Universidad Nacional de Córdoba

FACULTAD DE CIENCIAS EXACTAS FÍSICAS Y NATURALES

CARRERA INGENIERÍA EN COMPUTACIÓN

SISTEMA REALIMENTADO PARA CORRECCIÓN Y DETERMINACIÓN DE POLÍTICA EN SISTEMAS CONTROLADOS POR REDES DE PETRI

Proyecto Fin de Carrera, Ingeniería en Computación

Autores:
Casas, Nicolás
40574806, nicolas.casas@mi.unc.edu.ar, 3584305868
Ciarrapico, Nicolás Valentin
39880567, nicolas.ciarrapico@mi.unc.edu.ar, 2994840005

Director: Ing. Luis Ventre
Co-director: Dr. Ing. Orlando Micolini

14 de septiembre de 2023

Índice

1. Introducción **5**
1.1. Motivación . . . 5
1.2. Objetivos . . . 5
1.3. Requerimientos . . . 5
1.3.1. Lista de requerimientos . . . 5
1.4. Modelo de Desarrollo . . . 6
1.5. Gestión de riesgos . . . 6
1.5.1. Identificación de riesgos . . . 7
1.5.2. Análisis de riesgos . . . 8
1.5.3. Planeación de riesgos . . . 9

2. Marco Teórico **10**
2.1. Redes de Petri . . . 10
2.1.1. Red de Petri Generalizada . . . 10
2.1.2. Red de Petri Marcada . . . 11
2.1.3. Tipos de Arcos . . . 11
2.1.4. Matrices . . . 11
2.1.5. Sensibilizado de una transición . . . 12
2.1.6. Disparo de una transición . . . 12
2.1.7. Ecuación de estado . . . 12
2.1.8. Extensión de la ecuación de estado . . . 13
2.1.9. Propiedades de las Redes de Petri . . . 13
2.1.10. Invariantes . . . 15
2.2. Reinforcement Learning . . . 15
2.2.1. Componentes del RL . . . 15
2.2.2. Tipos de algoritmos de RL . . . 16
2.3. Redes de Bayes . . . 18
2.3.1. La estadística de Bayes . . . 18
2.3.2. Probabilidad: definición axiomática . . . 19
2.3.3. Teorema de Bayes . . . 19
2.3.4. Fórmula de Bayes . . . 19
2.3.5. Redes Bayesianas . . . 20

3. Desarrollo **22**
3.1. Iteraciones . . . 22
3.1.1. Planificación de las iteraciones . . . 22
3.2. Condiciones . . . 24
3.3. Iteración 1 . . . 25
3.3.1. Introducción . . . 25
3.3.2. Objetivos . . . 25
3.3.3. Desarrollo . . . 25
3.3.4. Conclusiones . . . 25
3.4. Iteración 2 . . . 26
3.4.1. Introducción . . . 26
3.4.2. Objetivos . . . 26
3.4.3. Desarrollo . . . 26
3.4.4. Conclusiones . . . 27
3.5. Iteración 3 . . . 28
3.5.1. Introducción . . . 28
3.5.2. Objetivos . . . 28
3.5.3. Desarrollo . . . 28

3.5.4. Conclusiones . . . 29
3.6. Iteración 4 . . . 30
3.6.1. Introducción . . . 30
3.6.2. Objetivos . . . 30
3.6.3. Desarrollo . . . 30
3.6.4. Test realizados . . . 31
3.6.5. Conclusiones . . . 38
3.7. Iteración 5 . . . 39
3.7.1. Introducción . . . 39
3.7.2. Objetivos . . . 39
3.7.3. Desarrollo . . . 39
3.7.4. Tests realizados . . . 40
3.7.5. Primeras conclusiones . . . 49
3.7.6. Casos límites y/o situaciones de no convergencia . . . 49
3.7.7. Análisis de los casos . . . 50
3.7.8. Conclusiones . . . 51
3.8. Iteración 6 . . . 52
3.8.1. Introducción . . . 52
3.8.2. Objetivos . . . 52
3.8.3. Desarrollo . . . 52
3.8.4. Tests realizados . . . 53
3.8.5. Conclusiones . . . 65
3.9. Iteración 7 . . . 66
3.9.1. Introducción . . . 66
3.9.2. Objetivos . . . 66
3.9.3. Desarrollo . . . 66
3.9.4. Test Realizados . . . 67
3.9.5. Conclusiones . . . 73
3.10. Iteración 8 . . . 74
3.10.1. Introducción . . . 74
3.10.2. Objetivos . . . 74
3.10.3. Desarrollo . . . 74
3.10.4. Conclusión . . . 79

4. Conclusiones **80**
4.1. Conclusiones . . . 80

5. Trabajos Futuros **82**
5.1. Propuestas de trabajos futuros . . . 82

6. Anexos **83**

A. Anexo 1: Redes de Petri utilizadas **83**
A.1. Canal de Panamá . . . 83
A.2. Huang . . . 84
A.3. Guanjun . . . 85
A.4. Hospital . . . 86
A.5. MFC . . . 87
A.6. POPN . . . 88
A.7. Zhao . . . 89
A.8. Portugal . . . 90

B. Anexo 2: Tests iteración n°4 **91**
B.1. Red 3 - Hospital . . . 91
B.1.1. Test 1 - Red de Bayes sin estado . . . 91
B.1.2. Test 2 - Red de Bayes con estado . . . 93
B.1.3. Conclusiones de los test . . . 94
B.2. Red 4 - Huang . . . 94
B.2.1. Test 1 - Red de Bayes sin estado . . . 94
B.2.2. Test 2 - Red de Bayes con estado . . . 95
B.2.3. Conclusiones de los test . . . 96
B.3. Red 5 - MFC . . . 96
B.3.1. Test 1 - Red de Bayes sin estado . . . 96
B.3.2. Test 2 - Red de Bayes con estado . . . 98
B.3.3. Conclusiones de los test . . . 100
B.4. Red 6 - Portugal . . . 100
B.4.1. Test 1 - Red de Bayes sin estado . . . 100
B.4.2. Test 2 - Red de Bayes con estado . . . 102
B.4.3. Conclusiones de los test . . . 103
B.5. Red 7 - Zhao . . . 103
B.5.1. Test 1 - Red de Bayes sin estado . . . 103
B.5.2. Test 2 - Red de Bayes con estado . . . 105
B.5.3. Conclusiones de los test . . . 106

C. Anexo 3: Tests iteración n°5 **107**
C.1. Red 3 - Hospital . . . 107
C.1.1. Test 1: Realimentación considerando batch de ponderación . . . 107
C.1.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 108
C.1.3. Test 3: Realimentación considerando función exponencial sobre resta con batch . 109
C.2. Red 4 - MFC . . . 110
C.2.1. Test 1: Realimentación considerando batch de ponderación . . . 110
C.2.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 111
C.2.3. Test 3: Realimentación considerando función exponencial sobre resta con batch . 112
C.3. Red 5 - Portugal . . . 113
C.3.1. Test 1: Realimentación considerando batch de ponderación . . . 113
C.3.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 114
C.3.3. Test 3: Realimentación considerando función exponencial sobre resta con batch . 115
C.4. Red 6 - Zhao . . . 116
C.4.1. Test 1: Realimentación considerando batch de ponderación . . . 116
C.4.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 117
C.4.3. Test 3: Realimentación considerando función exponencial sobre resta con batch . 118
C.5. Red 7 - POPN . . . 119
C.5.1. Test 1: Realimentación considerando batch de ponderación . . . 119
C.5.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 120
C.5.3. Test 3: Realimentación considerando función exponencial sobre resta con batch . 121

D. Anexo 4: Tests iteración n°6 **122**
D.1. Red 4: Red Hospital . . . 122
D.1.1. Test 1: Realimentación considerando la resta del histórico . . . 122
D.1.2. Test 2: Realimentación considerando batch de ponderación . . . 123

D.1.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 124
D.1.4. Test 4: Realimentación considerando función exponencial sobre el histórico . . . 125
D.2. Red 5: MFC . . . 126
D.2.1. Test 1: Realimentación considerando la resta del histórico . . . 126
D.2.2. Test 2: Realimentación considerando batch de ponderación . . . 127
D.2.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 128
D.2.4. Test 4: Realimentación considerando función exponencial sobre el histórico . . . 129
D.3. Red 6: Red Portugal . . . 129
D.3.1. Test 1: Realimentación considerando la resta del histórico . . . 129
D.3.2. Test 2: Realimentación considerando batch de ponderación . . . 130
D.3.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 131
D.3.4. Test 4: Realimentación considerando función exponencial sobre el histórico . . . 132
D.4. Red 7: Red Zhao . . . 132
D.4.1. Test 1: Realimentación considerando la resta del histórico . . . 132
D.4.2. Test 2: Realimentación considerando batch de ponderación . . . 134
D.4.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 135
D.4.4. Test 4: Realimentación considerando función exponencial sobre el histórico . . . 136
D.5. Red 8: Red POPN . . . 136
D.5.1. Test 1: Realimentación considerando la resta del histórico . . . 136
D.5.2. Test 2: Realimentación considerando batch de ponderación . . . 137
D.5.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación . . . 138
D.5.4. Test 4: Realimentación considerando función exponencial sobre el histórico . . . 139

E. Anexo 5: Tests iteracion n°7 141
E.1. Red 4: Hospital . . . 141
E.1.1. Test 1: Realimentación resta directa . . . 141
E.1.2. Test 2: Realimentación función exponencial de ponderación . . . 142
E.2. Red 5: MFC . . . 143
E.2.1. Test 1: Realimentación resta directa . . . 143
E.2.2. Test 2: Realimentación función exponencial de ponderación . . . 144
E.3. Red 6: Portugal . . . 145
E.3.1. Test 1: Realimentación resta directa . . . 145
E.3.2. Test 2: Realimentación función exponencial de ponderación . . . 146
E.4. Red 7: Zhao . . . 147
E.4.1. Test 1: Realimentación resta directa . . . 147
E.4.2. Test 2: Realimentación función exponencial de ponderación . . . 148
E.5. Red 8: POPN . . . 149
E.5.1. Test 1: Realimentación resta directa . . . 149
E.5.2. Test 2: Realimentación función exponencial de ponderación . . . 150

1. Introducción

1.1. Motivación

Los sistemas ciberfísicos (CPS) se refieren a sistemas que consisten en componentes cibernéticos (como implementaciones computarizadas) y físicos. La idea general es que los componentes cibernéticos y físicos se influyen mutuamente de tal manera que el componente cibernético puede hacer que el componente físico cambie de estado y que el cambio, a su vez, se retroalimente, lo que resultará en un cambio de estado en el componente cibernético[4]. Los mismos pueden entenderse como redes de procesos multi-físicos (mecánicos, eléctricos, bioquímicos, etc.) y computacionales (control, procesamiento de señales, inferencia lógica, planificación, etc.), que a menudo interactúan con un entorno altamente incierto y adverso, incluidos actores humanos y otros CPS. Los avances en la teoría y la práctica de estos sistemas aportan los medios para obtener un desempeño óptimo de los sistemas dinámicos, mejorar la productividad, aligerar la carga de muchas operaciones manuales, repetitivas y rutinarias, así como de otras actividades.

Los CPS entonces, tienen una interacción constante con el ambiente donde operan elpi. La manera en que toman decisiones para interactuar con el mismo es a través de lo que llamamos política. El problema, es que las diversas formas de implementar esta política en la actualidad (estáticas, aleatorias o round robin) resultan ser insatisfactorias cuando se aplican en sistemas reactivos, como los descriptos anteriormente. Es por esto que estudiaremos otras alternativas de realimentación que permitan obtener una política del sistema de manera dinámica, capaz de cumplir con requerimientos de usuario preestablecidos, y de esta forma optimizar los recursos empleados y el tiempo de respuesta del sistema.

1.2. Objetivos

El objetivo del presente trabajo es encontrar el compensador correcto para construir un sistema realimentado capaz de generar de manera dinámica una política que nos permita ordenar los disparos de una red de Petri de forma tal que se cumpla con requerimientos específicos definidos por el usuario. Además es importante poder realizar esto sin modificar el grafo de la red de Petri, teniendo en cuenta que la misma debe ser de tipo *S3PR* y no debe poseer deadlock. Para llevar a cabo esto estudiaremos las redes Bayesianas que usan la probabilidad para tratar la incertidumbre dentro de la inteligencia artificial y compararemos su rendimiento contra los compensadores más utilizados.

1.3. Requerimientos

Las definiciones de requerimientos del sistema especifican qué es lo que el sistema debe hacer (sus funciones) y sus propiedades esenciales y deseables[15]. Para crear definiciones de requerimientos del sistema requiere consultar con los clientes del sistema y con los usuarios finales, por lo que para obtener la información necesaria en esta etapa se llevaron adelante las siguientes tareas:

1. *Entrevistas*: Se realizaron una serie de entrevistas con el director de tesis Luis Ventre en conjunto con el co-director Orlando Micolini, en las cuales se definió de forma general los objetivos del proyecto. Además nos brindaron bibliografía de guía para estudiar posibles soluciones.

2. *Búsqueda y estudio de documentos*: Se analizó la bibliografía provista por los docentes, se buscó profundizar en material relacionado y se prepararon posibles soluciones que se discutieron en nuevas entrevistas.

A partir de esto se generó una lista de requerimientos para establecer las funcionalidades que tiene que cumplir el sistema.

1.3.1. Lista de requerimientos

1. El sistema debe ser capaz de controlar los disparos de una Red de Petri de forma tal que cumpla con requerimientos de invariantes de transición.

2. El sistema debe funcionar sin alterar o modificar el grafo de la red de Petri.
3. El sistema debe ser capaz de funcionar con cualquier Red de Petri S3PR.
4. Se debe implementar un modulo capaz de leer los archivos .html generados por Petrinator y generar las matrices de la red de petri en texto plano.
5. El sistema debe poder ejecutarse en Windows y Linux.
6. El usuario debe poder determinar los requerimientos por invariante de transición que debe seguir el sistema para una red en particular con el fin de generar las políticas de la misma.
7. El sistema debe ser capaz de ilustrar si el mismo convergió de forma correcta.

1.4. Modelo de Desarrollo

El modelo adoptado para el desarrollo del presente trabajo es el modelo iterativo. En este se realizan múltiples iteraciones en las que se desarrolla una versión del proyecto, se la expone a revisión por parte del cliente y se recolectan las correcciones del mismo. Estas se utilizan como entrada de la nueva iteración donde se genera una nueva versión repitiendo el proceso hasta obtener un sistema adecuado según los requerimientos planteados.

La elección de este modelo se fundamenta en que el modelo iterativo es una buena opción para equipos pequeños con requerimientos no definidos al inicio como es nuestro caso. Permite ajustes continuos y retroalimentación frecuente de clientes e interesados, lo que reduce el riesgo del proyecto y mejora la satisfacción del cliente. Además, fomenta la comunicación constante entre los miembros del equipo, lo que puede mejorar la eficiencia y efectividad del proceso de desarrollo de software.

Las actividades de especificación, desarrollo y validación están entrelazadas en vez de separadas, con rápida retroalimentación a través de las actividades. Ian Sommervile[15].

Figura 1: Desarrollo incremental.

En cada nueva versión se busca incorporar algunas de las funciones requeridas por el usuario. Es habitual comenzar por aquellas funcionalidades más importantes o urgentes. Esta interacción constante con el usuario permite controlar, desde una etapa temprana del desarrollo, que el producto es lo que se requiere. Si esto no se cumple, solo la ultima iteración se debe modificar y, probablemente, definir una nueva funcionalidad para próximas iteraciones.

1.5. Gestión de riesgos

De forma simple, se puede concebir un riesgo como una probabilidad de que una circunstancia adversa ocurra. Los riesgos son una amenaza para el proyecto, para el software que se está desarrollando y para la organización[15]. Se los puede agrupar en tres categorías:

- *Riesgos del proyecto*: Repercuten sobre el cronograma o los recursos del proyecto.
- *Riesgos del producto*: Afectan la calidad o el rendimiento del software que se quiere desarrollar.

- *Riesgos del negocio*: Influyen a la organización que desarrolla o comercializa el software.

Dado que las consecuencias de estos riegos pueden provocar que el proyecto fracase, es importante generar un *plan de gestión de riesgos* una vez definidos los requerimientos y antes de iniciar el desarrollo del primero. Esto cobra mayor importancia en los proyectos de software debido a las incertidumbres inherentes con las que se enfrentan muchos proyectos. Es preciso anticiparse a los riesgos: comprender el impacto de éstos en el proyecto y en el producto final, y considerar los pasos para evitarlos. En el caso de que ocurran, se deben crear planes de contingencia para que sea posible aplicar acciones de recuperación. Las etapas que comprenden el proceso de gestión de riesgos son:

1. ***Identificación de riesgos.*** Identificar los posibles riesgos para el proyecto o el producto.
2. ***Análisis de riesgos.*** Valorar las probabilidades y consecuencias de estos riesgos.
3. ***Planificación de riesgos.*** Crear planes para abordar los riesgos, ya sea para evitarlos o minimizar su impacto en el proyecto.
4. ***Supervisión riesgos.*** Valorar los riesgos de forma constante y revisar los planes para la mitigación de riesgos tan pronto como se tenga nueva información sobre los mismos.

1.5.1. Identificación de riesgos

Para la realización de esta etapa se aplicaron dos técnicas:

- *Brainstorming*: Es una técnica creativa grupal cuyo objetivo es la generación de nuevas ideas sobre un tema o problema determinado. Permite obtener una lista de riesgos que luego serán analizados con mayor nivel de detalle.
- *Checklist*: consiste en un análisis detallado de los riesgos listados, descartando aquellos irrelevantes.

A continuación se presentan los riegos identificados, sobre que impactan y una breve descripción de los mismos.

Riesgo	Repercute en	Descripción
Planificaciones demasiado optimistas	Proyecto	El ser humano es optimista por naturaleza y tiende a minimizar los tiempos requeridos para realizar una tarea. Esto es particularmente difícil de estimar en proyectos de investigación.
Problema de coordinación entre los miembros del grupo	Proyecto	La descoordinación entre los integrantes de las prácticas puede llevar a retrasos en el desarrollo de las mismas.
Problemas de comunicación entre los integrantes	Proyecto	Una mala comunicación puede llevar a código duplicado o al desarrollo de lineamientos incongruentes.
Subestimación de tamaño o complejidad	Proyecto	Los temas a investigar pueden ser más complejos y requerir más tiempo de lo previsto.
Investigaciones innecesarias en el desarrollo de las prácticas	Proyecto	Dedicar tiempo a contenidos que en principio parezcan pertinentes al proyecto o finalmente no lo sean o no se utilicen.
Abandono de integrante del equipo	Proyecto	Un integrante abandona las prácticas.
Cambio de requerimientos	Proyecto y producto	Cambios en los requerimientos pueden llevar a que trabajo realizado haya sido en vano y se requiera de más tiempo para implementar los nuevos.
Pérdida del trabajo realizado	Proyecto y producto	Pérdida de lo realizado por error en la nube o extravío del dispositivo en el que se almacene.

Cuadro 1: Riesgos identificados

1.5.2. Análisis de riesgos

Durante este proceso, se considera por separado cada riesgo identificado y se decide acerca de la probabilidad y la seriedad del mismo. No existe una forma fácil de hacer esto (recae en la opinión y experiencia del gestor del proyecto). No se hace una valoración con números precisos sino en intervalos[15]:

- La probabilidad del riesgo se puede valorar como muy bajo ($< 10\,\%$), bajo (10-25 %), moderado (25-50 %), alto (50-75 %) o muy alto ($>75\,\%$).
- Los efectos del riesgo pueden ser valorados como catastrófico, serio, tolerable o insignificante.

Este análisis se puede observar en el cuadro 2.

