UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

TESIS

APRENDIZAJE POR REFUERZO PARA LA TOMA DE DECISIONES EN PROBLEMAS CON ALTA INCERTIDUMBRE QUE SE PUEDAN MODELAR POR ETAPAS

PRESENTADA POR:

CARMEN CONSTANZA URIBE SANDOVAL

COMO REQUISITO PARCIAL PARA OBTENER EL GRADO DE:

DOCTOR EN INGENIERÍA

CON ESPECIALIDAD EN INGENIERÍA DE SISTEMAS

JULIO DE 2021

Universidad Autónoma de Nuevo León

Facultad de Ingeniería Mecánica y Eléctrica

Subdirección de Estudios de Posgrado

Aprendizaje por Refuerzo para la toma de decisiones en problemas con alta incertidumbre que se puedan modelar por etapas

por

Carmen Constanza Uribe Sandoval

como requisito parcial para obtener el grado de

doctorado en ingeniería

con especialidad en Ingeniería de Sistemas

Julio de 2021

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica
Subdirección de Estudios de Posgrado

Los miembros del Comité de Tesis recomendamos que la Tesis «Aprendizaje por Refuerzo para la Toma de Decisiones en Problemas con Alta Incertidumbre que se Pueden Modelar por Etapas», realizada por la alumna Carmen Constanza Uribe Sandoval, con número de matrícula 1935063, sea aceptada para su defensa como requisito para obtener el grado de Doctorado en Ingeniería con Especialidad en Ingeniería de Sistemas.

El Comité de Tesis

Dr. José Arturo Berrones Santos
Director

Paola A. Sánchez
Dra. Paola Andrea Sánchez Sánchez
Co-Director

Dra. Sara Verónica Rodríguez Sánchez
Revisor

Dra. Leticia Amalia Neira Tovar
Revisor

Dr. Francisco Javier Almaguer Martínez
Revisor

Dr. Romeo Sánchez Nigenda
Revisor

Vo. Bo.

Dr. Simón Martínez Martínez
Subdirector de Estudios de Posgrado

176

San Nicolás de los Garza, Nuevo León, julio de 2021

Ciudad Universitaria Pedro de Alba s/n, C.P. 66455. A.P. 076 Suc. "F"
San Nicolás de los Garza, Nuevo León, México. Tels: (81) 8332 0903 /
Conm.: 8329 4020 / Fax: (81) 8332 0904

A mi familia.

A la Universidad de Boyacá.

Índice general

Agradecimientos xii

Resumen xiii

1. Introducción 1

1.1. Descripción del problema científico 1

1.1.1. Sistematización del problema 3

1.2. Objetivo general 4

1.3. Objetivos específicos 4

1.4. Hipótesis 5

1.5. Novedad científica 5

1.6. Justificación 6

1.7. Resultados esperados 7

1.8. Metodología 7

1.8.1. Enfoque general del método 7

1.8.2. Técnicas específicas seleccionadas 8

1.8.3. Fases de la metodología propuesta 9

1.9. Estructura del documento 9

2. FUNDAMENTOS 11

2.1. Conceptos básicos y antecedentes 11

2.1.1. Conceptos básicos de Grafos 11

2.1.2. Aprendizaje automático 13

2.1.3. Los Bandits 14

2.1.4. Antecedentes sobre Grafos 15

2.1.5. Antecedentes sobre aprendizaje por refuerzo 17

2.2. Denominación del modelo propuesto 20

2.3. Fundamentos matemáticos 21

2.4. Proceso de negocio 22

2.4.1. Gestión de procesos de negocio 22

2.4.2. Proceso de negocio dinámico y caso 24

2.4.3. CMMN 24

3. Aplicación del método 26

3.1. Diseño del grafo 26

3.2. Diseño del modelo matemático 27

3.3. Diseño del modelo de aprendizaje 29

3.3.1. Exploración 30

3.3.2. Explotación 31

3.4. Construcción del algoritmo 31

3.4.1. Generador de bandits 33

3.4.2. Generador de la matriz de adyacencia 34

3.4.3. Encontrando la ruta óptima 37

3.4.4. Búsqueda con probabilidad uniforme 37

3.4.5. Búsqueda con probabilidades aprendidas 38

3.5. Pruebas 40

3.5.1. Grafo para las pruebas 40

3.5.2. Pruebas con probabilidad uniforme 41

3.5.3. Pruebas con probabilidades aprendidas 42

4. Resultados computacionales **43**

4.1. Parámetro Deltha puro 45

4.1.1. Prueba inicial 45

4.1.2. Segunda prueba 46

4.1.3. Experimentos 46

4.2. Parámetro Gamma 52

4.2.1. Primer caso de estudio 52

4.2.2. Experimentos 54

4.3. Generación de un grafo aleatorio 62

4.3.1. Prueba 64

4.4. Posibles aplicaciones del modelo 65

5. Conclusiones **67**

6. Discusión y trabajo futuro **70**

6.1. Los pros y los contras 70

6.2. Trabajo futuro 71

A. Propuesta para CMMN **72**

A.1. Modelo CMMN en un grafo por etapas 72

B. Breve reseña de la Gestión de Casos **75**

Índice de figuras

2.1. Modelado de un proceso dinámico 25

3.1. Grafo por etapas 27

3.2. Resultado de la selección de brazo usando *bandits* 34

3.3. Resultado de la matriz de adyacencia para un grafo por etapas 35

3.4. Grafo Modelo 40

3.5. Resultado con probabilidad uniforme 41

3.6. Resultado con probabilidad aprendida 42

4.1. Grafo para las pruebas de parámetros 44

4.2. Ejecución algoritmo Probabilidad Modelada 53

4.3. Diagramas BoxPlot para la variable Gamma 57

4.4. Diferencia normalizada de ganancias para T y γ 61

4.5. Histograma de diferencia normalizada de ganancia 61

4.6. Grafo generado en forma aleatoria 63

4.7. Convergencia de ganancia con un grafo generado en forma aleatoria . 65

A.1. Modelo CMMN 73

A.2. Caso en etapas. Parte 1. 74

A.3. Caso en etapas. Parte 2. 74

B.1. Garner iBPMS 2019 77

Índice de tablas

2.1. Clasificación de grafos según sus aristas 12

2.2. Tipos de aprendizaje automático 13

2.3. Revisión de literatura - Grafos 18

2.4. Revisión de literatura - Aprendizaje por refuerzo 20

4.1. Rangos de diferencia de ganancias con Deltha en décimas 46

4.2. Rangos de diferencia de ganancias con Deltha en otras unidades . . . 47

4.3. Promedios de los resultados variando décimas en Deltha 48

4.4. Experimento uno con Deltha 49

4.5. Experimento dos 51

4.6. Promedios de diferencias de ganancias usando Gamma 56

4.7. Medias y varianzas en Promedios de distancias usando Gamma 56

4.8. ANOVA para Distancias en ganancias usando Gamma 57

4.9. ANOVA para el Experimento dos con Gamma 59

4.10. ANOVA para el Experimento tres con Gamma 60

Agradecimientos

Primero doy gracias a mi Padre Celestial que ha permitido que yo viva esta magnífica experiencia y a mi familia terrena que ha apoyado esta aventura.

A mi director, el Dr. Arturo Berrones, responsable del impacto que ha tenido este proyecto, por todo el tiempo y conocimiento dedicado para llegar al fin esperado.

A mi codirectora, la Dra. Paola Sánchez, por su aporte significativo para que este proyecto sea presentado y socializado de forma asertiva.

Al ingeniero Luis Oliverio Chaparro Lemus, quien contribuyó con la construcción un ejemplo de la adaptación de un caso modelado en CMMN, al grafo propio de este trabajo.

A los doctores Leticia Neira, Sara Rodríguez y Romeo Sánchez, evaluadores de esta tesis, por el apoyo que manifestaron desde que conocieron el trabajo que aquí se presenta.

A cada uno de mis profesores del doctorado, por los conocimientos compartidos, por las palabras de ánimo para que siguiera adelante y diera lo mejor de mi, y por su paciencia en cada una de sus clases.

A la Universidad de Boyacá y especialmente, a la Dra. Rosita Cuervo Payeras por conseguir este tipo de alianzas y patrocinios educativos y por realizar los esfuerzos necesarios para mantener el apoyo económico y administrativo en mis estudios doctorales.

Resumen

Carmen Constanza Uribe Sandoval.

Candidato para obtener el grado de Doctorado en Ingeniería con especialidad en Ingeniería de Sistemas.

Universidad Autónoma de Nuevo León.

Facultad de Ingeniería Mecánica y Eléctrica.

Título del estudio: Aprendizaje por Refuerzo para la toma de decisiones en problemas con alta incertidumbre que se puedan modelar por etapas.

Número de páginas: 84.

Objetivos y método de estudio: El objetivo general es diseñar e implementar un algoritmo que apoye la toma de decisiones en el tiempo con pasos discretos, cuando la recompensa de la selección de cada una de las decisiones solo se conoce al final de n pasos, como puede ocurrir con procesos propios de la gestión de casos en los Business Process Management - BPMS. Este se divide en los siguientes objetivos específicos:

1. Establecer las bases teóricas y matemáticas que permitirán la creación del modelo *L-n-armed bandit*, mediante una revisión de las técnicas de Aprendizaje por Refuerzo.
2. Diseñar un modelo matemático que represente la distribución de probabilidad de cada una de las actividades del grafo, así como el modelo de aprendizaje a seguir.
3. Comprobar el funcionamiento del modelo, mediante su implementación en una herramienta tecnológica que permita evaluarlo con valores generados aleatoriamente.

4. Plantear el problema de la gestión de casos en los BPMS, como un grafo por etapas que se ajuste a los características de la solución propuesta.

En cuanto al método, se definieron las siguiente fases para el desarrollo de esta tesis:

Fase 1. Revisión de Investigaciones previas.

Fase 2. Diseño de prototipo.

Fase 3. Construcción de algoritmo.

Fase 4. Pruebas y Análisis de resultados.

Fase 5. Comunicación de resultados.

Contribuciones y conclusiones:

- El modelo propuesto es una implementación, hasta donde sabemos, nueva y muy simple en el campo de aprendizaje por refuerzo usando probabilidades. Se inscribe dentro del campo de los sistemas de decisión de Markov de estado finito y tiempo discreto (por episodios) con un número de tiempos en cada episodio también finito, específicamente para problemas dispuestos en un grafo por etapas, donde se requiere aprender de forma eficiente una trayectoria óptima que debe seguir un agente.
- La estructura del modelo de decisión como grafo dirigido por etapas permite construir un algoritmo en el que las probabilidades de transición entre estados se actualizan directamente a partir de la función de valor estimada para cada nodo, dado que la estructura del grafo permite un cálculo eficiente de la constante de normalización de dichas probabilidades.
- Se alcanzaron todos los objetivos, tanto general como específicos propuestos para esta tesis

Firma del asesor: _____

Capítulo 1

Introducción

En este capítulo se da un contexto del problema que se aborda en esta tesis y se describe la estructura de este documento.

1.1 Descripción del problema científico

Los problemas de decisión están presentes en diversos escenarios de la vida diaria de las personas o de las organizaciones y han sido tema de estudio de la Ingeniería de sistemas y de otras ciencias afines (Parnell *et al.*, 2011). Los problemas dinámicos de decisión se refieren a la elección de una entre varias posibles opciones, donde las decisiones son tomadas de acuerdo a algún orden temporal.

Este proyecto se enfoca en procesos sobre tiempos discretos con decisiones organizadas en un grafo dirigido, es decir, si un agente escoge una alternativa al tiempo t, ello condiciona sus opciones para el tiempo t+1. Además el entorno del agente es fuertemente estocástico, puesto que la ganancia o pérdida de cada decisión tomada se rige de acuerdo con una distribución de probabilidad no conocida. El agente no conoce el efecto de sus decisiones a cada paso sino hasta el final de su recorrido por todo el grafo.

Un ejemplo cotidiano de un proceso de decisión con dichas características pue-

de ser el de elección de hoja de ruta profesional. Una vez seleccionada una carrera profesional, la persona debe ir escogiendo su currículo, optar por lugares para ganar experiencia laboral, elegir entre varias propuestas de empleo y entre lugares geográficos o nichos económicos en los cuales ejercer su profesión. Evidentemente cada selección condiciona las opciones subsecuentes y la persona conoce el valor global de su secuencia de decisiones, tan sólo, luego de un tiempo considerable de ejercicio profesional.

Otro ejemplo más técnico, y con horizonte de tiempo más corto, es un proceso de planificación de secuencias de acciones que permiten alcanzar un objetivo; este proceso es conocido como "planificación automática" y se ubica dentro de las ramas de la inteligencia artificial; tal proceso, tradicionalmente, es modelado mediante un árbol And/Or, donde las acciones o estados de cada uno de sus niveles tienen precondiciones claramente demarcadas, así como un estado o nivel inicial y un estado final u objetivo (que puede ser un multiobjetivo) (Russell y Norvig, 2004).

Pero, la motivación del presente trabajo es, el problema que se tiene en los "casos" o procesos dinámicos que han de gestionar las BPMS (Bussines Process Management Suits), llamados así porque la secuencia de actividades a seguir no está totalmente establecida, como sí lo está para los procesos de negocio deterministas, y es porque cada caso tiene una solución particular. De esta manera, algunas BPMS han dado solución a esta necesidad convocando reuniones virtuales de expertos que recomiendan la secuencia de acciones a seguir de acuerdo al caso.

Si bien, las industrias de BPMS han evolucionado hacia las iBPMS, que son BPMS con características inteligentes, que han logrado apoyar la administración de casos de los negocios (Frece *et al.*, 2012), solo se ha logrado que algunas iBPMS puedan guiar a sus clientes en una secuencia óptima de actividades en dominios específicos de aplicación, como por ejemplo los préstamos para automóviles (Lakshmanan *et al.*, 2010) y el mantenimiento preventivo de automóviles (Khoshafian y Rostetter, 2015), pero se sigue trabajando en una técnica que apoye procesos dinámicos en

general. Pensar en este problema como en uno de decisión de acciones por etapas, donde su éxito o fracaso solo se conocerá al final de ellas, motivó esta investigación.

1.1.1 Sistematización del problema

Basados en la situación descrita, se requiere verificar, desde una base matemática, la posibilidad de dar manejo a la incertidumbre en la toma de decisiones para un problema susceptible de ser representado en un grafo por etapas, así como, establecer su modelo de aprendizaje y la forma en que se ha de implementar; de la misma manera, se hace necesario recomendar cómo el problema de la gestión de casos de los BPMS se puede adaptar al resultado de esta propuesta.

Ante la necesidad de automatizar este proceso de decisiones, se plantea la siguiente pregunta de investigación:

Pregunta problema:

¿Qué características debe tener un algoritmo que apoye la toma de decisiones en el tiempo, con pasos discretos, cuando las recompensas de la selección de cada una de las decisiones solo se conoce al final de l pasos, como puede ser el problemas de la gestión de casos en los BPMS?

Preguntas específicas

1. ¿Cuáles son las bases teóricas y matemáticas del problema *n-armed bandit* del Aprendizaje por refuerzo, que se deben tomar en cuenta para el algoritmo requerido?
2. ¿Cómo simular la toma de decisiones automática, para un problema que se ajuste a las características que se presentan en este documento?
3. Con qué restricciones se puede modelar un caso dinámico para ajustarlo a un problema de decisión como el que se presenta en este documento?

Para resolver estos interrogantes, se plantean los objetivos de la investigación.

1.2 Objetivo general

Diseñar e implementar un algoritmo que apoye la toma de decisiones en el tiempo, con pasos discretos, cuando la recompensa de la selección de cada una de las decisiones del problema solo se conoce al final de l pasos, como puede ser el problema de la gestión de casos en los BPMS.

1.3 Objetivos específicos

1.3.1. Establecer las bases teóricas y matemáticas que permitirán la creación de un modelo *L-n-armed bandit* mediante una revisión de las técnicas de Aprendizaje por Refuerzo. (En la sección 2.2 se explican las razones del nombre dado al modelo).

1.3.2. Diseñar un modelo matemático que represente la distribución de probabilidad de cada una de las actividades del grafo, así como el modelo de aprendizaje a seguir.

1.3.3. Comprobar el funcionamiento del modelo, mediante su implementación en una herramienta tecnológica que permita evaluarlo con valores generados aleatoriamente.

1.3.4. Plantear el problema de la gestión de casos en los BPMS, como un grafo por etapas que se ajuste a los características de la solución propuesta.

1.4 Hipótesis

El modelo basado en aprendizaje por refuerzo *multi-armed bandit* conseguirá dar recomendaciones buenas para la toma de decisiones secuenciales, en procesos con alta incertidumbre que se puedan modelar con grafos, cuyas actividades sean representadas por nodos que se clasifican en etapas disyuntas.

- Variables de entrada: Etapas, actividades por etapas, tareas, tareas predecesoras, objetivo.
- Variables de salida: Ruta de actividades o tareas que se deben seguir para alcanzar el objetivo con mayor certeza.

1.5 Novedad científica

Para México y Colombia, particularmente, se encuentran artículos donde se aplican o evalúan algoritmos de toma de decisiones automáticas como se puede apreciar en Kim *et al.* (2005) y en López Guarín *et al.* (2013), entre otros. Además, se cuenta con universidades y grupos de investigación que tiene como tema de estudio el aprendizaje automático, como son el Centro de Investigación en Física, matemáticas y ciencias de datos del Conacyt en México, el grupo de investigación MindLab de la Universidad Nacional de Colombia y el grupo de investigación MACC de la Universidad del Rosario, también de Bogotá Colombia.

No se encuentra información sobre una propuesta de investigación en la misma dirección del algoritmo que aquí se presenta como idea novedosa, por ser una solución sencilla que se ofrece para problemas de alta incertidumbre en los que se puede pensar en soluciones computacionalmente más costosas.

Lo que hace diferente y original a la propuesta de esta investigación es el grafo

por etapas, que obliga a decidir en cada una de ellas el nodo que se ha de seleccionar, teniendo en cuenta la experiencia de transición que se va adquiriendo desde el nodo de la etapa anterior que ya se haya seleccionado, lo que implica que las decisiones se vean como la selección de un arreglo de nodos, compuesto por un nodo de cada etapa, arreglo para el cual, realmente, se puede estimar su ganancia o pérdida asociada.

Como ya se ha planteado anteriormente, se espera que esta investigación tenga una aplicabilidad relevante e importante para la industria de las BPMS, en cuanto a la gestión de los procesos dinámicos.

1.6 Justificación

En los últimos años, la IA se ha utilizado en la toma de decisiones, para mejorar el desempeño de las organizaciones y ofrecer un creciente portafolio de herramientas para el entretenimiento. Para esta tesis, el aprendizaje juega el papel más importante, pues se constituye como la base para poder recomendar acciones a seguir.

Las técnicas de Aprendizaje Automático - AA vienen siendo perfeccionadas con los resultados de nuevas investigaciones, como es el caso del Deep Learning – DL, que mejoró el desempeño de las Redes Neuronales. El Aprendizaje por Refuerzo - AR también se ha venido utilizando para ofrecer apoyo en sistemas de recomendación, caracterizados por su alta incertidumbre.

