Redes de Petri, algoritmo para la construcción de árboles de mínima cobertura

Ing. Luis Orlando Ventre, Dr. Ing. Orlando Micolini, Ing. Gabriel Valenzuela, and Ing. Mauricio Ludemann

Laboratorio de Arquitectura de Computadoras, FCEFyN-Universidad Nacional de Córdoba Av. Velez Sarfield 1601, CP-5000, Córdoba, Argentina {luis.ventre, orlando.micolini, mauri.ludemann, gabriel.valenzuela}@unc.edu.ar

Resumen En el ámbito del diseño de sistemas embebidos críticos, donde predominan las reacciones a eventos y la lógica basada en acciones, las Redes de Petri emergen como una herramienta de modelado esencial. Mediante la implementación de una ecuación de estado extendida, es posible capturar la lógica de estos sistemas. El modelo lógico resultante se ejecuta a través de un monitor que integra la lógica (Red de Petri), la política (gestión de conflictos) y las acciones, formando un sistema heterogéneo. Esta integración mejora la capacidad del sistema para ser verificado mediante formalismos matemáticos basados en el modelo empleado. En este contexto, la verificación de propiedades importantes de las RdP, como la cobertura, la acotación, la alcanzabilidad y la vivacidad, se realiza utilizando árboles de cobertura, con un tiempo de verificación proporcional al tamaño del árbol. Por lo tanto, es crucial diseñar algoritmos que calculen el árbol de cobertura de manera eficiente. En este trabajo, presentamos una modificación del algoritmo de árbol de cobertura mínima, originalmente propuesto por Karp y Miller. Para hacer este cálculo, se memorizan las aceleraciones activándolas como transiciones ordinarias. Esta estrategia no solo asegura un límite predecible para el uso de memoria adicional —específicamente, un máximo de doble exponencial—, sino que también mejora la eficiencia operativa. La implementación de un prototipo de este algoritmo muestra su competitividad, presentando un uso reducido de memoria y tiempos de ejecución comparables a las herramientas más rápidas disponibles.

Keywords: Redes de Petri , Árboles de Cobertura , Sistemas Embebidos.

1. Introducción

En el diseño y desarrollo de sistemas concurrentes, reactivos, embebidos y distribuidos, desacoplar la lógica, la política (solución de conflictos) y las acciones facilita la solución y verificación del sistema. Esto da lugar a un **sistema modular**, **simple**, **mantenible**, **formal** y **flexible**. Además, se logra la verificación formal de la lógica en todas las fases del desarrollo. Para modelar la lógica

del sistema, se emplean RdP y se convierten en código ejecutable mediante la ecuación de estado generalizada [1] y un monitor [2]. Esta solución conserva las propiedades verificadas mediante el uso de formalismos matemáticos [3].

En el estudio de modelos utilizando RdP, la eficacia de los métodos se basa en el **grafo de alcanzabilidad** [4] y está directamente vinculada al tamaño de la red [5]. Los algoritmos que generan este árbol en las RdP son manejables desde el punto de vista computacional y resultan fundamentales para el análisis y supervisión de sistemas concurrentes [6,7]. Los **árboles de alcanzabilidad** representan el conjunto de todas las marcas accesibles a partir de una marca inicial, donde cada nodo del árbol corresponde a una marca generada mediante transiciones habilitadas. La complejidad computacional asociada al cálculo de un árbol de alcanzabilidad se ve principalmente determinada por el tamaño y la naturaleza de la red [8,9]. Por consiguiente, el modelado de sistemas extensos puede resultar costoso en términos de recursos computacionales, especialmente en RdP continuas, híbridas y discretas con distintos tipos de arcos y/o configuraciones de transiciones [10]. A pesar de esto, la importancia de estos árboles en la evaluación y verificación de sistemas concurrentes justifica los esfuerzos empleados.