Riesgo	Probabilidad	Efectos
Planificaciones optimistas	Alta	Tolerable
Problema de coordinación entre los miembros del grupo	Baja	Tolerable
Problemas de comunicación entre los integrantes	Baja	Tolerable
Subestimación de tamaño o complejidad	Alta	Serio
Investigaciones innecesarias en el desarrollo de las prácticas	Moderado	Tolerable
Abandono de integrante del equipo	Muy baja	Catastrófico
Cambio de requerimientos	Moderado	Tolerable
Pérdida del trabajo realizado	Muy baja	Catastrófico

Cuadro 2: Análisis de riesgos

1.5.3. Planeación de riesgos

El proceso de planificación de riesgos considera cada uno de los riesgos clave que han sido identificados, así como las estrategias para gestionarlos. Otra vez, no existe un proceso sencillo que nos permita establecer los planes de gestión de riesgos. Depende del juicio y de la experiencia del gestor del proyecto[15]. En esta etapa se seleccionaron los riesgos con mayor gravedad o mayor probabilidad de ocurrencia y se generaron los siguientes planes:

- *Plan de mitigación*: Tiene por objetivo reducir la probabilidad de ocurrencia del riesgo o el impacto que el mismo puede provocar. El mismo puede observarse en la tabla 3.
- *Plan de contingencia*: Esta comprendido por las acciones que se deben realizar en caso de que el riesgo se presente. Este puede apreciarse en la tabla 4

Riesgo	Estrategia
Subestimación de tamaño o complejidad	Se utilizarán estrategias con diferentes enfoques para lograr una buena estimación
Investigaciones innecesarias en el desarrollo de las prácticas	Se evaluará los pros y contras de investigar en el tema y qué impacto puede tener sobre el desarrollo del proyecto consultado con los directorios si les parece pertinente
Pérdida del trabajo realizado	Se guardarán copias en distintos dispositivos
Abandono de integrante del equipo	Continuación el trabajo de forma individual

Cuadro 3: Plan de mitigación

Riesgo	Estrategia
Subestimación de tamaño o complejidad	Se buscará dividir el problema en partes de menor complejidad y resolver las más relevantes. También se puede optar por buscar bibliografía adicional o consultar con un docente
Investigaciones innecesarias en el desarrollo de las prácticas	Cuestionar de manera crítica qué impacto o qué utilidad puede tener lo que se está investigando sobre el desarrollo del proyecto
Pérdida del trabajo realizado	Restaurar a la última versión de las copias de seguridad guardadas
Abandono de integrante del equipo	Se continuará el trabajo de forma individual

Cuadro 4: Plan de contingencia

2. Marco Teórico

2.1. Redes de Petri

Las Redes de Petri son modelos matemáticos utilizados para la representación de sistemas con paralelismo, concurrencia, sincronización e intercambio de recursos[10]. La red de Petri esencial fue definida por Carl Adam Petri. Son una generalización de la teoría de autómatas que permite expresar un sistema como eventos concurrentes.

Las redes de Petri están fuertemente asociadas a la teoría de grafos, ya que las mismas pueden representarse como un grafo dirigido bipartito compuesto por cuatro elementos:

- *Plazas*: Representan estados del sistema. El estado de una plaza está dado por la cantidad de marcas o tokens que esta contiene.
- *Token*: Figuran como puntos negros dentro de las plazas. Estos representan el valor específico de una condición o estado y generalmente se traducen en la presencia o ausencia de algún recurso del sistema.
- *Transiciones*: Representan el conjunto de sucesos cuya ocurrencia produce la modificación de los estados de las plazas, y en consecuencia del estado global del sistema.
- *Arcos*: Indican las interconexiones entre las plazas y las transiciones, estableciendo el flujo de tokens que sigue el sentido de la flecha.

2.1.1. Red de Petri Generalizada

Una Red de Petri generalizada no marcada es una cuádrupla $< P, T, Pre, Post >$ donde:

- $P = P_1, P_2, \ldots, P_n$ es un conjunto finito, no vacío, de plazas.
- $T = T_1, T_2, \ldots, T_m$ es un conjunto finito, no vacío, de transiciones, donde $P \cap T = \emptyset$, i.e. los conjuntos P y T son inconexos.
- $Pre : PxT \rightarrow \mathbb{N}^P$ es la función de incidencia de entrada y
- $Post : PxT \rightarrow \mathbb{N}^P$ es la función de incidencia de salida.

$Pre(p_i, t_j)$ contiene el peso del arco que va de P_i a T_j.
$Post(p_i, t_j)$ contiene el peso del arco que va de T_j a P_i.

2.1.2. Red de Petri Marcada

Una Red de Petri Marcada está definida por el par (N, M), donde N es una Red de Petri y $M : P \to \mathbb{N}^P$ (donde $|P| = p$ es una aplicación llamada **marcado**. $m(N)$ define el marcado de la RdP y m_{pi} indica el marcado de la plaza p_i, es decir, el número de tokens contenido en la plaza i. La marca inicial se denota m_0 y da la cantidad inicial de tokens en todas las plazas de la red, por lo que especifica el estado inicial del sistema.

2.1.3. Tipos de Arcos

- *Común*: Consume o produce una cierta cantidad de tokens de una plaza, de acuerdo al peso del mismo.
- *Inhibidor*: Siempre en la dirección plaza a transición. En el caso que el marcado de la plaza sea mayor o igual al peso del arco, la transición no está sensibilizada. No consume tokens al producirse el disparo de la transición asociada.
- *Lector*: Siempre en la dirección plaza a transición. La transición está sensibilizada si la marca en la plaza es mayor al peso del arco. No consume tokens al producirse el disparo de la transición asociada.
- *Reset*: Siempre en la dirección plaza a transición. Consume todos los tokens de la plaza al dispararse la transición.

2.1.4. Matrices

Para una red con n plazas y m transiciones, mas matrices tienen un tamaño nxm. Cada fila representa una plaza, mientras que cada columna representa una transición. Se conforman de la siguiente manera:

- *Matriz de Incidencia*: Está compuesta por las matrices I^+ e I^-, las cuales son función de los arcos comunes.

  En la matriz I^+, denominada matriz de incidencia de entrada o *post*, cada elemento $post(P_i, T_j)$ contiene el peso del arco que va desde T_j a P_i. Indica la cantidad de tokens generados al disparar la transición.

  En la matriz I^-, denominada matriz de incidencia de salida o *pre*, cada elemento $pre(P_i, T_j)$ contiene el peso del arco que va desde P_i a T_j. Indica la cantidad de tokens consumidos por la transición al realizar el disparo.

  Finalmente, la matriz de incidencia I se forma de la siguiente forma:

$$I = I^+ - I^- \tag{1}$$

- *Matriz de inhibición*: Contiene en cada uno de sus elementos $inh(P_i, T_j)$, el peso del arco de inhibición que va desde $P_i a T_j$
- *Matriz de Reset*: Contiene en cada uno de sus elementos $res(P_i, T_j)$ un 1 si existe un arco de reset que va desde P_i a T_j
- *Matriz de lectura*: Contiene en cada elemento $lec(P_i, T_j)$ el peso del arco lector que va desde P_i a T_j.

2.1.5. Sensibilizado de una transición

Una transición está sensibilizada si todas las plazas de entrada a la transición tienen una marca igual o mayor al peso del arco que une cada plaza con la transición.

Previo a expresar la condición de sensibilizado de manera general, son necesarias las siguientes definiciones:

- $\bullet T_j$ es el conjunto compuesto por las plazas entrante a T_j
- $T_j\bullet$ es el conjunto compuesto por las plazas salientes a T_j.
- $M_k(P_i)$ es el marcado de la plaza P_i antes de disparar la transición T_j.
- $M_{k+1}(P_i)$ es el marcado de la plaza P_i después de dispara la transición T_j.
- w_{ij} es el peso del arco $P_i \rightarrow T_j$
- w_{ji} es el peso del arco $T_j \rightarrow P_i$

De esta forma, el sensibilizado de una transición T_j se expresa como:

$$T_j \text{ está sensibilizada } sii\ \forall P_i \in\ \bullet T_j\ \rightarrow M_k(P_i) \geq w_{ij} \tag{2}$$

Lo definido anteriormente se cumple para redes cuyos arcos son todos comunes. Para redes con arcos inhibidores y/o de lectura cambia la condición de sensibilizado. De esta forma, dependiendo del tipo de arco deben cumplirse ciertas condiciones:

- *Arco de inhibición*: El marcado de la plaza de la cual parte debe es menor que el peso del arco.
- *Arco de lectura*: El marcado de la plaza de la cual parte debe ser mayor o igual al peso del arco.

Por su parte los arcos de reset no alteran la condición de sensibilizado de una transición.

2.1.6. Disparo de una transición

Dada una marca $M_k(P)$, cualquier transición que se encuentre sensibilizada puede ser disparada, este disparo nos llevará a una nueva marca $M_{k+1}(P)$ dada por:

$$M_{k+1}(P) = M_k(P) + I^+(P_i, T_j) - I^-(P_i, T_j) \forall P_i \in P \tag{3}$$

Al disparar la transición T_j se extraen tantos tokens de $\bullet T_j$ como indiquen los arcos que unen estas plazas con T_j. Se añaden a $T_j\bullet$ la cantidad de tokens que indiquen los arcos que unen a T_j con estas plazas. El disparo de una transición T_j se denota $M_k \rightarrow T_j \rightarrow M_{k+1}$.

2.1.7. Ecuación de estado

La ecuación de estado representa matemáticamente el comportamiento dinámico del sistema [8]. Esta permite obtener el estado del sistema luego del disparo de una transición. Se utiliza el marcado en un instante k para calcular el marcado de la red en un instante de tiempo $k+1$. Por lo tanto la ecuación de estado para una Red de Petri con n plazas y m transiciones se define de la siguiente forma:

$$M_{k+1} = M_k + I * \sigma \tag{4}$$

Siendo:

1. I la matriz de incidencia.
2. σ vector de disparo. Tiene dimensión m x 1 y contiene 1 en la posición de la transición que se quiere disparar.
3. M_k: vector de marcado actual.
4. M_{k+1}: vector de marcado del estado siguiente.

2.1.8. Extensión de la ecuación de estado

La ecuación de estado descrita previamente es útil únicamente en presencia de arcos comunes. Para representar matemáticamente la existencia de los demás arcos nombrados anteriormente se requiere de una matriz para indicar la conexión plaza transición para cada uno de estos.

- *Vector de transiciones des-sensibilizadas por arco inhibidor*: Es un vector de valores binarios de dimensión $m \times 1$, que indica con un cero cuáles transiciones están inhibidas por el arco y con un uno cuales no.
- *Vector de transiciones des-sensibilizadas por arco lector*: Es un vector de valores binarios de dimensión m x 1, que indica con un cero cuáles transiciones están inhibidas por el arco con un uno las que no.
- *Vector transiciones des-sensibilizadas por tiempo*: Es un vector de valores binarios de dimensión m x 1, que indica con un cero cuáles transiciones están inhibidas porque no se ha alcanzado o se ha superado el intervalo de tiempo transcurrido desde que la transición fue sensibilizada.
- *Vector de transiciones reset*: Es un vector de valores enteros de dimensión m x 1, que tiene el valor de la marca de la plaza que se quiere poner a cero, mientras que las otras componentes son uno.

Finalmente utilizando los vectores descriptos anteriormente se puede obtener el vector de sensibilizado extendido E_x:

$$E_x = E \ and \ B \ and \ L \ and \ Z \tag{5}$$

Para poder introducir el brazo reset hay que multiplicar elemento a elemento (#) al vector que resulte de la conjunción por A. Obteniendo así la ecuación de estado extendida:

$$M_{k+1} = M_k + I * ((\sigma \ and \ E_x)\#A) \tag{6}$$

2.1.9. Propiedades de las Redes de Petri

2.1.9.1 Propiedad de limitación

Dada una Red de Petri PN, se dice que una plaza P_i está **k-limitada** por una marcado inicial M_0 si hay un entero natural k tal que, para todas las marcas alcanzables desde M_0, el número de tokens en P_i no es mayor que k. Es decir:

$$\exists \ k \in \mathbb{N} \ / \ \forall M \in marcados(PN) \rightarrow M(P) \leq k \tag{7}$$

Una Red de Petri PN está limitada para un marcado inicial M_0 si todos los lugares están limitados para M_0. Es decir, PN está limitada por k si todas sus plazas están limitadas por k.

A partir de esta definición surgen varios conceptos, entre los cuales se encuentran los siguientes:

- Una Red de Petri es **segura** si todas sus plazas son **1-limitadas**.
- Una Red de Petri es **cíclica** si siempre existe la posibilidad de alcanzar el marcado inicial desde cualquier otro marcado alcanzable.
- Una Red de Petri es **repetitiva** si existe una secuencia de disparos que contiene a todas las transiciones y que lleva a la red desde el marcado actual al mismo marcado.
- Una Red de Petri es **conservativa** si se cumple que el número de tokens en el marcado es siempre el mismo.

2.1.9.2 Propiedad de vivacidad

La **vivacidad** de una transición indica que, en todo instante de la evolución de la red, su disparo es posible. Este concepto es particularmente relevante ya que determina si la ejecución de la red puede o no detenerse en un estado determinado. A partir de esto se puede definir la vivacidad de una red de Petri. Esta propiedad indica que una red es viva para un marcado si todas sus transiciones lo son.

Por otro lado, la **cuasi-vivacidad** de una transición expresa la posibilidad de dispararla al menos una vez a partir de un marcado inicial M_0. De la misma manera que para el caso de la vivacidad, una red de Petri es cuasi-viva si todas sus transiciones lo son.

Gracias a esta última definición, se puede definir la vivacidad en función de la cuasivivacidad de la siguiente manera: una transición es viva si la misma es cuasi-viva en la red para todo marcado alcanzable desde M_0.

La vivacidad está directamente asociada con la ausencia de **deadlock** o **interbloqueo**. En términos generales, el deadlock es el bloqueo permanente de un conjunto de procesos o hilos de ejecución en un sistema concurrente que compiten por recursos del sistema o bien se comunican entre ellos. En el caso de una red de Petri, esto suele ocurrir cuando dos o más transiciones esperan mutuamente por el disparo de la otra, produciendo el bloqueo permanente de esa porción de la red. Una red de Petri viva garantiza la ausencia de interbloqueo sin importar la secuencia de disparos.

2.1.9.3 Propiedad de alcanzabilidad

La **alcanzabilidad** de una Red de Petri es fundamental para el análisis de las propiedades dinámicas de un sistema. A grandes rasgos, permite determinar si el sistema modelado puede alcanzar un determinado estado.

Un marcado M_i es alcanzable desde el marcado inicial M_0 si existe una secuencia finita de disparos que me haga llegar a esa marca. Un marcado M_i es alcanzable desde M_0 si existe una secuencia finita de disparos σ tal que $M_0 \xrightarrow{\sigma} M_i$.

El conjunto de marcados alcanzables puede ser representado mediante un grafo, en el cual cada nodo representa un marcado. El arco entre estos nodos representa la transición que se necesita disparar para llegar de un marcado a otro.

Ejemplo:

Figura 2: Red de Petri de tres estados.

Esta red cuenta con tres estados:

1. Marcado inicial o al disparar T3 desde m_2: $m_0 = [1, 0, 0]$
2. Marcado al disparar T1 desde m_0: $m_1 = [0, 1, 0]$
3. Marcado al disparar T2 desde m_1: $m_2 = [0, 0, 1]$

Con estos estados es posible generar el siguiente grafo:

2.1.9.4 Cobertura de una Red de Petri

Si la red de Petri no es acotada habrá plazas cuya cantidad de tokens aumentará indefinidamente, lo que también se repetirá en los nodos del grafo de alcanzabilidad, por lo que el algoritmo para obtener el

Figura 3: Grafo de alcanzabilidad.

árbol de alcanzabilidad no convergerá. Para estos casos existe otro tipo de análisis denominado **grafo de cobertura**.

El grafo de cobertura consiste en graficar todas las marcas posibles de la red, pero colapsando en un único marcado genérico, representado con el sımbolo ω, aquellas plazas cuya cantidad de tokens crecerá infinitamente.

2.1.10. Invariantes

Las invariantes de una red son propiedades independientes tanto del marcado inicial como de la secuencia de disparos, y pueden asociarse a ciertos subconjuntos de plazas o de transiciones. De esta forma surgen dos conceptos:

- **Invariante de plaza** o **p-invariante**: Se llama así a un conjunto de plazas si dado un determinado vector de ponderación con enteros positivos o cero se cumple que:

  $$q_1 \text{ x } M(P_1) + q_2 \text{ x } M(P_2) + \cdots + q_n \text{ x } M(P_n) = K$$

  Siendo K constante para todo marcado. Es un componente conservativo independiente del marcado inicial. Sin embargo, el valor de la constante sí depende del marcado inicial.

- **Invariante de transición** o **t-invariante**: conjunto de transiciones que una vez disparadas secuencialmente generan el mismo marcado del cual habían partido, pudiendo disparar la secuencia indefinidamente.

2.2. Reinforcement Learning

Reinforcement Learning (RL) [17] es una rama del *machine learning* donde el aprendizaje se da interactuando con el ambiente. Es un aprendizaje orientado a objetivos, donde al algoritmo no se le enseña qué acciones debe tomar; sino que aprende de las consecuencias de sus acciones.

Este tipo de aprendizaje se complementa con el aprendizaje supervisado y no supervisado como se ve en la siguiente Fig.4:

El Reinforcement Learning entonces, intentará hacer aprender a la máquina basándose en un esquema de "premios y castigos" en un entorno en donde hay que tomar acciones y que está afectado por múltiples variables que cambian con el tiempo [3].

2.2.1. Componentes del RL

- *El agente:* Es el modelo que queremos entrenar para que aprenda a tomar decisiones
- *Ambiente:* Es el entorno en donde interactúa y "se mueve" el agente. El ambiente contiene las limitaciones y reglas posibles en cada momento.
- *Acción:* Son las posibles acciones que puede tomar en un momento determinado el Agente (cambiar de estado).
- *Estado:* Es el indicador de cómo se encuentran los diversos elementos que componen el ambiente en un momento dado.

Figura 4: Tipos de aprendizajes.

- *Recompensas:* Cada acción tomada por el Agente tendrá como consecuencia un refuerzo positivo o negativo que orientará al Agente hacia la forma correcta de comportarse.
- *Política:* Define el comportamiento del agente en el ambiente.

Figura 5: Interacción entre el agente y el entorno.

El agente se encuentra en un primer momento en un "estado inicial" y luego realiza una acción, lo cual genera que esto influya en el ambiente, luego de esto el agente obtiene dos cosas: Un nuevo estado y la recompensa. Si esta ultima es negativa el agente actuara de forma distinta frente a dicha situación y si es positiva reforzara este comportamiento. Esta interacción se ve mas claramente en la Fig.5.

Al finalizar la instancia de aprendizaje todo el conocimiento aprendido es almacenado en la poıtica.

Se debe lograr un equilibrio entre la recompensa y el agente, dado que si el agente recibe una recompensa lo suficientemente alta el mismo siempre realizará la misma acción no consiguiendo generalizar el sistema. Fig.6.

2.2.2. Tipos de algoritmos de RL

Existen dentro del aprendizaje por refuerzo ciertos algoritmos a emplear, algunos de ellos son:

2.2.2.1 Q-Learning(Value Learning):

El objetivo principal al entrenar el modelo a través de las simulaciones es ir completando una matriz de Políticas de manera que las decisiones que tome nuestro agente obtengan "la mayor recompensa" evitando el sobreajuste.

Figura 6: Interacción entre el agente y el entorno.

- A la política se la denomina Q.
- Q(estado, acción) nos indica el valor de la política para un estado y una acción determinados.

Para completar la matriz de políticas se utiliza la ecuación de Bellman Fig.7.

Figura 7: Ecuación de Bellman.

Esta ecuación determina cómo se irán actualizando las políticas Qˆ(s,a), en base a su valor actual más una recompensa recibida como consecuencia de dicha acción. Hay dos ratios que afectan a la manera en que influye esa recompensa: el ratio de aprendizaje, que regula "la velocidad" en la que se aprende, y la "tasa de descuento" que tendrá en cuenta la recompensa a corto o largo plazo.

2.2.2.2 Policy learning

Este método de aprendizaje por refuerzo tiene como principal diferencia a Value Learning (Q-Learning) que en esta última se busca tener una Red Neuronal/Política que aprenda a aproximar la función Q [1], para obtener un valor $Q_(s, a)$ de un estado dada una acción, y luego usamos este valor para inferir cuál es la mejor acción a tomar, esta es nuestra política.

Por otra parte **Policy Learning** busca directamente aprender la política pudiendo usar una NN o, en casos más simples, una matriz para luego poder alimentar la misma con una entrada y como salida se tendrá qué acción debemos tomar. Esto simplifica la situación, ya que para obtener la acción a tomar, es decir la acción que maximizar´a la recompensa dado un estado, simplemente se debe muestrear desde la función de política sin necesidad de realizar numerosas valuaciones.

En resumen en **Value Learning** se aproxima una función Q y se la usa para inferir la política óptima, mientras que en **Policy Learning** se optimiza la política de forma directa.

La política en esta metodología será alimentada con el estado actual, y como salida tendrá las probabilidades correspondientes a cada acción posible.

En la figura 8 se tiene como ejemplo las probabilidades de los distintos tipos de acciones que puede tomar el agente (la barra) con el fin de pegarle a la bola roja. Aquí la opción óptima es la de moverse

Figura 8: Policy gradient.

a la izquierda, pero dado que esta es una distribución de probabilidad, la vez que se muestree podría llegar a elegirse quedarse en el lugar (acción 2) dado que también cuenta con un valor no nulo de probabilidad.

Este método de RL puede trabajar con valores continuos. Siguiendo con el ejemplo previo, podría no solo obtenerse la dirección a la cual moverse, sino también la velocidad con la cual hacerlo. Esto puede realizarse visualizando la salida como una distribución probabilística dependiendo de cual se adapte más a la situación. En el ejemplo siguiente se toma una distribución gaussiana.

Figura 9: Policy gradient.

En la Fig.9 puede observarse que el agente debe moverse hacia la izquierda con una velocidad cercana a -1 m/s. Al muestrear en esta distribución, se puede notar que la velocidad en concreto que tomará el agente es de 0.8 m/s hacia la izquierda. De esta forma también se puede ver que a pesar de que la media de esta distribución es -1, no estamos limitados a ese número exacto.

Todo esto abre la posibilidad para escenarios donde puede llegar a tenerse un sin fin de acciones posibles a tomar.

