la exploración del árbol de alcance, se van detectando los nodos simétricos repetidos y se almacenan en una lista junto con la información del reloj global y los sellos de tiempo de los tokens que componen el marcado.
2) Análisis de la información temporal de los estados repetidos. El análisis se lleva a cabo entre los marcados repetidos, un marcado M_0 (almacenado como parte del árbol de alcance) y un M_1 (nodo repetido), evaluando la información temporal de ambos. Se lleva a cabo una comparación token a token de los sellos de tiempo entre los dos marcados obteniendo los tres posibles resultados para decidir si el marcado M_1 es mejor que el M_0:

Haciendo uso del algoritmo de optimización, se optimizaron los talleres de trabajo 3x3, 5x5 y 6x6 principalmente, los cuales son problemas de referencia con muy conocidos [106,107]; todo esto con el fin de evaluar la eficiencia computacional del algoritmo desarrollado.
La Tabla 3-2 presenta los resultados de las pruebas realizadas para evaluar la eficiencia de la etapa de análisis de los estados repetidos.

Tabla 3-2: Análisis de Desempeño

TIPO TALLER	T-T 3X3	T-T 5X5	T-T 6X6
No. De Estados Distintos	693	7,776	117,650
Estados Repetidos que aparecen	2,032	24,625	487,408
Nodos Actualizados después de la Evaluación	59	1,893	26,555
Nodos desechados después de la Evaluación	485	5,829	128,387
Nodos Imposibles de Decidir	1,488	16,903	332,475
Nodos Actualizados	11	10	35

La información recopilada es la siguiente:

- **No de estados distintos.** Son los estados diferentes entre sí que presenta el árbol de alcance para cada uno de los tipos de talleres; en este caso se contabilizan todos los estados distintos basándose únicamente en el número de tokens que conforman el estado del modelo así como los diferentes colores que los componen.
- **Estados repetidos.** Son los estados que sin tomar en cuenta el tiempo resultan iguales al realizarse la exploración del árbol de alcance
- **Nodos actualizados después de la evaluación.** Son el número de nodos que resultan mejor en sus características temporales al comparar dos estados iguales.
- **Nodos desechados después de la evaluación.** En número de nodos que resultan peores en sus características temporales al comparar dos estados iguales.
- **Nodos imposibles de decidir.** El número de veces que al realizar al comparación entre dos estado iguales, no es posible decidir (nodos irresolubles). Para resolver esta situación, es necesario el llevar a cabo una exploración en profundidad hasta alcanzar el estado objetivo y finalmente comparar los dos estados finales para decidir si se actualiza la rama con los sellos de tiempo mejores.
- **Nodos actualizados.** El número de veces que se tuvieron que actualizar los sellos de tiempo al llevar a cabo la exploración a profundidad al encontrar nodos *irresolubles* y concluir que efectivamente la mezcla de sellos de tiempo del estado en evaluación resultaba mejor que la que se tenía almacenada.

El desempeño del algoritmo resultó adecuado en el sentido que permitió obtener los valores óptimos para los problemas de referencia propuestos. Sin embargo a partir de los resultados se puede observar lo siguiente:

- Los *nodos actualizados después de la primera evaluación* resultan mínimos comparados con los otros dos posibles resultados.
- Los *nodos desechados* es un indicador de la adecuada selección de los nodos durante la exploración a profundidad. Mientras mejor seleccionado sea el nodo a evaluar el número de nodos con peores valores temporales aumentará.

- Los *nodos imposibles de decidir* son otro indicador importante a mejorar para incrementar el desempeño del algoritmo ya que cada uno de esos nodos ocasiona una exploración en profundidad hasta el nodo objetivo lo cual ralentiza el desempeño de la optimización.

Estos detalles fueron tomados en cuenta para el desarrollo de algoritmos más avanzados los cuales son presentados en la sección siguiente.

3.3 Exploración inteligente del árbol de alcance

Las dos etapas principales del algoritmo pueden ser abordadas con diferentes estrategias tanto de exploración como de análisis con el fin de mejorar el desempeño del simulador:

a) Etapa de generación de estados: El desarrollo del simulador se basa en la apertura del árbol de alcance a través de una exploración en profundidad y realizando la evaluación de los nodos descendientes de manera secuencial bajando nivel a nivel hasta que se llegue a un nodo objetivo o a marcas muertas. Esta etapa es factible de mejora a través de la selección adecuada de los nodos y ramas a explorar.
b) Etapa de análisis de estados simétricos repetidos: En esta etapa se cuenta con la información de los estados del sistema, por lo que si se selecciona adecuadamente cual o cuales estados repetidos se deben actualizar es factible el minimizar la cantidad de nodos irresolubles que se tendrán que evaluar a profundidad para llevar a cabo la optimización del modelo con lo cual el desempeño del algoritmo aumentará.

El uso del algoritmo en dos etapas permite la mejora progresiva del camino que parte del nodo inicial hasta el nodo objetivo. Este algoritmo es factible de mejora debido a que cuando se trata de problemas que generan un árbol de alcance muy grande no es factible el almacenado en memoria del total de los estados que puede tener un modelo en RdPC; más aún en ocasiones este problema se vuelve intratable debido a la infinita generación de estados del modelo [61].

El problema de la intratabilidad debido a la explosión combinatoria o cuando los sistemas modelados presentan un espacio de estado no acotado se puede afrontar con técnicas heurísticas. Estas técnicas permitirán la exploración de las ramas del árbol que en base a algún criterio se espere que produzcan los mejores resultados con base a los objetivos que se pretendan.

El algoritmo en dos etapas permite la implementación de heurísticas tanto en la primera etapa como en la segunda etapa.

3.3.1 Heurísticas para la etapa de generación de Estados.

En la primera etapa es posible dirigir la exploración implementando una función de utilidad que permita dirigir la búsqueda en profundidad realizada por el algoritmo de generación. Es importante desarrollar una función de utilidad f_k basada en tiempo, coste o funcionamiento que asigne un valor a cada nodo del árbol de alcance tomando en cuenta datos extraídos a partir de la información contenida en el marcado:

$$f_k : \mathrm{N}^k \rightarrow \mathrm{N} \tag{4.1}$$

La Figura 3-8 ilustra la lógica utilizada al llevar a cabo la implementación de una función de utilidad para la evaluación de los diferentes hijos a medida que se desciende en los niveles del árbol. Haciendo uso de la función de utilidad, se le asigna a cada nodo hijo del árbol de alcance un

valor calculado a partir de los valores temporales obtenidos al generar los correspondientes marcados hijos.

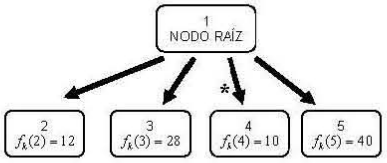


Figura 3-8: Función de utilidad para dirigir la búsqueda en profundidad

En el caso que no se implementara ninguna función de utilidad (como funciona originalmente el algoritmo), la selección del siguiente nivel a explorar sería completamente aleatoria o secuencial de izquierda a derecha o de derecha a izquierda en la figura. Al implementar la función de utilidad, es posible seleccionar aquel nodo que tenga el menor valor de la función de utilidad, (criterio de minimización). En el ejemplo de la Figura 3-8, el nodo 4 es el que tendría que ser evaluado al descender en el siguiente nivel. Esta implementación permite ir seleccionando de manera más inteligente los nodos que producen una mejora en la función del valor de utilidad.
El desempeño global del algoritmo se ve mejorado de manera sustancial, ya que la adecuada selección de los nodos a evaluar permitirá en la segunda etapa reducir el número de análisis en profundidad para determinar si la rama mejora o no la secuenciación encontrada. En este trabajo se probaron dos heurísticas para mejorar la primera etapa del algoritmo. La diferencia entre ambas técnicas yace en el criterio de selección; mientras que una selecciona en base a valores absolutos de los sellos de tiempo, la otra lo hace en base a la probabilidad de incrementar el reloj global al llevar a cabo un disparo.
Durante el desempeño del algoritmo de búsqueda en profundidad, la selección del nodo a explorar se lleva a cabo utilizando la función (4.1) y el nodo con el menor valor es el que se seleccionará para ser evaluado en el siguiente nivel.
Formalizando, dado un conjunto N de nodos sucesores, el siguiente nodo a ser evaluado es aquel que es seleccionado con el siguiente criterio:

$$min\left\{ f_k(M_j) \right\}_{j=1}^{N} \tag{4.2}$$

Donde el índice j representa el número de marcas sucesoras del nodo en evaluación.

3.3.1.1 Heurística basada en Valores Absolutos de Tiempo

La rama que conduce del estado inicial a un estado objetivo se ve mejorada (en lo referente a tiempos) al implementar una regla de selección de estados a evaluar en cada nivel del algoritmo de búsqueda en profundidad. Dicha selección consiste en escoger aquel estado hijo que probablemente adelante lo menos posible el reloj en posteriores evaluaciones. Al llevar a cabo esta selección, la primera rama evaluada del árbol de alcance contiene sellos de tiempos mejores que si se realizara una selección aleatoria. Como resultado de esta selección, al llevarse a cabo la segunda etapa del algoritmo, en la evaluación de los estados repetidos debería existir un menor número de nodos irresolubles, mejorando el desempeño del algoritmo debido a que se realizan menor número de exploraciones a profundidad. La reducción en el número de nodos irresolubles ocasiona la obtención de resultados en menor tiempo dado que el número de operaciones para llevar a cabo la optimización disminuye.

Dado que no se sabe de antemano que transición será la que se disparará en la siguiente evaluación, la heurística guía la selección en base a la suposición que mientras el marcado tenga valores más pequeños, existe menos probabilidad que el reloj sea incrementado a medida que se llevan a cabo subsecuentes evaluaciones. En esta heurística la función lleva a cabo la suma de los diferentes sellos de tiempo de cada uno de los marcados de los nodos sucesores y selecciona el que tenga el menor valor.
De manera formal, dados los T_i sellos de cualquier estado del conjunto de sucesores, sea F_k la función que suma los sellos de tiempo donde el índice j recorre desde el token 1 hasta el token k del marcado.

$$\begin{aligned} F_k: \quad \mathrm{N}^k &\rightarrow \mathrm{N} \\ T_i &\mapsto \sum_{j=1}^{k} T_{ij} \end{aligned} \tag{4.3}$$

La heurística se implementó para evaluar los talleres de trabajo previamente mencionados y se analizaron los valores de los indicadores de desempeño que se mencionaron en el capítulo anterior. La Tabla 3-3 presenta los resultados obtenidos con la implementación de la heurística.

Tabla 3-3: Resultados de la Simulación con la Heurística

TIPO TALLER	T-T 3X3	T-T 5X5	T-T 6X6
No. De Estados Distintos	693	7,776	117,650
Estados Repetidos que aparecen	2,032	24,625	487,408
Nodos Actualizados después de la Evaluación	0	1,330	18,725
Nodos desechados después de la Evaluación	2,032	10,506	213,966
Nodos Imposibles de Decidir	0	12,789	254,717
Nodos Actualizados	0	22	34

Se puede observar (contrastando los datos de la Tabla 3-2 con la Tabla 3-3) que se mejora considerablemente el desempeño del simulador, debido que se reduce el número de nodos irresolubles, de 332,475 a 254,717, lo que supone una reducción de casi 23% e implica el evitar realizar una exploración profunda de casi 80,000 nodos. La misma conclusión se puede obtener al observar que el número de nodos desechados aumenta al implementar la heurística pasando de 128,387 a 213,966.

Con estos resultados es posible confirmar que este tipo de heurística para la primera etapa del algoritmo el desempeño total del algoritmo de simulación/optimización es mejorado en su totalidad.

3.3.1.2 Heurística basada en Valores Probabilísticos

La idea de esta heurística es evaluar para cada marcado el número de sellos de tiempo que son mayores que el reloj global, y el número de ellos que son menores. Basado en esa cantidad se calcula la probabilidad de que el reloj global sea incrementado en la evaluación siguiente debido al uso de los sellos de tiempo que son mayores al reloj global; de manera más formal se plantea a continuación.

Dado un estado M_j del árbol de alcance, sea P_k la función que asigna a cada estado la probabilidad de incrementar el reloj global *Gc* cuando se evalúan los sellos de tiempo T_i desde 1 hasta k en el marcado correspondiente.

$$\begin{aligned} \Psi: \ \mathbf{R}\times\mathbf{R} &\rightarrow \{0,1\} \\ (x,y) &\mapsto \begin{cases} 0, x \geq y \\ 1, x < y \end{cases} \end{aligned} \tag{4.4}$$

$$P_k: \quad N^k \times N \rightarrow N$$
$$(T_i, Gc) \mapsto \frac{\sum_{j=1}^{k} \Psi(T_{ij}, Gc)}{k} \tag{4.5}$$

Usando la fórmula (4.4), la función Ψ asigna el valor de 1 a cada sello de tiempo que es mayor que el reloj global *Gc*, y 0 para aquellos que son menores; esta evaluación se lleva a cabo para todos los k sellos de tiempo.

Esta heurística se implementó siguiendo la misma idea planteada al principio del capítulo y se experimentó con los talleres mencionados previamente. La Tabla 3-4 muestra los resultados obtenidos con esta implementación.

Tabla 3-4: Resultados de la exploración a profundidad con una heurística probabilística

TIPO TALLER	T-T 3X3	T-T 5X5	T-T 6X6
No. De Estados Distintos	693	7,776	117,650
Estados Repetidos que aparecen	2,032	24,625	487,408
Nodos Actualizados después de la Evaluación	62	1,272	16,734
Nodos desechados después de la Evaluación	610	10,974	233,476
Nodos Imposibles de Decidir	1,360	12,379	237,198
Nodos Actualizados	4	21	30

Utilizando esta heurística los talleres de trabajo fueron resueltos obteniendo las soluciones óptimas. Se puede apreciar que el indicador *nodos imposibles de decidir* que el valor disminuyó en comparación con las implementaciones realizadas anteriormente. Debido a esto las búsquedas a profundidad fueron por tanto disminuidas lo cual ocasiona que la convergencia a la solución óptima se vea acelerada en comparación con los algoritmos previos. Es importante señalar también que en particular esta heurística aplicada en un sistema con un espacio de estados o árbol de alcance con un número reducido de estados como el taller 3x3 los resultados no son mejores que cuando se prueba con otro modelo que presenta un mayor número de estados. En particular en el caso del taller 6x6 el algoritmo en dos pasos con la heurística implementada reduce el número de *nodos imposibles de decidir* de 254,717 utilizando la heurística de tiempos absolutos a 237,198 con esta heurística. Lo cual es un indicador positivo en cuanto a la mejora del rendimiento total del algoritmo.

4 DESCRIPCION DE LOS PROBLEMAS ESTUDIADOS

Para evaluar el rendimiento de la herramienta implementada se llevaron a cabo pruebas con distintos problemas académicos, los cuales fueron implementados progresivamente con el desarrollo de las diferentes características de análisis del entorno de prueba del simulador/optimizador.

Con el objetivo de evaluar la etapa de generación del espacio de estados haciendo uso solo de las restricciones impuestas por los colores, el peso de los arcos y los guardas, se implementó un problema de paletizado en dos dimensiones que resulta adecuado debido a su alta complejidad. Este problema fue abordado en las etapas iniciales del desarrollo del simulador/optimizador y el cual se puede utilizar para resolver modelos en los cuales no sea necesario el análisis temporal.

Posteriormente al añadir la extensión temporal, se pudieron abordar los diferentes talleres de trabajo donde el objetivo ha sido alcanzar los resultados reportados en la literatura en un tiempo de cómputo adecuado.

La evaluación de los diferentes talleres de trabajo permitió evaluar la eficiencia del algoritmo haciendo uso de las redes de Petri coloreadas temporales, y a partir del análisis de estos problemas se pudieron desarrollar heurísticas que mejoraran el desempeño computacional del algoritmo y por ende de la herramienta de prueba desarrollada.

A continuación se presentan los modelos en RdPC de los diferentes casos evaluados, así como los resultados obtenidos al utilizar la herramienta.

4.1 Problema de Paletizado de Cajas

El problema consiste el empaquetado de objetos de diferentes dimensiones en algún o algunos espacios confinados (cajas, palets, cajas de tráileres, contenedores, etc.). Los objetos deben ser colocados de manera que se minimice el espacio utilizado por los mismos en el palet. Dada la importancia que tiene a nivel industrial el optimizar el espacio de almacenamiento, diferentes autores han abordado el problema con diferentes enfoques [26, 92, 59, 68, 100]. Para este estudio se ha codificado un modelo en RdPC [96] para modelar el proceso de paletizado de cajas. Dicho problema puede ser visto como un sistema a eventos discretos en el que se utiliza un nivel de abstracción en el cual los eventos representan la colocación de una caja en la superficie del palet tal cual como se muestra en la Figura 4-1.