En el presente documento, se propone un nuevo algoritmo eficiente en términos de tiempo y espacio para los casos en que el grafo es infinito. Además, se destacarán varios problemas que pueden surgir al calcular los árboles de alcanzabilidad de una RdP. Estos incluyen la explosión del espacio de estados, la complejidad computacional, redes no acotadas, dificultades en la visualización, precisión y exhaustividad, así como transiciones no deterministas. A pesar de la presencia de estos obstáculos, el cálculo de los árboles de alcanzabilidad continúa siendo esencial para comprender el comportamiento de las RdP que modelan sistemas concurrentes.

2. Conceptos preliminares

Las **RdP** son modelos ampliamente utilizados para representar sistemas concurrentes, que pueden tener estados tanto finitos como infinitos.

2.1. Redes de Petri ordinarias

Una RdP ordinaria (**PN**) se define como una quíntupla $PN = (P, T, I^-, I^+, M_0)$, donde:

- $P = \{P_1, P_2, \ldots, P_n\}$: es un conjunto finito y no vacío de plazas.
- $T = \{T_1, T_2, \ldots, T_m\}$: es un conjunto finito y no vacío de transiciones.
- Las plazas y las transiciones forman un conjunto bipartito, cumpliéndose que $P \cap T = \emptyset$ y $P \cup T \neq \emptyset$.
- I^- y I^+: son las matrices de incidencia que representan las relaciones de entrada y salida entre plazas y transiciones, respectivamente. En una red con m plazas y n transiciones, I^- e I^+ son matrices de enteros de dimensión $m \times n$.

- $M_0 = [M_0(P_1), M_0(P_2), \ldots, M_0(P_n)]$: es el marcado inicial de la red, indicando la cantidad de tokens en cada plaza.

La **matriz de incidencia** I se define como $I = I^+ - I^-$, facilitando la representación matemática de las interacciones en la red.

Así, la definición simplificada de una PN puede expresarse mediante una 4-tupla $PN = (P, T, I, M_0)$.

2.2. Definición del árbol de alcanzabilidad

Dada una RdP (N, M_0), comenzando desde una marca inicial M_0, es posible generar nuevas marcas en la medida en que las transiciones estén habilitadas. A partir de cada nueva marca, podemos generar aún más marcas. Este proceso resulta en un árbol de marcas, donde los nodos representan las marcas generadas desde M_0 (la raíz) y cada arco representa el disparo de una transición que transforma una marca en otra [4].

Para abordar problemas relevantes como la cobertura repetida, la verificación de modelos en *Lógica Temporal Lineal* (LTL), la acotación y la regularidad de las trazas, se introduce un conjunto de marcas ω, etiquetado por un conjunto $C \subseteq N_\omega^P$ (donde N_ω es el conjunto de números naturales ampliado con un límite superior ω, y P es el conjunto de plazas). Este conjunto C representa todas las posibles representaciones finitas del cierre infinito hacia abajo del conjunto de alcanzabilidad, permitiendo resolver múltiples instancias de cobertura de manera eficiente y evitando llamadas repetidas a algoritmos costosos.

2.3. Importancia del árbol de alcanzabilidad

La alcanzabilidad en una RdP es crucial para analizar las propiedades dinámicas de un sistema [9,11]. En términos generales, permite determinar si el sistema modelado puede alcanzar un estado específico. Un marcado M_i es alcanzable desde un marcado inicial M_0 si existe una secuencia finita de disparos ρ, tal que $M_0 \xrightarrow{\rho} M_i$. Los marcados alcanzables por la red pueden representarse como nodos en un grafo o árbol, donde los arcos están etiquetados con los disparos necesarios para alcanzar cada marcado. El algoritmo para determinar el árbol de alcanzabilidad de una RdP se explicará con más detalle posteriormente.

El grafo de alcanzabilidad A se define como el conjunto mínimo que cumple con las siguientes expresiones: dado un $M_0 \in A$; para todo M_i, si $M_0 \xrightarrow{\rho} M_i$ entonces $M_i \in A$. Esta condición implica que, para cualquier marcado alcanzado M_i, si a partir de este se puede alcanzar otro marcado M_j, entonces M_j forma parte del grafo.

La alcanzabilidad (T) es la propiedad más estudiada en las RdP, dado que a partir de esta es posible determinar si una red es: **acotada**, **finita**, **libre**, si sus trancisiones son **vivas** y la **alcanzabilidad**.