2.3. Redes de Bayes

2.3.1. La estadística de Bayes

Es necesario definir ciertos términos que se utilizan con frecuencia en el manejo de la probabilidad Bayesiana. Estos son:

- **Probabilidad previa o a priori:** Es la probabilidad que se tiene de que cierto modelo sea cierto, antes de obtener datos a través de observaciones. Es necesario advertir que las observaciones cambian lo que sabemos del modelo.
- **Probabilidad posterior o a posteriori:** Es la probabilidad de que el modelo sea cierto después de las observaciones.
- **Variables aleatorias:** Representa una "parte"del mundo cuyo estado se desconoce. Se representan en mayúscula y sus posibles valores en minúscula. Las v.a pueden ser de distintos tipos:
    - Booleanas: El conjunto de valores posible esta formado solo por dos valores: $true$ (verdadero) y $false$ (falso).

    - Notación: a y $\neg a$ son equivalentes a $A = true$ y $A = false$ respectivamente.
  - Discretas: Sus posibles valores componen un conjunto discreto. Incluyen a las booleanas.
  - Continuas: El conjunto de valores posibles es un conjunto no numerable.

- **Proposiciones:** Usando las conectivas proposicionales y las variables, podemos expresar proposiciones.
  - Conectivas: $\vee(or)$, $\wedge(and)$, $\sim(nop)(\neg)$
  - Se asignan probabilidades a las proposiciones para expresar el grado de creencia que se tiene en las mismas.

2.3.2. Probabilidad: definición axiomática

Una función de probabilidad es una función definida en el conjunto de proposiciones (respecto de un conjunto dado de variables aleatorias), verificando las siguientes propiedades:

- $0 \leq P(a) \leq 1$ para toda proposición a.
- $P(true) = 1$ y $P(false) = 0$
  - donde true y false representan a cualquier proposición redundante o insatisfacible, respectivamente.
- $P(a \vee b) = P(a) + P(b) - P(a \wedge b)$, para cualquier par de proposiciones a y b.

El calculo de probabilidades se construye sobre estos tres axiomas. Por ejemplo:

- $P(\neg a) = 1 - P(a)$
- $P(a \vee b) = P(a) + P(b)$, si a y b son independientes.
- $\sum_{i=0}^{n} P(D = d_i) = 1$, siendo D una variable aleatoria y $di, i = 1, 2, \ldots, n$ sus posibles valores.

2.3.3. Teorema de Bayes

Sea $\{A_1, A_2, \ldots, A_i, \ldots, A_n\}$ un conjunto de sucesos mutuamente excluyentes y exhaustivos tales que la probabilidad de cada uno de ellos es distinta de cero ($P[A_i] \neq 0$ para $i = 1, 2, \ldots, n$). Si B es n suceso cualquiera del que se conocen las probabilidades condicionales $P(B|A_i)$ entonces la probabilidad $P(A_i|B)$ viene dada por la expresión:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{P(B)}$$

donde:

- $P(A_i)$ son las probabilidades a priori.
- $P(B|A_i)$ son las probabilidades de B en la hipótesis A_i.
- $P(A_i|B)$ son las probabilidades a posteriori.

2.3.4. Fórmula de Bayes

Con base en la definición de probabilidad condicionada se obtiene la Fórmula de Bayes, también conocida como Regla de Bayes:

$$P(A_i|B) = \frac{P(B|A_i)P(A_i)}{\sum_{k=1}^{n} P(B|A_k)P(A_k)} \cdots$$

Esta fórmula nos permite calcular la probabilidad condicional $P(A_i|B)$ de cualquiera de los eventos A_i dado B.

2.3.5. Redes Bayesianas

Es un grafo que representa la relación entre las variables de un problema, permitiendo una representación compacta y son de ayuda en la toma de decisiones [5]. Están conformadas por:

- *Nodos:* círculos que representan una variable aleatoria.
- *Flechas:* relaciones causales entre las variables aleatorias (conexiones entre nodos).

Las redes Bayesianas nos permiten realizar inferencias[2], existen tres tipos:

- Razonamiento de predicción.
- Razonamiento de diagnostico.
- Razonamiento de justificación.

2.3.5.1 Razonamiento de predicción

Es explicado por la cadena de causalidad.

Se usa la causa (información dada) para inferir algo

Figura 10: Cadena de Causalidad.
Red de Bayes con tres nodos.
Nodo A causa Nodo B y Nodo B causa Nodo C.

Aquí se utiliza la información dada en la causa para poder inferir algo. Si se tiene conocimiento sobre A, entonces se puede inferir le probabilidad de que ocurra B, y a su vez, la probabilidad de que ocurra algo sobre C.

El evento A causa el evento B y el evento B causa el evento C

Por lo que si se conoce el se puede inferir C y no es necesario conocer A. Esto se llama **bloqueo**, el evento B *bloquea* el evento A.

2.3.5.2 Razonamiento de diagnostico

Es explicado por una red de causa común.

Figura 11: Red de causa común.
Nodo A causa Nodo B y también causa Nodo C.

2.3.5.3 Razonamiento de justificación

Dos o más eventos son causales de un tercero. Estas redes presentas características como:

- Independencia condicional
- Dependencia condicional

Entonces:

- Si A es verdadero es posible inferir la probabilidad de que sus padres A y B también sean verdaderos.
- Se puede explicar que ha causado C.

Cuando esto ocurre, si se conoce que uno de los eventos es verdadero, se puede inferir que la probabilidad que otro evento causante sea verdadero es menor. Esto es interesante dado que A y B son independientes.

En este tipo de redes también puede ocurrir un bloqueo. Existen casos donde un nodo puede bloquear varios nodos, esto es conocido con D-separación.

La figura 12, aclara estas ideas con un ejemplo, muestra si un vuelo puede estar retrasado; el retraso puede ser debido al mal tiempo o al exceso de trafico aéreo. La red esta compuesta por un nodo para cada variable y las relaciones entre estos (nodo Vr vuelo, nodo Mt mal clima, nodo Ta trafico aéreo). Las relaciones representada por flechas son causales donde los padres causan a los hijos.

En el ejemplo el retraso puede deberse al mal tiempo o al exceso de trafico aéreo y también por ambos.

Estas relaciones son probabilísticas y la fuerza de la causa depende del valor de la probabilidad que las relaciona. Por lo que en las aplicaciones, cuando sea posible conviene, que los eventos resultantes discretos sean mutuamente excluyentes, es decir tengamos solo un valor del grupo posible. Esto facilita el cálculo.

Para el desarrollo de este ejemplo se supone a todas las variables aleatorias como binarias (V,F). Ahora, con esta información, es posible saber si el vuelo esta retrasado o no.

Para esto es necesario construir una tabla de probabilidades condicionales para saber que valores son necesarios utilizar en el calculo de la probabilidad final, dependiendo del mal tiempo y del exceso de trafico.

La tabla de probabilidades condicionales del cuadro 5 debe construirse siempre para todos los nodos hijos de la red. Los nodos padres tendrán una tabla de probabilidad a priori.

Figura 12: Red de efecto común.
Nodo A y Nodo B son causa de Nodo C.

Valores de los padre		**Probabilidad para vuelos retrasados**	
Mal tiempo	Trafico aereo	Retrasado=Verdadero	Retrasado=Falso
v	v	$p(Vr \cap Mt \cap Ta)$	$p(\sim Vr \cap Mt \cap Ta)$
v	f	$p(Vr \cap Mt \cap \sim Ta)$	$p(\sim Vr \cap Mt \cap \sim Ta)$
f	v	$p(Vr \cap \sim Mt \cap Ta)$	$p(\sim Vr \cap \sim Mt \cap Ta)$
f	f	$p(Vr \cap \sim Mt \cap \sim Ta)$	$p(\sim Vr \cap \sim Mt \cap \sim Ta)$

Cuadro 5: Tabla de probabilidades condicionales para las variables Vr, Mt y Ta

3. Desarrollo

3.1. Iteraciones

Teniendo en cuenta lo expuesto previamente en la secciones Modelo de Desarrollo y Gestión de riesgos, se propone la siguiente planificación para el desarrollo del proyecto.

3.1.1. Planificación de las iteraciones

El desarrollo del proyecto se dividirá en las siguientes iteraciones:

- *Iteración 1*: Implementación de la lógica de ejecución de Redes de Petri y comunicación con Petrinator.
  1. Implementar la lógica para soportar Redes de Petri a partir de sus matrices.
  2. Implementar una herramienta que facilite la obtención de las matrices desde Petrinator.
- *Iteración 2*: Implementación del environ.
  1. Adaptación de la red, para poder ser implementada como un 'environ' propio del modelo de Reinforcement Learning.
  2. Implementar lógica de los métodos step y get_reward. Para este último considerar:
     - Puntos por poder disparar la transición
     - Puntos por utilizar un recurso
     - Puntos por liberar un recurso
     - Puntos por sensibilizar transiciones
     - Puntos por liberar un recurso que podría ser utilizado en la inmediatez
     - Puntos por completar invariantes
- *Iteración 3*: Implementar el agente y el sesgo inicial.

1. Implementar la lógica de un sesgo inicial: En una red de Petri es muy probable que la cantidad de transiciones que no estén sensibilizadas i.e. no se puedan disparar, sea un número proporcionalmente grande, llevando a que el sistema tarde en converger o bien, diverger.
2. Implementar la lógica de distintas políticas de decisión del autómata. Una determinística (la que genere mayor recompensa) y una probabilística (más probabilidad si mayor recompensa).

- *Iteración 4*: Implementar Redes de Bayes como sistema de control sobre conflictos.

  1. Implementar lógica para determinar situaciones de conflicto en Redes de Petri.
  2. Implementar lógica para obtener información para inicializar las tablas a priori de las redes Bayesianas.
  3. Implementar lógica para soportar redes de Bayes.
  4. Implementar lógica para la generación de políticas de la Red de Petri utilizando redes de Bayes para la resolución de conflictos.

- *Iteración 5*: Especificación de requerimientos por invariante de transición.

  1. Implementar una lógica para la especificación de requerimientos por invariante de transición.
  2. Implementar una lógica para contabilizar la cantidad de invariantes realizados.
  3. Implementar lógica de control para que la cantidad invariantes completados se ajuste a los requerimientos.
  4. Implementar una lógica para determinar si el sistema converge o no a los requerimientos especificados.

3.2. Condiciones

En esta sección detallaremos las condiciones que van a formar parte de la recompensa retornada por el ambiente al disparar una transición. Las mismas, son las siguientes:

- *Recompensa por poder dispararse*: Es la más simple de todas. Si dado un estado, puedo disparar una transición, va a tener esta componente. Sirve como caso base de prueba y equiponderar las transiciones disparables al deshabilitar el resto de las recompensas.
- *Recompensa por uso de recursos*: Si la transición disparada representa el uso de recurso, la recompensa otorgada por esta componente es directamente proporcional a la cantidad de recursos utilizados.
- *Recompensa por liberación de recursos*: Si la transición disparada representa la liberación de un recurso, la recompensa de esta componente es directamente proporcional a la cantidad de recursos liberados.
- *Recompensa por la liberación de recursos requeridos en la inmediatez*: Es un plus que se agrega a la recompensa anterior, en el caso en que el recurso liberado, sea necesario. i.e. permita que una nueva transición que tenga como plaza de entrada la que contenga el recurso liberado, se sensibilice por el disparo de la primer transición.
- *Recompensa por transiciones sensibilizadas*: Otorga recompensas si el disparo de la transición produce la sensibilización de otras que no lo estaban antes del disparo.
- *Recompensa por invariante completado*: La más importante de todas porque simboliza la finalización de un proceso o circuito. Otorga recompensa si la transición disparada, es la última necesaria para terminar un invariante. Notar el caso particular de que no sólo depende del disparo de una transición, si no de que se hayan disparado previamente el resto de transiciones pertenecientes a dicho invariante.

La recompensa obtenida será una suma ponderada de la siguiente forma:

$$R_t = \sum_{i=1}^{n} \alpha_i * R_i \tag{8}$$

donde:

- R_i es la recompensa iésima, y su valor representa la cantidad de veces que se cumplió esa condición,
- α_i es el peso o valor numérico de cumplir la recompensa iésima una vez y
- n es el número de condiciones que otorgan recompensas

3.3. Iteración 1

3.3.1. Introducción

En esta iteración se buscará implementar un ambiente capaz de representar los elementos de una red de Petri (plazas, arcos y transiciones) representados como matrices, y su lógica de disparo así como también una herramienta que nos permita obtener la información necesaria para dicha representación desde los archivos .html generados por Petrinator.

3.3.2. Objetivos

1. Implementar la lógica para soportar Redes de Petri a partir de sus matrices.
2. Implementar una herramienta que facilite la obtención de las matrices desde Petrinator.

3.3.3. Desarrollo

3.3.3.1 Lógica de redes de petri

En esta etapa se creara una clase que realice las operaciones necesarias para manipular redes de Petri. Indica el marcado de la red en un momento dado, cuales transiciones están sensibilizadas en dicho momento, permite disparar transiciones y actualiza el estado de la red.

El constructor de la clase toma como parámetros los elementos que caracterizan a una red de petri, tales como el marcado inicial y las matrices de incidencia.

3.3.3.2 HTML_Parser

Para la representación de las redes de Petri en el sistema se necesita obtener desde Petrinator los siguientes elementos de cada red:

- La matriz de incidencia de entrada W^-.
- La matriz de incidencia de salida W^+.
- El marcado inicial M_0.
- Los invariantes de transición.
- Entero natural k al que esta limitada la red.

Petrinator permite obtener estos datos y guardarlos en archivos con formato .html. Se decidió implementar la clase *HTML_Parser* capaz de interpretar el contenido de estos archivos y recuperar e imprimir los elementos listados en formato requerido por Python.

3.3.4. Conclusiones

En esta iteración se logró con éxito incorporar la lógica de las redes de Petri al sistema y generando un base que permitirá implementar sobre esta un sistema que genere dinámicamente la política de disparo de la red.

3.4. Iteración 2

3.4.1. Introducción

En esta iteración se buscará adaptar el código de la red de Petri para que pueda ser implementada como un 'environ' propio del modelo de Reinforcement Learning. Se implementará la lógica de recompensas y del método *step*.

3.4.2. Objetivos

1. Adaptación de la red de petri para poder ser implementada como un 'environ' propio del modelo de Reinforcement Learning.
2. Implementar lógica de los métodos step y get_reward. Para este último considerar por:
   - Poder disparar la transición
   - Utilizar un recurso
   - Liberar un recurso
   - Sensibilizar transiciones
   - Liberar un recurso que podría ser utilizado en el próximo disparo
   - Completar invariantes

3.4.3. Desarrollo

En el modelo de Reinforcement Learning, el environ es algo a partir del cual se ejecuta una acción y se observa qué sucede. En la práctica, esto se hace mediante el método *step*, cuyo prototipo es el siguiente:

$$state, reward, done, info\ step(action) \tag{9}$$

donde:

- *action*: Valor que recibe la función por parámetro. Es la acción a realizar sobre el ambiente. Va de 0 a $n_actions - 1$. En este caso, 'action' en el scope de Reinforcement learning es la transición a disparar en el scope de la red de Petri.
- *state*: Valor retornado por la función. Representa la información que se puede obtener del ambiente luego de realizar la acción. En este caso, es el nuevo estado de la red.
- *reward*: Valor retornado por la función. Es un indicador de qué tan bueno (o malo) fue el resultado de ejecutar la acción. Mientras mayor sea numéricamente hablando, mejor habrá sido la decisión tomada. En este caso, será una combinación lineal de una serie de factores a considerar, que se definirán en la próxima sección.
- *done*: Valor booleano que retorna la función. Indica que la ejecución terminó, sea porque se cumplió un objetivo o porque hay un fallo del que no se puede retornar. En el caso de la red de Petri, solo será True en caso de intentar disparar una transición no sensibilizada.
- *info*: Valor de retorno de la función. Información extra, en nuestro caso se omite este campo. Siempre será null, pero se incluye para mantener compatibilidad con el standard de environ de reinforcement learning.

3.4.3.1 Calculo de la Recompensa

Las condiciones que van a formar parte de la recompensa retornada por el ambiente al disparar una transición, son las siguientes:

- *Recompensa por poder dispararse*: Es la más simple de todas. Si dado un estado, puedo disparar una transición, va a tener esta componente. Sirve como caso base de prueba y equiponderar las transiciones disparables al deshabilitar el resto de las recompensas.
- *Recompensa por uso de recursos*: Si la transición disparada representa el uso de recurso, la recompensa otorgada por esta componente es directamente proporcional a la cantidad de recursos utilizados.
- *Recompensa por liberación de recursos*: Si la transición disparada representa la liberación de un recurso, la recompensa de esta componente es directamente proporcional a la cantidad de recursos liberados.
- *Recompensa por la liberación de recursos requeridos en la inmediatez*: Es un plus que se agrega a la recompensa anterior, en el caso en que el recurso liberado, sea necesario. i.e. permita que una nueva transición que tenga como plaza de entrada la que contenga el recurso liberado, se sensibilice por el disparo de la primer transición.
- *Recompensa por transiciones sensibilizadas*: Otorga recompensas si el disparo de la transición produce la sensibilización de otras que no lo estaban antes del disparo.
- *Recompensa por invariante completado*: La más importante de todas porque simboliza la finalización de un proceso o circuito. Otorga recompensa si la transición disparada, es la última necesaria para terminar un invariante. Notar el caso particular de que no sólo depende del disparo de una transición, si no de que se hayan disparado previamente el resto de transiciones pertenecientes a dicho invariante.

La recompensa obtenida será una suma ponderada de la siguiente forma:

$$R_t = \sum_{i=1}^{n} \alpha_i * R_i \tag{10}$$

donde:

- R_i es la recompensa iésima, y su valor representa la cantidad de veces que se cumplió esa condición,
- α_i es el peso o valor numérico de cumplir la recompensa iésima una vez y
- n es el número de condiciones que otorgan recompensas

3.4.4. Conclusiones

En esta iteración se logró desarrollar con éxito los métodos planteados, logrando incorporar a la red de petri los métodos propios de un agente en un modelo de reinforcement learning.

3.5. Iteración 3

3.5.1. Introducción

En esta iteración se llevará a cabo la implementación del agente y del sesgo inicial.

3.5.2. Objetivos

1. Implementar la lógica de un sesgo inicial para el agente.
2. Implementar la lógica de distintas políticas de decisión del agente: Una determinística (se dispara la transición que mayor recompensa genere) y una probabilística (más probabilidad si mayor recompensa).

3.5.3. Desarrollo

En esta iteración, se implementará el agente. Como ya se mencionó, el agente es quien toma acción sobre el ambiente. La decisión de qué acción realizar se lleva a cabo a través del método *choose_action()* que recibe por parámetro el estado actual y la política a utilizar. Esta última puede ser una de las siguientes:

- *eg_policy*: Política épsilon greedy. En base a parámetros de exploración, determina si se inclina por la transición que mayor recompensa otorgue o por una al azar. Utiliza tabla Q.
- *prob_eg_policy*: Política épsilon greedy probabilística. En base a parámetros de exploración, determina si para seleccionar la acción a realizar se realiza mediante una distribución de probabilidad que priorice a aquellas que más recompensa otorguen o si elige una al azar i.e. distribución equiprobable. En ambos casos, nunca seleccionará una transición que no se pueda disparar en el estado pasado por parámetro. Utiliza una tabla Q probabilística, que se explica más adelante en esta iteración.
- *Otras políticas*: Que se explicarán en otras iteraciones.

A diferencia del trabajo final en el que se basa este trabajo, donde las tablas de Q-learning se utilizaban en caso de conflictos y solo considerando las transiciones pertenecientes a dicho conflicto, la idea del presente es extender el sistema para controlar todas las transiciones, independientemente del estado de la red.

Al encarar este problema como un problema de Q learning tradicional, aparecen dos situaciones a resolver:

- *Determinar el tamaño de la tabla Q*
- *Tiempo en converger*

3.5.3.1 Tamaño de la tabla Q

Una tabla Q tiene dimensiones n_states x $n_actions$, donde $n_actions$ es el número de transiciones a disparar, y n_states es la cardinalidad del conjunto de marcados posibles de la red de petri, dado un estado inicial.

Para determinar los estados posibles, se desarrolló una búsqueda por amplitud (cuyo nodo raíz es el estado inicial) disparando todas las transiciones y generando un nodo a explorar en caso de poder disparar la transición e.g. environ.can_shoot(T_i) $== true$ y que el nodo no haya sido explorado previamente. Luego, el conjunto de los nodos explorados contiene el conjunto de estados posibles.

3.5.3.2 Sesgo inicial

Al haber redes grandes en las cuales, dado un estado, solo un subconjunto pequeño del conjunto de transiciones es factible de disparar, inicializar la tabla Q en ceros lleva a que el agente demore un tiempo considerable en poder aprender a disparar unas pocas transiciones. Surge entonces la idea de crear una tabla sesgo. Esto es, una tabla no inicializada en ceros.

Como el problema principal es que, dado un estado, la mayoría de las transiciones no están sensibilizadas, se optó por aprovechar la búsqueda por amplitud del punto anterior para completar una estructura que vinculara, $estado_{actual} \rightarrow estados_{alcanzables}$ (en dicho estado).