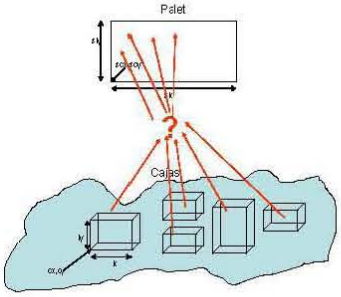


Figura 4-1: El paletizado de cajas de diferentes dimensiones

Los colores utilizados para este modelo se definen a continuación para describir la configuración del palet: las coordenadas en las cuales cada caja debe ser colocada en la superficie del palet junto con las coordenadas del espacio fragmentado como resultado de posicionar una caja en el palet. Como consecuencia de ubicar una caja en el palet, la superficie disponible se irá fragmentado a medida que nuevas cajas se vayan colocando, esta situación se puede observar en la Figura 4-2 en la cual se puede observar que se tienen dos superficies diferentes en el palet las cuales se evalúan para colocar una nueva caja que ocupe dichos espacios.

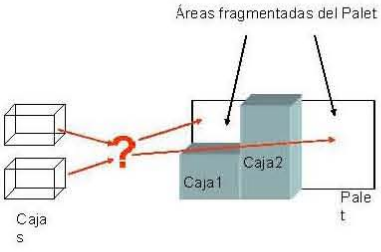


Figura 4-2: Áreas fragmentadas en el palet

Por lo tanto, eventos que describan las diferentes posibilidades para colocar una caja el palet se codifican de manera que se calcule la nueva disposición del espacio disponible en el palet una vez que una caja ha sido colocada: la posición y la orientación de la caja en el palet junto con el cálculo

de las dimensiones de los nuevos espacios fragmentados disponibles en el palet como consecuencia del colocado de cajas de diferentes dimensiones.

La Tabla 4-1 resume los colores utilizados para describir la información que se utiliza para llevar a cabo los cálculos antes mencionados y para poder implementar el nivel de abstracción introducido en esta sección.

Tabla 4-1: Especificación de colores

COLOR	DEFINICIÓN	SIGNIFICADO
Idc	Entero	Identificador de la caja
Cr	Entero	0:orientación original 1:rotada 90° sobre eje Z
Ce	Entero	0:No asignada 1: en proceso 2: colocada en el Palet
Cx	Real	Coordenada X donde la caja está localizada
Cy	Real	Coordenada Y donde la caja está localizada
Cz	Real	Coordenada Z donde la caja está localizada
Lx	Real	Longitud de la caja en el eje coordenado X
Ly	Real	Longitud de la caja en el eje coordenado Y
Lz	Real	Longitud de la caja en el eje coordenado Z
Scx	Real	Coordenada X donde la superficie libre está localizada
Scy	Real	Coordenada Y donde la superficie libre está localizada
Slx	Real	Longitud X de la superficie libre
Sly	Real	Longitud Y de la superficie libre
Ge	Entero	0: Una caja puede ser colocada en el palet 1: Caja a ser asignada 2: En busca de superficie 3: Evaluando las nuevas superficies fragmentadas
Gz	Entero	Indica el piso del palet
Gsf	Real	Superficie disponible en el palet
Gncv	Entero	Numero de cajas Virtuales
Pa	Producto idc*cx*cy*cz*lx*ly*lz*cr*ce	Información que describe las características físicas de cada caja
Ps	Producto scx*scy*slx*sly	Información que describe las características de las superficies libres
Pv	Producto idc*cx*cy*cz*lx*ly*lz*ce	Información que describe las características de las cajas virtuales
Pg	Producto ge*gz*gsf*gncv*gnc	Información global del modelo

La Figura 4-3 ilustra el evento que formaliza el colocado de una caja en un área disponible cuando la longitud de la caja es igual que la longitud del palet. Bajo estas circunstancias, se genera una nueva superficie la cual tiene nuevas dimensiones, por lo que cuando se utiliza un token para

realizar la actividad, la superficie original se ve disminuida restando la longitud de la caja de la longitud de la superficie en el eje coordenado Y (sly-ly).

Figura 4-3: Colocado de una caja con lx=slx

También existen otros eventos como el que se ilustra en la Figura 4-4 en que la caja colocada sea menor que la superficie seleccionada, bajo estas condiciones la nueva superficie fragmentada se ve aumentada por dos nuevos espacios rectangulares de dimensiones menores que la superficie original. Dichos espacios se utilizarán para colocar más cajas.
El nodo lugar P1 representa los tokens asociados a las cajas, y el nodo lugar P2 representa los tokens que describen los espacios disponibles en el palet, por lo que cuando se dispare la transición T12 (Figura 4-3) únicamente un token es generado para describir las nuevas áreas libres, pero en la transición T13 (Figura 4-4) dos nuevos tokens son generados para describir los dos espacios generados debido al colocado de la caja.

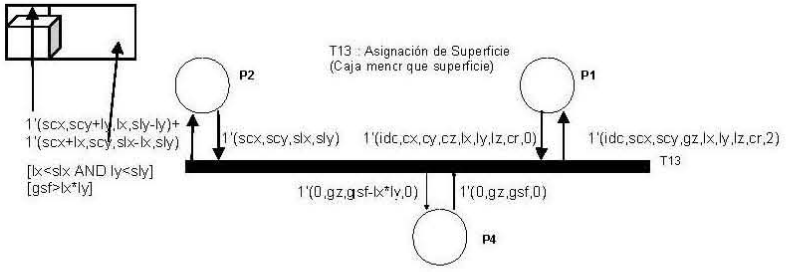


Figura 4-4: Colocado de una caja con lx<slx y ly<sly

Adicionalmente a los eventos que especifican como colocar las cajas en un espacio disponible del palet, existe un evento en particular que permite cambiar la orientación de la caja para de esa manera aprovechar al máximo los espacios disponibles en el palet. Este evento se puede disparar en cualquier momento, pero solo puede ser disparado una sola vez por cada caja. La figura 4-5 ilustra las expresiones de arco que describen el evento.

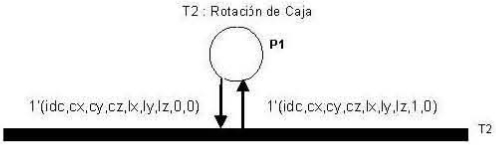


Figura 4-5: Rotación de una caja para ser colocada

Las figuras anteriores describen la idea principal del modelado de eventos en este ejemplo, el resto de las transiciones modeladas sufre una ligera variación en cuanto a las restricciones impuestas así como a las dimensiones de los espacios generados y valores que adquieren los colores para mantener la información del estado del sistema modelado.
El modelo final consta de 28 nodos transición y 4 nodos lugar. En el anexo E se presenta el modelo completo del paletizado de cajas.
A continuación se presenta un ejemplo del tipo de soluciones que es posible determinar cuando la herramienta no hace uso de la extensión temporal en los modelos de RdPC.

4.1.1 Paletizado de cajas con diferentes dimensiones

Este ejercicio se ejecutó para verificar el modelo desarrollado, donde se tiene un palet con un área determinada y se tienen un cierto número de cajas, las cuales tienes que ser colocadas de la mejor manera. Se tienen como restricciones físicas las dimensiones de las cajas así como el área disponible del palet. La figura 4-6 esquematiza los elementos de este problema.

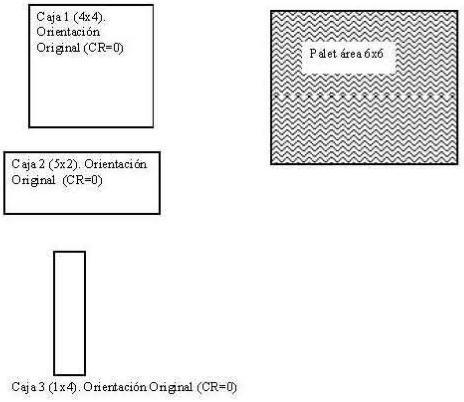


Figura 4-6: Problema del paletizado de cajas

Este problema aparentemente sencillo tiene diferentes variantes para la colocación de las diferentes cajas, dado que el simulador debe de probar todas las posibles configuraciones, y además debe de probarlas variando también la orientación de las cajas, la multiplicidad de soluciones se ve incrementada si se considera la orientación de las cajas.
El modelo se codificó haciendo uso de la herramienta para redes atemporales utilizando los colores y lugares anteriormente presentados. Se obtuvieron las diferentes soluciones a este problema, las cuales se presentan en Tabla 4-2 que contiene los diferentes estados del nodo lugar P1 que dan solución al modelo.

Tabla 4-2: Soluciones obtenidas por el simulador para el problema de 3 cajas.

Soluciones encontradas Lugar P1
1,0,0,1,2,5,4,1,2-1,2,0,1,4,1,4,1,2-1,2,1,1,4,4,4,0,2
1,0,0,1,2,5,4,1,2-1,2,0,1,4,4,4,0,2-1,2,4,1,4,1,4,1,2
1,0,0,1,2,5,4,1,2-1,0,5,1,4,1,4,1,2-1,2,0,1,4,4,4,0,2
1,0,0,1,4,1,4,1,2-1,0,1,1,4,4,4,0,2-1,4,0,1,2,5,4,1,2
1,0,0,1,4,4,4,0,2-1,0,4,1,4,1,4,1,2-1,4,0,1,2,5,4,1,2
1,0,0,1,4,4,4,0,2-1,0,4,1,5,2,4,0,2-1,4,0,1,1,4,4,0,2
1,0,0,1,5,2,4,0,2-1,0,2,1,4,4,4,0,2-1,5,0,1,1,4,4,0,2
1,0,0,1,5,2,4,0,2-1,0,2,1,1,4,4,0,2-1,1,2,1,4,4,4,0,2
1,0,0,1,5,2,4,0,2-1,0,2,1,4,4,4,0,2-1,4,2,1,1,4,4,0,2
1,0,0,1,1,4,4,0,2-1,0,4,1,5,2,4,0,2-1,1,0,1,4,4,4,0,2
1,0,0,1,1,4,4,0,2-1,1,0,1,5,2,4,0,2-1,1,2,1,4,4,4,0,2

Para obtener una versión gráfica de las soluciones, es posible graficar en un plano cartesiano las diferentes soluciones con las coordenadas de las cajas dentro de un área representando al palet de 6x6. Para este ejemplo, en la Figura 4-7 se presentan las soluciones de este problema.

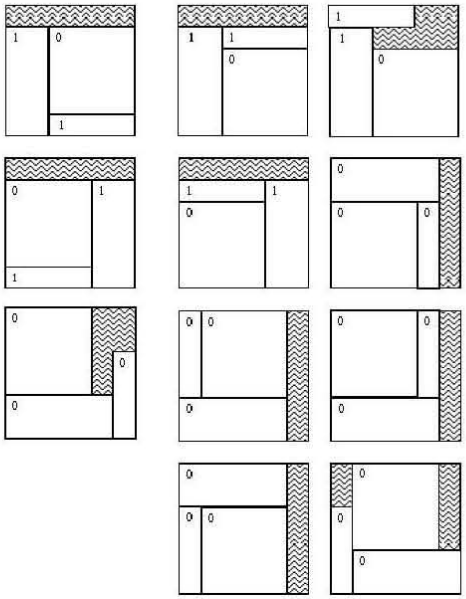


Figura 4-7: Soluciones al problema del Paletizado 3 cajas diferentes dimensiones.

Se puede observar que cada una de las cajas tiene un pequeño número, dicho número representa el valor del color CR que representa la orientación de la caja, por lo que en las cajas que no son cuadradas se ve claramente la orientación, pero en el caso de la caja 4x4 no es necesario rotarla, por lo que el valor para el color CR siempre tendrá el valor de 0.

4.2 Taller de Trabajo 3x3

La programación de tareas en un taller de trabajo no es un asunto trivial, se deben considerar diferentes combinaciones de operaciones de procesamiento en un conjunto de máquinas para llevar a cabo diferentes trabajos. El problema consiste en la producción de partes con diferentes características, las cuales son procesadas siguiendo una secuencia de tareas predefinidas. El problema del taller de trabajo consiste en un problema de asignación de actividades a recursos (máquinas) con capacidad finita y con ventanas de tiempo de ejecución.

Generalmente, las operaciones que afectan a las máquinas requieren respetar restricciones temporales, restricciones de precedencia entre operaciones, capacidad limitada de recursos, procesos con preferencia de recursos, etc. Se ha demostrado que el problema particular de taller de trabajo es un problema del tipo NP-Completo [41].

El caso del taller de trabajo 3x3 (tres trabajos y tres máquinas) es un problema de programación de rutas que consiste en asignar operaciones para cada una de las máquinas para llevar a cabo cada uno de los trabajos (*jobs*). Para algunos trabajos, la misma tarea se puede llevar a cabo en dos máquinas distintas aunque con tiempos de procesamiento distintos también. El objetivo de este taller de trabajo es, como se mencionó anteriormente, el minimizar la duración total para llevar a cabo los tres trabajos. La información que se necesita para definir este problema se presenta en la Tabla 4-3

Tabla 4-3: Secuencia de Tareas para cada Trabajo

SECUENCIA DE TAREAS	TRABAJO 1				TRABAJO 2				TRABAJO 3			
	Opcion1		Opcion2		Opción 1		Opción 2		Opcion1		Opción 2	
	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo
1	1	7	2	8	1	6	2	5	1	8	3	5
2	2	4			2	4	3	2	2	2		
3	1	7	3	4	1	6			2	6	3	4
4					1	3	2	2	1	4	2	2
5									1	2	3	3

De la tabla se puede apreciar que los tres trabajos varían en cuanto a número de tareas, el primer trabajo necesita llevar a cabo tres tareas, mientras que el segundo trabajo cuatro, y el tercero necesita realizar cinco tareas para estar terminado. Algunas de las tareas se pueden realizar en más de una máquina (la tarea 1 para el trabajo 1 puede ser realizada ya sea por la máquina 1 o por la máquina número dos), por lo que la selección de la mejor secuencia para llevar a cabo los tres trabajos no resulta trivial. Para resolver este problema se desarrolló el modelo en RdPC, posteriormente fue implementado en la interfase de usuario para finalmente llevar a cabo la simulación y optimización del modelo planteado.

4.2.1 Modelo en Redes de Petri Coloreadas

El modelo en RdPC se presenta en la Figura 4-8 el cual cuenta con tres nodos lugar, un nodo transición, seis arcos y tres guardas para la transición.

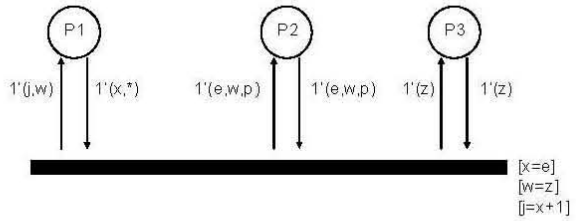


Figura 4-8: Modelo en RdPC del taller de trabajo 3x3

La Tabla 4-4 resume los colores utilizados para describir toda la información requerida.

Tabla 4-4: Especificación de colores taller de trabajo 3x3

COLOR	DEFINICIÓN	DESCRIPCIÓN
j	Entero 11..33	Identificador de trabajo y tarea
w	Entero 1..3	Máquina necesaria para la siguiente tarea
x	Entero 11..33	Trabajo y tarea en progreso
e	Entero 11..33	Identificador de trabajo y tarea
p	Entero	Tiempo de cada tarea
z	Entero 1..3	Máquinas disponibles

La Tabla 4-5 describe los lugares del modelo RdPC del sistema Taller de trabajo

Tabla 4-5: Descripción de los lugares del modelo

LUGAR	COLOR	DESCRIPCIÓN
P1	Producto j*w	El valor del color j representa el trabajo en turno y la tarea que se está realizando en un instante de tiempo. El color w representa la máquina que es necesaria para llevar a cabo la siguiente tarea del trabajo correspondiente
P2	Producto e*w*p	Representa la información de la secuencia lógica que deben seguir las diferentes actividades de los trabajos correspondientes. Los colores e,w y p representan respectivamente el trabajo y tarea en que se encuentra, la máquina siguiente al terminar la tarea del color e, y el tiempo de operación p que consumirá al llevar a cabo la tarea correspondiente en la máquina w.
P3	z	La información asociada con la máquina disponible.

Se ha considerado como estado inicial del sistema todas las máquinas disponibles y todas las tareas sin empezar ninguna operación, lo cual se ilustra en la Tabla 4-6.

Tabla 4-6: Información del marcado inicial del T-T 3x3

MODELO 3X3	LUGAR		
	P1	P2	P3
Marcado Inicial	1'(10,0)+1'(20,0)+1'(30,0)	1'(10,1,7)+1'(10,2,8)+1'(11,2,4)+1'(12,1,7) +1'(12,3,4)+1'(20,1,6)+1'(20,2,5)+1'(21,2,4) +1'(21,3,2)+1'(22,1,6)+1'(23,1,3)+1'(23,2,2) +1'(30,1,8)+1'(30,3,5)+1'(31,2,2)+1'(32,2,6) +1'(32,3,4)+1'(33,1,4)+1'(33,2,2)+1'(34,1,2) +1'(34,3,3)	1'(1)+1'(2)+1'(3)

Ejemplo 4.1: Interpretación del codificado de los tokens

La interpretación de la información en los colores de los tokens es la siguiente:

- Lugar P1. Utilizando el primer token 1'(10,0) , el primer color corresponde a un 10, el cual representa que el trabajo número 1 se encuentra realizando la actividad 0, y el segundo color representa la máquina en utilización.
- Lugar P2. Satisfaciendo la expresión de guarda [x=e] , se tiene en el lugar P2 el token 1'(10,1,7), el segundo color indica que cuando se realice la actividad 0 del trabajo 1, necesita a continuación la máquina 1 por una duración temporal de 7 unidades.
- Lugar P3. La restricción [w=z] se resuelve con el token 1'(1) que representa la disponibilidad de la máquina 1.