2.4. Obtención del árbol de alcanzabilidad

Para obtener los árboles de alcanzabilidad en una (RdP), existen varios algoritmos y enfoques. Destacamos algunos de estos: **Generación manual** [12], **herramientas de software** [13] y/o **métodos de búsqueda** [14].

Debemos hacer notar que la elección del algoritmo o método depende del contexto específico y de las características de la RdP que se esté analizando. Cada enfoque tiene sus ventajas y limitaciones, y es importante seleccionar el más adecuado para las necesidades particulares.

3. Árbol de cobertura

3.1. Cobertura de una red de Petri.

El algoritmo para obtener el árbol de alcanzabilidad de una RdP no converge si la red no es acotada. Esto se debe a que habrá plazas cuya cantidad de tokens seguirá aumentando indefinidamente, afectando la expansión de los nodos en el árbol de alcanzabilidad. Para abordar estos casos, existe otro tipo de análisis denominado árbol de cobertura [12]. En el árbol de cobertura, se representan gráficamente todas las marcas posibles de la red. Sin embargo, se colapsan en un único marcado genérico aquellas plazas cuya cantidad de tokens crecerá infinitamente a medida que la red evolucione. Para denotar este marcado, se utiliza el símbolo ω, ya mencionado en el apartado (2.2), que representa un valor entero arbitrariamente alto y no afecta la evolución de la red.

Ejemplo: Analizando la red ilustrada en la Fig. 1-a, compuesta por tres plazas y cuatro transiciones, podemos observar que la red es no acotada. Por ejemplo, en la plaza P2, cuando se disparan T3 o T4, el marcado de P2 aumenta (destacado por ω), mientras que cuando se disparan T1 o T2, su marcado permanece intacto. Esto implica que el marcado de la plaza P2 incrementa a medida que la red evoluciona, pero nunca disminuye.

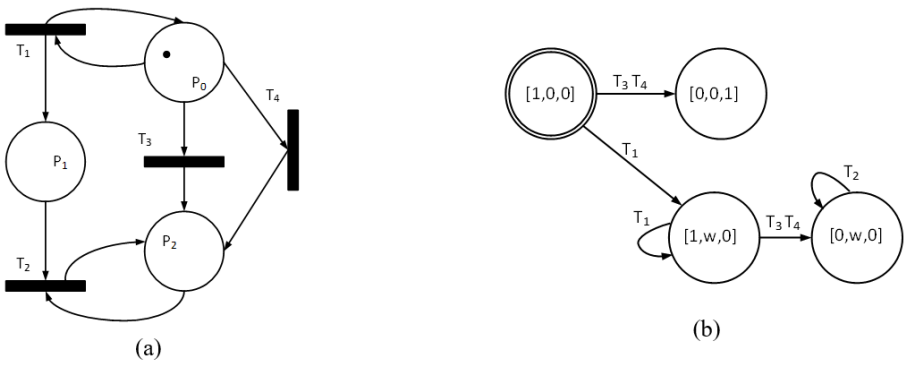


Figura 1: (a) muestra una RdP y en (b) el árbol de cobertura, notar que ω transforma una la máquina de estado infinita en finita.

El grafo de cobertura asociado a la red no acotada de la Fig. 1-a se muestra en la Fig. 1-b. En este grafo, el estado con un doble círculo representa el marcado inicial, y los arcos representan los disparos de las transiciones. Es importante aclarar que, cuando la RdP es acotada, el árbol de cobertura y el árbol de alcanzabilidad son exactamente iguales. El problema de complejidad para obtener la cobertura es EXPSPACE-completo [8], lo que significa que es computacionalmente costoso. En contraste, la accesibilidad no tiene una complejidad similar.