Una propuesta nueva que busque seguir dando apoyo a la toma de decisiones secuenciales con el mismo o un mejor desempeño, o con un menor costo computacional, será un aporte valioso a la tecnología y a la ciencia

La sencillez del modelo propuesto es, quizás, el aspecto más importante de la propuesta, ya que el algoritmo logra hacer búsquedas en escenarios con alta incertidumbre, que, eventualmente pueden requerir herramientas más robustas, ofreciendo de esta forma una alternativa para estos casos.

Finalmente, esta investigación puede sentar las bases de investigaciones futuras, donde quizás se pretendan solucionar problemas específicos de decisiones, cuya complejidad no haya permitido acercamientos satisfactorios para los usuarios.

1.7 Resultados esperados

Esta sección presenta los productos que se esperan como resultado de esta tesis.

- Entrega de un modelo validado con datos aleatorios, que sirva de insumo a otras investigaciones tecnológicas.
- Socialización de resultados en un artículo producto del trabajo colaborativo entre docentes vinculados a los procesos investigativos de la Universidad Autónoma de Nuevo León y la Universidad de Boyacá.

1.8 Metodología

1.8.1 Enfoque general del método

Se selecciona inicialmente el enfoque cualitativo, ya que se sigue una lógica inductiva donde, primero se explora una serie de teorías que pueden servir para solucionar el problema, segundo se describe cómo se utilizarán y, tercero se generan perspectivas teóricas concluyentes (Hernández Sampieri *et al.*, 2010). Teniendo ya la propuesta objeto de esta investigación, se procede con un enfoque cuantitativo, que aterriza la pregunta de investigación y la hipótesis del trabajo y que trata de deducir o generalizar la utilidad misma del modelo para dejarlo a disposición de la comunidad investigativa.

1.8.2 Técnicas específicas seleccionadas

Dentro del enfoque cualitativo, esta investigación se ubica en el ámbito los estudios exploratorios, donde, según Hernández Sampieri *et al.* (2010), se "investigan problemas poco estudiados", o se "indagan desde una perspectiva innovadora", que son dos de las características de este trabajo. Específicamente se utiliza aquí la lectura, la discusión y el análisis, como técnicas para avanzar en esta primera parte de la investigación de forma inductiva.

La parte que se enmarca dentro del enfoque cuantitativo, tiene que ver con los procesos deductivos o probatorios (Hernández Sampieri *et al.*, 2010), que permiten consolidar la hipótesis y la pregunta problema para esta investigación. El alcance se define como exploratorio, con un diseño experimental que utiliza la simulación para conocer el comportamiento de algunas variables de acuerdo a unos datos generados aleatoriamente, siguiendo una distribución de probabilidad normal.

La simulación se emplea para validar el desempeño del modelo con datos aleatorios, lo que permite la creación de escenarios de diferentes características y la comprobación de que el modelo no se ve afectado por tal aleatoriedad.

Se analizan las relaciones entre una variable independiente y dos dependientes, y los efectos causales mediante los resultados del análisis de varianza y las medias. Se tiene en cuenta la hipótesis planteada para la tesis.

No se utilizan datos de casos como los que inspiraron esta investigación, porque no es factible encontrar un conjunto de estos procesos, que se repitan, donde se establezca si se alcanzan o no los objetivos, pero se ofrece una aproximación a la forma en que se puede modelar un caso como un grafo como el propuesto.

1.8.3 Fases de la metodología propuesta

Fase 1. Revisión de Investigaciones previas. En esta etapa se pretende establecer la existencia de trabajos similares al que se pretende desarrollar y recopilar la información necesaria para establecer los fundamentos matemáticos que han servido de base al aprendizaje por refuerzo, para, de esta forma, establecer los elementos del sistema a modelar, como son sus variables de entrada, la salida y sus relaciones.

Fase 2. Diseño de prototipo. Aquí se hará el diseño de:

- El modelo gráfico que permitan representar el problema (grafo)
- El modelo de probabilidades de transición entre nodos
- El modelo de aprendizaje

Fase 3. Construcción de algoritmo. Aquí se hará la escogencia de las herramientas computacionales que faciliten la implementación requerida y se desarrollará la aplicación para el modelo.

Fase 4. Pruebas y Análisis de resultados. Aquí se definen los valores para las pruebas, se hace su aplicación y se hacen el análisis y la discusión correspondientes para establecer la funcionalidad de la aplicación.

Fase 5. Comunicación de resultados. Aquí se planteará el ajuste de un caso dinámico, al modelo expuesto en esta investigación y se escribirán los artículos de resultados de la investigación para que sean presentados a la comunidad científica.

1.9 Estructura del documento

El documento de tesis consta de esta introducción, donde se plasman los componentes que fundamentan el trabajo desarrollado, como son la presentación del

problema de investigación que se aborda, la justificación, la novedad científica, los objetivos, la hipótesis, los resultados esperados y la metodología que se sigue para este proyecto, entre otras.

Posteriormente, se presentan algunos antecedentes de la gestión de procesos de negocios, de forma que se entienda el origen de la idea de los grafos por etapas y de su solución con técnicas de Aprendizaje por refuerzo, en su caso básico de los *multi-armed bandit*.

Seguidamente, se desarrolla la metodología prevista que inicia con la revisión de literatura, y se continúa con los fundamentos matemáticos, así como los aspectos de diseño, tanto del grafo como del modelo de aprendizaje, finalizando con su implementación computacional.

Las pruebas realizadas al modelo y los resultados obtenidos se presentan como una serie de experimentos que dan cuenta de la forma como se avanza en la concreción de la propuesta y se finaliza con las conclusiones generales y las referencias bibliográficas.

Así, esta introducción ha presentado las bases de la propuesta de tesis que se desarrolla como requisito para alcanzar el título de doctorado.

Capítulo 2

FUNDAMENTOS

Se dedica este capítulo a exponer los conceptos necesarios para entender el problema de investigación que se aborda.

2.1 Conceptos básicos y antecedentes

En este aparte se presentan algunos conceptos que puedes ser básicos, pero que permiten la contextualización del problema de esta tesis, así como un resumen de las publicaciones que soportar esta investigación y su aplicación en el problema que se aborda.

Preferiblemente se tomaron las referencias de los últimos 5 años, pero no se descartan otras que son básicas en este campo de investigación.

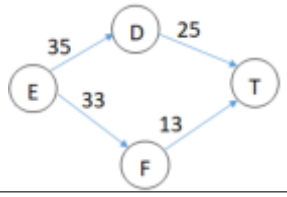
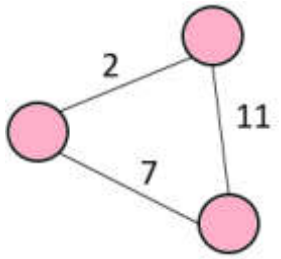
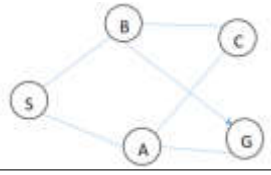
2.1.1 Conceptos básicos de Grafos

Un grafo no es otra cosa que un conjunto de vértices (al menos uno) conectados entre ellos con aristas que pueden ser líneas o flechas.

Los grafos pueden ser dirigidos o no dirigidos, lo que gráficamente significa que

sus aristas son flechas o líneas, respectivamente; si hay de ambas, se llaman grafos mixtos, como se aprecia en la tabla 2.1.

Tabla 2.1: Clasificación de grafos según sus aristas

TIPO	DIBUJO	CARACTERÍSTICA
Dirigido		Todas sus aristas están dirigidas (flechas), que se pueden representar por parejas ordenadas (x,y)
No dirigido		Todas sus aristas son no dirigidas (líneas), que se pueden representar por parejas no ordenadas {x,y}
Mixto		Contiene aristas dirigidas y aristas no dirigidas

Además, matemáticamente se hace una distinción en la representación de sus aristas, usándose parejas ordenas $(x, y) \in NxN$ para las aristas dirigidas y $x, y|(x, y) \in NxN$ para las aristas no dirigidas, lo que se puede interpretar como una doble flecha (x, y) y (y, x) (Tremblay y Grassmann, 1996).

También se habla de grafos conexos cuando no se puede separar en componentes o partes entre las que no aparecen aristas que las conecten, es decir que cada par de nodos debe estar conectado mediante un secuencia de aristas o mediante un camino.

Los grafos bipartitos son aquellos en los que se pueden diferenciar claramente dos grupos de nodos; en cada grupo de nodos no se encuentran aristas que los conecte, sino que estas van un nodo de un grupo al otro. Estos y otros conceptos sobre grafos se pueden ampliar en (Brandstädt *et al.*, 1999).

Estas características se cumplen en el grafo que se propone en esta tesis, además de una característica propia que se ha denominado «grafo por etapas», la cual consiste en que el grafo cuenta con sus nodos distribuidos en L etapas claramente diferenciadas, de forma similar a los grafos bipartitos, pero con más de dos grupos de nodos.

Esta estructura por etapas facilita el recorrido estocástico desde la primera hasta la última etapa, en orden y con probabilidades asociadas a las conexiones entre nodos, que se prestan para realizar búsquedas en el grafo, basadas en muestreo.

2.1.2 Aprendizaje automático

El Aprendizaje de Automático es una de las áreas de la Inteligencia Artificial que ha permitido extraer una importante cantidad de información, a partir de los datos que cada día se manejan en los negocios y en el mundo en general. Algunos trabajos, en este sentido, se han apoyado en diferentes técnicas como las redes Neuronales, los Algoritmos Genéticos, las Colonias de Hormigas, el Soporte de Máquina Vectorial y el Aprendizaje por Refuerzo, entre otras.

El Aprendizaje automático se encuentra catalogado en los textos como: aprendizaje supervisado, no supervisado o mixto, aunque el aprendizaje por refuerzo - AR no se encuentra realmente en ninguna de estos tipos (Russell y Norvig, 2004). En la tabla 2.2 se resume esta clasificación.

Tabla 2.2: Tipos de aprendizaje automático

APRENDIZAJE	**CARACTERÍSTICA**
Supervisado	Con datos de entrenamiento
No supervisado	Sin datos de entrenamiento
Por refuerzo	Premio o castigo

En el aprendizaje supervisado el modelo aprende de acuerdo a unos datos de

entrenamiento y posteriormente es capaz de reconocer patrones similares, mientras que, el aprendizaje no supervisado, no cuenta con datos de entrenamiento, sino que descubre automáticamente las características de los datos, logrando establecer, por ejemplo, las clases en las que pueden agruparse.

El AR se emplea en situaciones donde hay que tomar decisiones bajo incertidumbre, donde solo se conoce la consecuencia de una decisión después de haberla tomado. Aquí se trata de entrenar un agente inteligente, el cual debe escoger la política que mejor resultado le dé, de acuerdo a las recompensas que recibirá de un ambiente donde se encuentra inmerso (Sutton *et al.*, 1992).

2.1.3 Los Bandits

El Multiarmed-Bandit o *n-armed bandits* es un modelo inspirado en las máquinas de un casino que se conocen con este nombre, donde se tienen cierto número de brazos, de los que se espera jugar los que mejor resultado alcancen. Cada máquina tiene su propia distribución de probabilidad; el jugador no las conoce inicialmente, pero a medida que va jugando, se va dando cuenta del resultado y va aprendiendo.

El modelo básico es el *armed-bandit*, también conocido como *bandit*, concepto que se utiliza en esta tesis, cuya analogía corresponde a una máquina de casino con un solo brazo o palanca, que en adelante se llamará acción. Para este modelo se tienen fórmulas sencillas que facilitan su implementación.

Por ejemplo, el valor que tendrá seleccionar una acción a en el tiempo t se estima mediante un promedio sencillo de las recompensas que el autómata ha recibido del ambiente cada vez que seleccionó dicha acción a, recompensa que no necesariamente será la misma en cualquier otra oportunidad o tiempo. La fórmula que se emplea es la 2.1, donde R_i son los valores de recompensa que va recibiendo el

autómata en cada una de las $Nt(a)$ muestras en que la acción a fue seleccionada.

$$Q_t(a) = \frac{R_1 + R_2 + ... + R_{Nt(a)}}{N_t(a)} \tag{2.1}$$

Este valor asociado a cada acción, será actualizado durante el aprendizaje del autómata, es decir que para cada tiempo t puede cambiar y, de él depende que la acción tenga mayor posibilidad de ser seleccionada por el autómata en ese momento, de acuerdo a lo que muestra la figura 2.2 (Sutton *et al.*, 1998).

$$A_t = \underset{a}{argmax} Q_t(a) \tag{2.2}$$

Así, no es completamente seguro que la acción con mejor valor sea escogida, porque estos algoritmos, y en general los del AR, hacen un balance entre la exploración y la explotación y el agente deberá decidir entre elegir acciones que ya sabe que dan un buen resultado o acciones que no ha probado o que en el pasado, no han sido las mejores; en el primer caso, para mejorar la recompensa que esa acción le ofrece y, en el segundo caso, para darse la oportunidad de encontrar acciones con mejor recompensa (Sutton *et al.*, 1992). Una analogía de estos procesos es la búsqueda de un valor óptimo en una función; en el primer caso el agente se quedaría explorando un óptimo local y, en el segundo caso exploraría otros lugares de la función donde puede que aparezca el óptimo global.

2.1.4 Antecedentes sobre Grafos

Diferentes investigadores han propuesto aplicaciones de grafos en problemas particulares, así como la mejora o adaptación de algoritmos ya existentes.

Entre ellos, Zhou *et al.* (2019) proponen mejoras a algoritmos de enrutamiento de ruta más corta (Shortest Path Routing - SPR), mediante sondeos de rutas múlti-

ples y aprendizaje colaborativo. Citan, además, trabajos en esta misma área como el de Liu y Zhao (2012) quienes aplican los principios del Bandido Multi-Brazos con brazos independientes, al problema presentado por Liu y Zhao (2011), quienes pretenden recomendar la mejor ruta en un grafo, desde un origen hasta un destino, donde el costo de cada enlace individual no se puede ver y el costo total de extremo a extremo solo se puede observar al finalizar el recorrido y está dado por la suma de los costos de todos los enlaces en la ruta.

Un problema similar, conocido como *Online Shortest Path Problem*, es abordado como trabajo de grado por Ávila Cartes (2018), mediante el uso de un grafo dirigido y sin ciclos, con un nodo inicial y uno final o nodo sumidero. La solución aprovecha los alcances del *n-armed.bandit*, pero, además de minimizar el costo con el camino que se seleccione, busca disminuir la cantidad de veces que se escoge uno que presente fallas.

Otra tesis donde se recogen problemas de aplicación de grafos y *bandits*, es la de Valko (2016) quien se esfuerza por mostrar la aplicación de modelos que investigadores han desarrollado en problemas reales, valiéndose de los grafos y cómo hacer uso de los desarrollos en *n-armed-bandit*.

Es el caso de Tossou *et al.* (2017), ellos proponen un algoritmo para problemas de decisión secuenciales, con grafos que presentan altos grados de incertidumbre, basado en los enunciados de Thompson (1933), como alternativa para las decisiones que se toman basados en sencillos cálculos de probabilidades, donde uno de dos eventos queda aceptado y el otro descartado.

Se cita también una propuesta de investigadores en inteligencia artificial de IBM (Lakshmanan *et al.*, 2010) que modelan un caso semiestructurado de Gestión de Procesos de Negocios, mediante un grafo cuyos nodos son las actividades y sus aristas indica si hay flujo entre cada par de nodos. Se tiene la información de quienes son los vecinos de cada nodo y una probabilidad de pasar a cada uno de ellos, las cuales deben sumar 1; estas son las probabilidades que la aplicación va a ir modificando

hasta encontrar una sola secuencia de actividades para dar solución al caso; dicha modificación sigue las reglas de las feromonas de los algoritmos de Optimización con Colonia de Hormigas para encontrar cuál nodo ha de recibir la recompensa y para actualizar todas las probabilidades de acceder a los vecinos de un nodo. A medida que van disminuyendo los valores de las probabilidades para algunas aristas, también llegará el momento en que algunos nodos ya no sean tenidos en cuenta.

En este trabajo como en otros, el grafo propuesto no implica mayores características que ser un grafo conexo y dirigido, con un nodo origen y un nodo final establecidos, sin encontrar en dichos trabajos las características del grafo por etapas, objeto de esta tesis.

Un resumen de esta revisión bibliográfica se presenta en la tabla 2.3

2.1.5 Antecedentes sobre aprendizaje por refuerzo

Muchas investigaciones se han dado para mejorar o adaptar los métodos de aprendizaje por refuerzo a diferentes situaciones.

Una propuesta para mejorar el tiempo de decisión de un algoritmo *AQ-learnig* de aprendizaje por refuerzo con técnicas de inteligencia artificial de búsqueda por colonia de hormigas, se puede consultar en (Lee y Chung, 2005).

Se encuentra también la propuesta de Alon *et al.* (2017), quienes, mediante los principios de Regresión, crean un espectro de modelos cuya complejidad está entre la de los problemas donde el agente conoce únicamente la recompensa de las acciones luego de que las va eligiendo, y la de los problemas donde el agente tiene la información de lo que ocurrirá con cualquier acción candidata en cualquier instante de tiempo.

De otra parte, Alon *et al.* (2015) modelan el problema de un jugador aleatorio que se encuentra dentro de un ambiente adversario, de forma que después de cada

Tabla 2.3: Revisión de literatura - Grafos

FUENTE	DESCRIPCIÓN
Zhou *et al.* (2019)	Proponen mejoras a algoritmos de enrutamiento de ruta más corta (SPR), mediante sondeos de rutas múltiples y aprendizaje colaborativo.
Liu y Zhao (2011)	Recomendar la mejor ruta en un grafo, desde un origen hasta un destino donde el costo de cada enlace individual no se puede ver y el costo total de extremo a extremo solo se puede observar al finalizar el recorrido y está dado por la suma de los costos de todos los enlaces en la ruta. No hay etapas con nodos disyuntos.
Liu y Zhao (2012)	Aplican los principios del Bandido Multi-Brazos con brazos independientes al problema presentado por Liu y Zhao (2011)
Ávila Cartes (2018)	Problema *Online Shortest Path Problem*, donde se usa un grafo dirigido y sin ciclos, con un nodo inicial y uno final o nodo sumidero. Haciendo uso de *n-armed.bandit*, además de minimizar el costo con el camino que se seleccione, se desea disminuir la cantidad de veces que se escoge uno que presente fallas.
Valko (2016)	Recoge problemas de aplicación de grafos y *bandits*, mostrando la aplicación de modelos que investigadores han desarrollado en problemas reales
Tossou *et al.* (2017)	Algoritmo para problemas de decisión secuenciales, con grafos que presentan altos grados de incertidumbre, como alternativa para cuando las decisiones se tomaban basados en sencillos cálculos de probabilidades, quedando uno de dos eventos aceptado y el otro descartado.

acción, el jugador recibe de sus vecinos información binaria del efecto de su acción, que le permitirá mejorar sus próximas decisiones.

Gokcesu y Kozat (2018) presentan un algoritmo para *n-armed bandits* con complejidad lineal que crece en función de la cantidad de secuencias posibles y del número de rondas del juego, sin el conocimiento del brazo que se debería escoger y sin suposiciones estadísticas.