El token que se genera en el lugar P1 al disparar la transición es el 1'(11,1) que representa que el trabajo 1 está ejecutando la tarea 1 en la máquina 1, asociándole un sello de tiempo (7 unidades + reloj global) que indica el valor de tiempo en que se puede utilizar el token para otra evaluación. Este mismo proceso se continúa hasta que se hayan explorado todos los estados del sistema.

4.2.2 Codificación del Modelo en Interfase Usuario

El modelo descrito por la RdPC de la Figura 4-8 se ha codificado a través de hojas de cálculo utilizadas como interfase de usuario. La información se estructura en dos bloques de información para los dos niveles de agrupación: nivel color y nivel relación. Para el marcado de la Tabla 4-6 se utilizan dos hojas de cálculo y la información se introduce como se muestra en la Figura 4-9.

a) Nivel Color

	A	B	C	D	E	F
1		PLACE 1		PLACE 2		PLACE3
2	CARDINALITY	COLOR1	CARDINALITY	COLOR2	CARDINALITY	COLOR3
3	1,1,1	10,0-20,0-30,0	1,1,1,1,1,1,1,1,1	10,1,7-10,2,8-11,2,4-12,1,7-1	1,1,1,1	1-2-3-8

b) Nivel Relación

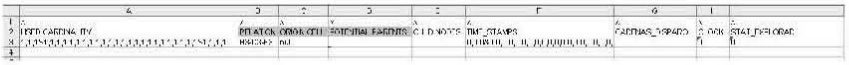


Figura 4-9: Codificación de M_0 en los dos niveles de información

En la hoja de cálculo de la Figura 4-9a se utilizan 6 columnas para definir los colores que aparecen y las cardinalidades asociadas a los mismos. Para almacenar la información correspondiente a cada

lugar, es necesario hacer el uso de dos columnas (columnas A-B, C-D y la E-F para los tres lugares del modelo) que corresponden a la combinación de color y sus cardinalidades
En la Figura 4-9b se ilustra la codificación del nivel de relaciones la cual se lleva a cabo con 9 columnas las cuales muestran la información de la estructura del árbol de alcance (relaciones padres–hijos, camino principal, cardinalidades utilizadas y marcado), la información temporal de los marcados (sellos de tiempo y reloj global) y un estatus de exploración.
La red de Petri se debe codificar en una tercera hoja de cálculo siguiendo una sintaxis en particular (la cual se detalla en el Anexo 8). En la Figura 4-10 muestra la hoja de cálculo para el codificado de la estructura de la RdPC.

	A	B	C	D	E	F	G
1							
2	**Relacion de Nodos Lugar-Transicion**						
3		**Nodos Lugar**	**Nodos Transicion**		**Numero Variables**		
4	**Cantidad**	3	1		5		
5							
6							
7	MATRIZ RELACION DE ARCOS		MATRIZ RELACIONES			**LUGARES**	
8	TRANSICION/LUGARES						
9	**Tiempo de Transicion**	**Transiciones**	1	2	3		
10	P	1	%-1'(X,*)|1'(J,W)	%-1'(E,W,P)|1'(E,W,P)	%-1'(Z)|1'(Z)		
11		2	*				
12							
13							

Figura 4-10: Codificado de la estructura de la RdPC

La estructura de la RdPC debe especificar número de nodos transición, numero de nodos, así como el número de variables que intervienen en el modelo.

Las relaciones entre los nodos transición y los nodos lugar se deben especificar utilizando una matriz donde la intersección del número de transición con el nodo lugar es una celda donde se codifica la expresión de arco.

En el ejemplo de la Figura 4-10 se puede observar que la relación existente entre la transición 1 con el lugar 1 viene dada por la expresión %-1'(X,*)|1'(J,W), la cual representa los dos arcos que relacionan la transición 1 con el lugar 1 (un arco de entrada y uno de salida).

En esta hoja de cálculo, se asocia también (para las redes de Petri temporizadas) el tiempo a la transición a través de un valor de tiempo constante o a través del valor de alguna variable, en el ejemplo es la variable P.
Haciendo uso de las tres hojas de cálculo (dos para definir el marcado inicial y una para la estructura de la RdPC) se define la estructura de la RdPC y el estado inicial del sistema.
Se debe utilizar una cuarta hoja para definir las características de las variables así como la relación existente entre ellas.

	A	B	C	D	E	F	G	H
1		VARIABLES A UTILIZAR						
2	NUMERO	NOMBRE	VALOR INICIAL	EXPRESION	RELACION (INT)	LUGAR ORIGEN	POS_COLOR	OPERADOR
3	1	X	-1	E	4	1	1	=
4	2	J	-1	X+1	1	-1	1	=
5	3	W	-1	Z	6	2	2	=
6	4	E	-1	X	1	2	1	=
7	5	P	-1	P	5	2	3	=
8	6	Z	-1	W	3	3	1	=

Figura 4-11: Codificado de las relaciones entre variables

Tal como se puede observar en la Figura 4-11, se utilizan 8 columnas, donde las primeras cuatro se utilizan para definir las variables a través de un identificador ordinario, los valores iniciales que toman las variables (-1), nombre de las variables y la expresión lógica de cada una de ellas.
Las siguientes tres columnas (relación, lugar origen y pos_color) sirven para establecer las relaciones entre las diferentes variables, y su posición dentro de la estructura de la RdPC, la columna de operador varía para cada variable y se obtiene de la expresión del guarda que relaciona cada una de las variables.

Ejemplo 4.2 Definición de la variable X

El guarda *[X=E]*, contiene en su expresión dos variables: variable X y variable E.
La variable X se define en la tabla de la Figura 4-11 de la siguiente manera:

- Columna A: Se define arbitrariamente como 1
- Columna B: El nombre identificador de la variable
- Columna C: Valor inicial de la variable
- Columna D: La Expresión de guarda "E"
- Columna E: Debido a que la variable E en se define en esta tabla como la cuarta variable, la relación de X con E tendrá el valor de 4.
- Columna F: En el modelo original (Figura 4-8) se observa que la variable X pertenece al arco de entrada que sale del lugar 1, y la posición de la variable dentro de los colores de la expresión es 1.
- Columna G: La posición de la variable en la expresión de arco.
- Columna H: Es el operador de la expresión de guarda.

4.2.2.1 Optimización y Resultados

A partir de la definición de la RdPC, relación de variables, restricciones y el marcado inicial, es posible utilizar la herramienta implementada para evaluar el funcionamiento del algoritmo.

Para el caso del taller de trabajo 3x3 se ha encontrado una ruta óptima que se presenta en la figura 4-12 a través de un diagrama de Gantt de la secuencia de operaciones para llevar a cabo las tareas de los tres trabajos.

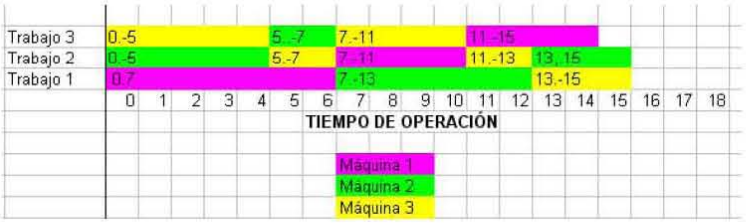


Figura 4-12: Diagrama de Gantt de la secuencia óptima para el T-T 3x3

La secuencia obtenida utilizando la herramienta es de 15 unidades de tiempo, la cual corresponde con la reportada en literatura [132].

Se reporta también un tiempo de solución de 5 minutos con un simulador para modelos de RdPC [88]. La solución con el algoritmo propuesto en este trabajo se encuentra en 32 segundos, lo cual contrasta favorablemente con lo reportado.

La información detallada de este modelo se presenta en el anexo 10.

4.3 Taller de trabajo 6x6

La codificación en la interfase de usuario de este modelo es similar, por lo que solo se presenta el modelo de RdPC, el diagrama de Gantt correspondiente así como los datos relevantes respecto al árbol de alcance y la simulación misma. Para el problema detallado revisar anexo 10.

Los datos para el taller de trabajo 6x6 se presentan en la Tabla 4-7.

Tabla 4-7: Secuencia de Operaciones para el Taller de trabajo 6x6

SEQ. TAREAS	TRABAJO 1		TRABAJO 2		TRABAJO 3		TRABAJO 4		TRABAJO 5		TRABAJO 6	
	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo
1	2	1	1	8	2	5	1	5	2	9	1	3
2	0	3	2	5	3	4	0	5	1	3	3	3
3	1	6	4	10	5	8	2	5	4	5	5	9
4	3	7	5	10	0	9	3	3	5	4	0	10
5	5	3	0	10	1	1	4	8	0	3	4	4
6	4	6	3	4	4	7	5	9	3	1	2	1

El modelo para este sistema tiene las restricciones que cada tarea de cada trabajo solo se puede realizar por una sola máquina, y las tareas deben ser realizadas en secuencia.

El modelo desarrollado en RdPC es similar al del taller de trabajo 3x3 con ligeras variaciones, el modelo se presenta en la figura 4-13

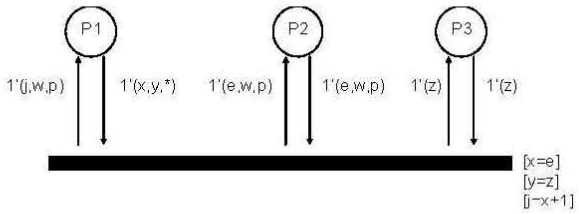


Figura 4-13: Modelo en RdPC del taller de trabajo 6x6

La Tabla 4-8 resume los colores utilizados para describir toda la información requerida por el modelo.

Tabla 4-8: Especificación de colores taller de trabajo 6x6

COLOR	DEFINICIÓN	DESCRIPCIÓN
J	Entero 11..66	Identificador de trabajo y tarea
W	Entero 1..6	Máquina necesaria para la siguiente tarea
X	Entero 11..66	Trabajo y tarea en progreso
E	Entero 11..66	Identificador de trabajo y tarea
P	Entero	Tiempo de cada tarea
Z	Entero 1..6	Máquinas disponibles
Y	Entero 1..6	Maquina en uso

La Tabla 4-9 presenta la información de los lugares del modelo.

Tabla 4-9: Descripción de los lugares taller de trabajo 6x6

LUGAR	COLOR	DESCRIPCIÓN
P1	Producto j*w*p	El valor del color j representa el trabajo en turno y la tarea que se está realizando en un instante de tiempo. El color w representa la máquina que es necesaria para llevar a cabo la siguiente tarea del trabajo correspondiente y el color p es el tiempo de duración de la operación realizada.
P2	Producto e*w*p	Representa la información de la secuencia lógica que deben seguir las diferentes actividades de los trabajos correspondientes. Los colores e,w y p representan respectivamente el trabajo y tarea en que se encuentra, la máquina siguiente al terminar la tarea del color e, y el tiempo de operación p que consumirá al llevar a cabo la tarea correspondiente en la máquina w.
P3	z	La información asociada con la máquina disponible.

En este modelo, los tokens del lugar P1 tienen un color extra (y) el cual sirve para especificar cuál es la máquina que sigue en la secuencia de tareas, también existe el guarda [y=z] el cual especifica la disponibilidad de la máquina del lugar P3. Lo que se refiere a la codificación en la interfase de usuario es muy similar así como también el estado inicial del sistema, por lo que no se vuelve a presentar en esta sección.

4.3.1.1 Optimización y Resultados

Este modelo presenta varias soluciones óptimas para la duración del tiempo total de procesamiento, en la Figura 4-14 se presenta una de las soluciones óptimas obtenidas.

Figura 4-14: Una solución óptima del taller de trabajo 6x6

La solución presentada es solo una de las posibles soluciones óptimas, ya que este problema presenta varias soluciones óptimas con una duración total de 55 unidades de tiempo.
Para el problema detallado revisar anexo 10.

4.4 Taller de trabajo 5x5

El modelo en RdPC para el problema del taller de trabajo 5x5 es exactamente el mismo que el de la Figura 4-13 para el taller de trabajo 6x6, solo que en este caso la multiplicidad del modelo es menor que la del modelo 6x6. La Tabla 4-10 presenta la información para este sistema.

Tabla 4-10: Secuencia de Operaciones para el taller de trabajo 5x5

SEQ. TAREAS	TRABAJO 1		TRABAJO 2		TRABAJO 3		TRABAJO 4		TRABAJO 5	
	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo
1	4	85	1	64	3	31	5	44	2	66
2	1	7	4	14	2	69	5	18	3	68
3	4	1	1	74	2	70	5	90	3	60
4	2	45	4	76	5	13	3	98	1	54
5	1	80	4	15	2	45	5	91	3	10

4.4.1.1 Optimización y Resultados

El óptimo encontrado para este modelo se presenta en el diagrama de Gantt del Figura 4-15.

Figura 4-15: Diagrama de Gantt de la secuencia óptima del T-T 5x5

Para detalles del modelo ver anexo 10

4.5 Máquina de Control Numérico

Un caso que resultó interesante por su complejidad fue el de una máquina de control numérico por que se dedicaba a rectificar y biselar lentes ópticas. Este problema tiene las características que el tamaño del árbol de alcance supera las capacidades de la memoria física de la computadora utilizada. Debido a su complejidad se propone este problema como un *benchmark* para llevar a cabo la prueba de algoritmos que hagan uso de las técnicas aquí presentadas.

4.5.1 Descripción del sistema

La maquina real llevaba a cabo diferentes operaciones con las lentes. La Figura 4-16 presenta un diagrama de la máquina real.

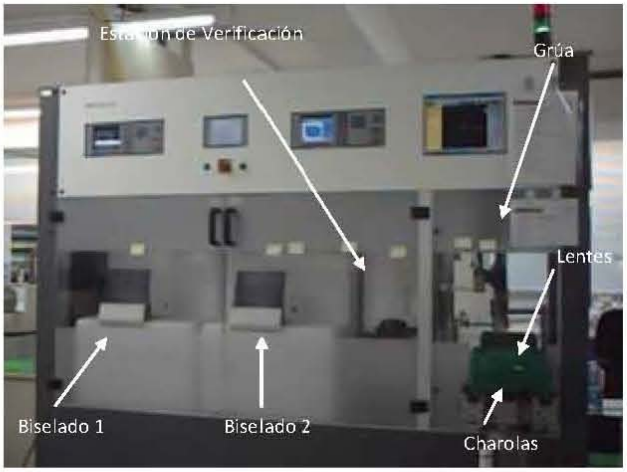


Figura 4-16: Esquema de la máquina de control numérico

La máquina cuenta con tres estaciones de trabajo. Una estación (**Ver**) checa inicialmente si los lentes tienen las dimensiones de acuerdo a especificación. La segunda y tercer estaciones realizan operaciones de trabajo en las lentes de manera que se ajustan perfectamente al tipo de marco en el cual deben ser ensambladas (**Biselado: B1 y B2**). Forma parte también del sistema una grúa con dos tenazas para transportar las lentes desde sus charolas a las estaciones de trabajo. Por otra parte cada uno de los anteojos se compone de dos lentes (lente derecha e izquierda). Las lentes deben seguir una secuencia a través de todo el proceso.
Primeramente deben ser verificadas y luego biseladas. La operación de biselado debe ser llevada a cabo para cada una de las lentes que compone los anteojos por lo que si por ejemplo se procesan tres anteojos entonces 6 lentes deben ser procesadas.
El problema consiste en encontrar la mejor secuencia de proceso de una carga de trabajo específica, considerando los movimientos de la grúa. Cabe hacer notar que hay tres tipos de operación de biselado dependiendo del tipo de lente y cada una tiene un tiempo de procesamiento diferente lo cual complica considerablemente el problema si además se requiere evitar los tiempos ociosos de la grúa y la máquina de verificación.
Los principales elementos del modelo en RdPC temporales se presentan a continuación.

4.5.2 Definición de Colores

Como se mencionó previamente el sistema utiliza modelos temporales de RdPC. La información acerca de los tiempos de proceso para las operaciones de biselado así como valores que sirven para determinar la secuencia de operaciones se codificaron en los colores de cada token. La Tabla 4-11

describe la información codificada en los colores. La primera columna define la variable utilizada por el color correspondiente, la segunda columna define los dominios de las correspondientes variables y la 3er columna da una descripción de la información almacenada en los colores.

Tabla 4-11: Definición de los colores del modelo

COLOR	DEFINICION	DESCRIPCION
l	{0,1}	Este color se utiliza para permitir solo un movimiento de la grua haciendo uso del guarda en el lugar G
ldc	Integer	Este color se utiliza como identificador
lid	Integer	Este color se usa como identificador del tipo de lente en el lugar VER.
lidl	Integer	Este color se utiliza como identificador de lente izquierda
lidr	Integer	Este color se utiliza como identificador de la lente derecha
ell	Integer	Este color se utiliza para identificar la operación realizada en la lente izquierda
elr	Integer	Este color se utiliza para identificar la operación realizada en la lente derecha
tvisll	Integer	Este color se utiliza para especificar el tiempo consumido por la operación de bicelado en la lente izquierda
tvislr	Integer	Este color se utiliza para especificar el tiempo consumido por la operación de bicelado en la lente derecha
posc	Integer	Este color se utiliza como identificador del número de lentes del mismo tipo
posg	Integer	Este color especifica la posición fisica de la grùa que mueve los lentes

Los nodos lugar utilizan conjuntos compuestos por los colores previamente definidos, la Tabla 4-12 describe los conjuntos de colores utilizados para los diferentes nodos lugar del modelo.