Para esto, cada nodo de la búsqueda por amplitud (recordemos que hay un nodo por estado posible) contenía en su estructura un vector de dimensión M i.e. número de transiciones posibles. El vector tiene en su posición iésima, el identificador del estado al que se llega desde el estado actual tras disparar la transición i. En caso de que la transición i no se pueda disparar, la posición contiene -1. Cabe destacar que el identificador de un estado alcanzable es un número entero no negativo.

Luego, a cada estado se le asigna un valor igual a la mayor recompensa que se puede obtener, considerando todas las posibles m^n combinaciones de disparos posibles partiendo de dicho estado. Donde n es un valor parametrizable y representa la profundidad deseada.

En la práctica, para cada estado se crea un tensor de rango igual a la profundidad de análisis elegida, donde cada elemento del tensor representa la recompensa obtenida tras disparar en orden las coordenadas del tensor, i.e. $tensor[0][2][1][3] = 67$ (tensor de rango 4), indica que dado un estado (al que pertenece el tensor), la recompensa de disparar las transiciones $0 - 2 - 1 - 3$ (en ese orden) es de 67. Luego, el valor que retornará el método $recursive_reward_getter()$ es el máximo valor encontrado en el tensor.

Vinculando esta información se pueden obtener vectores de dimensión M en los que, dado un estado, se obtiene la máxima recompensa esperada por el estado al que se llega tras disparar la transición i. Solo resta sumarle a cada elemento la recompensa de llegar desde el estado actual al nuevo estado.

Finalmente, al superponer estos vectores, se obtiene una tabla $Q(estado, transicion)$ tal que:

$$Q(estado, t_i) = \begin{cases} r > 0 & si\ t_i\ es\ disparable \\ r = 0 & en\ caso\ contrario \end{cases}$$

Superponiendo estos vectores se obtiene una *Q_value_table*, que no permite disparar transiciones no sensibilizadas y en la que al elegir la transición a disparar, elige la que mayor recompensa otorgue en el corto plazo.

3.5.3.3 Haciendo la tabla probabilística

Si bien es cierto que eligiendo siempre la transición cuya recompensa es la mayor, la recompensa global debería tender a ser la mayor, esto, en el escenario de representar un proceso productivo, podría llevar a realizar siempre el mismo proceso, condenando a los demás a no completarse nunca.

Para ello, se generó una tabla similar a la anterior, pero en la que se dividió cada elemento de Q_{ij} por la suma de la fila perteneciente al estado, i.e. $sum(Q_i)$. De esta forma:

$$Qprob_{ij} = \frac{Q_{ij}}{\sum_k Q_{ik}} \tag{11}$$

Con esta nueva tabla, se puede generar un número aleatorio, utilizando la fila Q_i como distribución de probabilidad y así elegir una transición con un peso ponderado a la recompensa de dispararla respecto de disparar las demás.

3.5.4. Conclusiones

En esta iteración se generó con éxito una tabla_Q con el objetivo de que el agente pueda, considerando los estados posibles, entrenar y ejecutarse sin perder tiempo persiguiendo trazas no realizables, concentrando tiempos y recursos en buscar qué caminos llevan a mayor recompensa.

3.6. Iteración 4

3.6.1. Introducción

En esta iteración se implementará una red de Bayes que represente de manera adecuada la dependencia de las variables de la red de Petri así como también un algoritmo para la generación automática de la política utilizando redes Bayesianas.

3.6.2. Objetivos

1. Implementar lógica para determinar situaciones de conflicto en Redes de Petri.
2. Implementar lógica para obtener información para inicializar las tablas a priori de las redes Bayesianas.
3. Implementar lógica para soportar redes de Bayes.
4. Implementar lógica para la generación de políticas de la Red de Petri utilizando redes de Bayes para la resolución de conflictos.

3.6.3. Desarrollo

3.6.3.1 Determinación de conflictos

Para la determinación de conflictos se decidió realizar una modificación a la definición de conflicto efectivo, a fines de poder considerar subconflictos dentro del mismo sin la necesidad de desglosarlos. Con la nueva consideración, la definición queda de la siguiente manera:

"Se define a un conflicto efectivo $K_e =< K, m >$ como la existencia de un conjunto estructural $K =< P_i, T1, T2, ... >$ y de una marca m, tal que de las transiciones sensibilizadas del conjunto $T1, T2, ...$ el número de tokens en P_i es menor que la suma de los pesos de los arcos de dichas transiciones habilitadas"

La implementación en código de la definición anterior es directa. Luego, se agrega un vector booleano de dimensión $n_conflictos$ x 1 que retorna si, dado el estado actual de la red, el conflicto i se encuentra efectivo o no.

3.6.3.2 Recolección de experiencias

Se utiliza el agente desarrollado en la iteración anterior para disparar la red de Petri y se generara un archivo de log donde se registrará en cada disparo la información necesaria para calcular las probabilidades a priori y condicionales de las tablas de cada red Bayesiana. Los datos registrados son:

- Estado de la red antes del disparo.
- La transición que se disparo.
- Si se resolvió un conflicto, el número del conflicto.
- Por cada condición que se haya cumplido, el número de condición.

3.6.3.3 Redes de Bayes

Con el objetivo de resolver los conflictos de la red de Petri con el uso de redes Bayesianas se decidió generar una de estas para cada conflicto. Se probaron dos diseños:

- Red de Bayes stateless: Representada en la figura 13, la misma está compuesta por un nodo que representa el conflicto pudiendo tomar valores entre 0 y el número de transiciones de la red de Petri que forman parte del conflicto, y un nodo por cada una de las condiciones listadas en la sección 3.2. Consideramos que el hecho de que las condiciones se cumplan (su valor sea 1) o no depende de la transición que resuelva el conflicto, es por eso que los arcos van desde el nodo del conflicto hacia los nodos de condiciones.

Figura 13: Red de Bayes sin estado.

- Red de Bayes stateful: Se puede observar en la figura 14. Su estructura es similar a la anterior pero se agrega un nodo que representa el estado de la red de Petri donde la tabla de este nodo tendrá tantas entradas como estados existan. El estado de la red de Petri es una variable observable por lo que su valor sera evidencia a la hora de calcular la probabilidad posterior conjunta del nodo C.

Figura 14: Red de Bayes con estado.

Para el soporte de la lógica bayesiana y su utilizacion para la resolución de conflictos fue necesario implementar 4 métodos encargados de generar la red de Bayes para cada conflicto, generar las tablas de cada nodo y cargarlas con los datos obtenidos de las experiencias, actualizar las tablas en tiempo de ejecución a medida que se tenga nueva información y seleccionar la transición con la que se debe resolver un conflicto en cada caso. Estos métodos son:

- *get_bns_list*: Método encargado de generar una red de Bayes para cada conflicto de la red de Petri. Aquí se generan los nodos de la red de Bayes, las tablas de los mismos indicando cuantos valores posibles tiene cada uno, y los arcos que representan la relación entre estos nodos. Como se requiere una red de Bayes para cada conflicto estas se almacenan en un arreglo llamado BN_list para facilitar su manejo.

- *read_log*: Método dedicado a recolectar la información generada en la sección 3.6.3.2. Los datos de estas experiencias permitirán calcular las probabilidades a priori y condicionales para llenar las tablas de las redes de Bayes.

- *update_tables*: Método donde se calculan las probabilidades a priori y condicionales para rellenar las tablas de cada nodo de la red de Bayes.

- *politica_bayesiana*: Método encargado de obtener la probabilidades posterior del conflicto a resolver a partir de la red de Bayes del mismo y con estos valores asignar una probabilidad de ser disparada a cada transición del conflicto. Finalmente utilizando estos valores como distribución de probabilidad se escoge la acción a tomar.

3.6.4. Test realizados

En esta sección se realizan diversos tests sobre distintas redes para corroborar si el funcionamiento del algoritmo se adecua a lo esperado. Se muestran los tests realizados sobre las redes Canal de Panama y Guanjun, el resto de las pruebas pueden encontrarse en Anexo 2: Tests iteración n°4.

3.6.4.1 Red 1 - Canal de Panamá

Analizando la estructura de la red y el cumplimiento de las condiciones definidas, se espera que el sistema priorice las siguientes transiciones:

- En el conflicto $K^1 < P19\{T1, T9\} >$, priorizar a T1 (por mayor uso de recursos).
- En el conflicto $K^2 < P20\{T3, T11\} >$, valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^3 < P21\{T5, T13\} >$, priorizar a T13 (por mayor uso de recursos).
- En el conflicto $K^4 < P22\{T1, T13\} >$, valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^5 < P23\{T1, T13\} >$, valores aproximadamente iguales entre ambas transiciones.

3.6.4.1.1 Test 1 - Red de Bayes sin estado

En los cuadros siguientes se pueden observar los resultados obtenidos con esta red de Bayes:

Ejecución	T1	T9
1	0,5247	0,4753
2	0,5247	0,4753
3	0,5247	0,4753
4	0,5247	0,4753
5	0,5247	0,4753
6	0,5247	0,4753
7	0,5247	0,4753
8	0,5247	0,4753
9	0,5247	0,4753
10	0,5247	0,4753

Cuadro 6: Conflicto 1

Ejecución	T3	T11
1	0,5179	0,4821
2	0,5179	0,4821
3	0,5179	0,4821
4	0,5179	0,4821
5	0,5179	0,4821
6	0,5179	0,4821
7	0,5179	0,4821
8	0,5179	0,4821
9	0,5179	0,4821
10	0,5179	0,4821

Cuadro 7: Conflicto 2

Ejecución	T5	T13
1	0,4585	0,5415
2	0,4585	0,5415
3	0,4585	0,5415
4	0,4585	0,5415
5	0,4585	0,5415
6	0,4585	0,5415
7	0,4585	0,5415
8	0,4585	0,5415
9	0,4585	0,5415
10	0,4585	0,5415

Cuadro 8: Conflicto 3

Ejecución	T1	T13
1	0,5283	0,4717
2	0,5283	0,4717
3	0,5283	0,4717
4	0,5283	0,4717
5	0,5283	0,4717
6	0,5283	0,4717
7	0,5283	0,4717
8	0,5283	0,4717
9	0,5283	0,4717
10	0,5283	0,4717

Cuadro 9: Conflicto 4

Ejecución	T1	T13
1	0,4423	0,5577
2	0,4423	0,5577
3	0,4423	0,5577
4	0,4423	0,5577
5	0,4423	0,5577
6	0,4423	0,5577
7	0,4423	0,5577
8	0,4423	0,5577
9	0,4423	0,5577
10	0,4423	0,5577

Cuadro 10: Conflicto 5

3.6.4.1.2 Test 2 - Red de Bayes con estado

En los cuadros siguientes se pueden observar los resultados obtenidos con esta red de Bayes:

Ejecución	T1	T9
1	0,5455	0,4545
2	0,5455	0,4545
3	0,5455	0,4545
4	0,5455	0,4545
5	0,5455	0,4545
6	0,5455	0,4545
7	0,5455	0,4545
8	0,5455	0,4545
9	0,5455	0,4545
10	0,5455	0,4545

Cuadro 11: Conflicto 1

Ejecución	T3	T11
1	0,4936	0,5064
2	0,4936	0,5064
3	0,4936	0,5064
4	0,4936	0,5064
5	0,4936	0,5064
6	0,4936	0,5064
7	0,4936	0,5064
8	0,4936	0,5064
9	0,4936	0,5064
10	0,4936	0,5064

Cuadro 12: Conflicto 2

Ejecución	T5	T13
1	0,4978	0,5022
2	0,4978	0,5022
3	0,4978	0,5022
4	0,4978	0,5022
5	0,4978	0,5022
6	0,4978	0,5022
7	0,4978	0,5022
8	0,4978	0,5022
9	0,4978	0,5022
10	0,4978	0,5022

Cuadro 13: Conflicto 3

Ejecución	T1	T13
1	0,4875	0,5125
2	0,4875	0,5125
3	0,4875	0,5125
4	0,4875	0,5125
5	0,4875	0,5125
6	0,4875	0,5125
7	0,4875	0,5125
8	0,4875	0,5125
9	0,4875	0,5125
10	0,4875	0,5125

Cuadro 14: Conflicto 4

Ejecución	T1	T13
1	0,4497	0,5503
2	0,4497	0,5503
3	0,4497	0,5503
4	0,4497	0,5503
5	0,4497	0,5503
6	0,4497	0,5503
7	0,4497	0,5503
8	0,4497	0,5503
9	0,4497	0,5503
10	0,4497	0,5503

Cuadro 15: Conflicto 5

3.6.4.1.3 Conclusiones de los tests

Analizando los resultados de la red de Bayes sin estado, podemos observar que los mismos se corresponden con lo esperado, a excepción de los conflictos 4 y 5 que se prioriza una transición por sobre la otra cuando son simétricas respecto de las condiciones. Además se observa que para el conflicto 1, si bien la transición con mayor probabilidad es la esperada, los valores de probabilidad de ambas transiciones del conflicto son cercanos al 0.5 para cada una, por lo que no se puede afirmar que el sistema de priorice realmente a una por sobre la otra. Esto se debe a que el sesgo inicial tiende a priorizar en situaciones como la descripta, aquel lazo con el que realiza la primer ejecución, por lo que al recolectar los datos iniciales para la carga de las tablas de la red de Bayes, la transición perteneciente al lazo favorecido presenta una mayor probabilidad a priori. Para solucionar este problema se propone añadir al sistema el control de invariantes de transición en la próxima iteración, lo que permitirá tener un mayor control sobre la cantidad de veces que es sistema recorre cada lazo.

La de red de Bayes con estado presenta mejores resultados para los conflictos 1 y 4, sin embargo en los conflictos 3 y 5 se presenta el mismo problema que se detallo en el párrafo anterior.

3.6.4.2 Red 2 - Guanjun

Analizando la estructura de la red y el cumplimiento de las condiciones definidas, se espera que el sistema priorice las siguientes transiciones:

- En el conflicto $K^1 < P1\{T2, T4\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^2 < P14\{T4, T9, T13\} >$, priorizar a T13 y luego a T4.
- En el conflicto $K^3 < P15\{T5, T7, T10, T12\} >$, priorizar mayormente a T10, siendo T12 el más desfavorecido.
- En el conflicto $K^4 < P19\{T1, T9, T12\} >$, priorizar a T1 y luego a T9, siendo T12 el más desfavorecido.
- En el conflicto $K^5 < P20\{T1, T9, T12\} >$, priorizar a T1 y luego a T9, siendo T12 el más desfavorecido.

3.6.4.2.1 Test 1 - Red de Bayes sin estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T2	T4
1	0,5302	0,4698
2	0,5302	0,4698
3	0,5302	0,4698
4	0,5302	0,4698
5	0,5302	0,4698
6	0,5302	0,4698
7	0,5302	0,4698
8	0,5302	0,4698
9	0,5302	0,4698
10	0,5302	0,4698

Cuadro 16: Conflicto 1

Ejecución	T4	T9	T13
1	0,1900	0,3662	0,4468
2	0,1900	0,3662	0,4468
3	0,1900	0,3662	0,4468
4	0,1900	0,3662	0,4468
5	0,1900	0,3662	0,4468
6	0,1900	0,3662	0,4468
7	0,1900	0,3662	0,4468
8	0,1900	0,3662	0,4468
9	0,1900	0,3662	0,4468
10	0,1900	0,3662	0,4468

Cuadro 17: Conflicto 2

Ejecución	T5	T7	T10	T12
1	0,1506	0,1494	0,3568	0,3433
2	0,1506	0,1494	0,3568	0,3433
3	0,1506	0,1494	0,3568	0,3433
4	0,1506	0,1494	0,3568	0,3433
5	0,1506	0,1494	0,3568	0,3433
6	0,1506	0,1494	0,3568	0,3433
7	0,1506	0,1494	0,3568	0,3433
8	0,1506	0,1494	0,3568	0,3433
9	0,1506	0,1494	0,3568	0,3433
10	0,1506	0,1494	0,3568	0,3433

Cuadro 18: Conflicto 3

Ejecución	T1	T9	T12
1	0,2995	0,2843	0,4162
2	0,2995	0,2843	0,4162
3	0,2995	0,2843	0,4162
4	0,2995	0,2843	0,4162
5	0,2995	0,2843	0,4162
6	0,2995	0,2843	0,4162
7	0,2995	0,2843	0,4162
8	0,2995	0,2843	0,4162
9	0,2995	0,2843	0,4162
10	0,2995	0,2843	0,4162

Cuadro 19: Conflicto 4

Ejecución	T1	T9	T12
1	0,3877	0,2863	0,3260
2	0,3877	0,2863	0,3260
3	0,3877	0,2863	0,3260
4	0,3877	0,2863	0,3260
5	0,3877	0,2863	0,3260
6	0,3877	0,2863	0,3260
7	0,3877	0,2863	0,3260
8	0,3877	0,2863	0,3260
9	0,3877	0,2863	0,3260
10	0,3877	0,2863	0,3260

Cuadro 20: Conflicto 5

3.6.4.2.2 Test 2 - Red de Bayes con estado

Ejecución	T2	T4
1	0,5340	0,4660
2	0,5340	0,4660
3	0,5340	0,4660
4	0,5340	0,4660
5	0,5340	0,4660
6	0,5340	0,4660
7	0,5340	0,4660
8	0,5340	0,4660
9	0,5340	0,4660
10	0,5340	0,4660

Cuadro 21: Conflicto 1

Ejecución	T4	T9	T13
1	0,1857	0,3597	0,4546
2	0,1857	0,3597	0,4546
3	0,1857	0,3597	0,4546
4	0,1857	0,3597	0,4546
5	0,1857	0,3597	0,4546
6	0,1857	0,3597	0,4546
7	0,1857	0,3597	0,4546
8	0,1857	0,3597	0,4546
9	0,1857	0,3597	0,4546
10	0,1857	0,3597	0,4546

Cuadro 22: Conflicto 2

Ejecución	T5	T7	T10	T12
1	0,1541	0,1490	0,3633	0,3336
2	0,1541	0,1490	0,3633	0,3336
3	0,1541	0,1490	0,3633	0,3336
4	0,1541	0,1490	0,3633	0,3336
5	0,1541	0,1490	0,3633	0,3336
6	0,1541	0,1490	0,3633	0,3336
7	0,1541	0,1490	0,3633	0,3336
8	0,1541	0,1490	0,3633	0,3336
9	0,1541	0,1490	0,3633	0,3336
10	0,1541	0,1490	0,3633	0,3336

Cuadro 23: Conflicto 3

Ejecución	T1	T9	T12
1	0,3461	0,3032	0,3507
2	0,3461	0,3032	0,3507
3	0,3461	0,3032	0,3507
4	0,3461	0,3032	0,3507
5	0,3461	0,3032	0,3507
6	0,3461	0,3032	0,3507
7	0,3461	0,3032	0,3507
8	0,3461	0,3032	0,3507
9	0,3461	0,3032	0,3507
10	0,3461	0,3032	0,3507

Cuadro 24: Conflicto 4

Ejecución	T1	T9	T12
1	0,3946	0,3069	0,2986
2	0,3946	0,3069	0,2986
3	0,3946	0,3069	0,2986
4	0,3946	0,3069	0,2986
5	0,3946	0,3069	0,2986
6	0,3946	0,3069	0,2986
7	0,3946	0,3069	0,2986
8	0,3946	0,3069	0,2986
9	0,3946	0,3069	0,2986
10	0,3946	0,3069	0,2986

Cuadro 25: Conflicto 5

3.6.4.2.3 Conclusiones de los tests

Observando los resultados obtenidos a lo largo de estos tests para ambas redes de Bayes, se puede determinar que el funcionamiento del algoritmo está lejos de ser el esperado. Al analizar el código del proyecto se detectó que la implementación de la política de selección de la transición a disparar y el cálculo de la probabilidad a posterior necesaria para esta tarea no otorga los resultados esperados por lo que se propone en una futura iteración repensar la manera en que estas se calculan, haciendo uso el cálculo de posteriores conjuntas que otorga la herramienta de redes Bayesianas utilizada en el proyecto.

3.6.5. Conclusiones

Se concluye que fue posible generar un sistema capaz de utilizar redes Bayesianas para la resolución de conflictos en una red de Petri y generar dinámicamente la política de disparo de la misma. Sin embargo los resultados de los test dan muestra de que el sistema no prioriza en los casos de conflicto la transición que se espera según las condiciones de recompensas planteadas. Como se destacó en las conclusiones de los test, para mitigar este problema se proponen dos soluciones:

- Añadir al sistema el control de invariantes de transición: esto implica añadir al sistema la capacidad de soportar la especificación de requerimientos por invariante de transición, contabilizar la cantidad de invariantes realizados, y controlar que la cantidad invariantes completados se ajuste a los requerimientos.
- Modificar el algoritmo de cálculo de la probabilidad a posterior de las transiciones del conflicto: Se propone modificar esta sección del código para que la probabilidad a posterior de las transiciones del conflicto se calculen haciendo uso el calculo de posteriores conjuntas que otorga la herramienta de redes Bayesianas, utilizada en el proyecto. Este cambio conllevaría a generar un nuevo modelo Bayesiano, es por esto que no se tratará en forma inmediata, sino que se retomará este problema y su consiguiente solución luego de incorporar las requerimientos de invariantes al sistema.