Así mismo, en (Xu *et al.*, 2017) se encuentra un modelado de un problema de aprendizaje por refuerzo, mediante grafos, no con el esquema tradicional de la

secuencia de los brazos seleccionado, sino que es un grafo en el que los nodos sí representan los brazos, pero las aristas tiene que ver con la similitud de sus valores de recompensas, lo que es aplicable a los sistemas de recomendación. El modelo les permitió presentar un algoritmo con baja complejidad.

Como Alon *et al.* (2017) y Xu *et al.* (2017), muchos autores han hecho propuestas para mejorar el desempeño de algoritmos existentes, en cuanto a su complejidad o al campo de solución que influyen, utilizando técnicas de aprendizaje por refuerzo en el escenario de los *bandits*. En general, se trabaja con un *m-armed bandit*, donde los nodos representan los brazos y, las aristas, dirigidas o no dirigidas, relacionan los nodos.

Y, finalmente, no se puede dejar de mencionar el trabajo de Silver *et al.* (2017) y los predecesores de *AlphaZero* (Silver *et al.*, 2016) que revolucionaron al mundo con la aplicación de técnicas de aprendizaje por refuerzo para mejorar el entrenamiento conseguido con redes neuronales de aprendizaje profundo, que logró derrotar a los mejores jugadores humanos de ajedrez y Go, entre otros juegos, cada vez con mejores resultados.

De esta forma se puede apreciar que en la revisión hecha, no se encuentra una propuesta que combine los aspectos del aprendizaje por refuerzo de los *m-armed bandit*, con un grafo que tenga como restricción que la acción seguida a ejecutar deba pertenecer a la etapa siguiente de la anterior, como lo contempla la propuesta de esta tesis.

Un resumen de esta revisión bibliográfica se presenta en la tabla 2.4

Luego de estas revisiones se procede a describir las bases matemáticas que soportan el modelo *L-n-armed-bandits*, para que se constituya en una propuesta novedosa que se sume a las muchas que se han generado para contribuir cada día a una mejor toma de decisiones, para las personas y para las aplicaciones mismas.

Tabla 2.4: Revisión de literatura - Aprendizaje por refuerzo

FUENTE	DESCRIPCIÓN
Lee y Chung (2005)	Propuesta para mejorar el tiempo de decisión de un algoritmo *AQ-learnig* de aprendizaje por refuerzo con técnicas de inteligencia artificial de búsqueda por colonia de hormigas.
Alon *et al.* (2017)	Mediante los principios de Regresión, crean un espectro de modelos cuya complejidad está entre la de los problemas donde el agente conoce únicamente la recompensa de las acciones luego de elegirlas, y la de aquellos donde el agente tiene la información de lo que ocurrirá con cualquier acción candidata en cualquier instante.
Alon *et al.* (2015)	Modelan el problema de un jugador aleatorio que se encuentra dentro de un ambiente que puede ser su adversario, quien después de cada acción recibe de sus vecinos información binaria del efecto de su acción.
Gokcesu y Kozat (2018)	Presentan un algoritmo para *n-armed bandits* con complejidad lineal que crece en función de la cantidad de secuencias posibles y del número de rondas del juego, sin el conocimiento del brazo que se debería escoger y sin suposiciones estadísticas.
Xu *et al.* (2017)	Modelado de un problema de aprendizaje por refuerzo, mediante grafos, con un grafo en el que los nodos sí representan los brazos, pero las aristas tiene que ver con la similitud de sus valores de recompensas, aplicable a los sistemas de recomendación con baja complejidad
Silver *et al.* (2017) y Silver *et al.* (2016)	Aplicación de técnicas de aprendizaje por refuerzo y redes neuronales de aprendizaje profundo, que lograron derrotar a los mejores jugadores humanos de ajedrez y Go, entre otros juegos, cada vez, con mejores resultados.

2.2 Denominación del modelo propuesto

El algoritmo propuesto se ha llamado *L-n-armed bandit* ya que se basa en el algoritmo *n-armed bandit* para un grafo por etapas (ya definido), cuya cantidad está representada por la letra L

Cabe resaltar que este grafo ha de contener la cantidad de etapas L que el

problema a modelar requiera, así como los nodos de cada etapa y las conexiones entre nodos de etapas seguidas; sin embargo, para el desarrollo de la propuesta se generarán de forma aleatoria algunos de estos valores, cuya digitalización puede resultaría tediosa en cada una de las pruebas a que se someta.

Esta estructura por etapas facilita el recorrido estocástico desde la primera hasta la última etapa, en orden y con probabilidades asociadas a las conexiones entre nodos, lo que se presta para realizar búsquedas en el grafo, basadas en muestreo. Para conseguir estas probabilidades se aprovecha la forma de operar del algoritmo *n-armed bandit*.

Cada uno de los nodos del grafo será en realidad un *bandit* con un valor o utilidad desconocido que solo se podrá ir estimando al terminar la selección de los L nodos de todas las etapas del grafo, ya que en la última etapa es donde se recibe algún refuerzo positivo o negativo, de acuerdo al conjunto de nodos seleccionados de cada etapa.

Las forma como se generan valores y se simulan los refuerzos, así como las bases matemática que soportan la propuesta se describen en seguida.

2.3 Fundamentos matemáticos

Como ya se ha mencionado, se pretende usar un modelo estructurado que aproxime una matriz de probabilidades de transición de estados, que se relacionen con la función de valor que en cada tiempo tengan los nodos.

Un modelo de probabilidades de transición de estados se adapta al problemas de búsqueda en el grafo por etapas con cualquier cantidad de nodos y remplazaría la forma de seleccionar una acción que se explicó para los *bandits* en las ecuaciones 2.1 y 2.2 (Sutton *et al.*, 1998).

También se ha de tener en cuenta que, la probabilidad total de que se dé una

secuencia muestral de nodos, equivale a multiplicar las probabilidades de ir de un nodo a otro hasta culminar la ruta, tal como se hace en un árbol Bayesiano y como se presenta en la fórmula 2.3.

$$P(i_0 \to i_1, ...i_{l-1} \to i_l, ...i_{L-1} \to i_L) = P(i_0 \to i_1)P(i_1 \to i_2)...P(i_{L-1} \to i_L), \quad (2.3)$$

donde i denota el nodo seleccionado en cada una de las etapas, su subíndice $l = 0, 1, ..., L$ denota la etapa a la que pertenece y L denota el número total de etapas.

De otra parte, para calcular dichas probabilidades, se debe usar una función de normalización que asegure valores entre 0 y1, como es característico de la distribución *softmax* (*Boltzmann*), que se visualiza en 2.4 (website, 2016).

$$P(i \to j) = \frac{e^{v_j}}{\sum_k A_{i,k} e^{v_k}}, \quad (2.4)$$

2.4 Proceso de negocio

Aunque la definición de proceso es algo que está en la mente de cualquiera de los lectores, se define aquí un proceso de negocio como un conjunto de actividades ejecutadas en una secuencia específica, es decir, que tiene un flujo determinado por la lógica del negocio, los eventos externos y las reglas del negocio (Hitpass, 2017), para dar un contexto inicial del contenido de este capítulo. A partir de este concepto se derivan los que aquí se exponen.

2.4.1 Gestión de procesos de negocio

BPM es el acrónimo de Business Process Management, en español: Gestión de Procesos de Negocio. Es una disciplina que integra un conjunto de principios,

métodos y tecnologías con el propósito de contribuir al mejoramiento continuo del funcionamiento empresarial.

La idea de BPM es hacer visible la gestión de los procesos de negocio y facilitar los cambios que sean requeridos (Smith y Fingar, 2003).

Según Garimella *et al.* (2008), BPM hace referencia a un conjunto de mejores prácticas de gestión de procesos, herramientas y tecnologías utilizadas para diseñar, representar, analizar y controlar los procesos del negocio, combinando las tecnologías de la información con metodologías de proceso y gobierno.

BPM incluye el soporte integral de las tecnologías de información para mejorar, innovar y gestionar los procesos que determinan los resultados del negocio, crean valor para el cliente y facilitan el logro ágil de los objetivos del negocio" (ABPMP, 2013).

2.4.1.1 Suite de gestión de procesos de negocio bpms

Un sistema o suite para la gestión de procesos de negocios (Business Process Management Suite (BPMS) "es un conjunto de herramientas de software que permiten modelar implementar y gestionar los procesos de negocio, que abarcan múltiples aplicaciones empresariales, departamentos y *partners*"(Smith y Fingar, 2003).

Una Suite de Gestión de Procesos de Negocio está conformada por herramientas de software para la gestión de los procesos de negocios (diseño de procesos, flujo de trabajo, aplicaciones, integración y supervisión), los cuales son automatizados favoreciendo a las organizaciones (Underdahl, 2013).

2.4.2 Proceso de negocio dinámico y caso

Es un proceso de negocio que no tiene un orden determinado para la ejecución de las actividades, ni la certeza de cuáles de ellas se han de ejecutar, requiriendo la intervención de un experto (Hitpass *et al.*, 2017).

El término *case*, que en Van der Aalst *et al.* (2005) se define como una situación que puede ocurrir en una organización, para la cual el procedimiento de resolución no está necesariamente predefinido.

A la tecnología que se encarga de la gestión de este tipo de procesos se le conoce como Gestión de "Casos"; el caso contiene toda la información sobre el proceso (Marin, 2016).

2.4.3 CMMN

Uno de los estándares emergentes para el modelado de casos es Case Management Model and Notation (CMMN), que utiliza un conjunto de símbolos gráficos, reglas de composición y artefactos para este propósito; una descripción completa de esta notación se encuentra en Group (2013), pero se enfatiza que las líneas de puntos alrededor de las actividades las hacen opcionales y esto revela la incertidumbre que caracteriza a los casos.

Case Management Model and Notation" – CMMN es una referenciación entregada por el grupo "Object Management Group" (OMG), como una notación gráfica para la gestión de casos y procesos dinámicos (Hauder *et al.*, 2014); algunas de sus principales diferencias con la notación de procesos de negocio deterministas se ilustra en Breitenmoser y Keller (2015). Auer *et al.* (2014) también fundamenta y explica la notación CMMN.

En Marin (2016) se da una aplicación de esta notación a un sistemas de atención

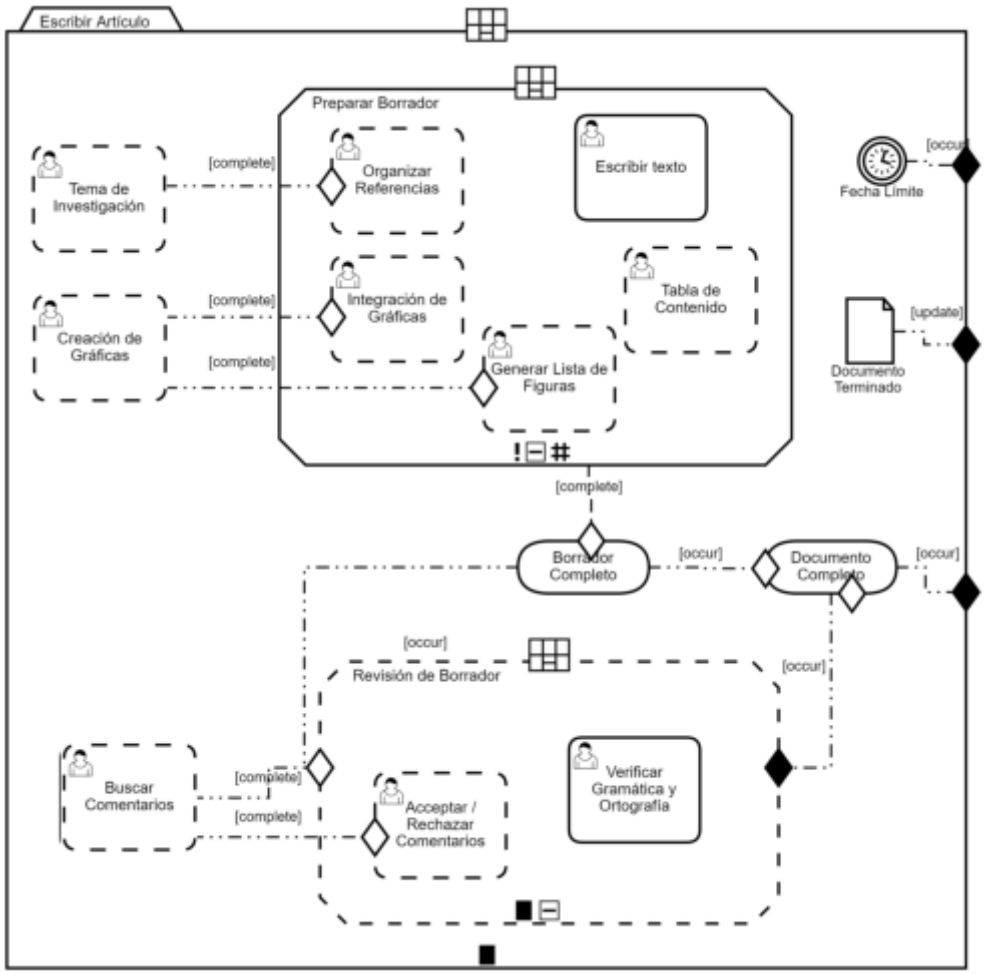


Figura 2.1: Modelado de un proceso dinámico con CMMN

de quejas. Breitenmoser y Keller (2015) presenta un modelo en notación CMMN para seleccionar candidatos que aplica a un proceso de naturalización en un gobierno local. Tanto en Auer *et al.* (2014) como en Omg *et al.* (2011) se expone una adaptación del ejemplo que se ve en la figura 2.1, el cual corresponde al proceso de escribir un documento.

Al hacer la comparación de la notación para el modelado de los procesos estructurados contra la notación de los procesos dinámicos, en el ejemplo que se adaptó de Group (2016), se establece que las líneas punteadas de la figura 2.1 representan la incertidumbre en cuanto a la secuencia que se puede seguir, e incluso a la presencia o no de algunas actividades en la solución a la que se llegue.

Capítulo 3

Aplicación del método

El desarrollo de la metodología propuesta en el proyecto es el capítulo central de este documento, aquí se presenta el diseño y la implementación de los algoritmos desarrollados en el proyecto.

3.1 Diseño del grafo

Inspirados en el grafo del caso de estudio descrito en la sección 2.4.3, se observó que en la notación de modelado CMMN (ver figura 2.1) se cuenta con un conjunto de tareas o actividades y de flujos entre ellas, que pueden ser opcionales. Se vio cómo esta situación podía modelarse en un grafo, en el que el problema a solucionar consiste en qué tareas realizar de **L** etapas consecutivas, para alcanzar al final el objetivo.

Un grafo dirigido y conexo permite modelar la estructura de procesos de decisión por etapas de estado y tiempo finitos, como el quw se aprecia en la figura 3.1, cuyos nodos, entre cada dos etapas, tienen las características de un grafo bipartito, es decir que, entre acciones de la misma etapa no deben existir aristas; además, como es conexo, se garantiza que exista al menos una arista que conecte dos etapas sucesivas, así, cada nodo, (excepto los de la última etapa), debe estar conectado al

menos con un nodo de la etapa siguiente.

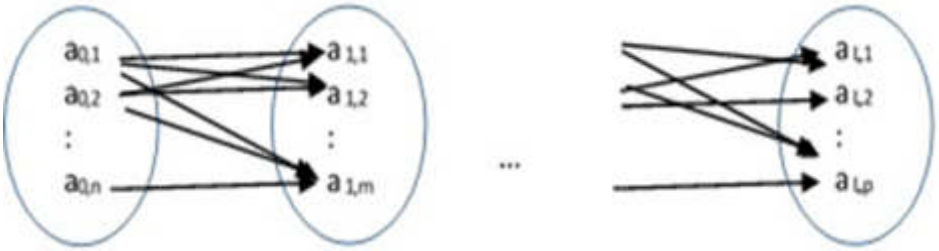


Figura 3.1: Grafo por etapas

El grafo es un «grafo por etapas», característica definida en la sección 2.1.1, y el problema que se quiera solucionar debe modelarse como este grafo, con número fijo L de etapas, para que sea susceptible de emplear la aplicación propuesta en esta tesis, por medio de la cual ha de encontrar un camino recomendable para alcanzar un objetivo dado.

3.2 Diseño del modelo matemático

Ya se ha dicho que cada uno de los nodos del grafo se constituye en un *bandit*, pero se desconoce la función de valor, o utilidad, que se obtendrá al pasar por cada uno de ellos y solo se podrá estimar al finalizar la secuencia total de acciones que se decida tomar, es decir, al final de las **L** etapas.

Además, como la probabilidad total de una ruta determinada, equivale a multiplicar las probabilidades de ir de un nodo a otro hasta culminar la ruta, tal como se hace en un árbol Bayesiano y como se presenta en la fórmula 3.1, al tener el valor de recompensa de haber seleccionado un camino, se decide afectar las probabilidades de pasar entre cada uno de sus nodos, lo que se reflejará en una matriz de probabilidades de transición de estados, donde los estados son los nodos del grafo.

$$P(i_0 \to i_1, ...i_{l-1} \to i_l, ...i_{L-1} \to i_L) = P(i_0 \to i_1)P(i_1 \to i_2)...P(i_{L-1} \to i_L), \quad (3.1)$$

donde i denota el nodo seleccionado en cada una de las etapas, su subíndice $l = 0, 1, ..., L$ denota la etapa a la que pertenece y L denota el número total de etapas.

Cuando $l = L$ se habrá acumulado una ganancia G, que puede ser positiva o negativa. Dado este valor, se hace una modificación a cada una de las probabilidades de transición involucradas y se constituye en el modelo de aprendizaje que se explicará en la sección 3.3.

Para el cálculo de estas probabilidades se propone la fórmula 3.2, que, como sucede con la distribución de *SoftMax*, permite generar las probabilidades, las que correspondan a cantidades positivas entre 0 y 1, pero con un fácil cálculo de la sumatoria del denominador, ya que los valores de A corresponden a los ceros y unos de la matriz de adyacencia del grafo.

$$P(i \to j) = \frac{e^{v_j}}{\sum_k A_{i,k} e^{v_k}}, \tag{3.2}$$

donde A es la matriz de adyacencia del grafo y, el valor binario de $A_{i,k}$ permite que solo se tengan en cuenta nodos alcanzables desde el nodo i; además, v es la función de valor asociado a cada uno de los nodos del grafo, la cual se va actualizando en cada una de las iteraciones τ como se muestra en la ecuación 3.3, valor que no necesariamente está relacionado con la recompensa asociada a ese nodo (*bandit*), sino que toma en cuenta la utilidad del nodo para alcanzar el objetivo.

La cantidad del numerador siempre será inferior o, a lo sumo, igual a la cantidad del denominador, dado que el numerador es un término de la sumatoria que aparece en el denominador, la cual está conformada por valores positivos; esto garantiza que los resultados estén en el rango entre 0 y 1; adicionalmente, la suma de todas las probabilidades para los k valores que puede tomar j será 1, lo que corresponde a las probabilidades de transición que se tienen al estar en un nodo i; y, la multiplicación por el valor de cada posición en la matriz de adyacencia A, garantiza que únicamente se sumen los exponenciales de nodos adyacentes.