Tabla 4-12: Conjuntos color en los lugares

LUGAR	GRUPOS COLOR	DESCRIPCION
Buckets	C=product idc*posc*lidl*lidr*ell*elr*tvisll*tvislr	Los tokens de este lugar registran la información de cada par de lentes definido por los valores de los colores.
Crane	G= product lidl*lidr*posg	Los tokens en este lugar modelan la información de la grúa que transporta las lentes en las diferentes estaciones.
G	I	Este lugar se utiliza paras limitar el movimiento de la gúa a través de los guardas correspondientes en las transiciones del movimiento de la grúa.
Ver	V= lidl	Los tokens de este lugar representan la máquina de verificación, debido a que solo hay una estación solo habrá un token.
Beveling	V=lidl	Los tokens the este lugar modelan las máquinas de bicelado, debido a que existen dos, habrá dos tokens.

4.5.3 Especificación de Transiciones

El modelo se codificó usando 4 grupos de transiciones; dos de esos grupos son para las operaciones de verificado y biselado y el resto de transiciones se utilizan para modelar el proceso de transporte de las lentes por medio de la grúa.

4.5.3.1 Transiciones para modelar operaciones

La operación de verificación se lleva a cabo con un grupo de cuatro transiciones, dos para los lentes izquierdos y dos para las lentes derechas. Estas transiciones modelan las actividades llevadas a cabo en las lentes durante los procesos de operación. Existen nodos lugar de entrada común a estas transiciones, los cuales son **Ver**, **Crane**, **Buckets** y **G**. La Figura 4-17 muestra un ejemplo del tipo de transiciones de este grupo.

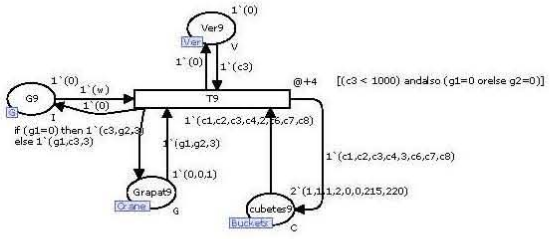


Figura 4-17: Ejemplo de una transición de la operación de verificación

La transición T9 quedará habilitada únicamente si las siguientes condiciones se satisfacen:

- Existen charolas que ya han sido verificadas una ocasión (ell=2)
- La grúa se encuentra en la posición 3 (posg=3), es decir existe un token del tipo 1'(g1,g2,3)
- Una de las pinzas de la grúa se encuentra disponible (g1=0 or else g2=0)
- La máquina de verificado se encuentra disponible (c3<1000)
- En este caso el lugar G no restringe de ninguna manera la habilitación de la transición pero ayuda a activar el movimiento de la grúa

Las operaciones de biselado son un grupo de cuatro transiciones también. Dos transiciones para las lentes izquierdas y dos para las lentes derechas. Estas transiciones también cuentan con cuatro nodos lugar de entrada: Bevelling, Crane, Buckets y G. La Figura 4-18muestra un ejemplo de una transición de este grupo.

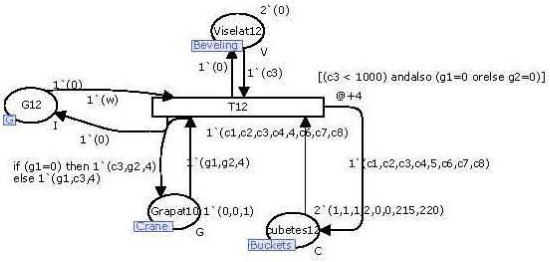


Figura 4-18: Ejemplo de una transición de biselado

Para que una transición de este tipo se active tiene que ocurrir lo siguiente:

- La grúa tiene que tener pinzas disponibles (g1=0 orelse g2=0)
- Las operaciones previas (verificación lente izq y verificación lente der) ya han sido llevadas a cabo. Esto es verificado con los identificadores **ell** o **elr.**
- La secuencia de las operaciones de biselado es la correcta(identificadores **lidr** o **lidl**)
- La grúa se encuentra en la posición correcta para las operaciones de biselado (posg=4).

Cada vez que alguna de estas operaciones se lleve a cabo, el color del token en el lugar G adquirirá el valor de 0 para permitir otro movimiento de la grúa.

4.5.3.2 Transiciones para modelar movimiento

El modelado de las operaciones de recogida y entrega de lentes por medio de la grúa se modela haciendo uso de cuatro transiciones, dos para las lentes izquierdas y dos para las lentes derechas. La Figura 4-19 es un ejemplo de una transición correspondiente para lentes izquierdas. Esta transición modela la recogida de la lente izquierda de las charolas.

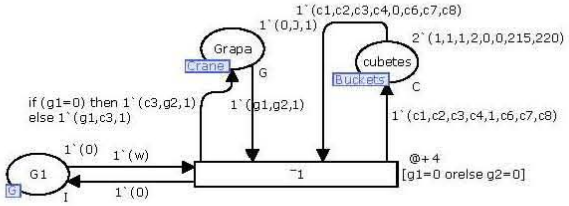


Figura 4-19: Las transiciones de recogida de lentes

Para llevar a cabo cualquier recogida de lente por medio de la grúa las siguientes condiciones deben ser satisfechas:

- La grúa se debe encontrar en la posición correcta (posg=1)
- Las lentes deben estar listas para la operación inicial (ell=0)
- Cualquiera de las pinzas de la grúa debe estar vacía (g1=0 orelse g2=0)

Para el caso de la entrega de lente a una estación, la correspondiente transición es similar y se deben de satisfacer las siguientes condiciones:

- La grúa se debe de encontrar en la posición de entrega
- La operación llevada a cabo en la lente izquierda tiene que ser correspondiente a la operación efectuada expresada por el color **ell**
- La grúa debe de transportar la lente correcta

Finalmente el movimiento de la grúa se modela con 4 transiciones. La Figura 4-20 da un ejemplo del grupo de transiciones utilizadas para modelar el movimiento de la grúa.

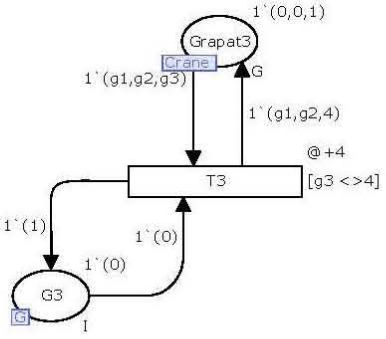


Figura 4-20: Movimiento de grúa

El movimiento de la grúa se modela por el token en el nodo lugar **Crane**, el color **posg** modela la posición actual de la grúa. Dos condiciones deben ser satisfechas para permitir el movimiento:

- la correspondiente posición no está actualmente ocupada por la grúa (g3<>4)
- La última operación efectuada no era un movimiento de grúa, es decir el token en el lugar **G** es 1'(0)

Las transiciones presentadas en esta sección son ejemplos de las transiciones del modelo completo el cual puede ser fácilmente construido con los ejemplos proporcionados.

4.5.4 Escenarios Implementados

Los anteojos se diferencian en el modelo por los identificadores y por el tiempo y tipo de trabajo que se debe de realizar en la operación de biselado. Con base en esta información se definieron tres tipos de lentes para el modelo en RdPC los cuales corresponden a los tokens presentes en el nodo lugar **Buckets**:

Lente tipo A: 1'(1,1,1,2,0,0,215,220)
Lente tipo B: 1'(2,1,3,4,0,0,120,120)
Lente tipo C: 1'(3,1,5,6,0,0,540,540)

Para analizar el sistema algunas cargas de trabajo se evaluaron. La Tabla 4-13 muestra la información de las cargas de trabajo. La primera columna muestra el tamaño y el tipo de lentes que se van a procesar. La segunda columna presenta los correspondientes estados iniciales para los modelos en RdPC.

Tabla 4-13: Escenarios de cargas de trabajo

Carga de Trabajo	Marcado Inicial
2 Lentes, 2 tipo A	Crane: 1'(0,0,1)@0 Buckets: 2'(1,1,1,2,0,0,215,220)@[0,0] G: 1'(0)@0 Ver: 1'(0)@0 Beveling: 2'(0)@[0,0]
3 Lentes, 3 tipo A	Crane: 1'(0,0,1)@0 Buckets: 3'(1,1,1,2,0,0,215,220)@[0,0,0] G: 1'(0)@0 Ver: 1'(0)@0 Beveling: 2'(0)@[0,0]
3 Lentes, 2 tipo A, 1 tipo B	Crane: 1'(0,0,1)@0 Buckets: 2'(1,1,1,2,0,0,215,220)@[0,0]+1'(2,1,3,4,0,0,120,120)@0 G: 1'(0)@0 Ver: 1'(0)@0 Beveling: 2'(0)@[0,0]
4 Lentes, 2 tipo A, 1 tipo B, 1 tipo C	Crane: 1'(0,0,1)@0 Buckets: 2'(1,1,1,2,0,0,215,220)@[0,0] +1'(2,1,3,4,0,0,120,120)@0 +1'(3,1,5,6,0,0,540,540)@0 G: 1'(0)@0 Ver: 1'(0)@0 Beveling: 2'(0)@[0,0]
22 Lentes (Carga Normal): 7 tipo A, 13 tipo B, 2 tipo C	Crane: 1'(0,0,1)@0 Buckets: 7'(1,1,1,2,0,0,215,220)[0,0,0,0,0,0,0] +13'(2,1,3,4,0,0,120,120)@[0,0,0,0,0,0,0,0,0,0,0,0,0] +2'(3,1,5,6,0,0,540,540)@[0,0] G: 1'(0)@[0] Ver: 1'(0)@[0] Beveling: 2'(0)@[0,0]

Como se puede apreciar directamente de la tabla, se comenzó con cargas de trabajo de baja complejidad. Primero se analizaron cargas de trabajo para una cantidad pequeña de lentes donde todas eran del mismo tipo (p. ej. 2 y 3 lentes del tipo A). Posteriormente se fue acrecentando la complejidad variando los tipos y numero de lentes a ser procesados. Finalmente el modelo fue probado con una carga de tamaño real (22 lentes).

Un objetivo de estas pruebas era el evaluar el tamaño de los diferentes árboles de alcance y el objetivo final era la obtención de una secuencia óptima o cercana al óptimo haciendo uso del enfoque previamente presentado.

4.5.5 Soluciones encontradas

El algoritmo en dos pasos fue utilizado para resolver el problema de la secuenciación de actividades con el objetivo de minimizar el tiempo total de procesamiento para las diferentes configuraciones presentadas en la Tabla 4-14. La primera columna muestra el estado final alcanzado junto con los valores de los sellos de tiempo correspondientes. La segunda columna muestra el valor del tiempo total de proceso en segundos. La tercer columna muestra la información referente a si la exploración realizada pudo analizar todos los estados diferentes del sistema o si se tuvo que realizar una exploración parcial. La cuarta columna muestra información particular del árbol de alcance explorado.

Tabla 4-14: Soluciones encontradas

Marcado Final	Makespan	No. Total de Nodos Analizados	Información Estructural
Crane: 1'(0,0,1)@616 Buckets: 2'(1,1,1,2,6,6,215,220) @[599,616] G: 1'(0)@591 Ver:1'(0)@376 Beveling:2'(0)@[599,616]	616 seg	Árbol completo	No. Nodos: 19,232 No. Arcos:59,610 No. OLD Nodos:15,431 Niveles: 53
Crane: 1'(0,0,1)@1228 Buckets: 3'(1,1,1,2,6,6,215,220) @[1109,1085,1228] G: 1'(0)@1143 Ver: 1'(0)@1104 Beveling: 2'(0)@[1088,1228]	1,228 seg	Árbol completo	No. Nodos: 172,242 No. Arcos:765,177 No. OLD Nodos:145,911 Niveles: 84
Crane: 1'(0,0,1)@183 Buckets: 2'(1,1,1,2,6,6,215,220) @[917,950] +1'(2,1,3,4,6,6,120,120)@954 G: 1'(0)@1183 Ver: 1'(0)@715 Beveling: 2'(0)@[909,942]	1,183 seg	Árbol completo	No. Nodos: 562,799 No. Arcos:1,800,951 No. OLD Nodos:471,939 Niveles: 81
Crane: 1'(0,0,1)@1898 Buckets: 2'(1,1,1,2,6,6,215,220) @[1483,1874] +1'(2,1,3,4,6,6,120,120)@1878 +1'(3,1,5,6,6,6,540,540)@1894 G: 1'(0)@1898 Ver: 1'(0)@1639 Beveling: 2'(0)@[1866,1886]	1,906 seg 1,898 seg	500,000 nodos 800,000 nodos	--- No. Nodos: --- No. Arcos:>2,541,330 No. OLD Nodos:>676,223 Niveles: 105
Crane: 1'(0,0,1)@10421 Buckets: 7'(1,1,1,2,6,6,215,220) @[8253,8512,8771,9030,9314,9598,9849] +13'(2,1,3,4,6,6,120,120) @[6062,6226,6390,6554,6718,6882,7046, 7210,7362,7452,7641,7830,7994] +2'(3,1,5,6,6,6,540,540)@[5898,10417] G: 1'(0)@[10421] Ver: 1'(0)@[9857] Beveling: 2'(0)@[9841,10409]	10,421 seg	800,000 nodos	No. Nodos: --- No. Arcos:>3,911,731 No. OLD Nodos:>675,161 Niveles: 614

Los resultados de los tres primero escenarios pudieron obtenerse analizando todo el árbol de alcance. Puede observarse que cuando la carga de trabajo es mayor a 3 tipos diferentes de lentes el árbol de alcance es mayor a 800,000 nodos, lo cual ocasiona una saturación de memoria. Por lo mismo para estos casos no se pudo llevar a cabo una exploración total del árbol de alcance; en su

lugar se exploró el árbol parcialmente. En el caso del problema con carga real de 22 lentes se resolvió con una exploración parcial obteniéndose un tiempo total de proceso de 10,421 segundos.

Es importante señalar que cuando se lleva a cabo una exploración parcial no es posible saber que tan cerca o lejos se encuentra el resultado del valor óptimo; pero dado que las heurísticas presentadas dieron buenos resultados con los modelos de Taller de trabajo se puede esperar que los resultados obtenidos sean cercanos al óptimo.

Para poder comparar las ventajas de utilizar este enfoque contra las que pudiera ofrecer una herramienta académica como CPNtools [27]. En particular CPNtools realiza una generación del árbol de alcance sin llevar a cabo ningún tipo de simplificación o aprovechando algún tipo de simetría presente en el modelo. La Tabla 4-15 presenta la comparación de los árboles de alcance realizada.

Tabla 4-15: Comparación de herramientas para modelos de RdPC

Carga de Trabajo	Modelo CPNTools	Algoritmo en Dos pasos
2 Lentes, 2 tipo A	Nodos: 389,884 Arcos: 457,992 Seg: 8558 Status: Completo	No. Nodos: 19,232 No. Arcos:59,610 No. Nodos Old:15,431 Niveles: 53
3 Lentes, 3 tipo A	Imposible de generar el árbol	No. Nodos: 172,242 No. Arcos:765,177 No. Nodos Old:145,911 Niveles: 84
3 Lentes, 2 tipo A, 1 tipo B	Imposible de generar el árbol	No. Nodos: 562,799 No. Arcos:1,800,951 No. Nodos Old:471,939 Niveles: 81

Esta tabla permite apreciar que con el algoritmo desarrollado el cálculo computacional se ve reducido. Se puede apreciar también que el tamaño del árbol de alcance generado por CPNtools para el análisis del árbol de alcance es considerablemente mayor que el que genera el algoritmo en dos pasos. Un detalle importante que hay que remarcar es que para árboles grandes CPNtools no puede arrojar una solución factible debido a la saturación temprana de memoria así como la técnica que utiliza para explorar el árbol de alcance mientras que el algoritmo en dos pasos a pesar que explora parcialmente el árbol permite dar solución al problema de secuenciación de operaciones.

5 CONCLUSIONES Y LINEAS DE INVESTIGACIÓN FUTURAS

En este trabajo se han presentado todos los elementos que se desarrollaron para la conceptualización y posterior codificación de un algoritmo que hace uso de las redes de Petri coloreadas tanto temporales como atemporales. El algoritmo utiliza el árbol de alcance también conocido como el espacio de estados de los modelos de redes de Petri con el objetivo de explorar las posibles configuraciones de un sistema industrial. El trabajo desarrollado sirve para obtener schedulings o secuenciaciones optimizadas de sistemas industriales minimizando a su vez el tiempo total de proceso. La mayoría del trabajo aquí presentado se ha presentado en congresos internacionales y en revistas científicas de nivel internacional. A continuación se presentan las principales aportaciones del trabajo así como las conclusiones y líneas futuras de investigación.