3.7. Iteración 5

3.7.1. Introducción

En esta etapa se incorporará el análisis de invariantes al control del sistema. Recordemos que completar un invariante representa culminar una actividad o proceso productivo de principio a fin. Esto es útil, porque el control relativo entre invariantes permite determinar cuántas veces se realiza una actividad respecto del total realizado.

3.7.2. Objetivos

1. Implementar una lógica para la especificación de requerimientos por invariante de transición.
2. Implementar una lógica para contabilizar la cantidad de invariantes realizados.
3. Implementar lógica de control para que la cantidad invariantes completados se ajuste a los requerimientos.
4. Implementar una lógica para determinar si el sistema converge o no a los requerimientos especificados.

3.7.3. Desarrollo

3.7.3.1 Contabilizar invariantes

Para llevar el recuento de invariantes, se creó un vector de contadores n_I de dimensión M, donde M es la cantidad de invariantes, donde $n_I[i]$ contiene el número de veces que se realizó el invariante i hasta el momento. A su vez, se optó por crear un método que al momento de la creación de la red genere en tiempo de ejecución una expresión regular que represente al conjunto de invariantes posibles para dicha red. Sumado a lo anterior, se procedió a generar un log con las transiciones disparadas comprobando al disparar, si se completó o no un invariante y así actualizando los contadores.

3.7.3.2 Control de los invariantes

En esta iteración se busca controlar a los invariantes a través de las situaciones de conflicto, a partir la idea de que para controlar a estos primeros es suficiente con controlar los conflictos pertenecientes a dichos invariantes i.e. sea un conflicto $K_i = \{P_x, T_y, T_z\}$ donde T_y pertenece al invariante A y T_z al invariante B, si se modifican las probabilidades a priori de elegir T_z sobre T_y cuando K_i esté efectivo, habrá más probabilidad de completar al invariante A respecto del invariante B.

Luego, la lógica de decisión utilizando las redes bayesianas de la iteración anterior, es la siguiente:

- Si en el estado actual hay un conflicto efectivo, se resuelve el conflicto utilizando la red de bayes.
- Si hay más de un conflicto efectivo, se elige uno al azar.
- Si no hay conflicto efectivo, se elige usando las probabilidades de la *tabla_Q*.

Respecto a cómo o cuánto modificar las tablas de probabilidades a priori, empleamos un delta. Si pensáramos el sistema con un sistema de control tradicional, el delta busca representar la idea de error. Dicho valor, se calculará de distintas maneras para analizar los diferentes comportamientos del sistema. Estas, son las que se presentan a continuación:

- $\Delta = REQ - ultimos_invariantes_realizados$
- $\Delta = REQ - invariantes_realizados + 10 * REQ - ultimos_invariantes_realizados$
- $\Delta = (REQ - ultimos_invariantes_realizados) * (1 - e^{-\alpha * |REQ - invariantes_realizados|})$

Donde:

- REQ: Vector de requerimientos de invariantes. Cada componente iésimo representa el requerimiento relativo del invariante iésimo, e.g. (0.25,0.35,0.40) representa que se pretende que del total de invariantes, el 25 % sean del invariante 0, el 35 % sean del invariante 1 y el 40 % del invariante 2. Este sería un caso posible de requerimientos de un sistema con 3 invariantes.
- *invariantes_realizados*: Vector de los invariantes realizados al momento. Se expresa de igual forma que el vector *REQ*
- *últimos_invariantes_realizados*: Vector que considera los últimos N disparos. Divide esos N disparos en ventanas de tamaño M ponderando con más peso a las ventanas correspondientes a los disparos más recientemente realizados. La explicación detallada de este punto se encuentra en
- $1 - e^{-\alpha * |REQ - invariantes_realizados|}$: Función exponencial de ponderación. Sirve como solución a sistemas diferenciales de primer orden de forma
$$v[i] = 1 - e^{(-\alpha * (|x_i - r_i|))}$$
Donde $v[i]$ es el valor de de la invariante i del vector de ponderación del error, α la pendiente, x el valor obtenido del contador de la invariante i, y r el valor especificado como requerimiento para esa invariante. La explicación de este punto se encuentra en

3.7.4. Tests realizados

Con el objetivo de ver si ahora la red converge y ver el impacto entre los distintos tests utilizados, se muestran a continuación los resultados de ejecuciones en las redes de Panamá, Huang y Guanjún. Las demás de las pruebas pueden encontrarse en Anexo 3: Tests iteración n°5.

3.7.4.1 Red 1: Canal de Panamá

3.7.4.1.1 Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 15: Contador invariante 1.

Figura 16: Contador invariante 2.

3.7.4.1.2 Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 17: Contador invariante 1.

Figura 18: Contador invariante 2.

3.7.4.1.3 Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 19: Contador invariante 1.

Figura 20: Contador invariante 2.

3.7.4.2 Red 2: Huang

3.7.4.2.1 Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 21: Contador invariante 1.

Figura 22: Contador invariante 2.

Figura 23: Contador invariante 3.

3.7.4.2.2 Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 24: Contador invariante 1.

Figura 25: Contador invariante 2.

Figura 26: Contador invariante 3.

3.7.4.2.3 Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 27: Contador invariante 1.

Figura 28: Contador invariante 2.

Figura 29: Contador invariante 3.

3.7.4.3 Red 3: Guanjun

3.7.4.3.1 Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 30: Contador invariante 1.

Figura 31: Contador invariante 2.

Figura 32: Contador invariante 3.

Figura 33: Contador invariante 4.

3.7.4.3.2 Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 34: Contador invariante 1.

Figura 35: Contador invariante 2.

Figura 36: Contador invariante 3.

Figura 37: Contador invariante 4.

3.7.4.3.3 Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 38: Contador invariante 1.

Figura 39: Contador invariante 2.

Figura 40: Contador invariante 3.

Figura 41: Contador invariante 4.

3.7.5. Primeras conclusiones

Como resultado de los primeros tests se concluye que las redes convergen mejor al considerar de alguna forma los valores histórico, sin embargo dependiendo de la red hay más facilidad o no para converger a ciertos requerimientos. Se observa que Panamá y Hospital convergen con precisión dentro de ciertos márgenes aceptables, pero otras, como Huang y Guanjun, presentan márgenes muy acotados de convergencia. Estudiaremos a continuación, los casos de no convergencia con el objetivo de intentar entender los motivos.

3.7.6. Casos límites y/o situaciones de no convergencia

Llamamos casos límites, a aquellos que llevan a los requerimientos a valores cercanos a 0 o a 1 para determinados requerimientos. Estos nos dan la pauta de hasta dónde es posible controlar el comportamiento de la red de petri con el sistema que estamos implementando. Buscamos esta información en aquellas redes en que, según lo visto en el punto anterior, convergen con más flexibilidad.

Figura 42: Contador invariante 1.

Figura 43: Contador invariante 2.

Hablamos de situaciones de no convergencia a aquellos valores donde alguna red no converge, aunque la lógica nos indica que debería ser algo viable. Es el caso del siguiente ejemplo.

Figura 44: Contador invariante 1.

Figura 45: Contador invariante 2.

Figura 46: Contador invariante 3.

3.7.7. Análisis de los casos

Con objetivo de entender lo que está pasando, se realizaron y analizaron gráficos de las probabilidades a priori de los confictos en estas situaciones. Recordemos que el método de control consiste en modificar dichas tablas con las que las redes de Bayes eligen qué transición disparar en conflicto efectivo. Los siguientes resultados corresponden al caso límite de la red de Panamá mostrados previamente:

Figura 47: Porcentaje de la transición correspondiente al invariante del 95 %

Figura 48: Porcentaje de la transición correspondiente al invariante del 5 %

Como se puede observar, aun llevando las probabilidades del conflicto a (1.0,0) donde 1.0 es la probabilidad a priori de la transición correspondiente al invariante que se quiere llevar a 95 %, es imposible lograr que el invariante supere el 75 %. Otro dato a considerar es que para el caso anterior las redes de bayes (nuestro sistema de control) toma decisiones sobre el 17.4 % de los disparos. Esto lleva a pensar que muy probablemente el sistema esté permitiendo realizar los invariantes que no son de interés, mientras los conflictos no están efectivos, es decir, donde no se tiene control.

Esto nos lleva a resolver que el solo control de conflictos efectivos, es insuficiente para controlar el sistema en su totalidad.

3.7.8. Conclusiones

Se concluye que se pudo realizar un control de invariantes basado en los conflictos. Sin embargo, esto funciona dentro de rangos acotados y nunca sobre extremos. Esto último se atribuye a priori porque en los casos donde un conflicto no se encuentre efectivo, la *tabla_Q* puede decidir disparar las transiciones del invariante no deseado. Dado que el control planteado hasta el momento no es suficiente, en la próxima iteración se buscará extender el control de invariantes sobre la totalidad de los estados y transiciones. Esto produce un cambio respecto de las iteraciones mencionadas en primera instancia. A continuación se lista el nuevo itinerario:

- *Iteración 6*: Extensión del control sobre todas los estados, no sólo casos de conflicto.
    1. Implementar la lógica que permita extender el control sobre todos los estados, no solo el conflicto.
    2. Realizar análisis de los resultados obtenidos respecto de la iteración anterior.

- *Iteración 7*: Análisis del comportamiento del transitorio.
    1. Implementar la lógica que permita obtener los parámetros de las gráficas de convergencia vistos como lazos de sistemas de control.
    2. Análisis del comportamiento de las gráficas al modificar parámetros del sistema o incorporar nuevos recursos.

3.8. Iteración 6

3.8.1. Introducción

En la etapa anterior se pudo observar que el control aplicado sólo a los conflictos no puede garantizar el cumplimiento de los requerimientos, ya que hay transiciones pertenecientes tanto a un conflicto como al invariante que no se pretende realizar, que pueden ser disparados en los momentos donde el conflicto no esté efectivo.

Con esto en mente, en esta etapa se buscará extender el control del sistema no solo a los casos de conflictos efectivos, si no a todas las transiciones de la red y a la totalidad de sus estados.

3.8.2. Objetivos

1. Implementar la lógica que permita extender el control sobre todos los estados, no solo en situaciones de conflicto efectivo.

2. Realizar análisis de los resultados obtenidos respecto de la iteración anterior.

3.8.3. Desarrollo

Al momento de extender el control del sistema a todos los estados de la red de Petri se tuvieron en cuenta distintos enfoques:

- Aplicar el control directamente sobre la tabla Q ya sea a la tabla sesgada o a la que genera el agente luego de la etapa de aprendizaje.

- Extender el control de todas las transiciones a una nueva red bayesiana. Este punto se desarrollará en la siguiente iteración

3.8.3.1 Control sobre tabla Q

Este primer enfoque consistió en utilizar el delta para controlar directamente sobre la *tabla_Q* generada por el agente. Para ello, se utilizó un vector fix_array de dimensión M (cantidad de transiciones de la red de petri) inicializado en ceros que se actualiza cada una cantidad de disparos parametrizable. La forma de actualización es la siguiente:

$$fix_array_{t+1} = fix_array_t + \sum_{i=0}^{I-1} invs_array_i * \Delta * gain \tag{12}$$

Donde:

- I : Cantidad de invariantes

- $invs_array$: Matriz de dimensión IxM, donde $invs_array_i$ es un vector fila de dimensión $1xM$ que indica en su posición j la cantidad de veces que se tiene que disparar la transición j para completar el invariante i. En otras palabras, si el vector fuera de la forma [0,0,1,0,1,1,0], nos indicaría que para completar ese invariante, hay que disparar las transiciones 2, 4 y 5 una vez. Aquellas cuyo valor es 0, no pertenecen al invariante.

- $gain$: Ganancia, constante parametrizable.

- $delta$: Muestra qué tan alejado está un invariante del requerimiento. En esta iteración usamos 4 posibles definiciones:

  - $\Delta = REQ - invariantes_realizados$
  - $\Delta = REQ - ultimos_invariantes_realizados$
  - $\Delta = REQ - invariantes_realizados + 10 * REQ - ultimos_invariantes_realizados$
  - $\Delta = (REQ - invariantes_realizados) * (1 - e^{-\alpha * |REQ - invariantes_realizados|})$

La explicación de los términos que arriba se ubican se encuentran en la sección 3.7.3.2. Notar que en la última ecuación, se utiliza en la resta *invariantes_realizados* en vez de *ultimos_invariantes_realizados* como se realizaba en dicha iteración.

Luego, al momento de decidir qué acción tomar, se usa como distribución de probabilidad:

$$p = tabla_Q[estado_actual] - fix_array \tag{13}$$

Cabe destacar que sobre lo anterior se toman las siguientes consideraciones:

- Solo se aplica la resta a los términos en los que $tabla_Q[estado_actual][i] > 0$, es decir, a las transiciones sensibilizadas en el estado actual.
- Si $p[i] < 0$ entonces $p[i]$ se lleva a cero (no pueden existir probabilidades negativas).
- Si $\sum p[i] = 0$ entonces $p = tabla_Q[estado_actual]$. Esto se hace para evitar la división por cero al normalizar. Se consideró hacer a p equiprobable sobre las transiciones sensibilizadas, pero esto condujo a peores resultados.
- Finalmente se normaliza el vector, para que pueda ser visto como una distribución de probabilidad

3.8.4. Tests realizados

Con el objetivo de ver si ahora la red converge y ver el impacto entre los distintos tests utilizados, se muestran a continuación los resultados de ejecuciones en las redes de Panamá, Huang y Guanjún, utilizando como requerimientos aquellos que más complicaciones mostraron. El resto de las pruebas pueden encontrarse en Anexo 4: Tests iteración n°6.

3.8.4.1 Red 1: Canal de Panamá

3.8.4.1.1 Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 49: Contador invariante 1.

Figura 50: Contador invariante 2.

3.8.4.1.2 Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 51: Contador invariante 1.

Figura 52: Contador invariante 2.

3.8.4.1.3 Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 53: Contador invariante 1.

Figura 54: Contador invariante 2.

3.8.4.1.4 Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 55: Contador invariante 1.

Figura 56: Contador invariante 2.

3.8.4.2 Red 2: Huang

3.8.4.2.1 Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 57: Contador invariante 1.

Figura 58: Contador invariante 2.

Figura 59: Contador invariante 3.

3.8.4.2.2 Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 60: Contador invariante 1.

Figura 61: Contador invariante 2.

Figura 62: Contador invariante 3.

3.8.4.2.3 Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 63: Contador invariante 1.

Figura 64: Contador invariante 2.

Figura 65: Contador invariante 3.

3.8.4.2.4 Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 66: Contador invariante 1.

Figura 67: Contador invariante 2.

Figura 68: Contador invariante 3.

Al igual que para la red anterior, el sistema converge sin problemas a los valores solicitados así como también, no presenta margenes de requerimientos a los que no pueda converger.

3.8.4.3 Red 3: Guanjun

3.8.4.3.1 Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 69: Contador invariante 1.

Figura 70: Contador invariante 2.

Figura 71: Contador invariante 3.

Figura 72: Contador invariante 4.

3.8.4.3.2 Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 73: Contador invariante 1.

Figura 74: Contador invariante 2.

Figura 75: Contador invariante 3.

Figura 76: Contador invariante 4.

3.8.4.3.3 Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 77: Contador invariante 1.

Figura 78: Contador invariante 2.

Figura 79: Contador invariante 3.

Figura 80: Contador invariante 4.

3.8.4.3.4 Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 81: Contador invariante 1.

Figura 82: Contador invariante 2.

Figura 83: Contador invariante 3.

Figura 84: Contador invariante 4.

3.8.5. Conclusiones

Observamos que en los casos en que el sistema considera el valor histórico converge sin problemas a los valores solicitados. Cabe aclarar que no presenta margenes de requerimientos a los que no pueda converger. Se destaca que al realimentar con una función exponencial de ponderación la curva de invariantes converge de manera mas suave, presentando menores sobrepasamientos. En la iteración siguiente, veremos si es posible lograr aplicar estas conclusiones a una nueva red bayesiana.

3.9. Iteración 7

3.9.1. Introducción

En esta iteración se extenderá el control con redes Bayesianas a todos los disparos de la red de Petri por lo que sera necesario diseñar una nueva red de Bayes capaz de representar a todos los movimientos del sistema y contrastar los resultados obtenidos con este nuevo sistema contra los resultados obtenidos de la iteración anterior a fin de poder comprobar si de esta manera obtenemos un mejor rendimiento.

3.9.2. Objetivos

1. Implementar la lógica que permita extender el control sobre todos los estados, no solo el conflicto.
2. Realizar análisis de los resultados obtenidos respecto de la iteración anterior.

3.9.3. Desarrollo

3.9.3.1 Nueva red de Bayes

Para poder extender el control a todos los disparos de transiciones de una Red de Petri fue necesario diseñar una nueva red de Bayes capaz de contemplar todas las transiciones de la red en un mismo conjunto y no solo aquellas que pertenecían a un conflicto y que sea capaz de analizar al momento de escoger la transición a disparar, a que invariante pertenecen cada una de ellas y escogiéndolas con el objetivo de cumplir los requerimientos del usuario así como también tener en cuenta cual de las transiciones disparadas tiene mayor probabilidad de ser la que otorgue mayor recompensa.

El diseño obtenido se puede observar en la figura85

Figura 85: Red de Bayes para controlar todos los disparos.

La lógica de este nuevo diseño es que para la elección de la transición a disparar se debe tener en cuenta la probabilidad de las transiciones de cumplir con las condiciones establecidas (brindar un mayor recompensa), el estado actual de la red (transiciones sensibilizadas) y los requerimientos del usuario (invariantes).

El nodo State representa los posibles estados de la red de Petri. Esta sera nuestra variable observable y en cada elección de una nueva transición a disparar el valor de este nodo sera un valor "evidencia" ya que podemos conocer el estado de la Red de Petri en cada nuevo disparo.

El nodo Reqs reflejara la distancia entre el requerimiento del usuario y el porcentaje de cada invariante completados hasta el momento. Su tabla de probabilidad sera de tamaño $1 * I$ siendo I en numero de invariantes de la Red de Petri. Estos valores se actualizaran cada un cierto numero de disparos y se analizaran diversas ecuaciones de realimentación a fin de comprobar cual brinda un mejor resultado.

El nodo Cond representa la probabilidad de que se cumpla cada una de las condiciones establecidas en secciones anteriores. Su tabla de probabilidad sera de tamaño $1*C$ siendo C el numero de condiciones contempladas en la ejecución.

Finalmente el nodo Choice es el que representa la elección de la transición a disparar y que combina en una sola tabla la información brindada por los demás nodos padres. Su tabla de probabilidad por lo tanto sera la que refleja para cada transición, en función del estado de la red, el invariante al que pertenece cada transición y la probabilidad de que se cumplan cada una de las condiciones, su probabilidad de ser disparada. La probabilidad posterior conjunta del nodo Choice sera la que devolverá

el vector con la probabilidad de elección de las transiciones en cada disparo y que sera utilizado por la política.

3.9.4. Test Realizados

En esta sección se llevara a cabo un análisis sobre las diferentes ecuaciones de realimentación para el cálculo del valor de ajuste del nodo de requerimientos con el propósito de encontrar la que genere que el sistema logre converger de la manera mas satisfactoria.

Se evaluaran dos ecuaciones:

- $\Delta = REQ - invariantes_realizados$
- $\Delta = (REQ - invariantes_realizados) * (1 - e^{-\alpha * |REQ - invariantes_realizados|})$

La explicación de los términos que arriba se ubican se encuentran en la sección 3.7.3.2. Notar que al igual que en la iteración anterior, en la última ecuación se utiliza en la resta *invariantes_realizados* en vez de *ultimos_invariantes_realizados* como se realizaba en la sección 3.7.3.2.

Se muestran algunas de las pruebas realizadas, las restantes se encuentran en Anexo 5: Tests iteracion n°6.

3.9.4.1 Red 1: Canal de Panamá

3.9.4.1.1 Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 86: Contador invariante 1.

Figura 87: Contador invariante 2.

3.9.4.1.2 Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 88: Contador invariante 1.

Figura 89: Contador invariante 2.

3.9.4.2 Red 2: Huang

3.9.4.2.1 Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 90: Contador invariante 1.

Figura 91: Contador invariante 2.

Figura 92: Contador invariante 3.

3.9.4.2.2 Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 93: Contador invariante 1.

Figura 94: Contador invariante 2.

Figura 95: Contador invariante 3.

3.9.4.3 Red 3: Guanjun

3.9.4.3.1 Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 96: Contador invariante 1.

Figura 97: Contador invariante 2.

Figura 98: Contador invariante 3.

Figura 99: Contador invariante 4.

3.9.4.3.2 Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 100: Contador invariante 1.

Figura 101: Contador invariante 2.

Figura 102: Contador invariante 3.

Figura 103: Contador invariante 4.

3.9.4.4 Conclusiones de los test

Al analizar los resultados obtenidos se puede concluir que si bien ambos métodos de realimentación llevan al sistema a converger de manera satisfactoria y sin presentar valores de requerimientos no alcanzables, es el método de realimentación con función exponencial de ponderación el que produce curvas con menores sobrepasamientos y mayor robustez al ruido. Seria importante definir de manera óptima los parámetros de función la exponencial a fin de obtener los mejores resultados posibles.