3.2. Importancia del árbol de cobertura en el ánálisis de sistemas concurrentes

El árbol de cobertura en el análisis de una RdP es una herramienta crucial para comprender y evaluar el comportamiento de sistemas concurrentes [4]. Su importancia se manifiesta en varios aspectos clave:

- **Visualización de estados alcanzables:** El árbol de cobertura muestra los estados alcanzables desde un estado inicial en la RdP, lo que permite visualizar cómo evoluciona el sistema a medida que se disparan transiciones. Identificar los estados alcanzables es fundamental para comprender las posibles secuencias de eventos y detectar problemas como bloqueos o estados inalcanzables.
- **Análisis de comportamiento:** Este árbol proporciona información sobre las transiciones que se pueden disparar desde cada estado, ayudando a analizar el comportamiento del sistema en términos de secuencias de eventos. Por ejemplo, se puede verificar si ciertos estados son inevitables o si existen estados de deadlock.
- **Identificación de estados críticos:** Algunos estados en el árbol de cobertura son especialmente importantes y pueden representar situaciones específicas en el sistema, como la finalización de una tarea o la presencia de conflictos. Estos estados críticos son útiles para detectar propiedades deseadas o indeseadas.
- **Optimización y reducción de modelos:** El árbol de cobertura también se utiliza para reducir y optimizar el modelo de la RdP. Al eliminar estados redundantes o inalcanzables, se simplifica el análisis y se obtiene un modelo más manejable, lo cual es especialmente valioso en sistemas grandes y complejos.
- **Verificación de propiedades:** Utilizando el árbol de cobertura, es posible verificar propiedades específicas del sistema, como la seguridad o la completitud en alcanzar todos los estados finales deseados. Esta verificación es esencial para asegurar la corrección y la confiabilidad del sistema.

En resumen, el árbol de cobertura es una herramienta poderosa para explorar, analizar y verificar sistemas concurrentes modelados mediante RdP.

3.3. Los algoritmos para obtener el árbol de cobertura de una red de Petri

Para obtener el árbol de cobertura en una RdP, se han desarrollado varios algoritmos: **Árbol de Karp-Miller** [15], **Algoritmo de Finkel** [9], **CoverProc** [16], **Algoritmo de Poda-Monótona** [17] y **Algoritmo MinCov** [17]

La elección del algoritmo depende del contexto específico y de las características de la RdP en particular.

3.4. Complejidad computacional y desidibilidad para obtener el árbol de cobertura

Calcular el árbol de cobertura en una RdP puede presentar diferentes niveles de complejidad computacional, dependiendo del algoritmo utilizado y de las características específicas de la red. A continuación, se describen algunos aspectos relevantes:

- **Explosión del espacio de estados:** La complejidad computacional está fuertemente influenciada por la explosión del espacio de estados.
- **Algoritmos utilizados:** Los algoritmos específicos para construir el árbol de cobertura pueden variar significativamente en su complejidad.
- **Características de la red:** La topología de la red, la cantidad de plazas y transiciones, y la presencia de transiciones no deterministas son factores críticos que afectan la complejidad.
- **Complejidad Espacial vs. Temporal:** La complejidad del cálculo puede medirse tanto en términos de tiempo (cuánto tiempo se requiere para calcular el árbol) como de espacio (cuánta memoria se necesita para almacenar el árbol).
- **Redes temporizadas:** Las RdP temporizadas, que incluyen información sobre tiempos asociados a las transiciones, pueden incrementar la complejidad del cálculo debido a la necesidad de gestionar estos tiempos adicionalmente.
- **Análisis cuantitativo de prestaciones:** La necesidad de realizar análisis cuantitativos de las prestaciones de la red también puede aumentar la complejidad computacional del proceso.

En resumen, no existe una única respuesta sobre la complejidad computacional para obtener el árbol de cobertura, ya que esta depende de múltiples factores inherentes a la estructura y funcionamiento de cada RdP.

4. Estado del arte en algoritmos de cobertura y alcanzabilidad en redes de Petri

La noción del conjunto de cobertura mínima emerge, como ya hemos mencionado, como una aproximación valiosa al conjunto de alcanzabilidad, definido a través de marcas ω, permitiendo abordar una amplia gama de problemas de decisión relacionados con las RdP.