5.1 Principales Aportaciones

Este trabajo de investigación tiene como principales aportaciones las siguientes:

- La generación de los diferentes estados de la red de Petri coloreada haciendo uso de técnicas por programación con restricciones permite la generación rápida y eficiente del árbol de alcance. Esta característica es necesaria para el desarrollo de herramientas de apoyo a la toma de decisiones a nivel industrial.
- La manera de administrar la información generada haciendo uso de un par de capas de información permite evitar la saturación de la memoria del computador de manera que es posible analizar espacios de estados más grandes que los que serían posibles almacenando la información del marcado sin implementar ningún tipo de técnica para ahorrar espacio de memoria.
- El uso de las simetrías para el desarrollo de un espacio de estados o árbol de alcance compacto permite aumentar aún más la capacidad de análisis de este tipo de herramientas.
- El algoritmo en dos etapas permite separar la generación de estados y análisis de estados llevada a cabo en la etapa de optimización del modelo lo cual resulta útil particularmente cuando se requiere únicamente verificar las propiedades u optimizar el sistema estudiado.
- Las heurísticas implementadas mejoran el rendimiento del algoritmo en dos pasos. Estas implementaciones permiten llevar a cabo de una manera más inteligente la exploración en profundidad (depth-first search) lo cual genera en la primera etapa una ruta factible con buenos valores de la función objetivo y ocasiona que la segunda etapa funcione más eficientemente para la optimización de modelos industriales.
- La segunda etapa permite analizar los nodos simétricos repetidos de manera que la marca objetivo o estado final es mejorada progresivamente con el análisis. En particular el algoritmo se enfocó en la obtención de schedulings con tiempo total de proceso mínimo en sistemas de manufactura.
- Los algoritmos desarrollados en este trabajo pueden ser utilizados para el desarrollo de herramientas de apoyo a la toma de decisiones a nivel operacional.

5.2 Conclusiones

En este trabajo se ha presentado el desarrollo de un algoritmo y heurísticas que hacen uso del formalismo de redes de Petri coloreadas en conjunto con el llamado espacio de estados o árbol de alcance. Estos se implementaron en un entorno de pruebas codificado en Visual.Net
Las principales conclusiones se presentan a continuación:

- El algoritmo desarrollado permite la generación eficiente del árbol de alcance haciendo uso de las implementaciones para la evaluación de transiciones.
- Las estructuras de información desarrolladas permiten el almacenado del árbol de alcance tanto acotado como no acotado de manera que el espacio físico necesario se ve minimizado incrementando con eso la capacidad de análisis de herramientas desarrolladas con este enfoque.
- El algoritmo tiene la particular ventaja respecto de herramientas disponibles que puede generar soluciones en tiempos cortos y las soluciones aportadas a pesar que no se puede asegurar sean óptimas se confía que son cercanas al óptimo debido al desempeño obtenido en pruebas con modelos académicos.
- Los algoritmos desarrollados permiten analizar sistemas industriales y generar soluciones en tiempos cortos lo cual va de acuerdo con el objetivo inicialmente planteado.
- Se puede concluir que los algoritmos e implementaciones desarrolladas apuntan en la dirección correcta para desarrollar herramientas más eficientes que sirvan como herramientas de apoyo a la toma de decisiones a nivel industrial.
- El algoritmo en dos fases es eficiente y permite el análisis de modelos así como la optimización de los mismo pero es factible de ser mejorado en ambas etapas tanto generación como análisis. Se han presentado heurísticas eficientes pero aun queda investigación por realizar para el desarrollo de técnicas más eficientes que utilicen este enfoque.

5.3 Direcciones Futuras

Como se mencionó anteriormente el algoritmo en dos etapas presentado puede ser mejorado tanto en la etapa de generación de estados como en la parte del análisis de los estados repetidos. A continuación se mencionan las líneas futuras de trabajo, algunas de las cuales ya están siendo investigadas pero desafortunadamente no pudieron ser incluidas en este documento:

- Se debe implementar una heurística para la primera etapa que tome en cuenta otros elementos además de los sellos de tiempo únicamente. Estos elementos deben ser parte de la estructura de la red de Petri para poder llevar a cabo una exploración donde exista una causalidad con los valores de los sellos de tiempo y en base a eso seleccionar de una mejor manera el siguiente nodo a evaluar.
- Los desarrollos aquí presentados se evaluaran utilizando diferentes funciones de coste para poder solucionar problemas que tengan objetivos diferentes al tiempo total de proceso.
- Lo anterior se puede aplicar también para la segunda etapa del algoritmo lo cual ocasionaría una mejor selección de los nodos a descartar durante el análisis.

- El desarrollo de una manera distinta de evaluación de los sellos de tiempo permitirá descartar aquellas ramas que *a priori* muestren poco potencial para la mejora de los valores del marcado objetivo.
- Estos algoritmos deben ser probados en arquitecturas distribuidas lo cual podría reducir drásticamente el tiempo en el cual se obtienen soluciones para problemas industriales así como se podría aprovechar la memoria compartida para el almacenado de un mayor número de estados.
-

Estas son solo algunas de las líneas de investigación que pueden dar continuación al trabajo presentado en esta tesis.

7 ANNEXO A: PUBLICACIONES

Las diferentes partes de este trabajo se presentaron en congresos y conferencias internacionales las cuales generaron diversas publicaciones en los *proceedings* o memorias correspondientes. Además parte del trabajo se ha publicado en revistas internacionales. Todo lo anterior se enumera a continuación.

Publicaciones en Revistas.

Mujica, M., Piera, M.A., Narciso, M., 2010, "Revisiting state space exploration of timed coloured petri net models to optimize manufacturing system's performance", in *Simulation Modelling Practice and Theory*, vol.8 (9), pp.1225-1241, Elsevier

Mujica, M., Piera, M.A., 2011, "A compact timed state space approach for the analysis of manufacturing systems: key algorithmic improvements", in *International Journal of Computer Integrated Manufacturing*, vol.2 (24), pp.135-153, Taylor & Francis

Mujica, M., Piera, M.A., 2010, "Performance optimisation of a CNC machine through exploration of timed state space", in *International Journal of Simulation and Process Modelling*, vol.6 (2), pp.165-174, Inderscience

Publicaciones como primer autor en congresos.

Mujica, M., Piera, M.A., 2006 "Building an Efficient Coloured Petri Net Simulator", in *Proc. of the International Mediterranean Modeling Multiconference*, p.p.153-158, October, Barcelona, Spain.

Mujica, M., Piera, M.A, 2007, "Data Management to Improve CPN Simulation Performance", in *Proc. of the International Mediterranean Modeling Multiconference*, p.p.53–58, October, Bergeggi, Italy.

Mujica, M., Piera, M.A., 2007,"Improvements for a Coloured Petri Net Simulator", in *Proc. of the 6th EUROSIM Congress on Modeling and Simulation*, p.p. 237, September, Ljubljana, Slovenia.

Mujica, M., Piera, M.A., Narciso M., 2008,"Optimizing Time Performance in Reachability Tree-Based Simulation", in *Proc. of the I3M Congress on Modeling and Simulation*, September, Briatico, Italy.

Mujica, M., Piera, M.A., 2009, "Optimizing Systems Performance Exploring the Timed State Space of Coloured Petri Nets", in *Proc of the OR-EUROCONFERENCE*, July,Bonn, Germany.

Mujica, M., Piera, M.A., 2009, "A Two Step Algorithm to Improve Systems Optimization based on the State Space Exploration for Timed Coloured Petri Nets", in Proc. of the TiSto 2009, June, Paris, France.

Mujica, M., Piera, M.A., 2009,"Performance Optimization of a CNC Machine Through Exploration of Timed State Space", in Proc. of the I3M 2009 Simulation Multiconference, September, Canary Islands, Spain.

Mujica, M., Piera, M.A., 2010,"Hybrid Search Algorithm to Optimize Scheduling Problems for TCPN Models", in Proc. of the SCSC Simulation Conference, July, Ottawa, Canada.

Publicaciones en co-autoría

M. A. Piera, L. Vilalta, M. Mujica, M. Narciso, T. Guasch, 2006, "An Efficient Modelling Approach to Combine Search Methods with Evaluation Methods", in Proceedings of the European Modelling Symposium, Londres (UK), 11-12 September 2006.

M. A. Piera, T. Guasch , L. Vilalta, M. Mujica, 2005, "Several Hints for A Master Search using Coloured Petri Net Simulators", in Proceedings of the SCSC, Calgary, Canada, 30 Julio – 2 Agosto 2006.

Piera M. A, Mujica M., Guasch, 2007, "An Efficient CN Modeling Approach to Tackle the Pallet Packing Problem", in *Proceedings of the 6th EUROSIM Congress on Modeling and Simulation,* p.p. 344, September 9-13, Ljubljana, Slovenia.

8 ANEXO B: INTERFASE DE USUARIO DEL MODELO DE RDPC

La implementación del algoritmo se ha llevado a cabo utilizando VBA [13] para utilizar EXCEL como interfase de usuario. Para la implementación de modelos de RdPC es necesario respetar una sintaxis diseñada que permite especificar los elementos de las RdPC.

Marcado Inicial

Para especificar el marcado inicial, es necesario descomponer la información de la RdPC para especificar el nivel color y el nivel relación del estado inicial. El nivel color y el nivel relación se especifican en una hoja de cálculo cada uno, siguiendo la siguiente sintaxis:

Hoja 1: Nivel Color

La información del modelo se especifica utilizando dos columnas para cada lugar del modelo y la información que se especifica es la cardinalidad y color.

- Columna 1 (Cardinalidades): La definición de esta columna se lleva utiliza tres elementos.
  - [Enteros]: Representan el número de tokens presentes de un mismo elemento.
  - [,] : Este símbolo se utiliza como separación entre cardinalidades
  - [&]: Este símbolo se utiliza para separar dos combinaciones de cardinalidades. Ver sección 2.4.1.
- Columna 2 (Color) : Se especifica a través de tres elementos:
  - [Enteros]: Representan el número de color del token
  - [,]: Elemento separador de colores
  - [&]:Elemento separador de tokens

Hoja 2: Nivel Relación

Esta hoja se utiliza para definir el nivel relación (ver sección 2.4.2). Se utilizan tres columnas para definir el marcado inicial y dos columnas más para especificar los tiempos iniciales y reloj global cuando se trata de redes temporales.

- Columna 1. Esta columna define que cardinalidades se utilizan en los diferentes lugares del modelo; la definición se lleva a cabo con los siguientes elementos:
  - [Enteros]:Representan el número de tokens de un mismo tipo que se utilizan
  - [,]: Elemento separador de cardinalidades
  - [&]: Elemento separador de cardinalidades entre dos lugares distintos

[13] VBA: Visual Basic for Applications

- Columna 2. Esta columna establece la relación entre los lugares del nivel color, y se define utilizando los siguientes elementos:
  - [referencia alfanumérica]: Se utiliza para hacer referencia a la posición de la información en la Hoja 1 del nodo lugar.
  - [-] : Elemento para separar referencias
- Columna 3. Indica el nodo original a través de la palabra "Null". En esta columna se almacena la referencia al estado que genera el marcado por primera vez o el estado que genera los mejores tiempos del marcado cuando se trata de redes temporales.
- Columna 4. Indica los sellos de tiempo del marcado inicial. Se utilizan los siguientes elementos para especificar los sellos de tiempo:
  - [,]: Elemento para separar sellos de tiempo de tokens en un lugar.
  - [&]: Elemento que separa sellos de tiempo de lugares diferentes
  - [Enteros]: Los valores enteros sirven para definir el valor inicial de los sellos de tiempo.
- Columna 5. Se utiliza para la especificación del reloj global del marcado. Hace uso de números enteros para los valores del reloj.

En esta hoja existen dos columnas que se utilizan para almacenar la información de la relación entre nodos padres y nodos hijos a medida que se va generando el árbol; estas columnas no son manipulables por el usuario.

Ejemplo 8.1 Definición de Marcado Inicial

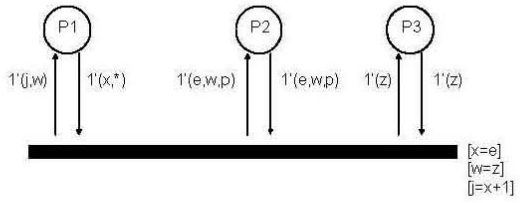


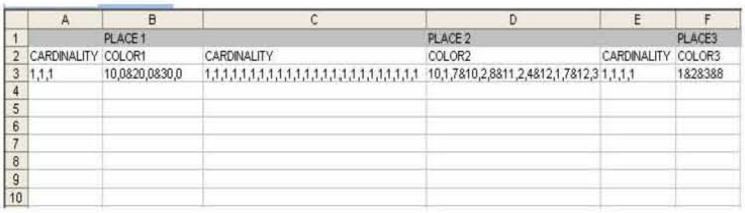
Figura 8-1: Modelo RdPC de Taller de trabajo 3x3

La figura 8-1muestra una RdPC para un sistema de manufactura, cuyo marcado inicial se describe en la tabla 8-1.

Tabla 8-1: Estado inicial modelo Taller de trabajo 3x3 con tiempos

MODELO 3X3	LUGAR		
	P1	P2	P3
Marcado Inicial	1'(10,0)+1'(20,0)+1'(30,0)	1'(10,1,7)+1'(10,2,8)+1'(11,2,4)+1'(12,1,7) +1'(12,3,4)+1'(13,8,1)+1'(20,1,6)+1'(20,2,5) +1'(21,2,4)+1'(21,3,2)+1'(22,1,6)+1'(23,1,3) +1'(23,2,2)+1'(24,8,1)+1'(30,1,8)+1'(30,3,5) +1'(31,2,2)+1'(32,2,6)+1'(32,3,4)+1'(33,1,4) +1'(33,2,2)+1'(34,1,2)+1'(34,3,3)+1'(35,8,1)	1'(1)+1'(2)+1'(3)+1(8)
Tiempos	[0,0,0]	[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]	[0,0,0]

La figura 8-2 y figura 8-3 muestran la especificación del marcado inicial utilizando las dos hojas de cálculo implementadas con este fin.



	A	B	C	D	E	F
1		PLACE 1		PLACE 2		PLACE3
2	CARDINALITY	COLOR1	CARDINALITY	COLOR2	CARDINALITY	COLOR3
3	1,1,1	10,0&20,0&30,0	1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1	10,1,7&10,2,8&11,2,4&12,1,7&12,3	1,1,1,1	1&2&3&8
4						
5						
6						
7						
8						
9						
10						

Figura 8-2: Hoja EXCEL utilizada para definir Nivel Color

En esta hoja se puede apreciar el uso de las dos columnas para almacenar la información de cada lugar (cardinalidades y colores).

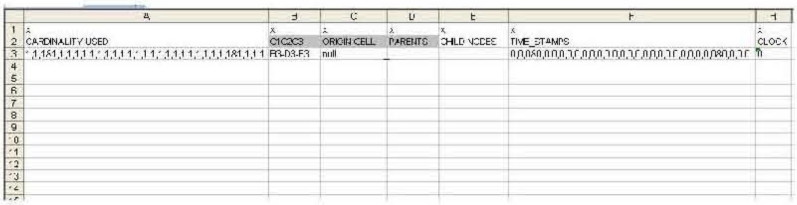


Figura 8-3: Hoja EXCEL para definir Nivel Relación

En esta hoja se aprecian las cinco columnas utilizadas para definir el marcado inicial del modelo, y también es posible apreciar las dos columnas que se utilizarán para mantener la relación entre los nodos que se generan durante la simulación.

Hoja 3: Estructura de la Red de Petri Coloreada

Se utiliza una hoja para especificar los siguientes elementos:

- Número de Nodos. Se utilizan números enteros en la celda indicada para la especificación.
- Número de Lugares. Se utilizan números enteros en la celda indicada para la especificación.
- Número de Variables. Se utilizan números enteros en la celda indicada para especificar la cantidad de variables a utilizar por el modelo
- Arcos. Se utiliza una matriz (lugar, transición) para indicar a través de una expresión alfanumérica cuando existe una relación entre los lugares con las transiciones, utilizando los siguientes elementos para la sintaxis de la expresión:
  - ' * ' : Indica que no existe relación entre el lugar y la transición
  - Expresión:[operador][Signo][Cardinalidad]['][Variables]|[Signo][Cardinalidad]['][Variables]
  - [Operador]: Cuando se tiene un arco de entrada y salida se coloca el símbolo [%]. No se escribe nada si solo es uno de salida o de entrada
  - [Signo]: [-] si es un arco de entrada, [+] si es un arco de salida a la transición
  - [Cardinalidad]: Es un entero que indica la cardinalidad del token y sea de entrada o salida
  - [Variables]: Se especifican los identificadores alfabéticos de las variables, o numéricos en caso que sea un valor constante, entre paréntesis y separados por comas [,].
  - [|] : Se utiliza únicamente cuando tiene un arco de entrada y de salida
- Tiempos de Transición (en redes temporales). Es un identificador al lado del identificador de la transición, y puede ser una variable o un valor constante entero.

Ejemplo 8.2 Definición de la Estructura de la RdPC

La definición de la estructura del modelo de la figura 8-1 se lleva a cabo utilizando una hoja de EXCEL, como se muestra en la figura 8-4.