3.9.5. Conclusiones

Se concluye de la presente iteración que extender el control a todos los disparos del sistema permite que el sistema pueda converger a los valores solicitados por el usuario con mayor velocidad, con menores sobrepasamientos, y obteniendo así una curva mas suave y un mejor rendimiento del sistema, sin incrementar de manera desfavorable el costo de implementación. También de forma experimental se pudo observar que, al igual que en la iteración anterior, es el método de realimentación con función exponencial de ponderación el que brinda mejores resultados, por lo que se propone incorporar una nueva iteración al proyecto con el objetivo de encontrar los parámetros de la función exponencial que brindan el mejor desempeño, y comparar el sistema con la nueva red de Bayes. Esto produce un cambio respecto de las iteraciones mencionadas en primera instancia. A continuación se lista el nuevo itinerario:

- *Iteración 8*: Comparativa entre los sistemas utilizando conceptos de teoría de control y búsqueda de parámetros óptimos.
    1. Implementar la lógica que permita obtener los parámetros de las gráficas de convergencia vistos como lazos de sistemas de control.
    2. Implementar una lógica que automatice la ejecución de los sistemas considerando toda combinación posible de requerimientos analizando si los mismos convergen o no sin la necesidad de observar la gráfica
    3. Comparar la convergencia de los sistemas de las iteraciones anteriores.
    4. Análisis del comportamiento de las gráficas al modificar los parámetros de la función exponencial de ponderación.

3.10. Iteración 8

3.10.1. Introducción

En esta iteración se analizarán los gráficos de convergencia de las iteraciones [6] y [7] con el objetivo de obtener de los mismos los parámetros de la función visto como un sistema de control. Esto se hace con el fin de estudiar el comportamiento de dichos sistemas, analizar las zonas de convergencia en todo el conjunto de combinaciones de requerimientos, comparar los sistemas y observar cómo afecta la modificación de las variables del sistema para lograr mejores resultados.

3.10.2. Objetivos

1. Implementar la lógica que permita obtener los parámetros de las gráficas de convergencia vistos como lazos de sistemas de control.
2. Implementar una lógica que automatice la ejecución de los sistemas considerando toda combinación posible de requerimientos analizando si los mismos convergen o no sin la necesidad de observar la gráfica
3. Comparar la convergencia de los sistemas de las iteraciones anteriores.
4. Análisis del comportamiento de las gráficas al modificar los parámetros de la función exponencial de ponderación.

3.10.3. Desarrollo

3.10.3.1 Aplicando conceptos de teoría de control

Inferir comportamientos de una función observando la gráfica es relativamente simple, vemos si la gráfica se acerca en el tiempo a la región de convergencia y si parece que se va a mantener en esa región en el tiempo. Si queremos automatizar este proceso, tenemos que valernos de una forma de poder inferir esto mismo pero a través de datos. Para esto es que nos valemos de algunos conceptos de la teoría de control.

Las definiciones que vamos a tomar son las siguientes:

- *Región de convergencia*: Espacio comprendido por el requerimiento considerando el margen de error ($\pm 5\,\%$);
- *Tiempo de subida*: Momento en el que la gráfica alcanza el requerimiento por primera vez. Se expresa tanto en cantidad de batchs (recordemos que los batchs son las situaciones donde el sistema realiza el control) como en tiempo (en segundos).
- *Tiempo de transitorio*: Momento en el que la gráfica, luego de entrar en la región de convergencia, no vuelve a salir de la misma.
- *Convergencia*: El criterio de convergencia elegido para la automatización de los resultados consiste en considerar que la función converge si, dadas N transiciones, la gráfica tiene su tiempo de transitorio en una transición t tal que $t < N/2$.
- *Máxima sobre-elongación*: Cociente entre el máximo valor alcanzado por la curva y el requerimiento deseado.

Figura 104: Contador invariante 1.

Figura 105: Contador invariante 2.

3.10.3.2 Paneo al conjunto de requerimientos posibles

Luego, con estos datos numéricos y el criterio de convergencia anteriormente mencionado, se hace un paneo por las distintas combinaciones de requerimientos posibles. Utilizando el sistema de la iteración 6 con un delta exponencial considerando histórico, utilizando la red de Huang disparando 30000 transiciones, y un cambio de requerimientos cada 10 % se obtiene una grilla como la que se muestra a continuación. Notar que las primeras tres columnas refieren al requerimiento del invariante I y las siguientes tres al tiempo de transitorio asociado. Estos últimos, así como el tiempo de ejecución, se expresan en segundos.

I0	I1	I2	tt_0	tt_1	tt_2	converge	tiempo_ejecucion
0	0	1	0.61	0.61	1.15	SI	10.95
0	0.1	0.9	0.70	2.99	2.52	SI	11.16
0	0.2	0.8	0.76	1.87	1.39	SI	10.47
0	0.3	0.7	1.21	2.65	0.68	SI	11.43
0	0.4	0.6	0.99	2.99	0.41	SI	11.53
0	0.5	0.5	0.78	1.94	1.20	SI	11.65
0	0.6	0.4	0.88	1.58	0.30	SI	11.57
0	0.7	0.3	0.69	0.95	0.35	SI	12.40
0	0.8	0.2	0.89	1.18	0.56	SI	12.53
0	0.9	0.1	0.94	1.72	1.07	SI	11.66
0	1	0	0.78	1.55	0.85	SI	11.57
0.1	0	0.9	3.20	0.88	2.81	SI	10.88
0.1	0.1	0.8	0.56	0.62	0.84	SI	11.13
0.1	0.2	0.7	0.64	0.38	0.70	SI	11.38
0.1	0.3	0.6	1.07	0.20	0.99	SI	11.88
0.1	0.4	0.5	0.90	1.36	0.69	SI	12.76
0.1	0.5	0.4	1.04	1.04	0.23	SI	13.45
0.1	0.6	0.3	0.57	0.86	0.07	SI	13.73
0.1	0.7	0.2	0.63	0.93	0.63	SI	14.01
0.1	0.8	0.1	0.50	1.04	0.96	SI	14.18

I0	I1	I2	tt_0	tt_1	tt_2	converge	tiempo_ejecucion
0.1	0.9	0	0.56	1.00	0.86	SI	14.96
0.2	0	0.8	1.89	0.62	0.41	SI	11.08
0.2	0.1	0.7	0.34	0.75	0.75	SI	11.46
0.2	0.2	0.6	0.37	0.44	0.64	SI	11.48
0.2	0.3	0.5	0.75	0.13	0.97	SI	11.78
0.2	0.4	0.4	0.33	0.27	0.14	SI	11.46
0.2	0.5	0.3	1.04	0.97	0.27	SI	11.84
0.2	0.6	0.2	0.83	1.11	0.48	SI	12.16
0.2	0.7	0.1	0.86	1.29	0.79	SI	12.29
0.2	0.8	0	0.74	1.27	1.03	SI	13.20
0.3	0	0.7	2.27	0.91	1.44	SI	10.38
0.3	0.1	0.6	0.32	1.15	1.15	SI	11.82
0.3	0.2	0.5	0.13	1.12	1.23	SI	11.26
0.3	0.3	0.4	0.19	0.33	0.47	SI	11.48
0.3	0.4	0.3	0.07	1.27	0.07	SI	11.50
0.3	0.5	0.2	0.06	1.15	0.75	SI	11.57
0.3	0.6	0.1	0.70	1.05	0.50	SI	11.71
0.3	0.7	0	0.31	1.16	0.72	SI	12.09
0.4	0	0.6	1.74	0.91	1.12	SI	11.15
0.4	0.1	0.5	1.32	0.75	0.56	SI	12.32
0.4	0.2	0.4	0.68	0.89	0.20	SI	11.39
0.4	0.3	0.3	0.21	0.21	0.06	SI	10.96
0.4	0.4	0.2	0.42	0.53	0.99	SI	10.39
0.4	0.5	0.1	0.38	1.33	1.23	SI	11.36
0.4	0.6	0	0.39	0.47	0.84	SI	11.31
0.5	0	0.5	1.33	0.84	0.84	SI	11.10
0.5	0.1	0.4	0.48	2.62	0.13	SI	12.33
0.5	0.2	0.3	1.05	0.91	0.27	SI	11.79
0.5	0.3	0.2	1.09	0.13	0.75	SI	11.43
0.5	0.4	0.1	0.36	0.71	0.85	SI	11.28
0.5	0.5	0	0.89	1.69	0.82	SI	11.44
0.6	0	0.4	0.97	0.78	0.13	SI	11.12
0.6	0.1	0.3	0.92	0.84	0.07	SI	13.34
0.6	0.2	0.2	1.12	0.76	0.56	SI	11.85
0.6	0.3	0.1	1.03	0.22	0.76	SI	10.91
0.6	0.4	0	1.85	0.32	1.08	SI	11.12
0.7	0	0.3	1.33	1.00	0.14	SI	11.33
0.7	0.1	0.2	1.01	0.58	0.87	SI	13.74
0.7	0.2	0.1	1.13	0.42	0.71	SI	11.97
0.7	0.3	0	0.96	0.13	0.83	SI	11.37
0.8	0	0.2	1.10	0.83	0.55	SI	11.05
0.8	0.1	0.1	1.07	0.83	0.61	SI	14.19
0.8	0.2	0	1.50	0.74	0.81	SI	12.67
0.9	0	0.1	0.97	0.63	0.63	SI	11.36
0.9	0.1	0	1.27	0.81	0.73	SI	14.92
1	0	0	1.71	0.77	0.77	SI	11.28

Realizando el mismo procedimiento, pero con el sistema de la iteración 7 obtenemos:

I0	I1	I2	tt_0	tt_1	tt_2	converge	tiempo_ejecucion
0	0	1	0.22	0.22	0.22	SI	31.68
0	0.1	0.9	0.14	0.14	0.14	SI	22.43
0	0.2	0.8	0.14	0.43	0.43	SI	22.55

I0	I1	I2	tt_0	tt_1	tt_2	converge	tiempo_ejecucion
0	0.3	0.7	0.15	0.44	0.44	SI	22.64
0	0.4	0.6	0.14	0.60	0.60	SI	23.49
0	0.5	0.5	0.15	0.30	0.30	SI	22.72
0	0.6	0.4	0.14	0.45	0.45	SI	22.80
0	0.7	0.3	0.16	0.16	0.16	SI	23.25
0	0.8	0.2	0.14	0.30	0.30	SI	22.99
0	0.9	0.1	0.15	0.30	0.30	SI	23.03
0	1	0	0.15	0.15	0.15	SI	23.18
0.1	0	0.9	0.15	0.15	0.15	SI	23.04
0.1	0.1	0.8	0.16	0.16	0.32	SI	22.78
0.1	0.2	0.7	0.15	0.15	0.15	SI	22.65
0.1	0.3	0.6	0.29	0.42	0.14	SI	23.29
0.1	0.4	0.5	0.15	0.46	0.46	SI	23.32
0.1	0.5	0.4	0.15	0.47	0.47	SI	24.59
0.1	0.6	0.3	0.14	1.35	0.44	SI	25.24
0.1	0.7	0.2	1.90	0.60	0.30	SI	24.59
0.1	0.8	0.1	0.28	0.71	0.14	SI	24.67
0.1	0.9	0	0.29	0.29	0.15	SI	25.24
0.2	0	0.8	0.29	0.15	0.29	SI	22.13
0.2	0.1	0.7	0.31	0.15	0.31	SI	22.91
0.2	0.2	0.6	0.30	0.30	0.45	SI	22.56
0.2	0.3	0.5	0.29	0.72	0.72	SI	23.07
0.2	0.4	0.4	0.30	0.16	0.60	SI	22.92
0.2	0.5	0.3	1.03	0.44	0.30	SI	23.18
0.2	0.6	0.2	0.43	0.29	0.14	SI	23.30
0.2	0.7	0.1	1.06	0.48	0.32	SI	23.71
0.2	0.8	0	0.44	0.44	0.15	SI	23.78
0.3	0	0.7	0.14	0.14	0.14	SI	22.14
0.3	0.1	0.6	0.29	0.14	0.29	SI	23.00
0.3	0.2	0.5	0.44	0.29	1.61	SI	23.57
0.3	0.3	0.4	0.30	1.16	0.14	SI	22.63
0.3	0.4	0.3	0.30	0.74	0.45	SI	22.75
0.3	0.5	0.2	1.02	0.43	0.14	SI	22.86
0.3	0.6	0.1	0.29	0.45	0.15	SI	23.07
0.3	0.7	0	0.31	0.31	0.15	SI	23.47
0.4	0	0.6	0.29	0.14	0.29	SI	22.37
0.4	0.1	0.5	0.58	1.30	0.13	SI	23.18
0.4	0.2	0.4	0.30	0.15	0.30	SI	23.16
0.4	0.3	0.3	0.14	0.59	0.14	SI	22.80
0.4	0.4	0.2	0.19	0.19	0.19	SI	22.69
0.4	0.5	0.1	0.61	0.61	0.41	SI	24.36
0.4	0.6	0	3.13	3.13	0.15	SI	22.88
0.5	0	0.5	0.72	0.20	0.72	SI	22.10
0.5	0.1	0.4	0.18	1.34	1.54	SI	24.44
0.5	0.2	0.3	0.61	0.41	0.41	SI	23.02
0.5	0.3	0.2	0.37	0.19	0.19	SI	22.76
0.5	0.4	0.1	0.39	0.39	0.20	SI	22.80
0.5	0.5	0	0.80	0.80	0.15	SI	22.92
0.6	0	0.4	0.44	0.15	0.44	SI	22.41
0.6	0.1	0.3	0.58	0.14	0.58	SI	24.29
0.6	0.2	0.2	0.45	0.14	0.14	SI	23.53
0.6	0.3	0.1	0.89	0.89	0.14	SI	23.00
0.6	0.4	0	0.30	0.30	0.15	SI	22.92

I0	I1	I2	tt_0	tt_1	tt_2	converge	tiempo_ejecucion
0.7	0	0.3	0.44	0.15	0.44	SI	23.51
0.7	0.1	0.2	0.61	0.31	0.15	SI	24.79
0.7	0.2	0.1	0.15	0.30	0.15	SI	23.58
0.7	0.3	0	1.57	1.57	0.21	SI	23.46
0.8	0	0.2	0.18	0.18	0.18	SI	22.44
0.8	0.1	0.1	0.15	0.15	0.15	SI	25.00
0.8	0.2	0	0.20	0.20	0.20	SI	23.69
0.9	0	0.1	0.19	0.19	0.19	SI	22.13
0.9	0.1	0	0.30	0.30	0.15	SI	25.27
1	0	0	0.23	0.23	0.23	SI	22.29

3.10.3.3 Comparativa de los sistemas

Como se puede observar, ambos sistemas convergen. Entre las diferencias se puede apreciar que el sistema que involucra a Bayes converge generalmente más rápido que el otro sistema (o a un tiempo similar), pero demora más en finalizar la ejecución. Como la cantidad de disparos es la misma, se deduce que necesita menos transiciones para converger y menos instantes de corrección. Como contraparte requiere de más cómputo.

Esta diferencia se acrecienta cuanto mayor es la cantidad de estados de la red de petri. Así se muestra en el caso de la red de Panamá que tiene más de mil estados posibles (mientras que Huang tiene cerca de 40):

Sistema de la iteración 6:

I0	I1	tt_0	tt_1	converge	tiempo_ejecucion
0	1	0.65	0.65	SI	9.45
0.1	0.9	0.31	0.31	SI	9.50
0.2	0.8	0.37	0.37	SI	9.36
0.3	0.7	0.37	0.37	SI	9.25
0.4	0.6	0.27	0.27	SI	8.80
0.5	0.5	0.06	0.06	SI	9.26
0.6	0.4	0.36	0.36	SI	9.61
0.7	0.3	0.47	0.47	SI	9.29
0.8	0.2	0.41	0.41	SI	8.82
0.9	0.1	0.40	0.40	SI	9.46
1	0	0.58	0.58	SI	9.32

Sistema de la iteración 7:

I0	I1	tt_0	tt_1	converge	tiempo_ejecucion
0	1	0.51	0.51	SI	74.34
0.1	0.9	2.25	2.25	SI	73.27
0.2	0.8	4.12	4.12	SI	72.89
0.3	0.7	0.46	0.46	SI	73.00
0.4	0.6	0.69	0.69	SI	72.79
0.5	0.5	0.42	0.42	SI	72.96
0.6	0.4	1.86	1.86	SI	72.85
0.7	0.3	1.34	1.34	SI	74.09
0.8	0.2	2.69	2.22	SI	73.64
0.9	0.1	1.85	1.85	SI	73.61
1	0	0.00	0.00	SI	71.51

3.10.3.4 Modificar los parámetros

Si bien esta iteración constaba de comparar los sistemas entre sí y utilizar diferentes valores para las variables de los mismos, considerando que los dispuestos en un principio funcionan bien (se logra convergencia en tiempos aceptables) y que ejecutar una grilla entera sí empieza a tener valores de tiempo considerable, por una relación esfuerzo-ganancia se descarta realizar cambios en los valores de dichas variables.

3.10.4. Conclusión

En esta iteración se pudo obtener valores de las gráficas pertenecientes a la teoría de control para automatizar el control de convergencia de los sistemas en todo el espectro de requerimientos. Además, se realizaron comparaciones entre los sistemas realizados en el trabajo concluyendo que, si bien el sistema de la iteración 6 funciona mejor en redes con mayor cantidad de estados que el sistema de la iteración 7, ambos muestran convergencia en todos los escenarios. A su vez, el sistema que involucra a Bayes presenta menores sobrepasamientos y funciona mejor en redes con menor cantidad de estados.

Debido a que el tiempo de simular las grillas con distintos parámetros, deltas y sistemas crece de forma exponencial y considerando que los valores actuales generan resultados aceptables, se decide culminar el análisis en este punto.

4. Conclusiones

4.1. Conclusiones

En el desarrollo de este trabajo, hemos logrado la creación e implementación de un algoritmo inteligente con la capacidad de realizar dos tareas fundamentales en el contexto de las Redes de Petri. Primero, el algoritmo puede generar automáticamente políticas para una Red de Petri sin requerir modificaciones en la estructura subyacente del grafo. Segundo, puede ajustar automáticamente las probabilidades de disparo de las transiciones para cumplir con requisitos específicos definidos por t-invariantes.

En el capítulo introductorio de este informe, se establecieron los objetivos clave para el desarrollo de este proyecto. Al completar las tareas propuestas, hemos llegado a varias conclusiones importantes.

En primer lugar, hemos creado una herramienta ejecutable que extrae las matrices necesarias para representar Redes de Petri en nuestro algoritmo a partir de los datos proporcionados por Petrinator. Luego, desarrollamos la lógica necesaria para detectar el marcado actual de la Red de Petri, identificar las transiciones sensibilizadas en ese momento y llevar a cabo disparos de transiciones, actualizando así el estado de la red con base en los datos importados previamente.

Posteriormente, adaptamos la Red de Petri para que se pueda implementar como un 'environ' en el modelo de Reinforcement Learning. Esto involucró la implementación de los métodos 'step' y 'get reward' para gestionar el disparo de transiciones y evaluar su impacto en función de las condiciones de recompensa definidas en este informe. Sobre esta base, construimos un agente capaz de tomar decisiones que influyen en la evolución de la Red de Petri. Esto nos permitió implementar dos políticas de decisión del agente, una determinística y otra probabilística, y establecer un sesgo inicial para el sistema.

Después de desarrollar este primer prototipo de algoritmo de decisión, implementamos una primera versión de un sistema de generación automática de políticas de Redes de Petri utilizando Redes Bayesianas como un mecanismo de control de conflictos. Sin embargo, durante las pruebas, observamos que el sistema no se comportaba como se esperaba, ya que solo funcionaba dentro de rangos limitados y nunca alcanzaba los extremos. Un análisis más profundo reveló que abordar solo los casos de conflicto no era suficiente para controlar completamente la evolución de la Red de Petri, ya que algunas transiciones que no estaban en conflicto tampoco se ajustaban a los requisitos.

Para abordar este problema, decidimos ampliar el control a todos los estados de la Red de Petri, no solo en situaciones de conflicto. Lo hicimos en dos etapas. Inicialmente, aplicamos el control directamente a la tabla Q del agente, ya sea la tabla sesgada o la generada después del aprendizaje. Este enfoque produjo resultados mucho más satisfactorios que la primera versión del sistema, logrando convergencia sin problemas hacia los valores requeridos, sin márgenes inalcanzables.

Luego, utilizamos este nuevo conocimiento para diseñar una nueva red de Bayes que representara todos los estados del sistema y sus relaciones. Con este enfoque, obtuvimos una versión mejorada del sistema de generación automática de políticas de Redes de Petri mediante Redes Bayesianas como mecanismo de control, demostrando su capacidad para lograr convergencia más rápida y suave, con un mejor rendimiento en comparación con la primera versión y sin aumentar desproporcionadamente el costo de implementación.

Además, demostramos que al realimentar el sistema con una función exponencial de ponderación, la convergencia de los invariantes presenta menos sobrepasamientos en comparación con otras opciones.