La función de valor v para los nodos se inicializa en cero, de forma que en la primera iteración la fórmula 3.2 calcula la misma probabilidad para pasar del nodo i a cada uno de sus siguientes, es decir, una probabilidad uniforme; de esta forma se garantiza que la decisión inicial de pasar del nodo i a cualquier otro nodo j, es totalmente aleatoria.

La probabilidad de transición del nodo i al nodo j va a estar en adelante influenciada por la función de valor v asociada a los nodos, como se explica en la sección 3.3.

3.3 Diseño del modelo de aprendizaje

El modelo busca orientar al agente para que sepa en cada instante cuál es el nodo de la etapa siguiente que a la larga ha de dar el mayor beneficio, lo que se constituye en una particularización de un problema de decisión de Markov donde se hace un muestreo de nodos en cada iteración y se aprovecha la estructura del Grafo por etapas

Cada uno de los nodos es un *bandit* con un valor asociado; en la etapa inicial se selecciona el nodo origen y en cada etapa siguiente se selecciona un nodo adyacente al ya seleccionado, inicialmente de forma aleatoria porque todas las decisiones inician con probabilidad uniforme, y posteriormente haciendo uso de una matriz de probabilidades de transición de estado, que el autómata irá modificando de acuerdo a su aprendizaje.

Al culminar la secuencia de nodos de las L etapas, se recibe una recompensa que indicará si se ha alcanzado o no el objetivo. De acuerdo a esta recompensa, la función de valor v de cada nodo de la ruta será ponderada para la siguiente iteración sumando un δ, que puede ser positivo o negativo, como se muestra en la fórmula 3.3

$$v_j(\tau + 1) = v_j(\tau) + \delta, \tag{3.3}$$

donde δ es un parámetro que controla la rata de aprendizaje, ya que $P(i \to j)_{\tau+1} \sim e^{\delta} P(i \to j)_{\tau}$ para pequeños valores de δ.

Luego de varias iteraciones, la matriz de probabilidades tiende a estabilizarse, es decir, sus valores no cambian considerablemente; en ese momento, el modelo ya ha aprendido una ruta específica que reconocerá como la recomendada.

Con la nueva función de valor v de los nodos se actualizan las probabilidades de selección, de acuerdo a la fórmula 3.2, y los nodos favorecidos por estas probabilidades ajustarán su función de valor v afectada por δ lo que influirá en los siguientes cálculos de la probabilidades de transición de estados que, finalmente guiará las secuencia de nodos que quedarán en la ruta solución.

3.3.1 Exploración

Este modelo de aprendizaje siempre deja una posibilidad de seleccionar en cada etapa un nodo, que no necesariamente sea el que ha venido dando el mejor resultado, dado que todos los nodos tendrán alguna posibilidad, aunque sea pequeña, de ser elegidos.

Este aspecto se conoce como la exploración, que consiste en dar un margen de duda, por si hay otras opciones que aún no han sido exploradas y que pueden dar una mejor recompensa.

En las iteraciones iniciales se dará mayor espacio a la exploración, puesto que se inicia con las probabilidades uniformes ya mencionadas y poco a poco se van diferenciando las probabilidades de transición de las opciones que hay desde un nodo i. Mientras estas diferencias no sean muy marcadas, se facilitará la exploración de otras decisiones que pueden dar resultados mejores o similares a los ya aprendidos.

3.3.2 Explotación

La posibilidad de seleccionar con mayor probabilidad los nodos que han estado involucrados en rutas que generan recompensas positivas, permite que el autómata pueda acercarse, cada vez con mayor exactitud, a los valores de los *bandits* que están asociados a cada uno de los nodos de la ruta que él considera que es la más opcionada.

Este aspecto es el que se conoce como explotación que, en términos de juegos de casino, es seguir jugando la misma opción que ha dado los mejores resultados, esperando mejorar aún más.

En el modelo de aprendizaje propuesto, la explotación tendrá mayor preferencia en las iteraciones finales, donde las probabilidades de transición desde un nodo i ya tienden a tener una acción elegida con una probabilidad cercana a 1 y las demás con probabilidades cercanas a 0. Esta diferencia ya será lo suficientemente grande como para que se siga tomando la misma decisión, de esa iteración en adelante.

3.4 Construcción del algoritmo

El desarrollo y ejecución de los algoritmos se hizo en el lenguaje de programación Python 3.7, en un computador personal con procesador Intel Core i3-6006U de 2.00 GHz con 4.0 GB de memoria RAM, con sistema operativo Windows 10, de 64 bits.

Para la pruebas finales se migró la aplicación a un computador de la Universidad de Boyacá con procesador Intel(R) Xeon(R) Core i3-6006U de 3.50 GHz con 16.0 GB de memoria RAM, con sistema operativo Windows 10 Pro, de 64 bits.

Para los cálculos de los algoritmos se utilizaron en Python las librerías numpy, operator y math, y para la visualización de las simulaciones se empleó la librería

matplotlib.

Para simular un problema de aplicación, donde se conocen la cantidad de etapas y la cantidad de tareas posibles para cada etapa, se diseña e implementa un algoritmo que recibe estos parámetros de entrada y genera aleatoriamente una matriz de adyacencia con estos datos y los valores de los *bandits* asociados a cada uno de los nodos. Se verifica que el número y disposición de las aristas sean tales que cada nodo se conecte con al menos uno de la etapa siguiente. La cantidad de aristas que se generen entre etapas es un parámetro que se tiene en cuenta para la generación aleatoria del grafo, el cual corresponde a la densidad del grafo.

Cada uno de los nodos del grafo, eventualmente, representa una actividad a desarrollar dentro de una ruta de actividades que debe contener exactamente un nodo de cada etapa del grafo. A cada uno de esos nodos se le ha asociado un *bandit*, que no es otra cosa que un número que indica la recompensa que se obtiene al jugar este *bandit* en un casino, cantidad que para el caso de la simulación será positiva o negativa; y alrededor de la cual se generan los valores distorsionada que el jugador realmente recibe al seleccionar ese *bandit* en el casino (que casi nunca será el valor real).

Cada *bandit* se genera con una distribución normal con media en cero y desviación estándar de uno, pero podrá adecuarse su magnitud, si es conveniente, tanto para el manejo computacional, como para la simulación de un ejemplo real; los valores que se obtienen se consideran los valores reales y, son los que debe descubrir el algoritmo. Este valor real será la base para la generación de los valores distorsionados de los *bandits* en cada una de las épocas o iteraciones, los cuales se generan tomando su valor inicial o real como media de otra distribución normal, también con desviación estándar de 1.

En cada iteración se calcula una ganancia real, con los valores reales que se generaron para cada uno de los nodos por donde pase la ruta seleccionada por el algoritmo. También se calcula, para cada iteración la ganancia con los valores

distorsionados de la misma ruta, para efectos de poder compararlas eventualmente. Con esta última ganancia se hacen los cálculos de probabilidades de selección de nodos.

En seguida se presentan las versiones de pseudocódigo que se fueron implementando y probando, las cuales se complementan para dar la propuesta final.

3.4.1 Generador de bandits

Se inicia con la creación y prueba de la selección del *bandit* que mejor probabilidad tenga asociada. El algoritmo aplica la fórmula del promedio de las recompensas sobre el número de veces que fue seleccionado 2.1.

Como se aprecia en el pseudocódigo 1 se generó para cada uno una media con distribución Normal (0,1) y a partir de cada valor se generaron otros valores distorsionados, adicionando un ruido, también generado por distribución Normal (0,1). Este código se probó varias veces y siempre encontró la acción a asociada al *bandit* con mejor probabilidad.

Algorithm 1 Genera-bandit(T=Iteraciones, n=Número de acciones, ϵ=ϵ-greedy)

```
for t = 1 to T do
    r = np.random.sample()
    if r > ε then
        a = np.argmax(Q)
    else
        a = np.random.randint(0,n)
    end if
    sQ[a] = sQ[a] + q[a] + np.random.randn()
    Nt[a] = Nt[a] + 1
    Q[a] = sQ[a] / Nt[a]
    Imprima('a ',a)
end for
```

La figura 3.2 es el resultado gráfico de una corrida de este algoritmo con 400 iteraciones, 10 acciones y ϵ=0.3. La línea roja representa la recompensa promedio

que va encontrando en cada iteración y la línea horizontal que está ubicada en 1 representa el número del *bandit* que en promedio obtiene la mejor ganancia al finalizar las iteraciones, respuesta que corresponde exactamente con los valores asignados inicialmente.

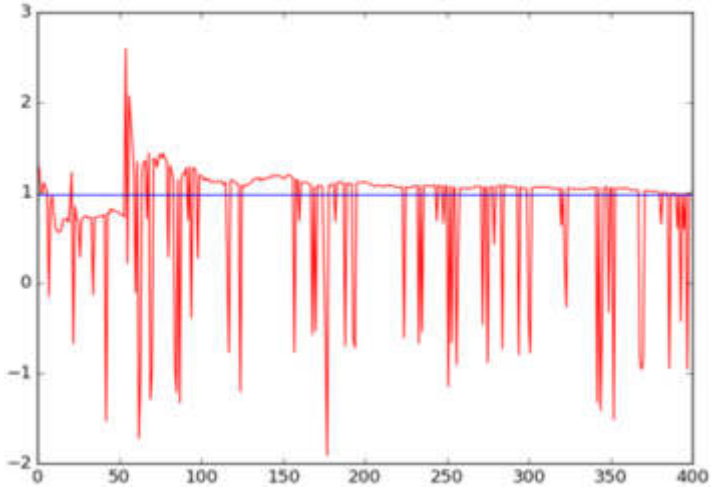


Figura 3.2: Selección de la mejor acción usando el algoritmo de los *bandits*

La oscilación de la línea roja se produce por la exploración que el algoritmo hace de otras alternativas diferentes a la que en cada momento parece ser la mejor. No se esperan los mismos valores cada vez que se esté probando el mismo *bandit*, porque los valores que recibe son los que fueron distorsionados con el ruido.

3.4.2 Generador de la matriz de adyacencia

Posteriormente se implementó un código que genera aleatoriamente matrices de adyacencia del grafo por etapas. Estas matrices deben ser escalonadas, deben obedecer a los requisitos de precedencia de nodos entre etapas y deben tener al menos un 1 en cada fila para garantizar la conexión de las etapas. El pseudocódigo 2 corresponde a tales características.

Este algoritmo recibe los datos de número de etapas y número de nodos por cada etapa que ya han de estar definidos con anterioridad.

Algorithm 2 Genera-matriz(L=Cantidad de etapas, M[L]=Nodos por etapa

```
Calculate: n=Cantidad de nodos
Generate: Ad[nxn] = zeros
nodoi=0
nodofin= nodoi+M[0]-1
for l = 1 to L-1 do
   for i in range(nodoi, nodofin+1) do
      nodohi=nodofin+1
      nodohfin= nodohi+M[l+1]-1
      sumafila=0
      for j in range(nodohi, nodohfin+1) do
         arg = np.random.randint(2, size=1)
         A[i,j] = arg
         sumafila += A[i,j]
         if umafila == 0 then
            A[i,j] = 1
         end if
      end for
   end for
   nodoi=nodohi
   nodofin=nodohfin
end for
Imprima('A ',A)
```

Este código también se probó varias veces y siempre arrojó resultados adecuados de matrices escalonadas, adecuadas para el grafo por etapas diseñado, como el ejemplo que se muestra en la figura 3.3.

```
A:
[[0. 0. 1. 1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 0. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 1. 1. 0. 1. 1. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 1. 1. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

Figura 3.3: Resultado de la matriz de adyacencia para un grafo por etapas

3.4.2.1 Generación de una ruta aleatoria

Se implementó la fusión del algoritmo para la generación de la matriz de adyacencia, con el que genera los valores asociados a cada nodo o *bandit*; el pseudocódigo correspondiente se probó con valores aleatorios, obteniendo siempre la matriz escalonada que se esperaba y los valores asociados a los *bandits*.

Se le adicionó una funcionalidad para que tomara una ruta factible, teniendo en cuenta la matriz de adyacencia para encontrar los vecinos de cada nodo, la cual se presenta en el algoritmo 3. Las pruebas que se hicieron funcionaron siempre correctamente.

Algorithm 3 Genera-rutas(L=Cantidad de etapas, A=Matriz de adyacencia

```
1: n[0]=0
2: for l = 1 to L do
3:    nzv = np.nonzero(A[o, ])
4:    nzv = np.reshape(nzv,-1) :vector de nodos adyacentes
5:    for i in range(o, len(nzv) do
6:       d = np.random.choice(nzv, 1)
7:       R[l]=d
8:    end for
9: end for
10: Imprima('Ruta ',R)
```

Los valores que se asignaron aleatoriamente a los *bandits* sirven para calcular una ganancia total por cada ruta que contenga nodos de las L etapas; esta ganancia equivaldría a la *ganancia real* que se obtendría si se supieran los valores asociados a cada *bandit*. Así mismo, en cada iteración que ejecute el código, el autómata recibirá una información distorsionada del valor de cada *bandit*, de acuerdo a la distribución Normal que ya se ha explicado; con estos valores el autómata ha de calcular la ganancia que realmente estará recibiendo al final de las L etapas, a la que en adelante se reconocerá como *ganancia distorsionada*.

El cálculo de la ganancia distorsionada se muestran en los algoritmos siguientes como $Gain_{path}$, la cual va recogiendo en cada etapa la ganancia del nodo seleccionado

Gan_{orig}, de acuerdo a los valores de los *bandit*, que realmente recibe el autómata y que se generan en cada iteración como Gan_n.

3.4.3 Encontrando la ruta óptima

Aquí se presentan dos momentos diferentes: Inicialmente aquel donde la escogencia del siguiente nodo de un camino, solo tiene que estar dentro del conjunto de vecinos del nodo actual, pero todos con la misma probabilidad de ser seleccionados; este modelo se reconocerá como Probabilidades Uniformes. Seguidamente se implementa la generación de la matriz de probabilidades de transición de pasar de un nodo a otro, la cual se basa en un fundamento matemático específico que se presentó en la sección 3.2, modelo que se espera que converja a una única respuesta con la mejor ruta encontrada; y este se reconocerá como Probabilidad Modelada.

3.4.4 Búsqueda con probabilidad uniforme

Inicialmente los nodos de una etapa que sean adyacentes al nodo que ha sido seleccionado en la etapa anterior, tienen una probabilidad uniforme de que sean seleccionados. Este algoritmo se construye básicamente, incluyendo un contador de iteraciones para que el algoritmo de generación de rutas factibles se ejecute una cantidad dada de veces determinada y una selección de la ruta que al final de las iteraciones hubiera obtenido la ganancia promedio, para entregarla como respuesta.

El pseudocódigo queda como se ve en el cuadro del algoritmo 4, que con los datos de entrada y los generados aleatoriamente para el grafo, calcula una ruta de nodos de cada etapa, solamente teniendo en cuenta que el siguiente nodo forme parte de los vecinos del presente.

Algorithm 4 L-n-bandit-Uniforme(L=Cantidad de etapas, M[L]=Nodos por etapa, n=Cantidad de nodos)