	A	B	C	D	E	F
1						
2	**Relacion de Nodos Lugar-Transicion**					
3		**Nodos Lugar**	**Nodos Transicion**		**Numero Variables**	
4	**Cantidad**	3	1		6	
5						
6						
7	MATRIZ RELACION DE ARCOS		MATRIZ RELACION			**LUGARES**
8	TRANSICION/LUGARES					
9	**Tiempo de Transicion**	**Transiciones**	1	2	3	
10	P	1	%-1'(X,'|1'(J,W)	%-1'(E,W,P)|1'(E,W,P)	%-1'(Z)|1'(Z)	
11		2	*			
12						
13						

Figura 8-4: Hoja EXCEL para definición de estructura de RdPC

Hoja 4: Definición de Variables

Se utiliza una hoja para la definición de las variables que intervienen en el modelo. La implementación se lleva a cabo en una tabla con 8 columnas con la información necesaria para el algoritmo de evaluación de transiciones:

- Número de Variable. Es un número entero que sirve como identificador numérico de la variable.
- Nombre. Es una cadena de caracteres, y funciona como identificador alfanumérico de la variable.
- Valor Inicial. Se especifica siempre como -1 y es el valor inicial de la variable.
- Expresión. Es la expresión obtenida a partir del guarda, y en caso de que no exista guarda asociado a la variable, es la variable misma.
- Relación. Es un número entero el cual hace establece la relación de la variable con las variables relacionadas por la expresión de guarda a través de su identificador numérico. Cuando hay más de una variable relacionada, se utiliza el separador ' , '.
- Lugar de origen. Es un número entero el cual especifica el lugar al que pertenece la expresión de arco donde aparece la variable. Cuando es un arco de salida va precedido por un signo '-'.
- Pos_color. Es un número entero, especifica la posición de color en la expresión de arco a la que pertenece la variable.
- Operador. Es una expresión aritmética [=, <,>, >=, <=, <>] que se obtiene directamente de la expresión de guarda. Cuando la variable no se relaciona con ningún guarda, el valor es '='.

Ejemplo 8.3 Definición de Variables en un Modelo de RdPC

La definición de la estructura del modelo de la figura 8-1 se lleva a cabo utilizando una hoja de EXCEL, como se muestra en la figura 8-5.

	A	B	C	D	E	F	G	H
1		VARIABLES A UTILIZAR						
2	NUMERO	NOMBRE	VALOR INICIAL	EXPRESION	RELACION (INT)	LUGAR ORIGEN	POS_COLOR	OPERADOR
3	1	X	-1	E	4	1	1	=
4	2	J	-1	X+1	1	-1	1	=
5	3	W	-1	Z	6	2	2	=
6	4	E	-1	X	1	2	1	=
7	5	D	-1	D	5	2	3	=
8	6	Z	-1	W	3	3	1	=

Figura 8-5: Hoja de EXCEL para definición de variables en un modelo de RdPC

9 ANEXO C: MODELO DE PALETIZADO DE CAJAS

El modelo en RdPC para el paletizado de cajas cuenta con cuatro lugares y 28 transiciones que se utilizan para llevar a cabo las diferentes acciones para modelar el proceso de colocado de cajas en el espacio disponible en un palet de dos dimensiones.
La tabla 9-1 y la tabla 9-2 describen los colores y los lugares utilizados

Tabla 9-1: Especificación de Colores

COLOR	DEFINICIÓN	SIGNIFICADO
Idc	Entero	Identificador de la caja
Cr	Entero	0:orientación original 1:rotada 90° sobre eje Z
Ce	Entero	0:No asignada 1: en proceso 2: colocada en el Palet
Cx	Real	Coordenada X donde la caja está localizada
Cy	Real	Coordenada Y donde la caja está localizada
Cz	Real	Coordenada Z donde la caja está localizada
Lx	Real	Longitud de la caja en el eje coordenado X
Ly	Real	Longitud de la caja en el eje coordenado Y
Lz	Real	Longitud de la caja en el eje coordenado Z
Scx	Real	Coordenada X donde la superficie libre está localizada
Scy	Real	Coordenada Y donde la superficie libre está localizada
Slx	Real	Longitud X de la superficie libre
Sly	Real	Longitud Y de la superficie libre
Ge	Entero	0: Una caja puede ser colocada en el palet 1: Caja a ser asignada 2: En busca de superficie 3: Evaluando las nuevas superficies fragmentadas
Gz	Entero	Indica el piso del palet
Gsf	Real	Superficie disponible en el palet
Gncv	Entero	Número de cajas Virtuales
Pa	Producto idc*cx*cy*cz*lx*ly*lz*cr*ce	Información que describe las características físicas de cada caja
Ps	Producto scx*scy*slx*sly	Información que describe las características de las superficies libres
Pv	Producto idc*cx*cy*cz*lx*ly*lz*ce	Información que describe las características de las cajas virtuales
Pg	Producto ge*gz*gsf*gncv*gnc	Información global del modelo

Tabla 9-2: Descripción de los lugares del modelo de paletizado en RdPC

LUGAR	COLOR	SIGNIFICADO
P1	Pa	Los tokens de este lugar representan las cajas y el color Pa la información de cada una de las cajas.
P2	Ps	Los tokens almacenados por este lugar representan las superficies libres en el palet, y Ps la información de la superficie.
P3	Pv	Los tokens de este lugar representan las cajas virtuales generadas durante el proceso de colocado de cajas. El color
P4	Pg	El token de este lugar almacena la información global del modelo.

El modelo no tiene extensión temporal, y las transiciones se agrupan en 5 bloques dependiendo del tipo de evento modelado. A continuación se describen las agrupaciones del modelo y un ejemplo de cada uno de los grupos de transiciones.

Rotación de Cajas

Este bloque utiliza una transición que modela el evento de rotar una caja 90°. La figura 9-1 muestra la transición utilizada para la rotación.

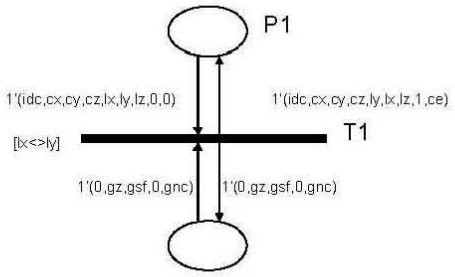


Figura 9-1: Rotación de una caja

Posicionamiento de cajas en Espacio Disponible 1

Este grupo de transiciones modela los eventos de posicionar una caja en un área libre del palet mayor o igual que las dimensiones de la caja, de manera que el espacio libre no utilizado se modela

fraccionándolo para que quede en disponibilidad para posicionamientos posteriores de mas cajas. La figura 9-2 muestra un ejemplo de una transición que modela esta actividad.

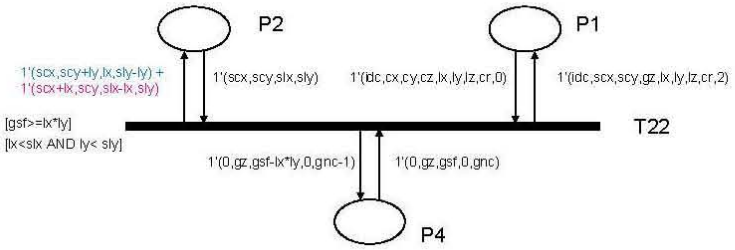


Figura 9-2: Posicionado de una caja con dimensiones menores a la superficie libre

Posicionamiento de cajas en Espacio Disponible 2

Este grupo de transiciones modela los eventos de colocar una caja en un área disponible en el palet cuando la caja excede las dimensiones del área disponible, en este caso se introduce el concepto de caja virtual[14]. La figura 9-3 presenta el ejemplo de una transición perteneciente a este grupo.

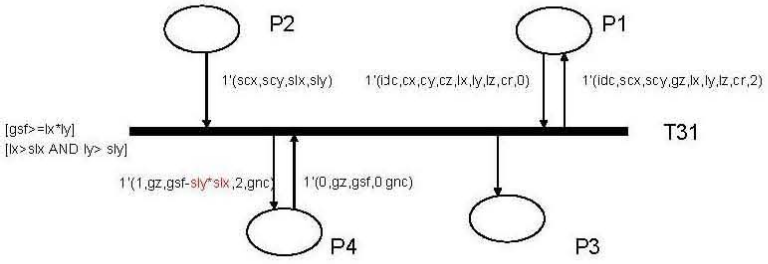


Figura 9-3: Posicionado de caja excediendo dimensiones de superficie libre

[14] Concepto introducido por Piera (2007)

Posicionamiento de cajas virtuales 1

Este grupo de transiciones modela el colocado de las cajas virtuales en los espacios disponibles cuando la superficie libre utilizada excede o iguala alguna dimensión de la caja virtual. Una actividad de colocado de una caja virtual se presenta en la figura 9-4.

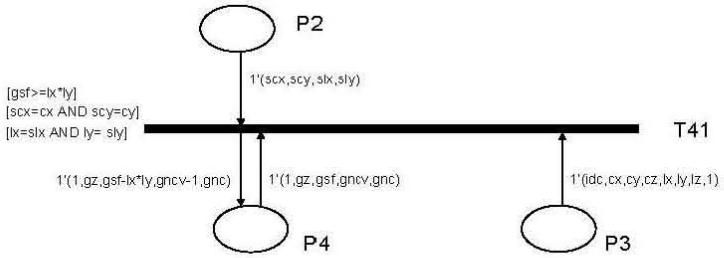


Figura 9-4: Posicionado de caja virtual en espacio disponible de la misma dimensión

Posicionamiento de cajas virtuales 2

Este grupo de transiciones modela el colocado de cajas virtuales en los espacios disponibles cuando la superficie libre es menor a la caja virtual. Un ejemplo de este grupo de transición es el presentado en la figura 9-5.

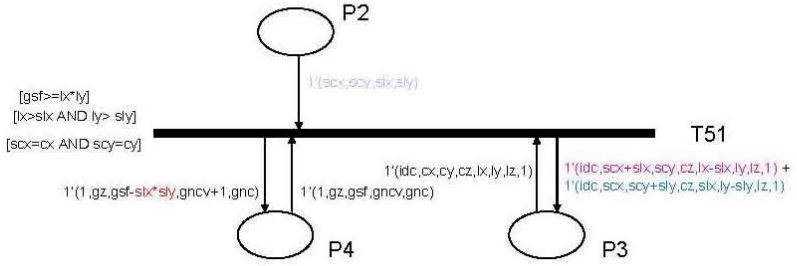


Figura 9-5: Posicionado de caja virtual en espacio disponible de menor dimensión que la caja

Liberación de cajas

Consta de la transición que posiciona la caja que previamente fue subdividida en cajas virtuales como se ilustra en la figura 9-6.

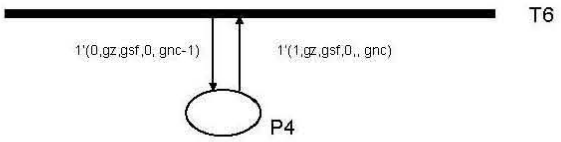


Figura 9-6: Liberación de la caja virtual

10 ANEXO D: MODELOS DE TALERES DE TRABAJO EN RDPC

La información de los modelos utilizados en los diferentes talleres de trabajo se presenta a continuación.

Taller de trabajo 3x3

Modelo en RdPC.

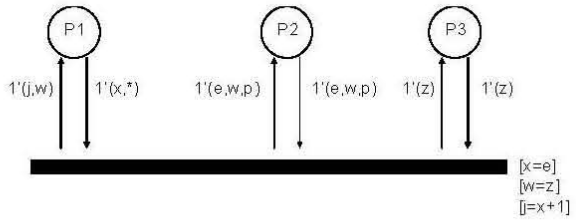


Figura 10-1: Modelo en RdPC del Taller de trabajo 3x3

Datos del Problema.

Tabla 10-1: Secuencia de Tareas para cada Trabajo

SECUENCIA DE TAREAS	TRABAJO 1				TRABAJO 2				TRABAJO 3			
	Opcion1		Opcion2		Opción 1		Opción 2		Opcion1		Opción 2	
	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo	Maq	Tiempo
1	1	7	2	8	1	6	2	5	1	8	3	5
2	2	4			2	4	3	2	2	2		
3	1	7	3	4	1	6			2	6	3	4
4					1	3	2	2	1	4	2	2
5									1	2	3	3

Definición de Colores.

Tabla 10-2: Especificación de Colores Taller de trabajo 3x3

COLOR	DEFINICIÓN	DESCRIPCIÓN
j	Entero 11..33	Identificador de trabajo y tarea
w	Entero 1..3	Máquina necesaria para la siguiente tarea
x	Entero 11..33	Trabajo y tarea en progreso
e	Entero 11..33	Identificador de trabajo y tarea
p	Entero	Tiempo de cada tarea
z	Entero 1..3	Máquinas disponibles

Descripción de Lugares.

Tabla 10-3: Descripción de los lugares del modelo

LUGAR	COLOR	DESCRIPCIÓN
P1	Producto j*w	El valor del color j representa el trabajo en turno y la tarea que se está realizando en un instante de tiempo. El color w representa la máquina que es necesaria para llevar a cabo la siguiente tarea del trabajo correspondiente
P2	Producto e*w*p	Representa la información de la secuencia lógica que deben seguir las diferentes actividades de los trabajos correspondientes. Los colores e,w y p representan respectivamente el trabajo y tarea en que se encuentra la máquina siguiente al terminar la tarea del color e, y el tiempo de operación p que consumirá al llevar a cabo la tarea correspondiente en la máquina w.
P3	z	La información asociada con la máquina disponible.

Estado Inicial.

Tabla 10-4: Información del Marcado Inicial del T-T 3x3

MODELO 3X3	LUGAR		
	P1	P2	P3
Marcado Inicial	1'(10,0)+1'(20,0)+1'(30,0)	1'(10,1,7)+1'(10,2,8)+1'(11,2,4)+1'(12,1,7) +1'(12,3,4)+1'(20,1,6)+1'(20,2,5)+1'(21,2,4) +1'(21,3,2)+1'(22,1,6)+1'(23,1,3)+1'(23,2,2) +1'(30,1,8)+1'(30,3,5)+1'(31,2,2)+1'(32,2,6) +1'(32,3,4)+1'(33,1,4)+1'(33,2,2)+1'(34,1,2) +1'(34,3,3)	1'(1)+1'(2)+1'(3)

Secuencia óptima del Problema.

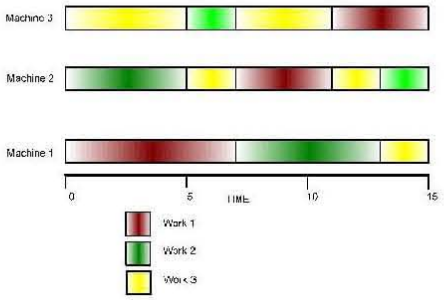


Figura 10-2: Diagrama de Gantt de la secuencia óptima para el T-T 3x3

Datos de Desempeño.

Tabla 10-5: Datos de Desempeño Taller de trabajo 3x3

NODOS DISTINTOS EXPLORADOS	693
Nodos Repetidos Analizados	680
Estados Padre Analizados	2,032
Nodos Actualizados en la 1ª evaluación	59
Nodos Descartados en la 1ª evaluación	485
Nodos Irresolubles	1,488
Búsquedas Binarias Realizadas	10,204

Taller de trabajo 5x5.

Modelo en RdPC.

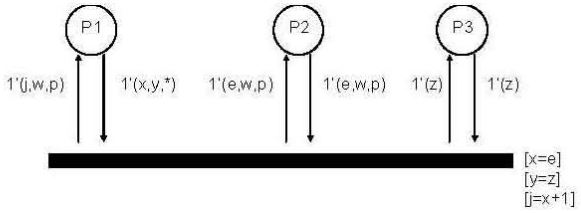


Figura 10-3: Modelo en RdPC del Taller de trabajo 5x5

Datos del Problema.

Tabla 10-6: Secuencia de Operaciones para el Taller de trabajo 5x5

SEQ. TAREAS	TRABAJO 1		TRABAJO 2		TRABAJO 3		TRABAJO 4		TRABAJO 5	
	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo
1	4	85	1	64	3	31	5	44	2	66
2	1	7	4	14	2	69	5	18	3	68
3	4	1	1	74	2	70	5	90	3	60
4	2	45	4	76	5	13	3	98	1	54
5	1	80	4	15	2	45	5	91	3	10

Definición de Colores.

Tabla 10-7: Especificación de colores Taller de trabajo 5x5

COLOR	DEFINICIÓN	DESCRIPCIÓN
j	Entero 11..55	Identificador de trabajo y tarea
w	Entero 1..5	Máquina necesaria para la siguiente tarea
x	Entero 11..55	Trabajo y tarea en progreso
e	Entero 11..55	Identificador de trabajo y tarea
p	Entero	Tiempo de cada tarea
z	Entero 1..5	Máquinas disponibles
y	Entero 1..5	Máquina en uso

Descripción de Lugares.

Tabla 10-8: Descripción de los lugares del modelo

LUGAR	COLOR	DESCRIPCIÓN
P1	Producto j*w*p	El valor del color j representa el trabajo en turno y la tarea que se está realizando en un instante de tiempo. El color w representa la máquina que es necesaria para llevar a cabo la siguiente tarea del trabajo correspondiente, y P el tiempo que consume la tarea.
P2	Producto e*w*p	Representa la información de la secuencia lógica que deben seguir las diferentes actividades de los trabajos correspondientes. Los colores e, w y p representan respectivamente el trabajo y tarea en que se encuentra la máquina siguiente al terminar la tarea del color e, y el tiempo de operación p que consumirá al llevar a cabo la tarea correspondiente en la máquina w.
P3	z	La información asociada con la máquina disponible.