Finalmente, hemos implementado la lógica esencial para realizar un análisis de los dos sistemas desarrollados en este proyecto desde la perspectiva de los sistemas de control. Además, hemos automatizado su ejecución, lo que nos ha permitido llevar a cabo una comparación detallada de su convergencia. Se ha observado que el sistema que emplea redes de Bayes como parte integral de su bucle de control generalmente presenta una convergencia más rápida en términos de cantidad de disparos, si bien implica un mayor costo computacional. Es por esto que el que emplea Bayes se prefiere para redes de petri con pocos estados y el de la iteración 6 para casos contrarios y/o sistemas limitados en recursos de hardware. El análisis minucioso de los parámetros asociados con la función exponencial de ponderación reveló que esta optimización resultaba ser excesivamente costosa en términos computacionales y no aportaba mejoras significativas en cuanto a la convergencia de los sistemas. En consecuencia, se ha

tomado la decisión de no continuar su desarrollo.

En resumen, partimos de la premisa de que existe una relación bidireccional entre las políticas y el sistema en las Redes de Petri. Sin embargo, a lo largo de este proyecto, descubrimos que esta relación no siempre es tan simple. Aprendimos que también hay una relación entre las limitaciones inherentes al sistema en ciertos escenarios y el grado de control que podemos ejercer sobre él. Esto significa que, en algunos casos, no es necesario modificar la estructura de la red, sino mejorar el control sobre los disparos de la Red de Petri. En este caso, lo logramos extendiendo el control sobre la totalidad de estados y transiciones.

Además, demostramos que extender el control a todos los disparos de la Red de Petri produce un sistema con una capacidad de convergencia superior y que presenta mejores resultados en términos de tiempo de ejecución. El resultado de todas estas iteraciones es una herramienta flexible y de alto rendimiento para la generación automática de políticas en redes de Petri del tipo S3PR

5. Trabajos Futuros

5.1. Propuestas de trabajos futuros

A continuación, se presentan algunas direcciones potenciales para la investigación futura, que pueden surgir como una extensión natural de este proyecto o como un resultado derivado del mismo. Además, se consideran alternativas a las metodologías empleadas en este estudio.

Entre las posibles áreas de investigación futura se incluyen:

- *Impacto de las Modificaciones en las Variables del Sistema*: Un área prometedora sería analizar cómo las modificaciones en las variables del sistema, como el tamaño de los lotes de corrección, los valores de ganancia 'k', la ponderación de las recompensas, entre otros, afectan la convergencia de los sistemas estudiados.

- *Desarrollo de una Capa Superior de Aprendizaje por Refuerzo*: Se podría investigar la implementación de una capa superior de aprendizaje por refuerzo que simule cómo las modificaciones en el grafo de las redes de Petri afectan el cumplimiento de los requisitos del usuario. Esta extensión permitiría un análisis más completo de la dinámica de sistemas complejos.

- *Soporte para Redes de Petri Temporales y Características Avanzadas*: Sería interesante explorar la incorporación de soporte para redes de Petri con componentes temporales y características avanzadas, como brazos de reset o lectores. Esto ampliaría significativamente la aplicabilidad de la plataforma desarrollada.

- *Integración en Otros Proyectos del Laboratorio*: Para maximizar el impacto de este algoritmo de aprendizaje por refuerzo, sería beneficioso considerar su integración en otros proyectos en desarrollo en el Laboratorio de Aprendizaje Computacional (LAC), lo que podría conducir a soluciones innovadoras en diversas áreas.

- *Validación con un Conjunto Más Amplio de Redes*: Ampliar el alcance de la validación mediante la inclusión de un mayor número y variedad de redes de Petri permitiría una evaluación más exhaustiva de la robustez y la aplicabilidad de las metodologías desarrolladas en este trabajo.

Estas direcciones potenciales de investigación pueden contribuir significativamente al avance del campo y al desarrollo de soluciones más efectivas en sistemas complejos basados en redes de Petri.

6. Anexos

A. Anexo 1: Redes de Petri utilizadas

En el desarrollo del presente trabajo se utilizan las siguientes redes:

A.1. Canal de Panamá

Esta red modela el comportamiento del control de tráfico de un sistema marítimo.

Figura 106: Red Canal de Panamá

A.2. Huang

Red simple que modela un sistema de procesos simples que compartes recursos.

Figura 107: Red Huang

A.3. Guanjun

Esta red busca modelar, similar a la anterior aunque un poco más compleja, procesos secuenciales que comparten recursos.

Figura 108: Red Guanjun

A.4. Hospital

Esta red modela un hospital. El mismo cuenta con una sala de recepción a partir de donde se derivan los pacientes o bien hacia una sala de consultas o hacia una sala de cirugías. Esta última se comparte con una entrada de emergencia.

Figura 109: Red Hospital

A.5. MFC

Esta red también modela procesos secuenciales simples con recursos compartidos. Puede ser un ejemplo de sistemas de manufactura.

Figura 110: Red MFC

A.6. POPN

Red que también puede ser un ejemplo de sistemas de manufactura. En este caso, se compone por cuatro máquinas y tres robots.

Figura 111: Red POPN

A.7. Zhao

Red que modela otro sistema con procesos secuenciales simples y recursos compartidos.

Figura 112: Red Zhao

A.8. Portugal

Red que modela un ejemplo de sistemas de manufactura flexible.

Figura 113: Red Portugal

B. Anexo 2: Tests iteración n°4

En esta sección se procede a mostrar el resto de los resultados obtenidos en las pruebas de la iteración 4.

B.1. Red 3 - Hospital

Analizando la estructura de la red y el cumplimiento de las condiciones definidas, se espera que el sistema priorice las siguientes transiciones:

- En el conflicto $K^1 < P1\{T2, T3\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^2 < P8\{T1, T9\} >$, valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^3 < P9\{T3, T8\} >$, valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^4 < P10\{T4, T5, T7\} >$, priorizar a T4.
- En el conflicto $K^5 < P14\{T1, T7\} >$, priorizar a T1.
- En el conflicto $K^6 < P15\{T1, T7\} >$, priorizar a T1.
- En el conflicto $K^7 < P16\{T1, T7\} >$, priorizar a T1.

B.1.1. Test 1 - Red de Bayes sin estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T2	T3
1	0,4929	0,5071
2	0,4929	0,5071
3	0,4929	0,5071
4	0,4929	0,5071
5	0,4929	0,5071
6	0,4929	0,5071
7	0,4929	0,5071
8	0,4929	0,5071
9	0,4929	0,5071
10	0,4929	0,5071

Cuadro 30: Conflicto 1

Ejecución	T1	T9
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 31: Conflicto 2

Ejecución	T3	T8
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 32: Conflicto 3

Ejecución	T4	T5	T7
1	0,5837	0,1518	0,2646
2	0,5837	0,1518	0,2646
3	0,5837	0,1518	0,2646
4	0,5837	0,1518	0,2646
5	0,5837	0,1518	0,2646
6	0,5837	0,1518	0,2646
7	0,5837	0,1518	0,2646
8	0,5837	0,1518	0,2646
9	0,5837	0,1518	0,2646
10	0,5837	0,1518	0,2646

Cuadro 33: Conflicto 4

Ejecución	T1	T7
1	0,5105	0,4895
2	0,5105	0,4895
3	0,5105	0,4895
4	0,5105	0,4895
5	0,5105	0,4895
6	0,5105	0,4895
7	0,5105	0,4895
8	0,5105	0,4895
9	0,5105	0,4895
10	0,5105	0,4895

Cuadro 34: Conflicto 5

Ejecución	T1	T7
1	0,6024	0,3976
2	0,6024	0,3976
3	0,6024	0,3976
4	0,6024	0,3976
5	0,6024	0,3976
6	0,6024	0,3976
7	0,6024	0,3976
8	0,6024	0,3976
9	0,6024	0,3976
10	0,6024	0,3976

Cuadro 35: Conflicto 6

Ejecución	T1	T7
1	0,5105	0,4895
2	0,5105	0,4895
3	0,5105	0,4895
4	0,5105	0,4895
5	0,5105	0,4895
6	0,5105	0,4895
7	0,5105	0,4895
8	0,5105	0,4895
9	0,5105	0,4895
10	0,5105	0,4895

Cuadro 36: Conflicto 7

B.1.2. Test 2 - Red de Bayes con estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T2	T3
1	0,4957	0,5043
2	0,4957	0,5043
3	0,4957	0,5043
4	0,4957	0,5043
5	0,4957	0,5043
6	0,4957	0,5043
7	0,4957	0,5043
8	0,4957	0,5043
9	0,4957	0,5043
10	0,4957	0,5043

Cuadro 37: Conflicto 1

Ejecución	T1	T9
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 38: Conflicto 2

Ejecución	T3	T8
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 39: Conflicto 3

Ejecución	T4	T5	T7
1	0,5837	0,1507	0,2656
2	0,5837	0,1507	0,2656
3	0,5837	0,1507	0,2656
4	0,5837	0,1507	0,2656
5	0,5837	0,1507	0,2656
6	0,5837	0,1507	0,2656
7	0,5837	0,1507	0,2656
8	0,5837	0,1507	0,2656
9	0,5837	0,1507	0,2656
10	0,5837	0,1507	0,2656

Cuadro 40: Conflicto 4

Ejecución	T1	T7
1	0,5205	0,4795
2	0,5205	0,4795
3	0,5205	0,4795
4	0,5205	0,4795
5	0,5205	0,4795
6	0,5205	0,4795
7	0,5205	0,4795
8	0,5205	0,4795
9	0,5205	0,4795
10	0,5205	0,4795

Cuadro 41: Conflicto 5

Ejecución	T1	T7
1	0,5899	0,4101
2	0,5899	0,4101
3	0,5899	0,4101
4	0,5899	0,4101
5	0,5899	0,4101
6	0,5899	0,4101
7	0,5899	0,4101
8	0,5899	0,4101
9	0,5899	0,4101
10	0,5899	0,4101

Cuadro 42: Conflicto 6

Ejecución	T1	T7
1	0,5205	0,4795
2	0,5205	0,4795
3	0,5205	0,4795
4	0,5205	0,4795
5	0,5205	0,4795
6	0,5205	0,4795
7	0,5205	0,4795
8	0,5205	0,4795
9	0,5205	0,4795
10	0,5205	0,4795

Cuadro 43: Conflicto 7

B.1.3. Conclusiones de los test

El sistema otorga una salida acorde a lo esperado, sin embargo en los conflictos 5 y 7 si bien la transición con mayor probabilidad es la esperada, la diferencia entre la probabilidad de ambas transiciones no es los suficientemente grande como para considerar que el sistema realmente este priorizando a una por sobre la otra.

B.2. Red 4 - Huang

Analizando la estructura de la red y el cumplimiento de las condiciones definidas, se espera que el sistema priorice las siguientes transiciones:

- En el conflicto $K^1 < P2\{T2, T3\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^2 < P13\{T1, T4, T5, T7, T9\} >$ priorizar a T1.

B.2.1. Test 1 - Red de Bayes sin estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T2	T3
1	0,5070	0,4930
2	0,5070	0,4930
3	0,5070	0,4930
4	0,5070	0,4930
5	0,5070	0,4930
6	0,5070	0,4930
7	0,5070	0,4930
8	0,5070	0,4930
9	0,5070	0,4930
10	0,5070	0,4930

Cuadro 44: Conflicto 1

Ejecución	T1	T4	T5	T7	T9
1	0,3062	0,1536	0,1524	0,1940	0,1938
2	0,3062	0,1536	0,1524	0,1940	0,1938
3	0,3062	0,1536	0,1524	0,1940	0,1938
4	0,3062	0,1536	0,1524	0,1940	0,1938
5	0,3062	0,1536	0,1524	0,1940	0,1938
6	0,3062	0,1536	0,1524	0,1940	0,1938
7	0,3062	0,1536	0,1524	0,1940	0,1938
8	0,3062	0,1536	0,1524	0,1940	0,1938
9	0,3062	0,1536	0,1524	0,1940	0,1938
10	0,3062	0,1536	0,1524	0,1940	0,1938

Cuadro 45: Conflicto 2

B.2.2. Test 2 - Red de Bayes con estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T2	T3
1	0,4982	0,5018
2	0,4982	0,5018
3	0,4982	0,5018
4	0,4982	0,5018
5	0,4982	0,5018
6	0,4982	0,5018
7	0,4982	0,5018
8	0,4982	0,5018
9	0,4982	0,5018
10	0,4982	0,5018

Cuadro 46: Conflicto 1

Ejecución	T1	T4	T5	T7	T9
1	0,3083	0,1536	0,1546	0,1918	0,1918
2	0,3083	0,1536	0,1546	0,1918	0,1918
3	0,3083	0,1536	0,1546	0,1918	0,1918
4	0,3083	0,1536	0,1546	0,1918	0,1918
5	0,3083	0,1536	0,1546	0,1918	0,1918
6	0,3083	0,1536	0,1546	0,1918	0,1918
7	0,3083	0,1536	0,1546	0,1918	0,1918
8	0,3083	0,1536	0,1546	0,1918	0,1918
9	0,3083	0,1536	0,1546	0,1918	0,1918
10	0,3083	0,1536	0,1546	0,1918	0,1918

Cuadro 47: Conflicto 2

B.2.3. Conclusiones de los test

El sistema otorga una salida acorde a lo esperado, priorizando aquellas transiciones que en el análisis previo se identificaron como las que mayores recompensas otorgarían.

B.3. Red 5 - MFC

Analizando la estructura de la red y el cumplimiento de las condiciones definidas, se espera que el sistema priorice las siguientes transiciones:

- En el conflicto $K^1 < P11\{T1, T11\} >$ priorizar a T1.
- En el conflicto $K^2 < P12\{T2, T10\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^3 < P13\{T3, T9\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^4 < P14\{T4, T8\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^5 < P15\{T5, T7\} >$ priorizar a T5.
- En el conflicto $K^6 < P20\{T1, T7\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^7 < P21\{T1, T7\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^8 < P22\{T1, T7\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^9 < P23\{T1, T7\} >$ valores aproximadamente iguales entre ambas transiciones.

B.3.1. Test 1 - Red de Bayes sin estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T1	T11
1	0,5634	0,4366
2	0,5634	0,4366
3	0,5634	0,4366
4	0,5634	0,4366
5	0,5634	0,4366
6	0,5634	0,4366
7	0,5634	0,4366
8	0,5634	0,4366
9	0,5634	0,4366
10	0,5634	0,4366

Cuadro 48: Conflicto 1

Ejecución	T2	T10
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 49: Conflicto 2

Ejecución	T3	T9
1	0,5763	0,4237
2	0,5763	0,4237
3	0,5763	0,4237
4	0,5763	0,4237
5	0,5763	0,4237
6	0,5763	0,4237
7	0,5763	0,4237
8	0,5763	0,4237
9	0,5763	0,4237
10	0,5763	0,4237

Cuadro 50: Conflicto 3

Ejecución	T4	T8
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 51: Conflicto 4

Ejecución	T5	T7
1	0,4183	0,5817
2	0,4183	0,5817
3	0,4183	0,5817
4	0,4183	0,5817
5	0,4183	0,5817
6	0,4183	0,5817
7	0,4183	0,5817
8	0,4183	0,5817
9	0,4183	0,5817
10	0,4183	0,5817

Cuadro 52: Conflicto 5

Ejecución	T1	T7
1	0,5461	0,4539
2	0,5461	0,4539
3	0,5461	0,4539
4	0,5461	0,4539
5	0,5461	0,4539
6	0,5461	0,4539
7	0,5461	0,4539
8	0,5461	0,4539
9	0,5461	0,4539
10	0,5461	0,4539

Cuadro 53: Conflicto 6

Ejecución	T1	T7
1	0,5162	0,4838
2	0,5162	0,4838
3	0,5162	0,4838
4	0,5162	0,4838
5	0,5162	0,4838
6	0,5162	0,4838
7	0,5162	0,4838
8	0,5162	0,4838
9	0,5162	0,4838
10	0,5162	0,4838

Cuadro 54: Conflicto 7

Ejecución	T1	T7
1	0,5244	0,4756
2	0,5244	0,4756
3	0,5244	0,4756
4	0,5244	0,4756
5	0,5244	0,4756
6	0,5244	0,4756
7	0,5244	0,4756
8	0,5244	0,4756
9	0,5244	0,4756
10	0,5244	0,4756

Cuadro 55: Conflicto 8

Ejecución	T1	T7
1	0,4882	0,5118
2	0,4882	0,5118
3	0,4882	0,5118
4	0,4882	0,5118
5	0,4882	0,5118
6	0,4882	0,5118
7	0,4882	0,5118
8	0,4882	0,5118
9	0,4882	0,5118
10	0,4882	0,5118

Cuadro 56: Conflicto 9

B.3.2. Test 2 - Red de Bayes con estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T1	T11
1	0,5949	0,4051
2	0,5949	0,4051
3	0,5949	0,4051
4	0,5949	0,4051
5	0,5949	0,4051
6	0,5949	0,4051
7	0,5949	0,4051
8	0,5949	0,4051
9	0,5949	0,4051
10	0,5949	0,4051

Cuadro 57: Conflicto 1

Ejecución	T2	T10
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 58: Conflicto 2

Ejecución	T3	T9
1	0,4913	0,5087
2	0,4913	0,5087
3	0,4913	0,5087
4	0,4913	0,5087
5	0,4913	0,5087
6	0,4913	0,5087
7	0,4913	0,5087
8	0,4913	0,5087
9	0,4913	0,5087
10	0,4913	0,5087

Cuadro 59: Conflicto 3

Ejecución	T4	T8
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 60: Conflicto 4

Ejecución	T5	T7
1	0,4033	0,5967
2	0,4033	0,5967
3	0,4033	0,5967
4	0,4033	0,5967
5	0,4033	0,5967
6	0,4033	0,5967
7	0,4033	0,5967
8	0,4033	0,5967
9	0,4033	0,5967
10	0,4033	0,5967

Cuadro 61: Conflicto 5

Ejecución	T1	T7
1	0,5335	0,4665
2	0,5335	0,4665
3	0,5335	0,4665
4	0,5335	0,4665
5	0,5335	0,4665
6	0,5335	0,4665
7	0,5335	0,4665
8	0,5335	0,4665
9	0,5335	0,4665
10	0,5335	0,4665

Cuadro 62: Conflicto 6

Ejecución	T1	T7
1	0,5021	0,4979
2	0,5021	0,4979
3	0,5021	0,4979
4	0,5021	0,4979
5	0,5021	0,4979
6	0,5021	0,4979
7	0,5021	0,4979
8	0,5021	0,4979
9	0,5021	0,4979
10	0,5021	0,4979

Cuadro 63: Conflicto 7

Ejecución	T1	T7
1	0,5123	0,4877
2	0,5123	0,4877
3	0,5123	0,4877
4	0,5123	0,4877
5	0,5123	0,4877
6	0,5123	0,4877
7	0,5123	0,4877
8	0,5123	0,4877
9	0,5123	0,4877
10	0,5123	0,4877

Cuadro 64: Conflicto 8

Ejecución	T1	T7
1	0,4808	0,5192
2	0,4808	0,5192
3	0,4808	0,5192
4	0,4808	0,5192
5	0,4808	0,5192
6	0,4808	0,5192
7	0,4808	0,5192
8	0,4808	0,5192
9	0,4808	0,5192
10	0,4808	0,5192

Cuadro 65: Conflicto 9

B.3.3. Conclusiones de los test

Es evidente que ninguno de los modelos de Redes de Bayes sometidos a prueba logra adecuarse a los parámetros deseados en lo que respecta a los conflictos 1 y 5. Además, el modelo de Bayes sin estado tampoco satisface las expectativas establecidas para los conflictos 3 y 6.

B.4. Red 6 - Portugal

Analizando la estructura de la red y el cumplimiento de las condiciones definidas, se espera que el sistema priorice las siguientes transiciones:

- En el conflicto $K^1 < P14\{T2, T4\} >$ priorizar a T2.
- En el conflicto $K^2 < P15\{T8, T10\} >$ priorizar T8.
- En el conflicto $K^3 < P13\{T1, T5, T7, T11\} >$ priorizar T11 y T5 por sobre T1 y T7.
- En el conflicto $K^4 < P16\{T1, T4, T7, T10\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^5 < P17\{T1, T4, T7, T10\} >$ valores aproximadamente iguales entre ambas transiciones..