```
Generate: Ad[nxn] = Matriz de adyacencia
Generate: B[n] = Bandits reales
for t = 1 to T do
    Generate: Gan_n
    Gain_path = 0
    Initialize: orig = 0
    Add: orig in path
    for l = 1 to L-1 do
        Generate: nvz_orig = vecinos de orig
        Random: dest ∈ nvz_orig
        orig = dest
        Add: orig in path
        Gain_path =+ Gan_orig
    end for
    Imprima: path
end for
```

3.4.5 Búsqueda con probabilidades aprendidas

Posteriormente se implementó el cálculo de la matriz de probabilidades de transición con las fórmulas explicadas en la sección 3.2 cuyo pseudocódigo se muestra en el cuadro algorítmico 5. A diferencia del anterior, la selección del nodo siguiente al presente en cada ruta, no solo va a tener en cuenta la vecindad con este último, sino la probabilidad de transición a cada nodo vecino, consignada en la matriz de probabilidad de transición que aquí se genera y se actualiza, de acuerdo a las ganancias que reciba el camino, en cada iteración, como se explica en seguida.

Como se explicó en el diseño del modelo de aprendizaje, para la primera iteración se maneja una probabilidad uniforme y para las demás iteraciones, se calculan las probabilidades de transición entre nodos. En cada iteración se selecciona un nodo de la etapa inicial, se calculan sus vecinos o nodos alcanzables y se selecciona el que mayor probabilidad de transición presente, para guardarlo en la ruta y proceder a hacer la búsqueda de sus vecinos en la siguiente etapa, hasta llegar a la última, usando siempre el mismo criterio de selección.

Algorithm 5 L-n-bandit(L=Cantidad de etapas, M[L]=Nodos por etapa)

Calculate: n=Cantidad de nodos
Initialize: v[n] = 0
Generate: Ad[nxn] = Matriz de adyacencia
Generate: B[n] = Bandits reales
for t = 1 **to** T **do**
  Initialize: orig = 0
  Add: orig in path
  $P(i \to j) = \frac{e^{v_j}}{\sum_k A_{i,k} e^{v_k}}$,
  Generate: Gan_n
  $Gain_{path} = 0$
  for l = 1 **to** L-1 **do**
    Generate: nvz_{orig} = vecinos de orig
    Select: $dest \in nvz_{orig}$ by $P(i \to j)$
    $orig = dest$
    Add: orig in path
    $Gain_{path}$ =+ Gan_{orig}
  end for
  if $Gain_{path} > 0$ **then**
    v[n+1] = v[n] + δ by i $\in$ path
  else
    v[n+1] = v[n] - δ by i $\in$ path
  end if
  Imprima: path
end for

Una vez se conozca la ganancia o pérdida al final de la iteración, se actualizan la función de valor v de cada nodo de esa ruta, de acuerdo a la fórmula 3.3, este valor se utiliza para calcular las nuevas probabilidades de transición con la fórmula 3.2, que se tendrán en cuenta en la siguiente iteración. Cabe anotar que al final de cada etapa solo se afectan las probabilidades de transición de los nodos involucrados en la ruta seleccionada, pero que al iniciar cada iteración se cuenta con todas las modificaciones que hayan sufrido estas probabilidades en las iteraciones anteriores de esta simulación.

Se recuerda que la selección del siguiente vecino está fuertemente influenciada por las probabilidades de transición, constituyéndose en la explotación de los caminos preferidos, pero siempre deja un porcentaje de posibilidad de escoger cualquier nodo que no sea el de mayor probabilidad de transición asociada, lo que permite resolver

el problema de la exploración, propio de la teoría de los *bandits*, que garantiza la búsqueda de mejores soluciones que no se han considerado aún.

Al final de las iteraciones definidas, el simulador debe converger a la mejor ruta que haya estudiado, que puede ser la óptima, como se verá en la siguiente sección.

3.5 Pruebas

3.5.1 Grafo para las pruebas

Aunque el algoritmo se ha diseñado para generar grafos aleatorios, se hace necesario generar uno particular para poder comparar los valores resultantes de las ejecuciones del algoritmo con este. Tal grafo debe ser de un tamaño adecuado para poder calcular todas las posibles rutas desde un nodo inicial hasta uno final con sus respectivas ganancias; además, los nodos deberán tener asociados valores de ganancias, de forma que se pueda establecer claramente cuál es la ruta óptima.

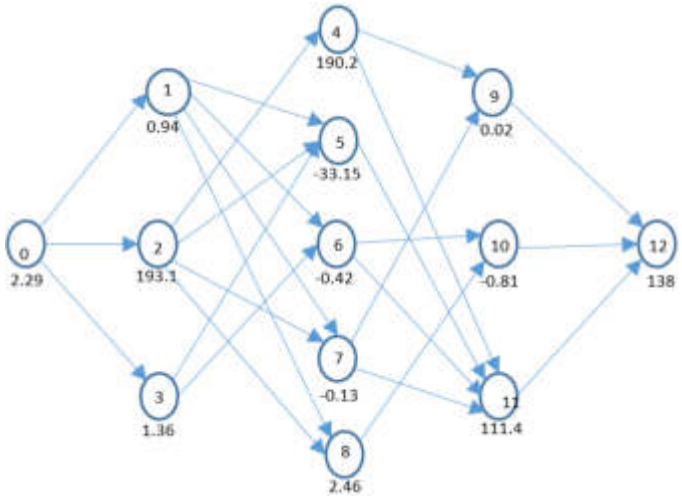


Figura 3.4: Grafo Modelo con 5 etapas y 13 nodos

Se realizan estas pruebas entonces, con un grafo de 5 etapas y 13 nodos, dispuestos como se ve en la figura 3.4, inicialmente con matriz de adyacencia aleatoria,

así como los valores de sus *bandits*, pero obligando a que uno de ellos por etapa tenga un mayor valor que los demás, para identificar con facilidad la mejor ruta y así verificar su funcionamiento.

3.5.2 Pruebas con probabilidad uniforme

El algoritmo cuyo pseudocódigo se presentó en el cuadro 4, se ejecutó con 99 intervalos de tiempo. La figura 3.5 muestra uno de los resultados.

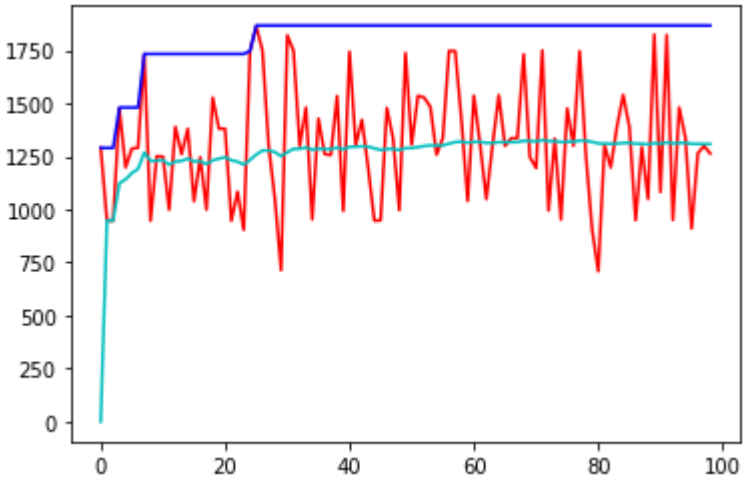


Figura 3.5: Resultado con probabilidad uniforme

En esta gráfica, la curva superior azul corresponde a la ganancia real de la mejor ruta probada hasta el momento, calculada con el valor asignado a los *bandits*; en la curva roja, la más inestable, se presentan las ganancias distorsionadas obtenidas en cada instante de tiempo por la ruta seleccionada, calculada con los valores distorsionados con ruido para los *bandits*; y en la curva inferior verdosa se muestra el promedio acumulado de estas ganancias, el cual converge a un único valor a través del tiempo, pero no trata de acercarse al valor real.

3.5.3 Pruebas con probabilidades aprendidas

Con 33 iteraciones (T=33) el algoritmo 5 generó la gráfica de la figura 3.6, en la cual se pueden apreciar los valores que van tomado su ganancia y su ganancia promedio, la cual convergió rápidamente hacia el valor de la mejor ganancia real. La mejor ganancia real se calculó para la mejor ruta que el mismo algoritmo haya encontrado: esto permite que el software sea el que estime dicha ruta y su ganancia, sin una intervención manual.

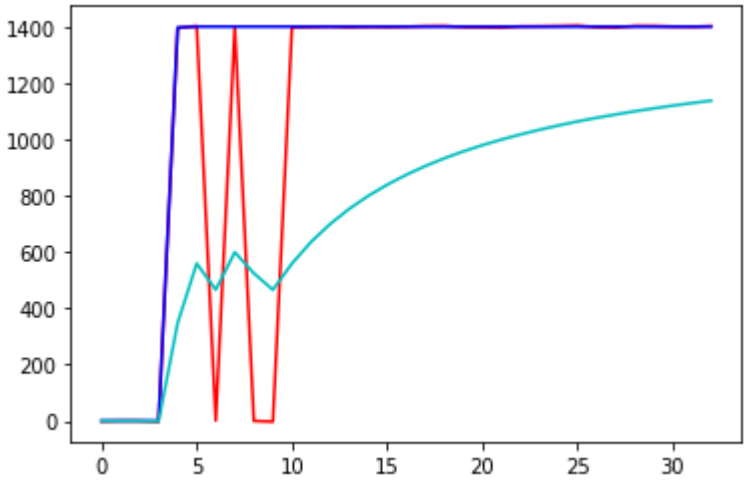


Figura 3.6: Resultado con probabilidad aprendida

En esta gráfica, al igual que en la del modelo anterior, la curva superior azul corresponde a la ganancia de la mejor ruta probada hasta el momento, calculada con el valor real asignado a cada uno de los *bandits*; en la curva más inestable se presentan las ganancias distorcionads obtenidas en cada instante de tiempo por la ruta seleccionada; y en la curva inferior verdosa se muestra el promedio acumulado de estas ganancias, el cual, trata de acercarse a la curva de la ganancia real.

De otra parte, para estimar el comportamiento del algoritmo se dejan fijos en el grafo los datos de la matriz de adyacencia y del vector de valores reales de los *bandits*, para ejecutar los experimentos que se explican en el siguiente capítulo.

Capítulo 4

Resultados computacionales

En este capítulo se presentan los resultados de los experimentos que se ejecutaron para ajustar algunos parámetros del algoritmo y para verificar el efecto de la aleatoriedad en el mismo. Para ello se toma el grafo de 5 etapas y 13 nodos con valores fijos para sus respectivos *bandits*, los cuales tiene la característica que, en cada etapa, uno de ellos es suficientemente mejor que los demás y hará que, seguramente, tal nodo forme parte del vector de acciones seleccionadas al final de las 5 etapas.

Los valores que se fijaron para la matriz de adyacencia y para los valores reales de los *bandits* del grafo que se usará en los experimentos descritas en este capítulo, son los siguientes:

$$\mathrm{A} = \begin{bmatrix} 0,1,1,1,0,0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,1,1,1,1,0,0,0,0 \\ 0,0,0,0,1,1,0,1,1,0,0,0,0 \\ 0,0,0,0,0,1,1,0,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0,0,1,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,1,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,1,0,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0 \end{bmatrix}$$

B = [2.29, 0.94, 193.06, -1.36, 190.22, -33.15, -0.42, -0.13, 2.46, 0.02, -0.81, 111.38, 1.38]

Es necesario tener en cuenta que aún se han dejando en forma aleatoria la selección inicial de los nodos y los valores distorsionados que se generan para los *bandits* en cada iteración.

El grafo, con las aristas indicadas en la matriz de adyacencia y los valores de los *bandits* del vector B, se aprecia en la figura 4.1

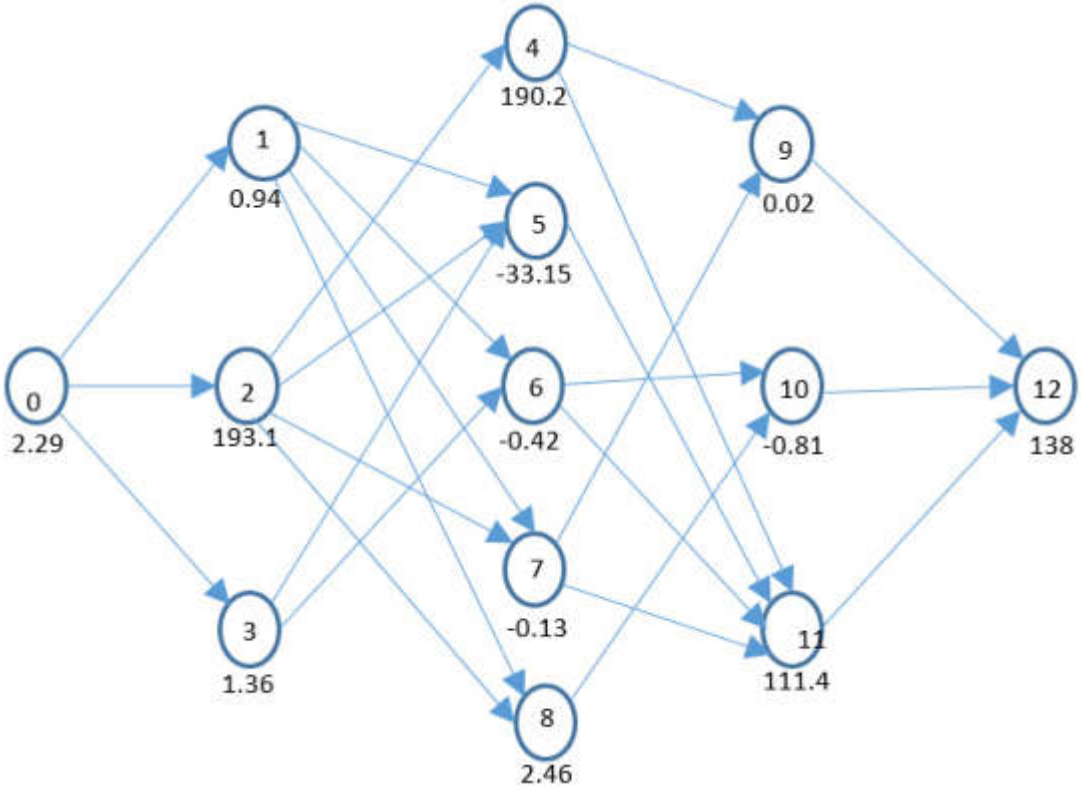


Figura 4.1: Grafo para las pruebas de parámetros

Para estimar el desempeño del algoritmo se verificará la convergencia de los promedios de las ganancias distorsionadas hacia la ganancia real. Para facilitar la visualización de este comportamiento se hace una diferencia ente los valores de estas dos magnitudes, pero se normaliza para producir valores entre 0 y 1, con una sencilla división en el valor máximo posible, siendo el mejor valor el que más se acerque a la cero.

4.1 Parámetro Deltha puro

Se retoman aquí las fórmulas con las que se consigue que el autómata aprenda y genere una matriz de probabilidades de transición entre nodos, para contextualizar el **parámetro de aprendizaje** δ que permite la actualización de una iteración a otra de la función de valor de los nodos, como se se muestra en la fórmula 4.1.

$$v_j(\tau + 1) = v_j(\tau) \pm \delta, \tag{4.1}$$

Estos valores son los exponentes que se usan en la fórmula de cálculo de probabilidades 4.2, que también se va actualizando para todos los pares de nodos en cada iteración.

$$P(i \to j) = \frac{e^{v_j}}{\sum_k A_{i,k} e^{v_k}}, \tag{4.2}$$

4.1.1 Prueba inicial

Se realiza inicialmente una prueba con 9 valores para δ entre 0.1 y 0.9, sumando, en caso de ganancia y restando, en caso de pérdida, para motivar o desmotivar al autómata para que en el paso siguiente seleccione los nodos que reciben el refuerzo; la cantidad de iteraciones T se varió entre 100 y 1000, con intervalos de 100. Las diferencias normalizadas que se encontraron entre la ganancia real y cada una de las ganancias que en promedio se obtuvieron se ven en la tabla 4.1.

Los rangos de diferencia superan el 50 %, independientemente de la cantidad de iteraciones que se ejecuten, por lo que se procede a experimentar con valores de δ de órdenes diferentes.

Tabla 4.1: Rangos de diferencia de ganancias con Deltha en décimas

Deltha	Iteraciones									
	100	200	300	400	500	600	700	800	900	1000
0,1	0.715	0.715	0.715	0.715	0.715	0.715	0.715	0.715	0.715	0.715
0,2	0.695	0.695	0.695	0.695	0.695	0.695	0.695	0.695	0.695	0.695
0,3	0.676	0.676	0.676	0.676	0.676	0.676	0.676	0.676	0.676	0.676
0,4	0.656	0.656	0.656	0.656	0.656	0.656	0.656	0.656	0.656	0.656
0,5	0.637	0.637	0.637	0.637	0.637	0.637	0.637	0.637	0.637	0.637
0,6	0.618	0.618	0.618	0.618	0.618	0.618	0.618	0.618	0.618	0.618
0,7	0.598	0.598	0.598	0.598	0.598	0.598	0.598	0.598	0.598	0.598
0,8	0.579	0.579	0.579	0.579	0.579	0.579	0.579	0.579	0.579	0.579
0,9	0.559	0.559	0.559	0.559	0.559	0.559	0.559	0.559	0.559	0.559

4.1.2 Segunda prueba

Se ejecuta la aplicación con valores de δ en magnitudes de centésimas, milésimas y diezmilésimas y con los mismos 10 valores para la cantidad de iteraciones T, cuyos resultados se aprecian en la tabla 4.2.

Aquí se puede apreciar que, a pesar de disminuir notablemente los valores de δ, se siguen obteniendo valores muy altos para la diferencia de las ganancias, con errores superiores al 40 %. Los promedios de estas diferencias se presentan en la tabla 4.3, donde además se aprecia que no hay una diferencia relevante para las diferentes magnitudes de δ.

Con estos resultados, se formulan experimentos con una mayor variación para δ y para la cantidad de Iteraciones T, que permita un análisis estadístico de la influencia de estas dos variables en los resultados esperados.

4.1.3 Experimentos

4.1.3.1 Variable dependiente

Se define como variable dependiente para este conjunto de experimentos a la diferencia entre la ganancia real máxima y la ganancia distorsionada media obtenida

Tabla 4.2: Rangos de diferencia de ganancias con Deltha en otras unidades

Deltha	Iteraciones									
	100	200	300	400	500	600	700	800	900	1000
0.01	0.697	0.697	0.671	0.695	0.675	0.677	0.661	0.694	0.654	0.692
0.02	0.707	0.686	0.628	0.583	0.565	0.605	0.661	0.751	0.760	0.809
0.03	0.615	0.698	0.763	0.744	0.742	0.734	0.777	0.512	0.813	0.819
0.04	0.646	0.651	0.533	0.783	0.812	0.800	0.505	0.495	0.815	0.820
0.05	0.816	0.720	0.780	0.795	0.504	0.484	0.483	0.798	0.817	0.492
0.06	0.607	0.475	0.806	0.555	0.809	0.537	0.459	0.817	0.819	0.476
0.07	0.725	0.764	0.774	0.810	0.824	0.514	0.824	0.831	0.837	0.822
0.08	0.654	0.806	0.450	0.472	0.492	0.476	0.490	0.421	0.471	0.466
0.09	0.756	0.765	0.502	0.604	0.486	0.820	0.780	0.775	0.827	0.835
Deltha	Iteraciones									
	100	200	300	400	500	600	700	800	900	1000
0.001	0.690	0.699	0.705	0.713	0.705	0.704	0.708	0.727	0.713	0.698
0.002	0.702	0.710	0.706	0.715	0.716	0.696	0.682	0.680	0.696	0.714
0.003	0.647	0.719	0.682	0.692	0.699	0.700	0.709	0.691	0.687	0.659
0.004	0.711	0.721	0.697	0.694	0.688	0.687	0.671	0.691	0.681	0.693
0.005	0.763	0.691	0.710	0.682	0.688	0.686	0.667	0.664	0.684	0.689
0.006	0.692	0.726	0.693	0.692	0.716	0.689	0.654	0.680	0.647	0.605
0.007	0.730	0.695	0.696	0.680	0.680	0.656	0.674	0.626	0.706	0.656
0.008	0.728	0.651	0.708	0.669	0.668	0.686	0.681	0.710	0.692	0.665
0.009	0.707	0.695	0.644	0.691	0.640	0.717	0.714	0.614	0.703	0.692
Deltha	Iteraciones									
	100	200	300	400	500	600	700	800	900	1000
0.0001	0.708	0.689	0.728	0.718	0.693	0.709	0.700	0.694	0.709	0.699
0.0002	0.684	0.697	0.718	0.696	0.710	0.734	0.700	0.706	0.708	0.699
0.0003	0.718	0.683	0.693	0.699	0.706	0.715	0.699	0.707	0.715	0.696
0.0004	0.748	0.717	0.704	0.707	0.710	0.745	0.705	0.705	0.707	0.713
0.0005	0.686	0.692	0.705	0.733	0.700	0.706	0.709	0.691	0.704	0.717
0.0006	0.649	0.742	0.703	0.711	0.699	0.685	0.717	0.706	0.707	0.706