En este trabajo, presentamos una modificación al algoritmo de árbol de cobertura mínima [12,14]. Nos enfocamos en el cálculo del conjunto mínimo de cobertura, tradicionalmente obtenido mediante los árboles de Karp y Miller. Estos árboles incorporan aceleraciones de ciclos para asegurar la completitud del análisis, aunque no sin incurrir en redundancias. Investigaciones previas han propuesto mejoras significativas mediante un algoritmo capaz de realizar podas entre las ramas del árbol de Karp y Miller, demostrando su corrección a pesar de la complejidad introducida por este mecanismo de poda.

Trabajos anteriores también mostraron la corrección de mejoras en los algoritmos, basado en una invariante que simplifica y abrevia considerablemente el argumento previo [15,9]. Esta mejora no solo es válida para este algoritmo, sino que también facilita su aplicación a variantes más generales de las RdP, siendo las bases para futuras generalizaciones.

Se introdujo el algoritmo Monotone-Pruning (MP), diseñado para optimizar el cálculo del conjunto de cobertura mínima en RdP. MP mejora la estrategia del algoritmo de Karp y Miller mediante una poda inter-ramas más efectiva, superando las limitaciones del Árbol de Cobertura Mínima (MCT). Su eficacia ha sido teórica y empíricamente confirmada, mostrando mejoras significativas frente al método tradicional [16].

Además, Finkel y Khmeniltsky [18] han desarrollado un algoritmo innovador, denominado MinCov, que contribuye significativamente a las estrategias de verificación para RdP. MinCov evita la reconstrucción de partes del conjunto de cobertura mínima, optimizando el proceso de poda. Su efectividad se maximiza mediante un enfoque de exploración en profundidad o priorizando nodos con mayor número de tokens, usando una técnica de fusión de historiales para mantener la integridad del conjunto sin eliminar vértices.

De manera notable, el algoritmo MinCov opta por generar un grafo en lugar del tradicional árbol de cobertura, marcando una desviación fundamental de las metodologías convencionales. Esta estrategia se ve potenciada por las mejoras introducidas por Piipponen y Valmari, quienes, a través del diseño de estructuras de datos especializadas y la implementación de una heurística dirigida para la exploración, logran reducir significativamente el tamaño del grafo resultante [19]. Este conjunto de innovaciones subraya el potencial del algoritmo MinCov para establecer un nuevo estándar en la eficiencia de la verificación de RdP.

5. MinCovVec: Avances en vectorización para la cobertura mínima en Redes de Petri

El presente trabajo introduce el algoritmo MinCovVec (Minimal Coverability Vectorized), una innovación que capitaliza las fortalezas observadas en los algoritmos Monotone-Pruning (MP) y MinCov. Este algoritmo se distingue por su fundamentación en la ecuación de estado extendida y está implementado en C++20, aprovechando las capacidades de este lenguaje para optimizar el desempeño.

MinCovVec se beneficia del uso de disparos en paralelo, lo que permite una ejecución más eficiente mediante políticas avanzadas de gestión de procesos. A diferencia de los algoritmos anteriores, MinCovVec comienza con la creación de un nodo raíz y procede a generar y procesar candidatos de manera eficiente. Este proceso se optimiza evitando redundancias a través de un sistema de filtrado basado en hashes, lo que mejora significativamente el rendimiento al reducir los cálculos innecesarios y facilitar la gestión de estados en sistemas concurrentes y paralelos.

Este enfoque innovador no solo mejora la eficacia del análisis de alcanzabilidad en Redes de Petri, sino que también establece un nuevo estándar en la verificación de sistemas complejos gracias a su capacidad para manejar grandes volúmenes de datos y su alta escalabilidad.