Estado Inicial.

Tabla 10-9: Información del Marcado Inicial del T-T 5x5

MODELO 5X5	LUGAR		
	P1	P2	P3
Marcado Inicial	1'(11,4,85)+1'(21,1,7)+1'(31,4,1) +1'(41,2,45)+1'(51,1,80)	1'(11,1,64)+1'(12,3,31)+1'(13,5,44)+1'(14,2,66) +1'(21,4,14)+1'(22,2,69)+1'(23,5,18)+1'(24,3,68) +1'(31,1,74)+1'(32,2,70)+1'(33,5,90)+1'(34,3,60) +1'(41,4,76)+1'(42,5,13)+1'(43,3,98)+1'(44,1,54) +1'(51,4,15)+1'(52,2,45)+1'(53,5,91)+1'(54,3,10)	1'(1)+1'(2)+1'(3) +1'(4)+1'(5)

Secuencia óptima del Problema.

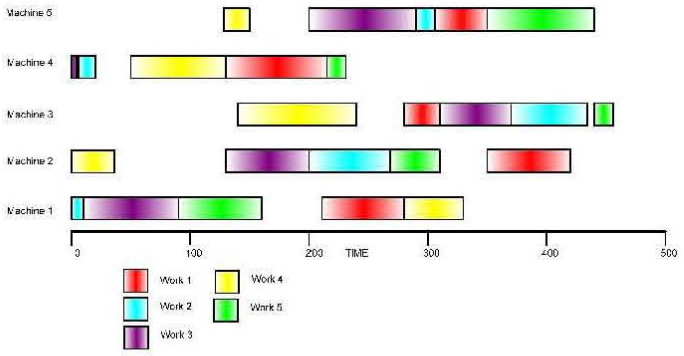


Figura 10-4: Diagrama de Gantt de la secuencia Óptima del T-T 5x5

Datos de Desempeño.

Tabla 10-10: Datos de Desempeño Taller de trabajo 5x5

NODOS DISTINTOS EXPLORADOS	7,776
Nodos repetidos analizados	7,750
Estados Padre Analizados	24,625
Nodos Actualizados en la 1ª evaluación	1,893
Nodos Descartados en la 1ª evaluación	5,829
Nodos Irresolubles	16,903
Búsquedas Binarias Realizadas	121,825

Taller de trabajo 6x6.

Modelo en RdPC.

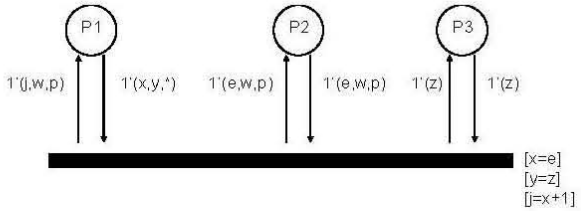


Figura 10-5: Modelo en RdPC del Taller de trabajo 6x6

Datos del Problema.

Tabla 10-11: Secuencia de Operaciones para el Taller de trabajo 6x6

SEQ. TAREAS	TRABAJO 1		TRABAJO 2		TRABAJO 3		TRABAJO 4		TRABAJO 5		TRABAJO 6	
	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo	Maq.	Tiempo
1	2	1	1	8	2	5	1	5	2	9	1	3
2	0	3	2	5	3	4	0	5	1	3	3	3
3	1	6	4	10	5	8	2	5	4	5	5	9
4	3	7	5	10	0	9	3	3	5	4	0	10
5	5	3	0	10	1	1	4	8	0	3	4	4
6	4	6	3	4	4	7	5	9	3	1	2	1

Definición de Colores.

Tabla 10-12: Especificación de colores Taller de trabajo 6x6

COLOR	DEFINICIÓN	DESCRIPCIÓN
j	Entero 11..66	Identificador de trabajo y tarea
w	Entero 1..6	Máquina necesaria para la siguiente tarea
x	Entero 11..66	Trabajo y tarea en progreso
e	Entero 11..66	Identificador de trabajo y tarea
p	Entero	Tiempo de cada tarea
z	Entero 1..6	Máquinas disponibles
y	Entero 1..6	Maquina en uso

Descripción de Lugares.

Tabla 10-13: Descripción de los lugares Taller de trabajo 6x6

LUGAR	COLOR	DESCRIPCIÓN
P1	Producto j*w*p	El valor del color j representa el trabajo en turno y la tarea que se está realizando en un instante de tiempo. El color w representa la máquina que es necesaria para llevar a cabo la siguiente tarea del trabajo correspondiente y el color p es el tiempo de duración de la operación realizada.
P2	Producto e*w*p	Representa la información de la secuencia lógica que deben seguir las diferentes actividades de los trabajos correspondientes. Los colores e, w y p representan respectivamente el trabajo y tarea en que se encuentra la máquina siguiente al terminar la tarea del color e, y el tiempo de operación p que consumirá al llevar a cabo la tarea correspondiente en la máquina w.
P3	z	La información asociada con la máquina disponible.

Estado Inicial.

Tabla 10-14: Información del Marcado Inicial del T-T 6x6

MODELO 6X6	LUGAR		
	P1	P2	P3
Marcado Inicial	1'(11,2,1)+1'(21,1,8)+1'(31,2,5) +1'(41,1,5)+1'(51,2,9)+1'(61,1,3)	1'(11,0,3)+1'(12,1,6)+1'(13,3,7)+1'(14,5,3) +1'(15,4,6)+1'(21,2,5)+1'(22,4,10)+1'(23,5,10) +1'(24,0,10)+1'(25,3,4)+1'(31,3,4)+1'(32,5,8) +1'(33,0,9)+1'(34,1,1)+1'(35,4,7)+1'(41,0,5) +1'(42,2,5)+1'(43,3,3)+1'(44,4,8)+1'(45,5,9) +1'(51,1,3)+1'(52,4,5)+1'(53,5,4)+1'(54,0,3) +1'(55,3,1)+1'(61,3,3)+1'(62,5,9)+1'(63,0,10) +1'(64,4,4)+1'(65,2,1)	1'(0)+1'(1)+1'(2)+1(3) +1'(4)+1'(5)

Secuencia óptima del Problema.

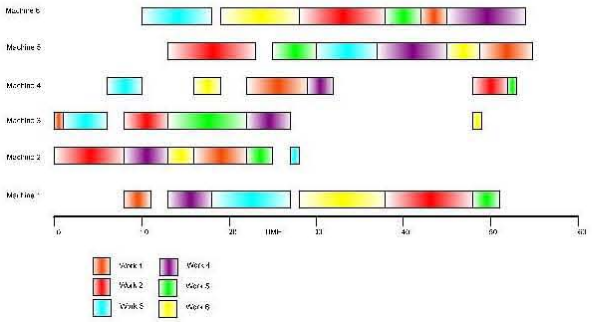


Figura 10-6: Una solución óptima del Taller de trabajo 6x6

Datos de Desempeño.

Tabla 10-15: Datos de Desempeño Taller de trabajo 6x6

NODOS DISTINTOS EXPLORADOS	117,650
Nodos repetidos analizados	117,612
Estados Padre Analizados	487,408
Nodos Actualizados en la 1ª evaluación	26,555
Nodos Descartados en la 1ª evaluación	128,387
Nodos Irresolubles	332,475
Búsquedas Binarias Realizadas	2,302,579

6 BIBLIOGRAFÍA

[1] Adballah, I.B., Elmaraghy H., Elmekkawy T., 1998, "An efficient search algorithm for deadlock free scheduling in FMS using Petri nets", in IEEE Intl Conf. Robotics Automation, vol. 2, pp. 1793–1798.

[2] Andersson A., Icking C., Klein R., Ottman T., (1990), "Binary Search Trees of Almost Optimal Height", in *Journal of Acta Informatica*, V.28 (2), p.p. 165-178

[3] Andersson, A. , (1993), "Balanced Search Trees Made Simple", in *Proceedings of the Workshop on Algorithms and Data Structures*, Springer –Verlag, p.p. 60 -71

[4] ARENA Homepage < http://www.arenasimulation.com/>.

[5] Banks, J., Carson, J., (1996), "Discrete-Event System Simulation", Prentice Hall, New Jersey.

[6] Beasley, J., OR-Library <http://people.brunel.ac.uk/~mastjjb/jeb/info.html>.

[7] Berliner, H.J., 1979, "The B* Tree Search Algorithm: A best-first proof procedure." In Artificial Intelligence, vol.12, pp. 23-40.

[8] Berliner, H.J., McConnell, C., 1996, "B* probability based search", in Artificial Intelligence, vol.86, pp. 97-156.

[9] Bierwirth, C., Kopfer, H., Mattfeld, D.C., Rixen, I., 1995, "Genetic Algorithm based Scheduling in a dynamic manufacturing environment", in Proc. of the IEEE International Conference on Evolutionary Computation, Pert.

[10] Bisgaard, Haag, T., Rudmose T, 1994 "Optimizing a Coloured Petri Net Simulator.", *University of Aarhus.*

[11] Bowden, F.D.J., 2000, "A Brief Survey and Synthesis of the Roles of Time in Petri Nets", in Mathematical and Computer Modelling, vol.31(10-12), pp. 55-68.

[12] Borret J. E., Tsang E., (2001), "A Context for Constraint Satisfaction Problem Formulation Selection", in *Constraints*, vol. 6 (4), Springer Netherlands

[13] Brady T., McGarvey, 1998, "Heuristic Optimization using Computer Simulation: A Study of Staffing Levels in a Pharmaceutical Manufacturing Laboratory", in Proc. of the 1998 WSC, Washington D.C.

[14] Camurri, A.P., Franchi P, Gandolfo F., (1991),"A timed Colored Petri Nets Approach to Process Scheduling", in *proceedings of the CompuEuro '91 5th Annual European Computer Conference*, Bologna, Italy p.p. 304-309

[15] Chan, F.T.S., Chaube, A., Mohan, V., Arora, V., Tiwari M.K., 2010, "Operation allocation in automated manufacturing system using GA-based approach with multifidelity models", in Robotics and Computer Integrated Manufacturing, vol. 26(5), pp.526-534.

[16] Chang, T.S., 1996, "A Framework of Using Linear Programming for Manufacturing Scheduling", in Proc. of 35th Conference on Decision and Control, pp. 3843-3846.

[17] Chiola G., 1993, "Simulation framework for Timed and Stochastic Petri Nets", *Journal of Computer Simulation*, vol1, No.2, August.

[18] Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S., 1997, "A Symbolic Reachability Graph for Coloured Petri Nets", in Journal of Theoretical Computer Science, vol.176 (1-2), pp. 39-65.

[19] Christensen, S., Kristensen, L.M., Mailund, T., 2001, "Condensed State Spaces for Timed Petri Nets", in Lecture Notes in Computer Science, vol. 2075, Springer.

[20] Christensen, S., Jensen, K., Mailund, T., Kristensen, L.M., 2001, "State Space Methods for Timed Coloured Petri Nets", in Proc. of 2nd International Colloquium on Petri Net Technologies for Modelling Communication Based Systems, pp. 33-42, Berlin.

[21] Christensen, S., Kristensen, L.M., Mailund, T., (2001),"A Sweep-Line Method for State Space Exploration"in *TACAS,* Springer-Verglag, Berlin-Heidelberg

[22] Christensen, S, Mailund T., (2002), "A Generalized Sweep-Line Method for Safety Properties", in FME, Springer- Verlag, Berlin- Heidelberg

[23] Christensen S., Kristensen L. M., Mailund T., (2001), "Condensed State Spaces for Timed Petri Nets", in *Proceedings of the 22nd Internacional Conference on Application an theory of Petri Nets*, p.p. 101-120, Springer-Verlag , London UK

[24] Clarke, E.M., Enders, R., Filkorn, T., Jha, S., 1996, "Exploiting Symmetries in temporal logic model checking", in Formal Methods in System Design, Vol. 9(1-2), Kluwer Academic Publishers, pp. 77-104.

[25] Coolahan, J.E., Roussopoulos, N., 1983, "Timing Requirements for Time Driven Systems Using Augmented Petri Nets", in IEEE Transactions on Software Engineering, Vol. SE-9 (5), pp. 603-616.

[26] Coffman. E.G.; Galambos, G.; Martello, S.; Vigo, D., (1998), "Bin Packing Approximation Algorithms: Combinatorial Analysis", in *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers

[27] CPNTools Homepage. <http://www.daimi.au.dk/~cpntools/>.

[28] Dashora, Y., Kumar, S., Tiwari, M.K., Newmann S.T., 2007, "Deadlock-free scheduling of an automated manufacturing system using an enhanced coloured time resource Petri-net model-based evolutionary endosymbiotic learning automata approach" in International Journal of Flexible Manufacturing Systems, vol.19 (4), pp. 486-515.

[29] Dauzére-Peres, S., Lasserre, J.B., (1994), "An Integrated Approach in Production Planning and Scheduling", *in Lecture Notes and Mathematical Systems,* Springer-Verlag, Berlin.

[30] David R., Alla H., (1992), "Petri Nets & Grafcet: Tools for modeling discrete events systems", Prentice Hall International, United Kingdom.

[31] Dechter, R., Pearl, J., 1985, "Generalized best-first search strategies and the optimality of A*", in Journal of the ACM, vol.32, pp. 505-536.

[32] Dengiz, B., Alabas, C., 2000, "Simulation Optimization using Tabu Search", in Proc. of the 2000 WSC, Orlando, USA.

[33] Duttheillet, C., Haddad S., (1993), "Conflict Sets in Colored Petri Nets", in *Proceedings of the 5th Intenational workshop*, Tolouse, France, p.p. 76-85

[34] Emerson, E.A., Sistla, A.P., 1996, "Symmetry and Model Checking", in Formal Methods in Systems Design, Vol. 9(1-2), Kluwer Academic Publishers, pp. 105-131.

[35] Evangelista, S., Westergaard, M., Kristensen, L.M., 2009, "The ComBack Method Revisited: Caching Strategies and Extension with Delayed Duplicate Detection", in LNCS, vol. 5800, Springer, Heidelberg 2009.

[36] Evans, J.R; Minieka, E., (1992), "Optimization Algorithms for Networks and Graphs", *Marcel Dekker*, New York.

[37] Fang, H, Ross P., Corne D., (1993), "A Promising Genetic Algorithm Approach to Job-Shop Scheduling, Rescheduling, and Open-Shop Scheduling Problems" in *Proceedings of the Fifth International Conference on Genetic Algorithms,* San Mateo: Morgan Kaufmann, p.p. 375-382.

[38] Finkel, A., 1993,"The minimal Coverability Graph for Petri Nets", in Lecture Notes in Computer Science, Springer, vol. 674, pp. 210-243.

[39] Gaeta, R. 1996, "Efficient Discrete-event Simulation of Coloured Petri Nets". In Transactions on software Engineering, vol. 22(9), pp. 629-639.

[40] Gambin A, Piera M. A, Riera D., (1999),"A Petri Nets Based Object Oriented Tool for the Scheduling of Stochastic Flexible Manufacturing System" IEEE.

[41] Garey, M.; Johnson, D., (1979), "Computers and Intractability: A Guide to the theory of NP-Completeness", *Freeman and Company*, San Francisco.

[42] Ghezzi C., Mandrioli D., Morasca S., Pezze M., (1989), "A General Way to Put Time in Petri Nets", in *ACM SIGSOFT Software Engineering Notes, Vol 4 (3), p.p. 60-67*

[43] Gonnet, G.H., (1984), "Handbook of Algorithms and Data Structures", *Addison- Wesley*, London

[44] Gradišar D., Mušič G, 2007, "Production-process modelling based on production-management data: a Petri-net approach", in International Journal of Computer Integrated Manufacturing, vol. 20(8), pp. 794-810.

[45] Guasch, A.; Piera, M.A.; Casanovas J.; Figueras J., (2002), "Modelado y Simulación: Aplicación a procesos logísticos de fabricación y servicios", *Edicions UPC*, Barcelona.

[46] Harrell, C.R., Ghosh, B.K, Bowden, R.O., 2000, "Simulation Using Promodel", McGRAW-HILL, 3rd edition, New York.

[47] Hart, P.E., Nilsson, N.J., Raphael, B., 1968, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", in IEEE Transactions of Systems Science and Cibernetics, vol.ssc-4(2), pp. 100-107.

[48] Hillion, H.P., Proth, J.P., 1989, "Performance Evaluation of Job-Shop systems using Timed Event Graphs", in IEEE Transactions on Automatic Control, vol.34, pp.3-9.

[49] Holzmann, G.J., (1998), "An analysis of Bitstate Hashing", in *Formal Methods in System Design*, V13 (3), p.p.301-314, November.

[50] Huber, P., Jensen, A.M., Jepsen, L.O., Jensen, K., 1986, "Reachability Trees for High-Level Petri Nets", in Theoretical Computer Science, vol. 45, pp. 261-292.

[51] Jard C., Jeron, T., (1991),"Bounded –memory algorithms for Verification On-the Fly", in *Proceedings of CAV 1991*, vol575 of *Lecture Notes in Computer Science*. Springer-Verlag.