0.0007	0.752	0.723	0.733	0.706	0.700	0.701	0.688	0.709	0.697	0.722
0.0008	0.682	0.725	0.704	0.706	0.720	0.690	0.700	0.712	0.705	0.709
0.0009	0.718	0.680	0.708	0.691	0.694	0.702	0.702	0.714	0.688	0.698

en la ejecución, normalizada de forma que se obtengan valores entre cero y uno. Se busca minimizar esta diferencia.

4.1.3.2 Variables independientes

Los experimentos se diseñan teniendo en cuenta que las variables independientes corresponden a:

- Factor de aprendizaje δ
- Número de iteraciones T

Tabla 4.3: Promedios de los resultados variando décimas en Deltha

PROMEDIOS DE ERROR EN GANANCIAS			
Décimas	**Centésimas**	**Milésimas**	**Diezmilésimas**
0.637	0.675	0.690	0.706

4.1.3.3 Validez

Para el control o validez interna se ha de comprobar que las variables, aquí definidas como independientes, son las que realmente influyen en las dependientes, y no factores externos como la aleatoriedad. Se toman, entonces, los resultados de 100 experimentos con cada uno de los valores de control sobre las variables independientes.

4.1.3.4 Técnica

La técnica que se usa en este experimento es la simulación. Es un instrumento objetivo, confiable y válido. La validez es alta, puesto que el computador siempre tratará los valores de la misma manera; se espera que con un mismo valor de variables independientes se obtengan resultados similares en las variables dependientes.

4.1.3.5 Herramientas

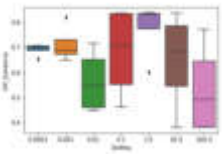
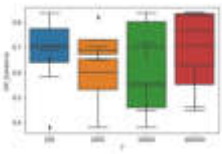
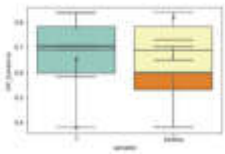
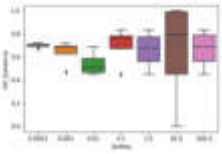
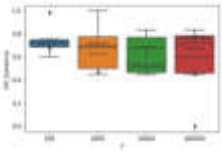
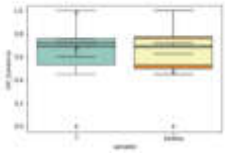
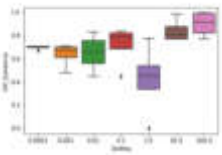
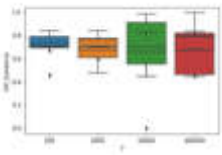
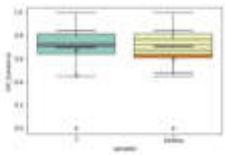
Se realizan los cálculos de medias, varianzas y de análisis de varianza (ANOVA) que permitan concluir la forma en que cada variable independiente influye en la dependiente.

4.1.3.6 Experimento uno

Se diseña un experimento con las variables independientes: T y δ, correspondientes a la cantidad de iteraciones y al factor de aprendizaje que ha de incentivar la selección o no de un nodo, ya que afecta la matriz de probabilidades de transición; la variable dependiente corresponde a la diferencia normalizada de las ganancias, ya descrita.

Se usan valores para T de 100, 1000, 10000 y 100000 iteraciones y valores de δ de 10^2, 10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3} y 10^{-4}. Este experimento se ejecuta varias veces, encontrando diferentes valores en sus resultados para los mismo valores de T y de δ, lo que se explica por la influencia de la aleatoriedad en estos resultados y se corrobora con el valor del estadístico de prueba F y el valor de P de las tablas ANOVA. Las gráficas correspondientes a tres ejecuciones de este experimento y los resultados del ANOVA correspondiente, se presentan en la tabla 4.4.

Tabla 4.4: Experimento uno con Deltha

GRÁFICA PARA δ	**GRÁFICA PARA T**	**GRÁFICA PARA T y δ**	**ANOVA**
			F PR(>F) δ 1.334392 0.292932 T 1.012468 0.410188
			F PR(>F) δ 0.246477 0.954510 T 1.111843 0.370257
			F PR(>F) δ 3.122896 0.028147 T 0.343180 0.794422

En las tres filas de la tabla 4.4 se notan cambios en las medias y las varianzas obtenidos de una ejecución a otra, y en las tablas ANOVA se puede verificar que la probabilidad $PR(> F)$ se mantuvo con valores superiores a 0.05, en casi todos los casos; en pero en el tercer caso, para la variable δ sí se obtuvo un valor menor de 0.05 en dicha columna de probabilidad. Estos resultados no nos permiten concluir que las variables independientes tengan un alto grado de significancia sobre la variable dependiente; además, se esperaría que, en general, el estadístico de prueba F tuviera siempre un valor distante de la unidad y los seis valores obtenidos son relativamente cercanos a la unidad.

Además estas gráficas muestran que, independientemente de la cantidad de iteraciones que se hagan (variable T, eje horizontal), se obtienen resultados muy altos para la diferencia normalizada de ganancias, con medias que llegan a estar por encima del 70 %. Coincide en las tres ocasiones que la varianza en las cajas para el menor valor de T es la más pequeña; esto se debe a que en estos casos, y especialmente cuando se tienen valores grandes para δ, el algoritmo tiende a aprender una de las primeras respuestas que, aleatoriamente, debió validar y se queda con ella convergiendo rápidamente a cualquier respuesta, que no es necesariamente la óptima. Se procede entonces a verificar lo que sucede con valores grandes para T y pequeños para δ.

Se quiere observar si algunos de los valor de T y de *delta* que en la gráfica muestran mejores resultados, influyen efectivamente en la optimización de la diferencia entre ganancias.

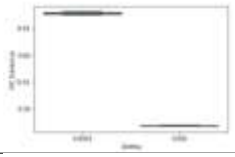
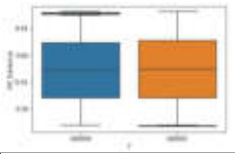
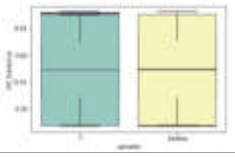
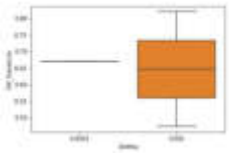
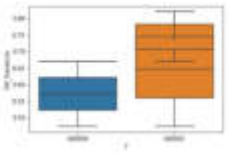
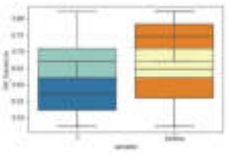
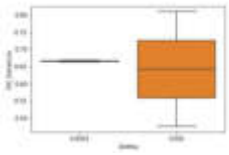
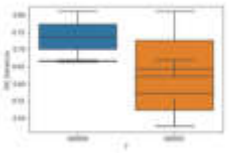
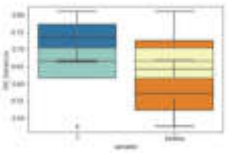
4.1.3.7 Experimento dos con Deltha

Se usan, en esta ocasión, valores para T de 100000 y 100001 (se deja casi la misma cantidad, para estimar básicamente el comportamiento de la variable δ) y valores de *delta* de 10^{-3} y 10^{-4}. Este experimento se ejecuta varias veces, encontrando

diferentes valores en sus resultados para los mismos valores de T y de δ, lo que se explica como una alta influencia de la aleatoriedad.

Tres resultados con los mismos valores de la variables independientes se muestran en la figura 4.5

Tabla 4.5: Experimento dos

GRÁFICA PARA δ	**GRÁFICA PARA T**	**GRÁFICA PARA T y δ**	**ANOVA**
			F PR(>F) δ 2905.033 0.011810 T 0.286967 0.686915
			F PR(>F) δ 0.018279 0.914447 T 0.985740 0.502286
			F PR(>F) δ 0.018889 0.913049 T 0.939678 0.509901

Los resultados de las tablas ANOVA en la ejecución de la fila 1 de la tabla 4.5 presentan un valor para el estadístico de prueba F mucho mayor que 1 y el valor de la probabilidad $PR(> F)$ menor a 0.05, para la variable δ, lo que indicaría que esta variable independiente influye notoriamente en la variable dependiente; pero en las dos ejecuciones que se muestran al final de la tabla, no se mantuvo este resultado para δ, obteniendo valores en $PR(> F)$ superiores a 0.05; así mismo para la variable T. Estos resultados no nos permiten, nuevamente, concluir que las variables independientes tengan un alto grado de significancia sobre la variable dependiente.

Se decide, entonces trabajar con una variable δ dependiente de la magnitud de la ganancia o pérdida que se reciba al final de cada ruta probada, como se explica en el siguiente experimento.

4.2 Parámetro Gamma

Se procede a cambiar δ por un valor que cambiará en función de la ganancia que se recibe al final de la ruta probada, de forma que, inclusive, el signo se ajustará cuando hayan pérdidas, sin hacerse necesario decidir entre suma y resta. Para ello se involucra un factor de aprendizaje Gamma (γ) que dosificará la retribución que tendrá la ganancia en el ajuste de la función de valor de los *bandits* asociados a los nodos de la ruta probada, como se aprecia en la ecuación 4.3, al multiplicar γ por la ganancia ($Gain$) o recompensa recibida al final del camino.

$$v_j(\tau + 1) = v_j(\tau) + \gamma Gain \tag{4.3}$$

El valor de γ ha de ser lo suficientemente pequeño para garantizar que, al multiplicarlo por la ganancia, la cantidad se reduzca a valores menores de 1 en su valor absoluto.

4.2.1 Primer caso de estudio

Inicialmente se realizó una prueba para establecer el porcentaje de ejecuciones en que el algoritmo realmente converge, ejecutando varias veces el algoritmo con el grafo ya fijo. En esta, se ha encontrado una tendencia a que solo en 7 de cada 10 ocasiones se logra la convergencia hacia el valor real. Las gráficas de la figura 4.2 corresponden a 10 corridas del algoritmo de probabilidades aprendidas con 1000 iteraciones, un δ de aprendizaje que se obtuvo de multiplicar $\gamma = 0.0003$ por el valor de la ganancia o recompensa de la ruta utilizada.

El hecho de que no todas las ejecuciones con los mismos valores de entrada, presenten la misma convergencia, refleja la influencia que sigue presentando la aleatoriedad en este modelo, lo que no es muy deseable.

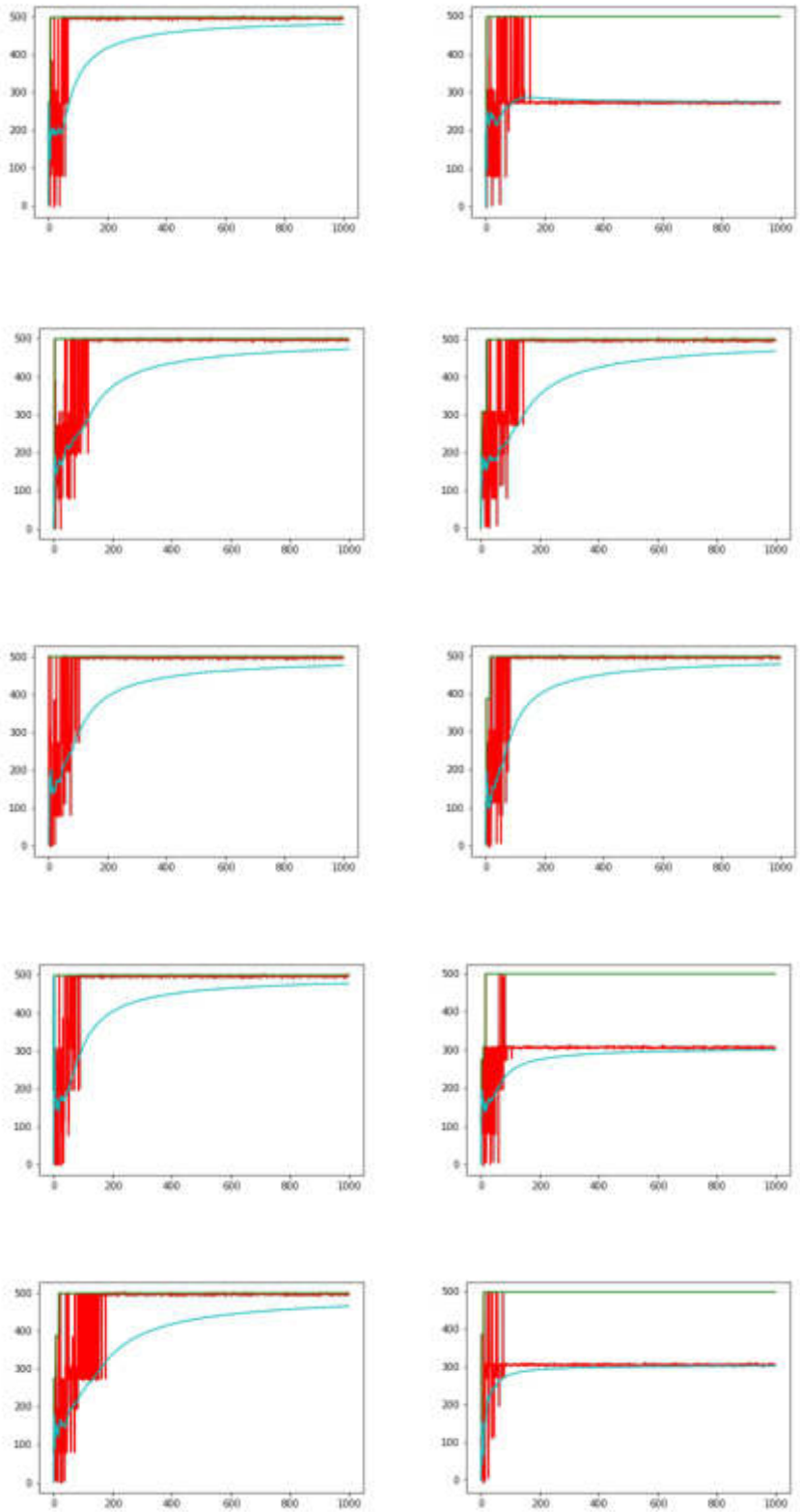


Figura 4.2: Ejecución algoritmo Probabilidad Modelada

Dado que esta prueba se hizo con un único valor para T y un único valor para γ, se procedió a explorar otros valores para dichas variables que puedan mejorar este porcentaje de éxito.

4.2.2 Experimentos

Al igual que para los experimento que hicieron con la variable δ, se explican loe elementos en seguida.

4.2.2.1 Variable dependiente

Se define como variable dependiente para este conjunto de experimentos a la diferencia entre la ganancia real máxima y la ganancia distorsionada obtenida en la ejecución, diferencia que se normaliza, de forma que se obtengan valores entre cero y uno.

4.2.2.2 Variables independientes

Los experimentos se diseñan teniendo en cuenta que las variables independientes corresponden a:

- Coeficiente del factor de aprendizaje γ
- Número de iteraciones T

4.2.2.3 Validez

Para el control o validez interna se ha de comprobar que las variables, aquí definidas como independientes, son las que realmente influyen en la variable depen-

diente, y no factores externos como la aleatoriedad. Se toman, entonces, los resultados de 100 experimentos con cada uno de los valores de control sobre las variables independientes.

4.2.2.4 Técnica

La técnica que se usa en este experimento es la simulación. Es un instrumento objetivo, confiable y válido. La validez es alta, puesto que el computador siempre tratará los valores de la misma manera; se espera que con un mismo valor de variables independientes se obtengan resultados similares en las variables dependientes.

4.2.2.5 Herramientas

Se realizan los cálculos de medias, varianzas y de análisis de varianza (ANOVA) que permitan concluir la forma en que cada variable independiente influye en la dependiente.

4.2.2.6 Experimento uno

Se asignan valores a las variables dependientes así: T toma valores de 100, 300, 500, 700, 900 y 1000; y γ valores de 0.00001, 0.00025, 0.00049, 0.00075, 0.0001, con los que se pretende cubrir los valores de γ del orden de 10^{-4}.

Los valores de los promedios en la diferencia de las ganancias que se obtuvieron se aprecian en la tabla 4.6.

Tabla 4.6: Promedios de diferencias de ganancias usando Gamma

GAMA	ITERACIONES					
	100	300	500	700	900	1000
0,00001	0,368	0,360	0,344	0,328	0,322	0,272
0,00025	0,102	0,044	0,034	0,066	0,052	0,042
0,00050	0,050	0,078	0,082	0,072	0,082	0,074
0,00075	0,080	0,092	0,086	0,090	0,078	0,090
0,00099	0,092	0,102	0,084	0,084	0,100	0,106

Los resultados son considerablemente mejores que los que se obtuvieron con δ, obteniendo un promedio en el error para esta tabla del 13 % (o del 8 %, si se elimina la primera fila donde se encuentran los valores más altos). También, se puede notar, en esta tabla, que los errores están influidos tanto por el valor de γ, como por el número de iteraciones que se ejecutan.

Utilizando los datos de la tabla 4.6, se obtienen promedios y varianza para cada fila y para cada columna, como se presenta en la tabla 4.7.

Tabla 4.7: Medias y varianzas en Promedios de distancias usando Gamma

	PROMEDIO	***VARIANZA***
GAMMA		
0.00001	0.3323	0.001188
0.00025	0.0567	0.000611
0.00050	0.0730	0.000144
0.00075	0.0860	0.000034
0.00099	0.0947	0.000089
ITERACIONES		
100	0.1384	0.016855
300	0.1352	0.016273
500	0.126	0.015322
700	0.128	0.012590
900	0.1268	0.012201
1000	0.1168	0.008087

Aquí se puede notar que el valor de γ, no afecta significativamente las diferencias entre ganancias, mientras que cada vez que se aumentó el número de iteraciones T en el experimento, los cambios fueron menores en la variable dependiente, lo que se apoya con el análisis de los resultados de las varianzas.

Se obtienen valores de varianzas pequeños, lo que es deseable al momento de tomar conclusiones sobre las medias dadas.

Este comportamiento se puede visualizar en el diagramas BoxPlot para Gamma que se ve en la figura 4.3, donde se aprecia, además que el incremento o decremento en los valores de gamma, no influye directamente en la diferencia entre ganancias, solo entre el primer valor, el cual no es recomendable, y los cuatro finales.

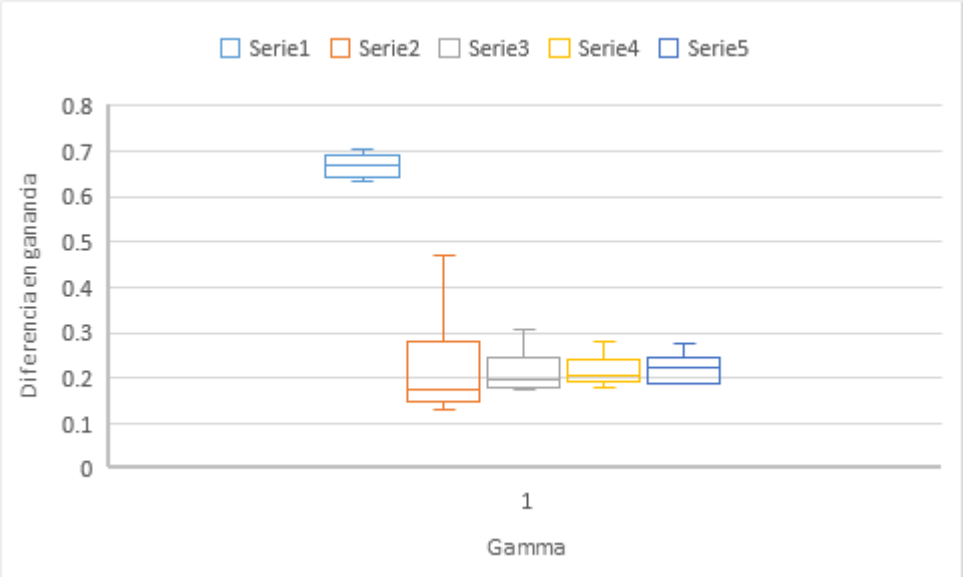


Figura 4.3: Diagramas BoxPlot para la variable Gamma

Para corroborar esta observación se genera el análisis de varianza ANOVA para los datos de la tabla 4.6, cuyo resultado se presenta en la tabla 4.8.

Tabla 4.8: ANOVA para Distancias en ganancias usando Gamma

Origen de las variaciones	***Promedio de los cuadrados***	***F***	***Probabilidad***	***Valor crítico para F***
Gamma	0.244513	118.074	1.297E-13	2.86608
Iteraciones	0.013214	6.381	0.00107	2.71089

Para la primera variable, *Gamma*, se obtiene el valor de F mayor al de F *crítica* y una probabilidad muy baja ($P < 0.05$), lo que indica que existe una diferencia significativa en Gamma que afecta la variable dependiente.

Para la segunda variable, *Iteraciones*, se obtiene el valor de F mayor al de F *crítica* y una probabilidad baja ($P < 0.05$), lo que indica que también existe una

diferencia significativa en el número de iteraciones que afecta la variable dependiente, pero no de la misma forma que la variable *Gamma*, tal como se había planteado al analizar las gráficas BoxPlot.