5.1. Pseudocódigo del algoritmo y diagrama de secuencia

1: **Inicialización:**
2: Generar el nodo raíz con la marca inicial de la RdP.
3: **Selección del Modo de Recorrido:**
4: Elegir entre BSF (Búsqueda por Amplitud), DSF (Búsqueda por Profundidad),
5: MOST_TOKENS (Mayor número de tokens) para el recorrido del grafo de cobertura.
6: **Obtención de Transiciones:**
7: Identificar las transiciones habilitadas en la RdP.
8: **Generación de Nuevas Marcas:**
9: **for** cada nodo **do**
10: Evaluar la posibilidad de generar una nueva marca a partir de las transiciones disponibles.
11: **if** la nueva marca no existe en el filtro (verSet) **then**
12: Agregarla como un nuevo nodo.
13: **end if**
14: **end for**
15: **Cálculo de Aceleraciones:**
16: **for** cada elemento en el vector de aceleraciones **do**
17: **if** es posible disparar transiciones ω desde la marca actual **then**
18: Si la nueva marca resultante es mayor, proceder a la aceleración del nodo.
19: **end if**
20: **end for**
21: **Verificación y Eliminación en verSet:**
22: **if** la marca acelerada ya existe en verSet **then**
23: Considerar la eliminación de dicho nodo.
24: **end if**
25: **Poda de Nodos:**
26: Buscar y eliminar aquellos nodos cuyas marcas sean menores a la marca actual.
27: **Actualización de verSet y Aceleración de la Marca:**
28: Añadir el nodo actual a verSet y acelerar la marca del nodo según las marcas de sus ancestros.
29: **if** es posible **then**
30: Determinar el nodo padre acelerado y proceder con la eliminación de nodos con marcas menores a la del nodo actual.
31: **end if**=0