B.4.1. Test 1 - Red de Bayes sin estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T1	T7
1	0,2820	0,7180
2	0,2820	0,7180
3	0,2820	0,7180
4	0,2820	0,7180
5	0,2820	0,7180
6	0,2820	0,7180
7	0,2820	0,7180
8	0,2820	0,7180
9	0,2820	0,7180
10	0,2820	0,7180

Cuadro 66: Conflicto 1

Ejecución	T1	T7
1	0,2543	0,7457
2	0,2543	0,7457
3	0,2543	0,7457
4	0,2543	0,7457
5	0,2543	0,7457
6	0,2543	0,7457
7	0,2543	0,7457
8	0,2543	0,7457
9	0,2543	0,7457
10	0,2543	0,7457

Cuadro 67: Conflicto 2

Ejecución	T1	T5	T7	T11
1	0,5202	0,0104	0,4570	0,0124
2	0,5202	0,0104	0,4570	0,0124
3	0,5202	0,0104	0,4570	0,0124
4	0,5202	0,0104	0,4570	0,0124
5	0,5202	0,0104	0,4570	0,0124
6	0,5202	0,0104	0,4570	0,0124
7	0,5202	0,0104	0,4570	0,0124
8	0,5202	0,0104	0,4570	0,0124
9	0,5202	0,0104	0,4570	0,0124
10	0,5202	0,0104	0,4570	0,0124

Cuadro 68: Conflicto 3

Ejecución	T1	T4	T7	T10
1	0,2567	0,2564	0,2244	0,2625
2	0,2567	0,2564	0,2244	0,2625
3	0,2567	0,2564	0,2244	0,2625
4	0,2567	0,2564	0,2244	0,2625
5	0,2567	0,2564	0,2244	0,2625
6	0,2567	0,2564	0,2244	0,2625
7	0,2567	0,2564	0,2244	0,2625
8	0,2567	0,2564	0,2244	0,2625
9	0,2567	0,2564	0,2244	0,2625
10	0,2567	0,2564	0,2244	0,2625

Cuadro 69: Conflicto 4

Ejecución	T1	T4	T7	T10
1	0,2567	0,2564	0,2244	0,2625
2	0,2567	0,2564	0,2244	0,2625
3	0,2567	0,2564	0,2244	0,2625
4	0,2567	0,2564	0,2244	0,2625
5	0,2567	0,2564	0,2244	0,2625
6	0,2567	0,2564	0,2244	0,2625
7	0,2567	0,2564	0,2244	0,2625
8	0,2567	0,2564	0,2244	0,2625
9	0,2567	0,2564	0,2244	0,2625
10	0,2567	0,2564	0,2244	0,2625

Cuadro 70: Conflicto 5

B.4.2. Test 2 - Red de Bayes con estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T1	T7
1	0,2832	0,7168
2	0,2832	0,7168
3	0,2832	0,7168
4	0,2832	0,7168
5	0,2832	0,7168
6	0,2832	0,7168
7	0,2832	0,7168
8	0,2832	0,7168
9	0,2832	0,7168
10	0,2832	0,7168

Cuadro 71: Conflicto 1

Ejecución	T1	T7
1	0,2774	0,7226
2	0,2774	0,7226
3	0,2774	0,7226
4	0,2774	0,7226
5	0,2774	0,7226
6	0,2774	0,7226
7	0,2774	0,7226
8	0,2774	0,7226
9	0,2774	0,7226
10	0,2774	0,7226

Cuadro 72: Conflicto 2

Ejecución	T1	T5	T7	T11
1	0,5075	0,0129	0,4639	0,0157
2	0,5075	0,0129	0,4639	0,0157
3	0,5075	0,0129	0,4639	0,0157
4	0,5075	0,0129	0,4639	0,0157
5	0,5075	0,0129	0,4639	0,0157
6	0,5075	0,0129	0,4639	0,0157
7	0,5075	0,0129	0,4639	0,0157
8	0,5075	0,0129	0,4639	0,0157
9	0,5075	0,0129	0,4639	0,0157
10	0,5075	0,0129	0,4639	0,0157

Cuadro 73: Conflicto 3

Ejecución	T1	T4	T7	T10
1	0,2529	0,2538	0,2303	0,2630
2	0,2529	0,2538	0,2303	0,2630
3	0,2529	0,2538	0,2303	0,2630
4	0,2529	0,2538	0,2303	0,2630
5	0,2529	0,2538	0,2303	0,2630
6	0,2529	0,2538	0,2303	0,2630
7	0,2529	0,2538	0,2303	0,2630
8	0,2529	0,2538	0,2303	0,2630
9	0,2529	0,2538	0,2303	0,2630
10	0,2529	0,2538	0,2303	0,2630

Cuadro 74: Conflicto 4

Ejecución	T1	T4	T7	T10
1	0,2529	0,2538	0,2303	0,2630
2	0,2529	0,2538	0,2303	0,2630
3	0,2529	0,2538	0,2303	0,2630
4	0,2529	0,2538	0,2303	0,2630
5	0,2529	0,2538	0,2303	0,2630
6	0,2529	0,2538	0,2303	0,2630
7	0,2529	0,2538	0,2303	0,2630
8	0,2529	0,2538	0,2303	0,2630
9	0,2529	0,2538	0,2303	0,2630
10	0,2529	0,2538	0,2303	0,2630

Cuadro 75: Conflicto 5

B.4.3. Conclusiones de los test

El sistema presenta una salida claramente en desacuerdo con las expectativas, dando prioridad a las transiciones que, en el análisis previo, se habían identificado como aquellas que proporcionarían las recompensas más bajas. Como resultado, se anticipaba que estas transiciones tenían una probabilidad significativamente menor de ser disparadas.

B.5. Red 7 - Zhao

Analizando la estructura de la red y el cumplimiento de las condiciones definidas, se espera que el sistema priorice las siguientes transiciones:

- En el conflicto $K^1 < P1\{T2, T6\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^2 < P5\{T2, T6\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^3 < P15\{T1, T11\} >$ valores aproximadamente iguales entre ambas transiciones.
- En el conflicto $K^4 < P16\{T2, T6, T10\} >$ priorizar T2 y T6 con valores similares.
- En el conflicto $K^5 < P17\{T3, T7, T9\} >$ priorizar T3 y T7 con valores similares.
- En el conflicto $K^6 < P18\{T1, T9\} >$ priorizar a T1.

B.5.1. Test 1 - Red de Bayes sin estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T2	T6
1	0,4952	0,5048
2	0,4952	0,5048
3	0,4952	0,5048
4	0,4952	0,5048
5	0,4952	0,5048
6	0,4952	0,5048
7	0,4952	0,5048
8	0,4952	0,5048
9	0,4952	0,5048
10	0,4952	0,5048

Cuadro 76: Conflicto 1

Ejecución	T2	T6
1	0,4952	0,5048
2	0,4952	0,5048
3	0,4952	0,5048
4	0,4952	0,5048
5	0,4952	0,5048
6	0,4952	0,5048
7	0,4952	0,5048
8	0,4952	0,5048
9	0,4952	0,5048
10	0,4952	0,5048

Cuadro 77: Conflicto 2

Ejecución	T1	T11
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 78: Conflicto 3

Ejecución	T2	T6	T10
1	0,4952	0,5048	0,0000
2	0,4952	0,5048	0,0000
3	0,4952	0,5048	0,0000
4	0,4952	0,5048	0,0000
5	0,4952	0,5048	0,0000
6	0,4952	0,5048	0,0000
7	0,4952	0,5048	0,0000
8	0,4952	0,5048	0,0000
9	0,4952	0,5048	0,0000
10	0,4952	0,5048	0,0000

Cuadro 79: Conflicto 4

Ejecución	T1	T3	T17
1	0,3333	0,3333	0,3333
2	0,3333	0,3333	0,3333
3	0,3333	0,3333	0,3333
4	0,3333	0,3333	0,3333
5	0,3333	0,3333	0,3333
6	0,3333	0,3333	0,3333
7	0,3333	0,3333	0,3333
8	0,3333	0,3333	0,3333
9	0,3333	0,3333	0,3333
10	0,3333	0,3333	0,3333

Cuadro 80: Conflicto 5

Ejecución	T1	T9
1	0,4605	0,5395
2	0,4605	0,5395
3	0,4605	0,5395
4	0,4605	0,5395
5	0,4605	0,5395
6	0,4605	0,5395
7	0,4605	0,5395
8	0,4605	0,5395
9	0,4605	0,5395
10	0,4605	0,5395

Cuadro 81: Conflicto 6

B.5.2. Test 2 - Red de Bayes con estado

En los cuadros siguientes se pueden observar los resultados obtenidos para esta red de Bayes:

Ejecución	T2	T6
1	0,4939	0,5061
2	0,4939	0,5061
3	0,4939	0,5061
4	0,4939	0,5061
5	0,4939	0,5061
6	0,4939	0,5061
7	0,4939	0,5061
8	0,4939	0,5061
9	0,4939	0,5061
10	0,4939	0,5061

Cuadro 82: Conflicto 1

Ejecución	T2	T6
1	0,4939	0,5061
2	0,4939	0,5061
3	0,4939	0,5061
4	0,4939	0,5061
5	0,4939	0,5061
6	0,4939	0,5061
7	0,4939	0,5061
8	0,4939	0,5061
9	0,4939	0,5061
10	0,4939	0,5061

Cuadro 83: Conflicto 2

Ejecución	T1	T11
1	0,5000	0,5000
2	0,5000	0,5000
3	0,5000	0,5000
4	0,5000	0,5000
5	0,5000	0,5000
6	0,5000	0,5000
7	0,5000	0,5000
8	0,5000	0,5000
9	0,5000	0,5000
10	0,5000	0,5000

Cuadro 84: Conflicto 3

Ejecución	T2	T6	T10
1	0,4939	0,5061	0,0000
2	0,4939	0,5061	0,0000
3	0,4939	0,5061	0,0000
4	0,4939	0,5061	0,0000
5	0,4939	0,5061	0,0000
6	0,4939	0,5061	0,0000
7	0,4939	0,5061	0,0000
8	0,4939	0,5061	0,0000
9	0,4939	0,5061	0,0000
10	0,4939	0,5061	0,0000

Cuadro 85: Conflicto 4

Ejecución	T1	T3	T17
1	0,3333	0,3333	0,3333
2	0,3333	0,3333	0,3333
3	0,3333	0,3333	0,3333
4	0,3333	0,3333	0,3333
5	0,3333	0,3333	0,3333
6	0,3333	0,3333	0,3333
7	0,3333	0,3333	0,3333
8	0,3333	0,3333	0,3333
9	0,3333	0,3333	0,3333
10	0,3333	0,3333	0,3333

Cuadro 86: Conflicto 5

Ejecución	T1	T9
1	0,4656	0,5344
2	0,4656	0,5344
3	0,4656	0,5344
4	0,4656	0,5344
5	0,4656	0,5344
6	0,4656	0,5344
7	0,4656	0,5344
8	0,4656	0,5344
9	0,4656	0,5344
10	0,4656	0,5344

Cuadro 87: Conflicto 6

B.5.3. Conclusiones de los test

Los resultados revelan que el sistema no alcanza la convergencia deseada en los conflictos 6 y 5. Asimismo, se evidencia que el conflicto 4 nunca se manifiesta como efectivo, incluso cuando se incorpora la transición T10.

C. Anexo 3: Tests iteración n°5

En esta sección se procede a mostrar el resto de los resultados obtenidos en las pruebas de la iteración 5.

C.1. Red 3 - Hospital

C.1.1. Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 114: Contador invariante 1.

Figura 115: Contador invariante 2.

Figura 116: Contador invariante 3.

C.1.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 117: Contador invariante 1.

Figura 118: Contador invariante 2.

Figura 119: Contador invariante 3.

C.1.3. Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 120: Contador invariante 1.

Figura 121: Contador invariante 2.

Figura 122: Contador invariante 3.

C.2. Red 4 - MFC

C.2.1. Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 123: Contador invariante 1.

Figura 124: Contador invariante 2.

C.2.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 125: Contador invariante 1.

Figura 126: Contador invariante 2.

C.2.3. Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 127: Contador invariante 1.

Figura 128: Contador invariante 2.

C.3. Red 5 - Portugal

C.3.1. Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 129: Contador invariante 1.

Figura 130: Contador invariante 2.

Figura 131: Contador invariante 3.

Figura 132: Contador invariante 4.

C.3.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 133: Contador invariante 1.

Figura 134: Contador invariante 2.

Figura 135: Contador invariante 3.

Figura 136: Contador invariante 4.

C.3.3. Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 137: Contador invariante 1.

Figura 138: Contador invariante 2.

Figura 139: Contador invariante 3.

Figura 140: Contador invariante 4.

C.4. Red 6 - Zhao

C.4.1. Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 141: Contador invariante 1.

Figura 142: Contador invariante 2.

Figura 143: Contador invariante 3.

C.4.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 144: Contador invariante 1.

Figura 145: Contador invariante 2.

Figura 146: Contador invariante 3.

C.4.3. Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 147: Contador invariante 1.

Figura 148: Contador invariante 2.

Figura 149: Contador invariante 3.

C.5. Red 7 - POPN

C.5.1. Test 1: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 150: Contador invariante 1.

Figura 151: Contador invariante 2.

Figura 152: Contador invariante 3.

Figura 153: Contador invariante 4.

C.5.2. Test 2: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 154: Contador invariante 1.

Figura 155: Contador invariante 2.

Figura 156: Contador invariante 3.

Figura 157: Contador invariante 4.

C.5.3. Test 3: Realimentación considerando función exponencial sobre resta con batch

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 158: Contador invariante 1.

Figura 159: Contador invariante 2.

Figura 160: Contador invariante 3.

Figura 161: Contador invariante 4.

D. Anexo 4: Tests iteración n°6

En esta sección se procede a mostrar el resto de los resultados obtenidos en las pruebas de la iteración 6.

D.1. Red 4: Red Hospital

D.1.1. Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 162: Contador invariante 1.

Figura 163: Contador invariante 2.

Figura 164: Contador invariante 3.

D.1.2. Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 165: Contador invariante 1.

Figura 166: Contador invariante 2.

Figura 167: Contador invariante 3.

D.1.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 168: Contador invariante 1.

Figura 169: Contador invariante 2.

Figura 170: Contador invariante 3.

D.1.4. Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 171: Contador invariante 1.

Figura 172: Contador invariante 2.

Figura 173: Contador invariante 3.

D.2. Red 5: MFC

D.2.1. Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 174: Contador invariante 1.

Figura 175: Contador invariante 2.

D.2.2. Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 176: Contador invariante 1.

Figura 177: Contador invariante 2.

D.2.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 178: Contador invariante 1.

Figura 179: Contador invariante 2.

D.2.4. Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 180: Contador invariante 1.

Figura 181: Contador invariante 2.

D.3. Red 6: Red Portugal

D.3.1. Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 182: Contador invariante 1.

Figura 183: Contador invariante 2.

Figura 184: Contador invariante 3.

Figura 185: Contador invariante 4.

D.3.2. Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 186: Contador invariante 1.

Figura 187: Contador invariante 2.

Figura 188: Contador invariante 3.

Figura 189: Contador invariante 4.

D.3.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 190: Contador invariante 1.

Figura 191: Contador invariante 2.

Figura 192: Contador invariante 3.

Figura 193: Contador invariante 4.

D.3.4. Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 194: Contador invariante 1.

D.4. Red 7: Red Zhao

D.4.1. Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

Figura 195: Contador invariante 2.

Figura 196: Contador invariante 3.

Figura 197: Contador invariante 4.

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 198: Contador invariante 1.

Figura 199: Contador invariante 2.

Figura 200: Contador invariante 3.

D.4.2. Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 201: Contador invariante 1.

Figura 202: Contador invariante 2.

Figura 203: Contador invariante 3.

D.4.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 204: Contador invariante 1.

Figura 205: Contador invariante 2.

Figura 206: Contador invariante 3.

D.4.4. Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 207: Contador invariante 1.

Figura 208: Contador invariante 2.

Figura 209: Contador invariante 3.

D.5. Red 8: Red POPN

D.5.1. Test 1: Realimentación considerando la resta del histórico

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 210: Contador invariante 1.

Figura 211: Contador invariante 2.

Figura 212: Contador invariante 3.

Figura 213: Contador invariante 4.

D.5.2. Test 2: Realimentación considerando batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 214: Contador invariante 1.

Figura 215: Contador invariante 2.

Figura 216: Contador invariante 3.

Figura 217: Contador invariante 4.

D.5.3. Test 3: Realimentación considerando suma ponderada entre el histórico y el batch de ponderación

Resultados obtenidos

- Ganancia K: 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 218: Contador invariante 1.

Figura 219: Contador invariante 2.

Figura 220: Contador invariante 3.

Figura 221: Contador invariante 4.

D.5.4. Test 4: Realimentación considerando función exponencial sobre el histórico

Resultados obtenidos

- Ganancia K: 1
- Alpha : 1
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 222: Contador invariante 1.

Figura 223: Contador invariante 2.

Figura 224: Contador invariante 3.

Figura 225: Contador invariante 4.

E. Anexo 5: Tests iteracion n°7

E.1. Red 4: Hospital

E.1.1. Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 226: Contador invariante 1.

Figura 227: Contador invariante 2.

Figura 228: Contador invariante 3.

E.1.2. Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 229: Contador invariante 1.

Figura 230: Contador invariante 2.

Figura 231: Contador invariante 3.

E.2. Red 5: MFC

E.2.1. Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 232: Contador invariante 1.

Figura 233: Contador invariante 2.

E.2.2. Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 234: Contador invariante 1.

Figura 235: Contador invariante 2.

E.3. Red 6: Portugal

E.3.1. Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 236: Contador invariante 1.

Figura 237: Contador invariante 2.

Figura 238: Contador invariante 3.

Figura 239: Contador invariante 4.

E.3.2. Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 240: Contador invariante 1.

Figura 241: Contador invariante 2.

Figura 242: Contador invariante 3.

Figura 243: Contador invariante 4.

E.4. Red 7: Zhao

E.4.1. Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 244: Contador invariante 1.

Figura 245: Contador invariante 2.

Figura 246: Contador invariante 3.

E.4.2. Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 247: Contador invariante 1.

Figura 248: Contador invariante 2.

Figura 249: Contador invariante 3.

E.5. Red 8: POPN

E.5.1. Test 1: Realimentación resta directa

Resultados obtenidos

- Ganancia K: 0.5
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 250: Contador invariante 1.

Figura 251: Contador invariante 2.

Figura 252: Contador invariante 3.

Figura 253: Contador invariante 4.

E.5.2. Test 2: Realimentación función exponencial de ponderación

Resultados obtenidos

- Ganancia K: 1
- Alpha : 0.8
- Intervalo de confianza: 0.05
- Cantidad de transiciones por batch: 200
- Número de batches: 200

Figura 254: Contador invariante 1.

Figura 255: Contador invariante 2.

Figura 256: Contador invariante 3.

Figura 257: Contador invariante 4.

Referencias

[1] Alexander Amini. Deep reinforcement learning. mit, 6:s191, 2019.

[2] Nicolas Arrioja Landa Cosio. *Inteligencia Artificial, Sistemas Inteligentes con C#*. USERS, 2011.

[3] Juan Ignacio Bagnato. Aprendizaje por refuerzo. `http://aprendemachinelearning.com/aprendizaje-por-refuerzo/`.

[4] Paulo Carreira, Vasco Amaral, and Hans Vangheluwe, editors. *Foundations of multi-paradigm modelling for cyber-physical systems*. Springer Nature, Cham, Switzerland, 1 edition, September 2020.

[5] José Luis Iglesias Feria. Ia probabilidad - redes bayesianas. `https://www.youtube.com/playlist?list=PLYWD-VqrD5BCr-QeESSOvpEqkAcvQOrQ0`, 2018.

[6] George D. Greenwade. The Comprehensive Tex Archive Network (CTAN). *TUGBoat*, 14(3):342–351, 1993.

[7] Edward A. Lee. Cyber physical systems: Design challenges. In *11th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, pages 363–369, 2008.

[8] Micolini Orlando, Cebollada Marcelo, Eschoyez Maximiliano, Ventre Luis O., and Schild Marcelo. Ecuación de estado generalizada para redes de petri no autónomas y con distintos tipos de arcos. *XXII Congreso Argentino de Ciencias de la Computación (CACIC 2016)*, 2016.

[9] Mario A. Perez, Analía Perez Hidalgo, and Elisa Perez Berenguer. Introducción a los sistemas de control y modelo matemático para sistemas lineales invariantes en el tiempo. Universidad Nacional de San Juan. Facultad de Ingeniería. Departamento de Electrónica y Automática, 2007.

[10] J.L. Peterson. *Petri Net Theory and the Modeling of Systems*. Independently Published, 2019.

[11] Augusto Pichetti and Lucas Antonio Salse. Generación dinámica de la política de una red de petri. Proyecto fin de carrera, ingenieria en computacion, Facultad de Ciencias Exactas, Fisicas y Naturales, UNC, 2022.

[12] Python.org. 3.11.0 documentation.la biblioteca estándar de python.servicios de procesamiento de texto.re — operaciones con expresiones regulares. `https://docs.python.org/es/3/library/re.html#id1`, 2022.

[13] Sudharsan Ravichandiran. *Hands-on reinforcement learning with Python: master reinforcement and deep reinforcement learning using OpenAI gym and tensorFlow*. Packt Publishing Ltd, 2018.

[14] Stuart J. Russell and Peter Norvig. *Artificial Intelligence. A Modern Approach*. Pearson, third edition edition, 2009.

[15] Ian Sommerville. *Ingenieria del software*. Pearson Educacion, 7 edition, 2005.

[16] the NumPy community. *NumPy User Guide. Release 1.23.0*. NumPy, 2022.

[17] Simonini Thomas. Deep reinforcement learning course. `https://simoninithomas.github.io/deep-rl-course/`.

[18] Stefano Zanero. Cyber-physical systems. *Computer*, 50(4):14–16, 2017.