Se quiere entonces, encontrar un valor de Gamma que ofrezca buenos resultados, y estimar hasta donde la cantidad de iteraciones puede disminuir considerablemente el valor de la variable dependiente, que corresponde a la diferencia entre las ganancias obtenidas y la ganancia real, para lo que se plantean los experimentos siguientes.

4.2.2.7 Experimento dos

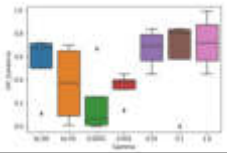
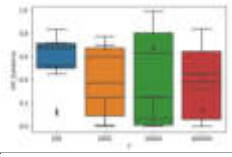
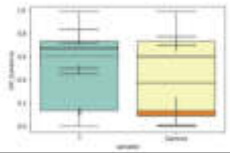
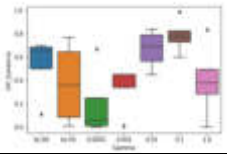
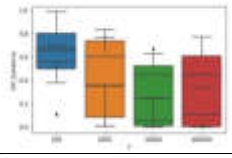
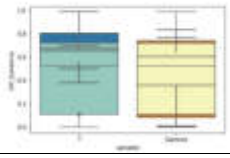
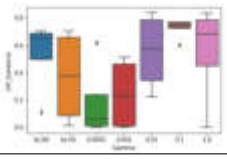
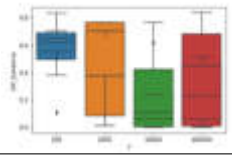
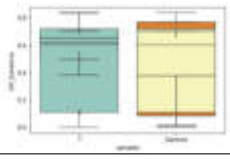
Para este experimento se usan valores para T de 100, 1000, 10000 y 100000 iteraciones y valores de γ de 10^0, 10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}. Al ejecutar este experimento varias veces, se encuentra que los valores de sus resultados para los mismo valores de T y de γ, tienen tendencias marcadas, lo que permite establecer que la influencia de la aleatoriedad en estos resultados es menor que la que se observó en los experimentos anteriores.

Tres de estos resultados se pueden ver en la tabla 4.9, donde, aunque el valor del estadístico de prueba F de las tablas ANOVA aún no presenta valores lejanos a la unidad, el ejemplo de la segunda fila muestra valores para $PR(> F)$ menores a 0.05, para ambas variables.

Estos cambios, que se ocasionan por la influencia de la aleatoriedad que aún se presenta, no permiten, una vez más, concluir que los valores de alguna de estas variables independientes influye efectivamente en los de la variable dependiente.

Aún así, en las gráficas de la tabla 4.9 se observa que se obtienen resultados de la diferencia normalizada de ganancias con medias que están por debajo del 80 % y que llegan a tomar valores cercanos al 0 %, especialmente con los valores más altos

Tabla 4.9: ANOVA para el Experimento dos con Gamma

GRÁFICA PARA γ	**GRÁFICA PARA T**	**GRÁFICA PARA T y γ**	**ANOVA**
			F PR(>F) δ 1.692930 0.180107 T 0.618445 0.612032
			F PR(>F) δ 2.886622 0.037647 T 3.463922 0.038132
			F PR(>F) δ 2.112502 0.102301 T 2.664158 0.078977

de la variable T y más bajos para la variable γ. Lo anterior sugiere que con pequeños pasos y mayor cantidad de ellos, se puede llegar a obtener mejores resultados, lo que se verifica en el siguiente experimento.

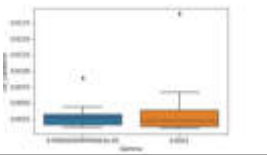
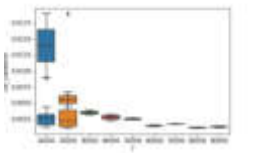
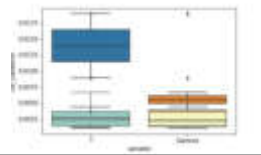
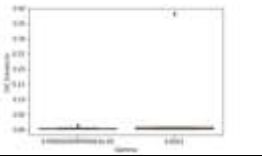
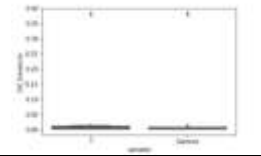
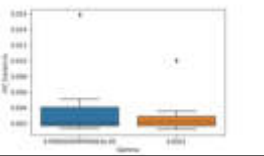
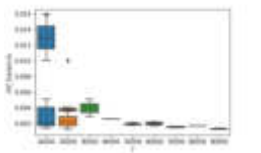
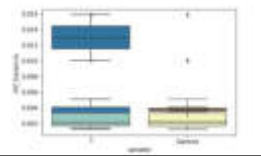
4.2.2.8 Experimento tres

Para este experimento se usan valores para T de 10000 a 100000 iteraciones con intervalos de 10000 y valores de γ del orden de 10^{-4}, teniendo en cuenta que fue la cifra que mostró los valores más bajos para la variable dependiente.

Las gráficas correspondientes a tres ejecuciones de este experimento con sus gráficas y resultados del ANOVA, se presentan en la tabla 4.10.

Aunque las gráficas generadas al ejecutar este experimento varias veces, con los mismo valores de T y de γ, parecen diferentes, mantienen valores para la variable independiente muy cercanos a cero, con valores inferiores al 2 % para todos los casos probados, lo que permite establecer una baja influencia de la aleatoriedad.

Tabla 4.10: ANOVA para el Experimento tres con Gamma

GRÁFICA PARA γ	**GRÁFICA PARA T**	**GRÁFICA PARA T y γ**	**ANOVA**
			F PR(>F) δ 1.146208 0.315578 T 5.622997 0.012393
			F PR(>F) δ 0.973408 0.352728 T 0.975696 0.513452
			F PR(>F) δ 2.185366 0.177585 T 14.020808 0.000584

Los valores del estadístico de prueba F y de $PR(> F)$ de las tablas ANOVA muestran que no hay una dependencia entre los valores de la variable γ y los resultados de la variable dependiente, puesto que F toma valores cercanos a 1 y, $PR(> F)$ es mayor a 0.05.

La independencia aparente de la variable γ con los valores que toma la variable dependiente (diferencia promedio de ganancias distorsionadas) resulta ser un aspecto deseable para el experimento, puesto que los dos valores tomados para γ solo difieren en $1 * 10^{-5}$, como se aprecia más claramente en la figura 4.4, porque lo esperado era obtener resultados similares.

De otra parte, en dos de los tres casos hay influencia de los valores que se dan a la variable T sobre los obtenidos en la variable dependiente, ya que el valor de la probabilidad $PR(> F)$ es menor que 0.05; aspecto que resulta positivo para el modelo, porque lo acercan a un control inicial de los efectos de la aleatoriedad.

De aquí que, con cualquier valor de T superior a 50000 iteraciones se obtuvieron excelentes resultados, con medias en la diferencias promedio de ganancias inferiores a

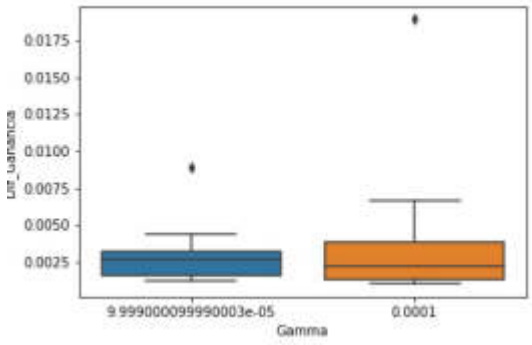


Figura 4.4: Diferencia normalizada de ganancias para T y γ

0.003 y varianzas muy pequeñas. Se elige entonces el valor de 100000 para la variable T y de 10^{-4} para la variable γ, para generar el histograma en el que se apreciará cómo, efectivamente, la mayor parte de los valores de la variable dependiente se acerca a cero. Dicho histograma se muestra en la figura 4.5.

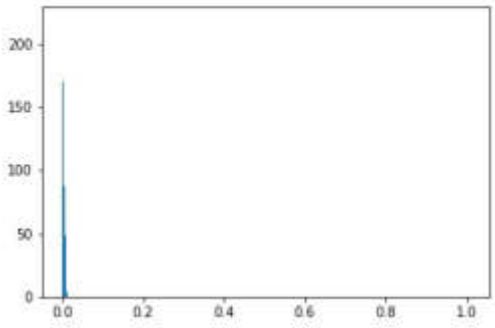


Figura 4.5: Histograma de diferencia normalizada de ganancia

La generación de histogramas como el de la figura 4.5 se realizó varias veces, encontrando siempre resultados similares, lo que permite concluir que los valores de T y de γ, probados en este experimento, permiten obtener valores para las ganancias distorsionadas muy cercanos a los valores reales y con muy poca variación, valores obtenidos a pesar de la aleatoriedad presente en la construcción de las rutas que prueba el algoritmo.

Teniendo establecidos los parámetros de número de iteraciones y del factor de aprendizaje γ, se procede a probar el desempeño del algoritmo con grafos generados

de forma aleatoria.

4.3 Generación de un grafo aleatorio

De acuerdo al diseño explicado en la sección 3.4, se ejecutó el programa de forma que aceptara como datos de entrada, además de los ya implementados, la cantidad de etapas del grafo y la cantidad de nodos en cada etapa. La generación de la matriz de adyacencia ha previsto para que cada nodo tenga al menos un nodo adyacente de la siguiente etapa, excepto si es de la ultima etapa, así como una densidad de conectividad entre los nodos del 66.67 %.

Los valores de los *bandits* se generan también de forma aleatoria con media en cero y desviación estándar de una unidad, pero se modifican computacionalmente para que uno de los nodos de cada etapa, elegido al azar, tenga asociada una ganancia positiva superior a la de los demás nodos de su etapa.

Los valores generados para los *bandits* se ajustaron para que las ganancias promedio quedaran del orden de las centenas, como las del grafo de prueba, ya que el valor de γ depende de la magnitud de la ganancia que se obtenga en cada iteración.

El programa arroja resultados favorables con cantidad de etapas L entre 5 y 12, observados en la convergencia del promedio de sus ganancias hacia la mayor ganancia real, dentro de los caminos probados por el algoritmo, como la que se aprecia en la figura 4.7. Uno de los casos se verifica manualmente, obteniendo el grafo a partir de la matriz de adyacencia generada aleatoriamente y de los valores generados para sus *bandits*, con un valor para L de 9 etapas y los siguientes valores para la cantidad de nodos por etapa: [1. 3. 4. 4. 3. 1.].

La gráfica se muestra en la figura 4.6 y su matriz de adyacencia es la siguiente:

$$A = \begin{bmatrix} 0,0,1,1,0,0,0,0,0,0,0,0,0,0,0,0 \\ 0,0,0,0,1,1,1,0,0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,1,0,0,0,0,0,0,0,0,0,0 \\ 0,0,0,0,1,0,1,1,0,0,0,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0,1,1,1,0,0,0,0,0 \\ 0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0 \\ 0,0,0,0,0,0,0,0,1,1,0,1,0,0,0,0 \\ 0,0,0,0,0,0,0,0,0,0,1,1,0,0,0,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1,0 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1 \\ 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0 \end{bmatrix}$$

Así mismo el vector de *bandits* generado de forma aleatoria fue el siguiente:

$B = [196.02255721, 9.32455363, 47.72564534, 66.21935444, 275.7996602, 6.62172573, 190.49629308, 13.33418851, 210.36419278, 37.39029137, 191.40773576, 117.1717095, 40.02228911, 220.45436046, 32.32878409, -125.16559689]$

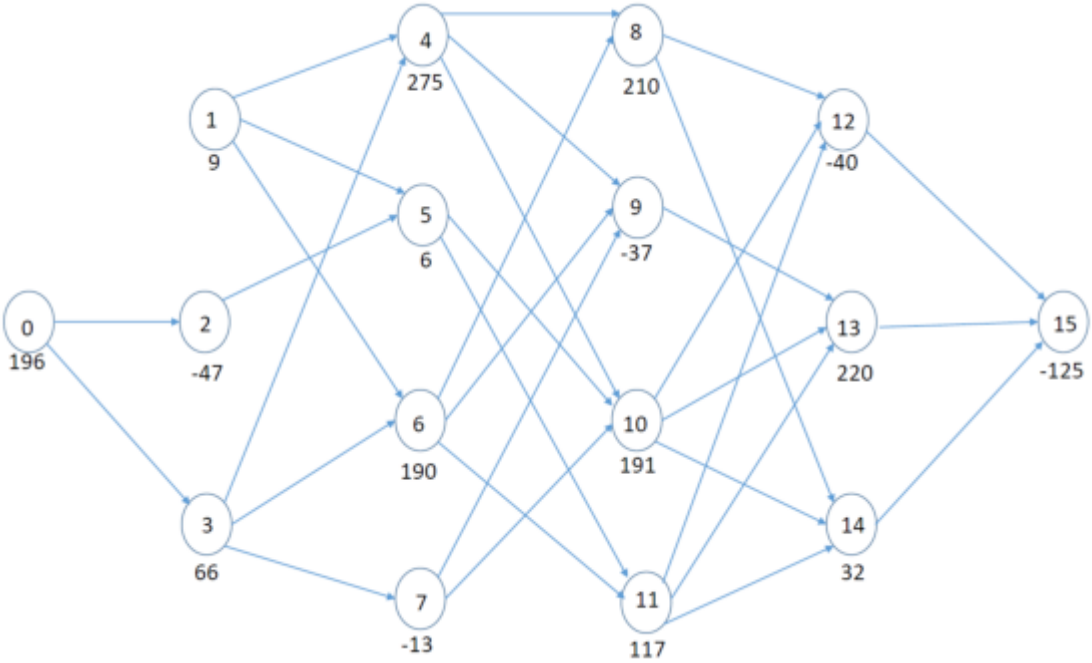


Figura 4.6: Grafo generado en forma aleatoria

4.3.1 Prueba

Se recuerda que las variables independientes son: la cantidad de Iteraciones T y el factor de aprendizaje γ. El valor asignado al parámetro de T es de 80000 iteraciones y el de γ es de 0.000003.

De otra parte, en esta prueba, además de la variable dependiente que se ha manejado en los experimentos anteriores, que ha sido el promedio de diferencias de ganancias normalizadas, se adiciona un indicador de salida que cuenta las veces que el vector solución generado por la aplicación corresponde con el vector óptimo.

Se mantiene la funcionalidad de la aplicación para establecer los valores óptimos del vector solución y de su ganancia real asociada, los que se obtienen guardando el valor de la ganancia real máxima que se vaya generando y su argumento (es decir el vector objeto de dicha ganancia).

Luego de la 80000 iteraciones con los parámetros ya explicados se obtuvieron las siguientes salidas:

- Promedio de la diferencia entre las ganancias obtenidas en cada iteración y la ganancia óptima: 0.019839554951030753, que es un valor inferior al 2 %.
- Porcentaje de aciertos entre los vectores generados en cada iteración y el vector óptimo: 0.9531375, valor que supera el 95 % de coincidencias.
- Vector óptimo = [0, 3, 4, 10, 13, 15], que corresponde al calculado del grafo de forma manual.

Además se generó la gráfica de convergencia de la ganancia que se obtienen en cada iteración y el resultado se muestra en la figura 4.7. En esta figura el eje horizontal corresponde al número de la iteración y el eje vertical a la ganancia obtenida en cada iteración.

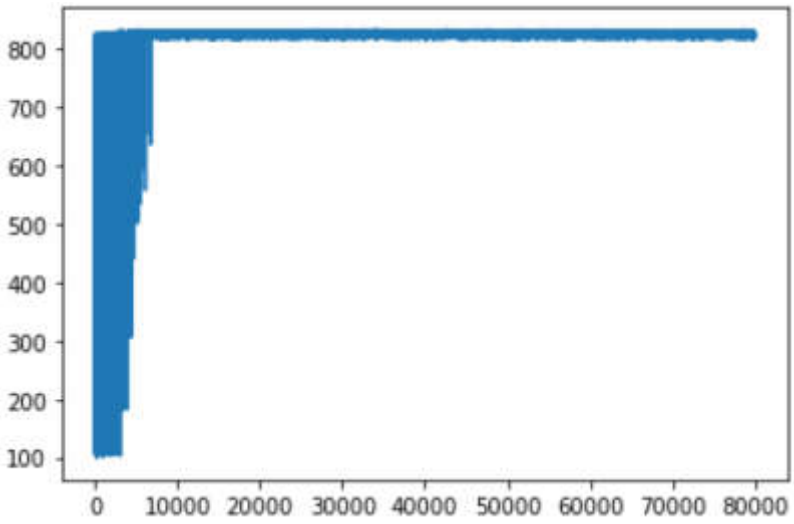


Figura 4.7: Convergencia de ganancia con un grafo generado en forma aleatoria

Es importante aclarar que, no se espera que el promedio de diferencias entre ganancias llegue a ser cero ni que el porcentaje de aciertos del vector generado con el vector óptimo llegue a ser uno, puesto que el autómata toma un tiempo en aprender la ruta óptima y solo se tendrán en cuenta los valores que el autómata haya tenido que probar durante las iteraciones.

Finalmente se resalta el hecho de que la gráfica se estabiliza después de solo 10000 iteraciones, pero es necesario ejecutar cerca de 100000 para obtener resultados aceptable en el promedio normalizado de diferencias entre ganancias y en el porcentaje de aciertos con el vector óptimo; conseguir un criterio de parada para el algoritmo que disminuya el número de iteraciones será parte de la continuidad de esta investigación.

4.4 Posibles aplicaciones del modelo

Como se ha expresado en apartes anteriores, todo modelo de toma de decisiones que se pueda modelar con el grafo por etapas diseñado, del que se deba seleccionar una actividad por etapas, será susceptible de convertirse en una aplicación de esta

tesis.

En el anexo A se muestra, con un caso específico relacionado con la elaboración de un documento, la adaptación del mismo a un grafo por etapas, a partir del modelado usual para casos CMMN descrito en la sección 2.4.3.

Además, en el anexo A se presentan recomendaciones que ayudan a decidir los nodos y las arista del grafo por etapas, teniendo en cuenta que el el modelo CMMN original se puede extraer información sobre posibles secuencialidades entre acciones y sus prerrequisitos entre otras.

Capítulo 5

Conclusiones

Al finalizar el trabajo de tesis, se establece que se ha dado solución a lo planteado en la sistematización del problema científico, y por ende, en los objetivos propuestos, así:

- Se logró con el modelo propuesto, dar manejo a la incertidumbre en la toma de decisiones para problemas que se puedan representar con el grafo por etapas propuesto, desde una base matemática y teórica. Esto se obtuvo luego de probar los parámetros T y γ, correspondientes a la cantidad de iteraciones y al coeficiente de aprendizaje, hasta que la variable dependiente alcanzó un valor deseable con baja variabilidad, a pesar de los componentes aleatorios del modelo.

- Se construyó un modelo de aprendizaje y se probó la convergencia de la ganancia hallada hacia una ganancia real, con valores aleatorios para el grafo propuesto. El modelo se basó en el cálculo de una matriz de probabilidades de transición entre estados, siendo los estados cada uno de los nodos del grafo. Estas probabilidades se actualizaron en cada una de las iteraciones o búsquedas, de acuerdo a una función de valor que iba cambiando para cada nodo involucrado en la búsqueda anterior, valor que indica la preferencia por ese nodo.

- La implementación de este modelo tuvo en cuenta las características del grafo por etapas, el modelo de probabilidades y función de valor para el aprendizaje, la generación de una ruta factible en cada uno de los tiempos de ejecución T y el valor de un factor de aprendizaje γ, como dosificador de la magnitud de la ganancia, que sirve de retroalimentación para el aprendizaje en cada iteración T. Su evaluación se llevó a cabo con la variación controlada de parámetros como T y γ y con la generación aleatoria tanto de los nodos que conforman una ruta, especialmente en las primeras iteraciones (donde se da campo a la exploración), como de valores asociados a cada uno de los nodos, con los que se podía estimar una ganancia para cada ruta. El funcionamiento fue tal que a partir de cierta cantidad de iteraciones, la aplicación seguía escogiendo la mejor ruta, con la que iba mejorando la estimación de la ganancia que realmente se había asignado a esta solución, lo que corresponde a la explotación en el aprendizaje por refuerzo.