Algoritmo 1: Algoritmo de Cobertura en Redes de Petri

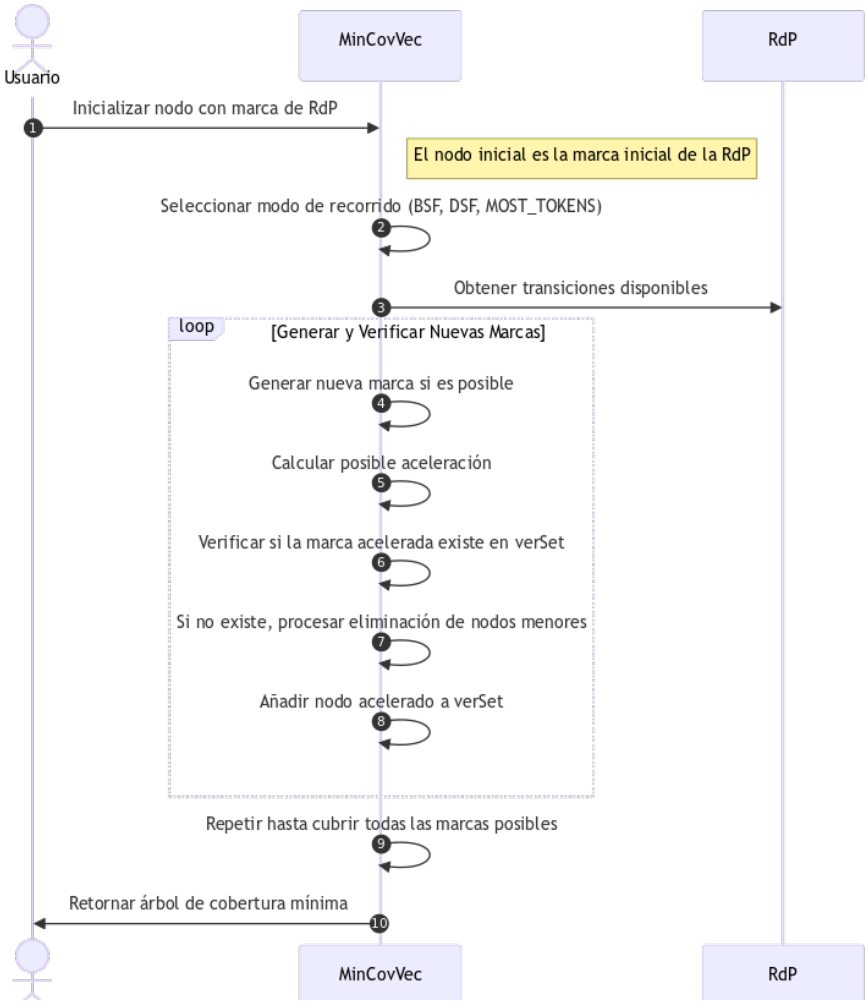


Figura 2: Diagrama de Secuencia

5.2. Resultados obtenidos

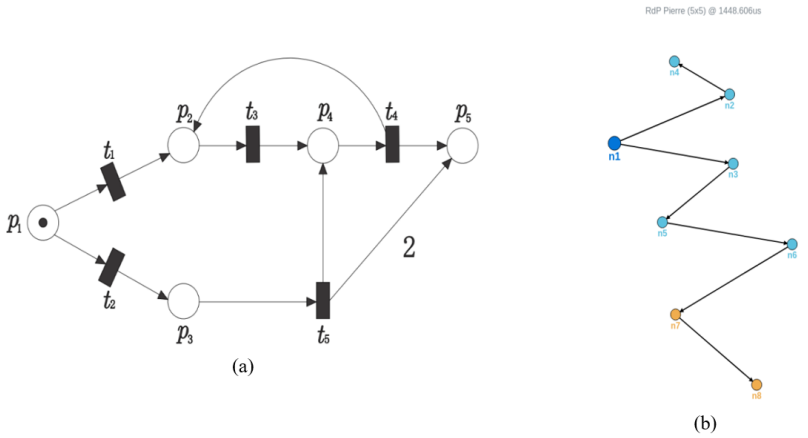
MinCovVec ha demostrado ser superior a métodos tradicionales en términos de rendimiento, logrando reducciones significativas en tiempo de cómputo y uso de recursos, como se evidencia en los resultados de las pruebas presentadas en la Tabla 1. Estos avances son particularmente notables en el análisis de alcanzabilidad y cobertura de sistemas complejos y concurrentes, estableciendo nuevos estándares en la verificación de RdP.

Tabla 1: Resultado comparativo de pruebas de algoritmo de cobertura mínima

RdP	**MP**	**MinCov**	**MinCovVec**	**Petrinator**
lamport	24@10000	14@1000	15@3324	14@17000
peterson	35@2000	20@2000	32@3528	32@34700
kanban	114@10000	160@32000	160@174065	160@NaN
fms	809@280000	37@8000	28@4846	NaN@NaN
multipool	2004@4900000	224@70000	226@232165	NaN@NaN
pncsacover	1604@1600000	92@44000	35@18418	NaN@NaN
basicME	5@10000	3@1000	4@133	NaN@NaN
csm	102@30000	19@3000	23@2344	NaN@NaN
mesh2x2	6241@18100000	268@140000	51@15546	NaN@NaN

La descripción del resultado sigue el formato #Nodos@TiempoEmpleado donde el tiempo se mide en microsegundos (us). La última columna detalla la cantidad de nodos generados por el algoritmo Petrinator, cuando aplicable.

Diversas pruebas realizadas en distintas configuraciones de RdP corroboran la eficacia de MinCovVec. Por ejemplo, en la Fig. 3 se muestra un caso emblemático para comparar el desempeño de los distintos algoritmos.



(a) (b)

Figura 3: (a) RdP de prueba. (b) nodos del árbol de cobertura de la RdP de prueba.

El algoritmo MinCovVec destaca por su capacidad para adaptarse y optimizar eficazmente la exploración del espacio de estado en diversas RdP, evidenciando su versatilidad y potencia.

6. Conclusiones

En este artículo, se ha presentado el algoritmo MCV, lo que constituye una mejora sustancial con respecto a los algoritmos MP y MinCov. Donde MCV implementa una estrategia de poda que equilibra de manera eficaz la precisión y la eficiencia, garantizando la fiabilidad de los resultados obtenidos. Se ha logrado optimizar significativamente la búsqueda de estados dentro de las RdP, resultando en una notable mejora en la velocidad de procesamiento.

El enfoque simplificado de MCV con respecto a MP, no solo facilita su adopción y adaptabilidad, sino que también mantiene la flexibilidad en la selección de estrategias de exploración de redes. Esto ha permitido el uso de métodos variados como la búsqueda en profundidad, en amplitud, o exploraciones aleatorias. Los resultados empíricos presentados en la sección de resultados muestran que nuestro algoritmo mejora a otros métodos establecidos, como MP, Karp & Miller (K&M), y CoverProc.