[52] Jensen, K., 1996,"Condensed State Spaces for Symmetrical Coloured Petri Nets", in Formal Methods in System Design, vol.9 (1-2), Springer Netherlands.

[53] Jensen, K., (1997), "Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use". Vol. 1 Springer-Verlag. Berlin.

[54] Jensen, K., (1997), "Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use", Vol 2, Springer-Verlag. Berlin

[55] Jensen, K., 1997. "Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use". vol. 1 Springer-Verlag, Berlin.

[56] Kalasky, D.R, (1996), "Manufacturing Systems: Modeling and Simulation", in *Systems Modeling and Computer Simulation*, cap. 7, Marcel Dekker, New York

[57] Karp, R.M., Miller, R.E., 1969, "Parallel program schemata", in Journal of Computer and Systems Sciences, vol.3(2), pp. 147-195.

[58] Korf, R.E., Chickering, D.M., 1993, "Best-First Minimax Search: First Results", in Proc. of the AAAI Fall Symposium on Games: Planning and Learning, Raleigh, pp. 39-47.

[59] Korf, R.E., (2004), "Optimal Rectangle Packing: New Results", *In Proceedings of the 14th International Conference on Automated Planning and Scheduling*.

[60] Kristensen, L.M., 2000,"State Space Methods for Coloured Petri Nets", PhD Dissertation, Dept. Computer Science, University of Aarhus, Denmark.

[61] Kristensen, L.M., Mailund, T., 2002, "A Generalized Sweep-Line Method for Safety Properties", in FME, Springer-Verlag, pp. 549-567, Berlin- Heidelberg.

[62] Kumar, R.R., Singh A., Tiwari, M.K., 2004, "A fuzzy based algorithm to solve the machine-loading problem of a FMS and its neuro fuzzy petri net model", in The International Journal of Advanced Manufacturing Technology, vol. 23(5), pp. 318-341.

[63] Kumar V., (1992), "Algorithms for Constraint Satisfaction Problems: A Survey". *In AI magazine 13(1)*

[64] Lalas, C., Mourtzis, D., Papakostas, N., Chryssolouris, G., 2006, "A simulation-based hybrid backwards scheduling framework for manufacturing systems", in International Journal of Computer Integrated Manufacturing, vol. 19(8), pp. 762-774.

[65] Law, A.M., Kelton, W.D., 2000, "Simulation Modeling and Analysis", 3rd edition, Mc Graw-Hill.

[66] Law, A.M., McComas, M.G., 2002, "Simulation Based Optimization", in Proc. of the 2002 Winter Simulation Conference, pp. 46-49.

[67] Lee, D.Y., DiCesare, F., 1994, "Scheduling Flexible Manufacturing Systems Using Petri Nets and Heuristic Search", in Transactions of Robotics and Automation, vol. 10(2), pp.123-132.

[68] Lel, V.T.; Creighton, D.; Nahavandi, S., (2005), "A heuristic algorithm for carton to pallet loading problem," Industrial Informatics, 2005. *INDIN '05. 2005 3rd IEEE International Conference* on , vol., no., pp. 593-598, 10-12 Aug. 2005

[69] Liu, C.M., Wu, F.C., 1993, "Using Petri Nets to solve FMS problems", in Int. Journal of Computer Integrated Manufacturing, vol.6(3), pp. 175-185.

[70] Liu, S. Q., Ong, H.L., Ng, K.M., 2005, "Metaheuristics for minimizing the makespan of the dynamic shop scheduling problem", in Advances in Engineering Software, vol.36(3), pp.199-205.

[71] Mailund, T., 2003, "Sweeping the State Space—a sweep-line state space exploration method.", PhD Thesis, University of Aarhus.

[72] McAllester, David, (1992), "Constraint Satisfaction Search", In Artificial Intelligence Lecture Notes.

[73] Mehlom, K., (1984), "Data Structures and Algorithms2, Graph Algorithms and NP-Completeness", Springer-Verlag, Berlin.

[74] Mejia, G., 2003, "Timed Petri Net Modeling and Optimization with Heuristic Search for Flexible Manufacturing Workstations", in Proc. of the IEEE ETFA'03, pp. 211-217.

[75] Mejía G.; Montoya, C., (2007), "A Petri Net based algorithm for minimizing total tardiness in flexible manufacturing systems", in *Annals of Operations Research*, Springer-Verlag

[76] Merkuryev, Y., Merkuryeva, G., Piera, M.A., Guasch, T., 2009,"Simulation-Based Case Studies In Logistics: Education and Applied Research", Springer.

[77] Merlin, P.M., Farber, D.J., 1976, "Recoverability of Communication Protocols – Implications of a Theoretical Study", in IEEE Transactions on Communications, Vol. 24(9), pp. 1036-1043.

[78] Moore, K.E., Gupta, S.M., 1996, "Petri Net Models of Flexible and Automated Manufacturing Systems: A Survey", in International Journal of Production Research, Vol. 34(11), pp. 3001-3035.

[79] Mortensen, K.H., 2001, "Efficient Data Structures and Algorithms for a coloured Petri Nets simulator", in CPN workshop Proceedings, Denmark, pp. 57-74.

[80] Mujica, M.A., Piera, M.A., 2007, "Improvements for a Coloured Petri Net Simulator", in Proc. of the 6th Eurosim congress on Modelling and Simulation, Ljubljana, Slovenia, pp. 237.

[81] Mujica, M.A., Piera M.A., Narciso, M., 2008, "Optimizing Time Performance in Reachability Tree-Based Simulation", in Proc. of the International Mediterranean Modelling Multiconference (I3M), Campora Sant Giovanni, pp.800-805, Italy.

[82] Mujica, M.A., Piera M.A., 2009, "A Two Step Algorithm to Improve Systems Optimization based on the State Space Exploration for Timed Coloured Petri Net Models", in Proc. of the TiStoWorkshop, Paris, France, pp.47-61.

[83] Mujica, M.A., Piera M.A., 2009, "Performance Optimization of a CNC Machine Through Exploration of Timed State Space", in Proc. of the I3M2009 Multiconference, pp. 20-25, Tenerife, Spain.

[84] Mujica, M.A., Piera, M.A., Narciso M., 2010, "Revisiting state space exploration of timed coloured petri net models to optimize manufacturing system's performance", in Simulation Modelling Practice and Theory, vol.8, no 9, pp. 1225-1241.

[85] Music, G., 2009, "Petri Net Based Scheduling Approach Combining Dispatching Rules and Local Search", in Proc. of the I3M2009 Multiconference, Tenerife, Spain.

[86] Narciso, M., Piera, M.A., Guasch A., 2009, "A Methodology for Solving Logistic Optimization Problems through Simulation", in Simulation: Transactions of the Society for Modeling and Simulation International, vol. 86 (5-6), pp. 369-389.

[87] Narciso, M., Piera, M.A., Guasch A., Casanovas, J., 2003, "A Methodology to Improve simulation optimization approach for process scheduling through Petri net formalism" in Proc. of Summer Computer Simulation Conference'03, Canada.

[88] Narciso M., (2007), "Metodología para la Resolución de Problemas de Optimización Mediante la Exploración de Estados Generado a Partir de Modelos de Redes de Petri Coloreadas", Tesis de Doctorado, Universitat Autónoma de Barcelona.

[89] Narciso, M., Piera, M.A., y Figueras J., (2005), "Optimización de Sistemas Logísticos Mediante Simulación: Una Metodología Basada en Redes de Petri Coloreadas", *Revista Iberoamericana de Automática e Informática Industrial*, Vol. 2(4), p.p. 54-65IAII, Valencia, España.

[90] Nilsson, N. J., (1980),"Principles of Artificial Intelligence",*Tioga Publishing*, Palo Alto, CA

[91] OptQuest < http://www.arenasimulation.com/Products_OptQuest.aspx >.

[92] Pardalos, P.M., (1998), "Bin Packing Approximation Algorithms: Combinatorial Analysis", in *Handbook of Combinatorial Optimization*, Kluwer Academic Publishers

[93] Paulussen, R.R., Somers, L.J., 1995, "Simulation of distributed simulation with timed colored Petri nets", in Proc. of Eurosim'95, pp. 547-552.

[94] Petri, C.A., 1962, "Kommunikation mit Automaten", Rheinisch-Westfälischen Institutes für Instrumentelle Matematik an der Universität Bonn, Nr.2.

[95] Petri, C.A., 1967,"Grundsätzliches zur Beschreibung diskreter Prozesse", in Colloquium über Automatentheorie, Birkhäuser Verlag, pp. 121-140.

[96] Piera M. A, Mujica M., Guasch, 2007, "An Efficient CN Modeling Approach to Tackle the Pallet Packing Problem", in Proc. of the 6th EUROSIM Congress on Modeling and Simulation, p.p. 344, September 9-13, Ljubljana, Slovenia.

[97] Piera, M.A., Music, G., 2009, "Coloured Petri Nets: timed state space exploration examples and some simulation shortages", in Proc. of the MathMod 09, Viena, Austria.

[98] Piera, M.A., Narciso, M., Guasch, A., Riera, D., 2004, "Optimization of Logistic and Manufacturing Systems through Simulation: A Colored Petri Net-Based Methodology" in Simulation: Transactions of the Society for Modeling and Simulation International, vol. 80 (3), pp. 121-129.

[99] Pinedo, M.L., 2005, "Planning and Scheduling in Manufacturing and Services", Springer, New York.

[100] Pisinger D., Sigurd M., (2005), "The Two-dimensional bin packing problem with variable bin sizes and costs", in *Discrete Optimization*, Volume 2, Issue 2.

[101] Pooch, W. U.; Wall, J.A., (1993), "Discrete Event Simulation", CRC Press, United States

[102] PROMODEL Homepage <http://www.promodel.com/products/promodel/>.

[103] Proth, J.M., Xie, X., 1997, "Petri nets : a tool for design and management of manufacturing systems", John Wiley & Sons, Chichester.

[104] Rajabinasab, A., Mansour, S., 2010, "Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach", in The international Journal of Advanced Manufacturing Technology, DOI 10.1007/s00170-010-2986-7.

[105] Renna, P., 2010, "Job Shop Scheduling by pheromone approach in a dynamic environment", International Journal of Computer Integrated Manufacturing, vol.23(5), pp. 412-424.

[106] Reyes, A., Yu, H., Kelleher, G., Lloyd, S., 2002, "Integrating Petri Nets and hybrid heuristic search for the scheduling of FMS", in Computers in Industry, vol.47(1), pp. 123-138.

[107] Sanders, M. J., 2000, "Efficient Computation of Enabled Transition Bindings in High – Level Petri Nets", in Proc. IEEE Int. Conf. on Systems, Man. and Cybernetics, vol.4, pp. 3153-3158.Tsang; E., (1993), "Foundations of Constraint Satisfaction", *Academic Press Limited*, London

[108] Seo, M., Iida, H., Uiterwijk, J., 2001, "The PN*-search algorithm: Application to tsume-shogi.", in Artificial Intelligence, vol.29, pp. 253-277.

[109] Shao, X., Li, X., Gao, L., Zhang, C., 2009, "Integration of process planning and scheduling: a modified genetic algorithm-based approach", in Computers and Operations Research, vol. 36(6), pp. 2082-2096.

[110] Silva, M., Valette, R., 1989, "Petri Nets and Flexible Manufacturing", in Lecture Notes in Computer Science, Advances in Petri Nets, pp.274-417, Springer-Verlag.

[111] SimRunner < http://www.promodel.com/products/simrunner/>.

[112] Störrle, H., 1998. "An Evaluation of High-End Tools for Petri Nets.", L.M.U.,Munich.

[113] Taha, H.A.,2007, "Operations Research: An Introduction", 8th edition, Pearson/Prentice Hall, New Jersey.

[114] Tarjan R.E, (1983), "Data Structures and Network Algorithms", Society for Industrial and Applied Mathematics, Philadelphia Pennsylvania, 1983

[115] Tarjan R.E., Polya G., Woods D.R., (1983), "Notes on Introductory Combinatorics", Birkhäuser, Boston, 1983

[116] Tchako, J.F.N., Beldjilali, B., Trentesaux, D., Tahon, C., 1994, "Modelling with coloured Petri nets and simulation of a dynamic and distributed management system for a manufacturing cell", in Computer Integrated Manufacturing, vol. 7(6), pp. 323-339.

[117] Tsang, E., 1993, "Foundations of Constraint Satisfaction", Academic Press.

[118] Tsai, C.F., Lin, F.C., 2003, "A New Hybrid Heuristic Technique for Solving Job-Shop Scheduling Problem", in Proc.of the IEEE Int. Workshop on Intel. Data Acq. and Adv. Comp. Systems, pp. 53-58.

[119] Valmari, A., 1991, "Stubborn Sets for Reduced State Space Generation", in Proc. Of the 10th International Conference on Applications and Theory of Petri Nets: Advances in Petri Nets 1990.

[120] Valmari, A.,1998, "The State Explosion Problem", in Lecture Notes in Computer Science, vol. 1491, Springer-Verlag, London, pp. 429-528.

[121] Van Der Aalst, W.M.,P., 1992, "Timed Coloured Petri Nets and their applications to logistics", PhD Thesis, Eindhoven University of Technology, Eindhoven.

[122] Van Der Aalst, W.M.,P., 1993, "Interval Timed Coloured Petri Nets and their Analysis", in Lecture Notes in Computer Science, vol.691, pp. 453-472, Springer.

[123] Van Der Aalst, W.M.,P., Odijk, M.A., 1995, "Analysis of Railway Stations by means of interval timed coloured Petri nets", in Real-Timed Systems, vol.9, pp. 241-263.

[124] Van der Aalst,W.M.P, 1995, "Petri Net based Scheduling". In Computing Science Reports, Eindhoven University of Technology.

[125] Wallace M. G., (2000),"Search in AI - Escaping from the CSP Straightjacket", *in Proceedings of 14th European Conference on Artificial Intelligence (ECAI 2000)*, Springer-Verlag.

[126] Wang J., Deng Y., Xu G., (2000), "Reachability Analysis of Real-Time Systems Using Time Petri Nets", in Transactions on systems, man and cybernetics, V.30 (5), p.p 725- 736

[127] Wells, L., 2002, "Performance Analysis Using Coloured Petri Nets", in Proc. of the 10th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'02), pp.0217.

[128] Westergaard, M., Kristensen, L.M., Brodal, G.S., Arge, L., 2007, "The ComBack method-extending hash compaction with backtracking.", in Lecture Notes in Computer Science Proc. of Application and Theory of Petri Nets, vol.4546, pp. 445-464, Springer.

[129] Witness Homepage <http://www.lanner.com/en/witness.cfm>.

[130] Wolf, K., 2007, "Generating Petri Net State Spaces", in Proc. of the 28th int. conf. on applications and theory of Petri nets and other models of concurrency, pp.29-42, Springer.

[131] Yogeswaran, M., Ponnambalam, S.G., Tiwari, M.K.,2009, "An efficient hybrid evolutionary heuristic using genetic algorithm and simulated annealing algorithm to solve machine loading problem in FMS", in International Journal of Production Research, vol.47(19), pp. 5421-5448.

[132] Yu, H., Reyes, A., Cang, S., Lloyd, S., 2003, "Combined Petri net modelling and AI based heuristic hybrid search for flexible manufacturing systems – Part 1. Petri net modelling and heuristic search", in Computers and Industrial Engineering, vol. 44(4), pp. 527-543.

[133] Zhang, H., Gu, M., 2009, "Modeling job shop scheduling with batches and setup times by timed Petri nets", in Mathematical and Computer Modelling, vol.49 (1-2), pp., 286-294.

Mujica, M., Piera, M.A., 2009, "Optimizing Systems Performance Exploring the Timed State Space of Coloured Petri Nets", in *Proc of the OR-EUROCONFERENCE*, July,Bonn, Germany.

Mujica, M., Piera, M.A., 2009, "A Two Step Algorithm to Improve Systems Optimization based on the State Space Exploration for Timed Coloured Petri Nets", in Proc. of the TiSto 2009, June, Paris, France.

Mujica, M., Piera, M.A., 2009,"Performance Optimization of a CNC Machine Through Exploration of Timed State Space", in Proc. of the I3M 2009 Simulation Multiconference, September, Canary Islands, Spain.

Mujica, M., Piera, M.A., 2010,"Hybrid Search Algorithm to Optimize Scheduling Problems for TCPN Models", in Proc. of the SCSC Simulation Conference, July, Ottawa, Canada.

Publicaciones en co-autoría

M. A. Piera, L. Vilalta, M. Mujica, M. Narciso, T. Guasch, 2006, "An Efficient Modelling Approach to Combine Search Methods with Evaluation Methods", in Proceedings of the European Modelling Symposium, Londres (UK), 11-12 September 2006.

M. A. Piera, T. Guasch , L. Vilalta, M. Mujica, 2005, "Several Hints for A Master Search using Coloured Petri Net Simulators", in Proceedings of the SCSC, Calgary, Canada, 30 Julio – 2 Agosto 2006.

Piera M. A, Mujica M., Guasch, 2007, "An Efficient CN Modeling Approach to Tackle the Pallet Packing Problem", in *Proceedings of the 6th EUROSIM Congress on Modeling and Simulation,* p.p. 344, September 9-13, Ljubljana, Slovenia.