Como mejora para este algoritmo se está desarrollando un modelo avanzado, en el laboratorio de arquitectura de computadoras de la FCEFyN de la UNC, centrado en la segmentación de RdP y la aplicación de ecuaciones de estado extendidas a diversas redes.

Referencias

1. O. Micolini, M. Cebollada, M. Eschoyez, L. O. Ventre, and M. Schild, "Ecuación de estado generalizada para redes de Petri no autónomas y con distintos tipos de arcos," in *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016)*, 2016.
2. M. F. Peter A. Buhr, "Monitor Classification," *ACM Computing Surveys*, vol. 27, no. 1, pp. 63-107, 1995.
3. D. I. O. M. Ing. Luis Orlando Ventre, Ing. Mauricio Ludemann, Agustín Carranza, David D'Andrea, Enzo Candotti, "Caso de estudio: metodología para el diseño y desarrollo de sistemas embebidos distribuidos," 2024.
4. M. Diaz, *Petri Nets Fundamental Models, Verification and Applications*. NJ, USA: John Wiley & Sons, Inc, 2009.
5. P. Darondeau, S. Demri, R. Meyer, and C. Morvan, "Petri net reachability graphs: Decidability status of first order properties," *Logical Methods in Computer Science*, vol. 8, 2012.
6. M. V. Iordache, "Methods for the supervisory control of concurrent systems based on Petri net abstractions," 2003.
7. A. I. Rabinovich, L. O. Ventre, and O. Micolini, "Baboon, framework conducido por red de Petri para sistemas reactivos dirigidos por eventos," in *XXV Congreso Argentino de Ciencias de la Computación (CACIC) (Universidad Nacional de Río Cuarto, Córdoba, 14 al 18 de octubre de 2019)*, 2019.
8. W. Czerwiński, S. Lasota, R. Lazić, J. Leroux, and F. Mazowiecki, "The reachability problem for Petri nets is not elementary," *Journal of the ACM (JACM)*, vol. 68, no. 1, pp. 1-28, 2020.
9. A. FINKEL, "The Minimal Coverability Graph for Petri Nets," presented at the Advances in Petri Nets 1993, 1993.

10. R. David and H. Alla, *Discrete, continuous, and hybrid Petri nets*. Springer Science & Business Media, 2010.
11. F. Lu, Q. Zeng, M. Zhou, Y. Bao, and H. Duan, "Complex reachability trees and their application to deadlock detection for unbounded Petri nets," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 6, pp. 1164-1174, 2017.
12. T. Murata, "Petri Nets: Properties, Analysis and Applications," *Proceedings of the IEEE*, vol. 77, no. 4, pp. 541-580, 1989.
13. "Welcome to the Petri Nets World." Accessed: 05-04-2024.
14. P.-A. Reynier and F. Servais, "Minimal coverability set for Petri nets: Karp and Miller algorithm with pruning," in *International Conference on Application and Theory of Petri Nets and Concurrency*, 2011: Springer, pp. 69-88.
15. R. M. a. M. KARP, R.E., "Parallel Program Schemata," *Journal of Computer and System Sciences* vol. 3, pp. 147-195, 1969.
16. G. Geeraerts, J.-F. Raskin, and L. Van Begin, "On the efficient computation of the minimal coverability set for Petri nets," in *International Symposium on Automated Technology for Verification and Analysis*, 2007: Springer, pp. 98-113.
17. S. Antón Fernández, "Algoritmos para el conjunto de mínima cobertura de las Redes de Petri," 2023.
18. Finkel, A., Haddad, S., & Khmeniltsky, I. (2019, November). "A Tool for the Coverability Problems in Petri Nets." In MSR 2019-12ème Colloque sur la Modélisation des Systèmes Réactifs, Nov 2019, Angers, France.
19. Piipponen, A., & Valmari, A. (2016). "Constructing minimal coverability sets. Fundamenta Informaticae", 143(3-4), 393-414.