- Se logra modelar en el grafo por etapas el manejo de casos dinámicos por parte de las BPMS, partiendo de la notación CMMN, notación gráfica que maneja varios elementos entre los que se cuentan las líneas punteadas para las acciones o para los flujos no-deterministas, como factor primordial de la incertidumbre presente en el caso. Tal transición hizo necesaria la copia de algunas acciones en diversas etapas, la creación de actividades nulas o ficticias en cada etapa y el posterior análisis de los flujos para respetar las precedencias de las acciones del modelo CMMN, pero dando prioridad a que estas conexiones fueran suficientes para posibiliten todas las rutas de acciones o nodos posibles.

El modelo propuesto es una implementación, hasta donde sabemos, nueva y muy simple en el campo de aprendizaje por refuerzo usando probabilidades. Se inscribe dentro del campo de los sistemas de decisión de Markov de estado finito y tiempo discreto (por episodios) con un número de tiempos en cada episodio también finito, específicamente para problemas dispuestos en un grafo por etapas, donde se requiere aprender de forma eficiente una trayectoria óptima que debe seguir un

agente.

La estructura del modelo de decisión como grafo dirigido por etapas permite construir un algoritmo en el que las probabilidades de transición entre estados se actualizan directamente a partir de la función de valor estimada para cada nodo, dado que la estructura del grafo permite un cálculo eficiente de la constante de normalización de dichas probabilidades.

Capítulo 6

Discusión y trabajo futuro

Es importante abrir una discutir sobre los aspectos positivos y negativos del trabajo de tesis y las aportaciones que han hecho a la ciencia, entre otros, así como presentar algunas ideas de temas de investigación que pueden ampliar este proyecto.

6.1 Los pros y los contras

Siendo el aporte principal de este trabajo un modelo computacional en el área de la optimización, no se evidencias connotaciones éticas ni aspectos que afecten directamente a la humanidad, pero como todos los proyectos que hacen parte de la Inteligencia Artificial, podrá ser fuente de discusión sobre su uso adecuado en el futuro.

Este proyecto, como otros diseñados para apoyar la toma de decisiones, busca, básicamente hacer predicciones sobre lo que se espera que ocurra en determinado contexto; y, es la predicción uno de los aspectos que la Inteligencia Artificial ha conseguido ahondar, con resultados impresionantes para el años 2021, en que nos encontramos.

Tales resultados podrán aportar positivamente a la humanidad, como es el caso de la aplicabilidad que se prevé para este trabajo en el apoyo a casos, siendo

los del área de la medicina los casos más frecuentes, por su incertidumbre y baja repetitividad.

Pero queda abierta la posibilidad, como ha sucedido con diferentes avances científicos, de que los resultados se adapten para beneficios egoístas o para ideas maliciosas que en este momento no se puedan visualizar.

Ahora bien, la tesis aquí expuesta pretende hacer una porte positivo a la ciencia y espera que tenga provecho para la humanidad.

En seguida se proponen algunos aspectos que darían continuidad a este estudio.

6.2 Trabajo futuro

Aunque la aproximación de las probabilidades de transición de entre nodos, generó resultados satisfactorios, es un trabajo en el que se puede avanzar en varios aspectos:

- Modificar la aplicación para que pueda mantener un resultado confiable, con un menor número de iteraciones.
- Diseñar e implementar aplicaciones con diferentes bases técnicas y teóricas, para el problema de búsquedas en el grafo por etapas propuesto.
- Diseñar e implementar una aplicación para el grafo por etapas que permita la no selección de un nodo en alguna de las etapas.
- Modificar la aplicación para permitir que la selección de un nodo en alguna de las etapas, esté apoyada por reglas o condiciones, o por heurísticas.
- Modelar un problema de planificación automática con el grafo por etapas, adaptándolo desde las características propias del grafo de tipo And/Or.

Apéndice A

Propuesta para CMMN

Se retoma en la figura A.1 el grafo que se genera al modelar un Caso con la notación CMMN. De tal esquema surge esta propuesta.

A.1 Modelo CMMN en un grafo por etapas

Es importante recordar que las líneas punteadas de la figura A.1 representan que la secuencia que se puede seguir o la presencia de algunas actividades es posible, pero no determinista.

Se vislumbra entonces, cómo esta notación CMMN puede llevarse al grafo con etapas propuesto, donde se pueda modelar la incertidumbre presente en la selección de tareas, disponiendo en cada etapa el conjunto de actividades posibles, para que el algoritmo de aprendizaje recomiende la tarea a seguir de la siguiente etapa.

Para esto se propone:

- Si dos tareas b y c están después de una tarea a, pero b y c no son disjuntas, entonces se deben ofrecer b y c en cualquiera de las etapas siguientes.
- Colocar una actividad nula en cada etapa para cuando no se ha de tomar alguna.

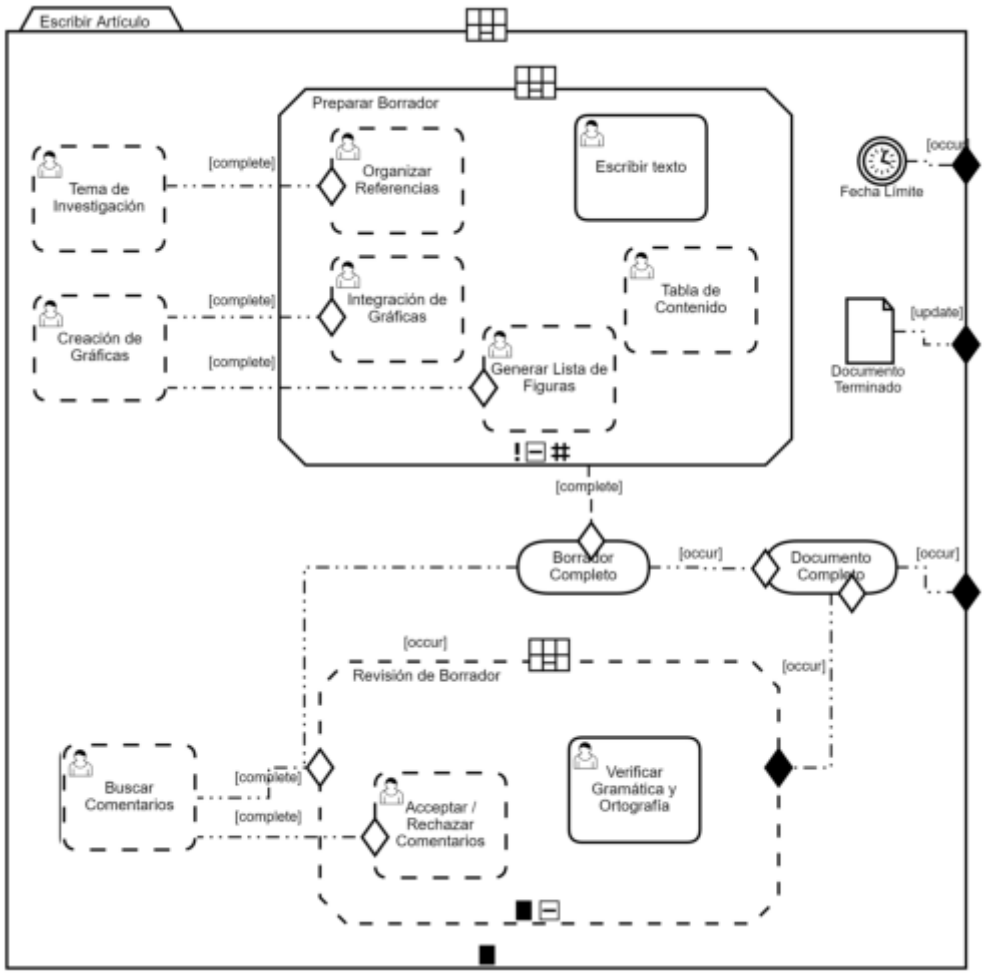


Figura A.1: Grafo en CMMMN que se llevará al modelo propuesto

- Dejar las aristas suficientes para las posibles conexiones entre nodos, sin ocuparse demasiado por los prerrequisito, puesto que el algoritmo ha de aprender cuando los caminos no sean viables.

La figura A.1 representa las tareas para escribir un artículo. En ella se pueden diferenciar claramente dos partes concernientes a la preparación del borrado y a su revisión. Estas partes permiten separar en dos figuras el grafo resultante de la propuesta de esta tesis.

La primera parte del grafo que se generaron a partir de esas apreciaciones se presentan en la figura A.2.

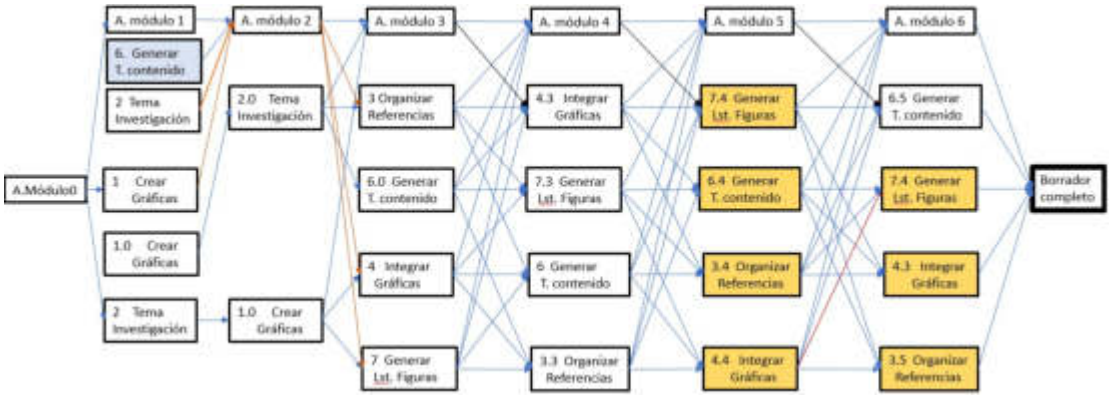


Figura A.2: Caso en etapas. Parte 1.

La segunda parte, la que corresponde a la parte inferior del esquema original, se puede apreciar en la figura A.3.

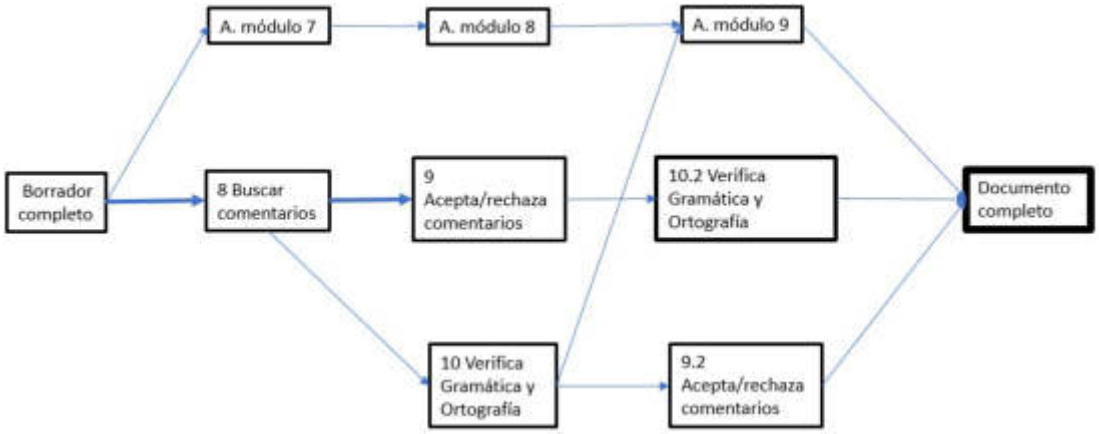


Figura A.3: Caso en etapas. Parte 2.

También se ha considerado, para cuando se hace necesario traer de alguna forma la información de actividades pasadas, que son predecesoras de otras, activar una bandera que lo indique. Así mismo se analizó la posibilidad de fusionar actividades no disyuntas en un nodo ficticio, con la salvedad de que no es fácil esclarecer con certeza sus prerrequisitos.

Apéndice B

Breve reseña de la Gestión de Casos

La administración de los procesos ha sido una preocupación constante de los investigadores que han estado en la búsqueda de la optimización de los mismos, específicamente del flujo de los procesos conocido como *Workflow*, para proporcionar mayor eficiencia a las empresas que los ejecutan. Se encuentra en Van der Aalst *et al.* (2005) una revisión juiciosa de sistemas que administran los flujos de los procesos, y cómo, desde las fechas en que se proponen, en general, del siglo pasado, se diseñan modelos que permiten la flexibilidad de los mismos, atendiendo a las características dinámicas que han presentado los procesos y que hoy sigue ocupando a la comunidad científica. En este aparte se revisan técnicas de Inteligencia Artificial para establecer su aplicación en la adaptación de los BPMS ante la incertidumbre en los procesos.

El término Proceso de Negocio o BP (*Business Process*) se refiere al conjunto de actividades que se ejecutan dentro de un proceso, en un orden determinado y de acuerdo a unos eventos iniciales. Así mismo, la Gestión de Procesos de Negocio o el BPM (*Business Process Management*), viene gestionando estos procesos de negocios, mediante suites llamadas BPMS, que integran un conjunto de facilidades tecnológicas para el dueño del negocio.

Varias de las funcionalidades con que cuentan las BPMS han sido mejoradas en los últimos años con la ayuda de técnicas de Inteligencia Artificial, para ofrecer mejor servicio a los clientes, acuñándose la sigla iBPM, que significa BPM Inteligente o iBPMS como sigla para las Suites o Sistemas para la Gestión de Procesos de Negocios Inteligentes.

Una BPMS se apoya en un conjunto de Reglas de Negocio - RN que contiene el conocimiento de los conocedores del negocio y que guían las decisiones en la ejecución de los procesos. En la mayoría de los casos se cuenta con un motor de RN que puede manejar las reglas en forma independiente de los procesos.

De otra parte, dado que los procesos de un negocio pueden ser dinámicos en el tiempo, se tiene la Gestión de Casos Adaptativos o *ACM - Adaptive Case Management* (Pucher, 2010), que pretende manejar la complejidad de los procesos donde no es fácil predecir el flujo de acciones o tareas que se requieren y en cuyas herramientas también se han vinculado técnicas de Inteligencia Artificial para lograr sus objetivos.

Los casos, que se pretende apoyar con esta investigación, se definen en Van der Aalst *et al.* (2005) como las situaciones que pueden ocurrir en el negocio y para las que no necesariamente se tienen predefinidos unos procedimientos a seguir. Allí se citan investigaciones basadas en Inteligencia Artificial que, es su momento, trabajaron en algoritmos de planificación y coordinación de procesos.

Para continuar con los antecedentes en BPM que permiten entender mejor el problema que apoya esta investigación, se revisó el cuadrante mágico de Garner de los *iBPMS*, para establecer que Pegasystem, Appian e IBM son los fabricantes de suites más visionarios, que vienen liderando la oferta de servicios con mayor habilidad de ejecución, y que se plasma en la figura B.1. Las características inteligentes de estas tres suites se resumen en seguida.

Pegasystem da uso a las tecnologías de la inteligencia artificial permitiendo a las empresas predecir el comportamiento de sus clientes para mejorar sus resultados.

Figura B.1: Cuadrante mágico de Garner para Suits de Gestión de Procesos de Negocio Inteligentes 2019

Además, refiere que ofrece soluciones inteligentes en BPM que incluye administración de casos, reglas de negocios y desarrollo de aplicaciones móviles, entre otras funcionalidades (Pega, 2013).

IBM Knowledge Center (2017) muestra su inteligencia con IBM BPM Advancer, con la cual ofrece a las empresas la facilidad de la Gestión de Casos, mediante el aplicativo Basic Case Management, donde se tiene en cuenta lo impredecible de la secuencia de actividades en la gestión de casos y da las instrucciones al dueño del negocio para hacer uso de esta funcionalidad. Adicionalmente, cuenta con la herramienta IBM Watson que brinda funcionalidades de inteligencia artificial a las empresas, entre las que se cuenta el aprendizaje dinámico (IBM Sala de Prensa, 2017).

Appian a su vez, expone que, mediante el Aprendizaje de Máquina, las BPMS están actuando sobre los grandes volúmenes de datos con que cuentan las empresas gracias al Internet de las Cosas; de aquí se descubren los gustos y tendencias de

los clientes y con ello se apoya la toma de decisiones de las empresas; muestra la aplicación del Aprendizaje de Máquina en la automatización de procesos como una tendencia futura (Potrzeba, 2016). Algunas de las facilidades que ofrece la BPMS Appian son relacionadas por Alarcón Matta (2007), así: gestión de documentos, gestión del contenido, herramientas de colaboración y soporte a comunidades de conocimiento.

Así, se aprecia que, en los últimos años, las BPMS han desarrollado facilidades que involucran técnicas de inteligencia Artificial o Machine Learning, para brindar un servicio más eficiente a las empresas que deben lidiar con procesos altamente dinámicos.

De otra parte, Business Process Model and Notation (BPMN) y Case Management Model and Notation (CMMN) son estándares para el modelado de procesos; el primero modela procesos estáticos y el segundo procesos dinámicos o Casos. De acuerdo con Auer *et al.* (2014), estas notaciones deben combinarse para atender los requerimientos cambiantes de los negocios aprovechando la simplicidad del BPM.

El Aprendizaje Automático o *Machine Learning* es una de las áreas de la Inteligencia Artificial que ha permitido un manejo eficiente de la información, teniendo en cuenta el volumen de datos que cada día se manejan en los negocios y en el mundo en general, apoyada en diferentes técnicas como las redes Neuronales, los Algoritmos Genéticos, las Colonias de Hormigas y el Soporte de Máquina Vectorial, entre otras. Técnicas como estas se vienen aplicando desde las investigaciones en los Workflow, como la referenciada por Herbst (2000) quienes aplican técnicas de Machine Learning a los grafos de los procesos e implementan técnicas de inferencia gramatical que están restringidas a flujos de trabajo secuenciales en Worflow concurrentes, dejando abierta la investigación para otros interesados.

Bibliografía

ABPMP, B. C. (2013), «v3. 0», *Guide to the Business Process Management Common Body of*.

Alarcón Matta, J. L. (2007), «Modelo de gestión del conocimiento aplicado a la gestión de procesos de negocio», Universidad Nacional Mayor de San Marcos. Programa Cybertesis PERÚ.

Alon, N., N. Cesa-Bianchi, O. Dekel y T. Koren (2015), «Online learning with feedback graphs: Beyond bandits», en *Annual Conference on Learning Theory*, tomo 40, Microtome Publishing.

Alon, N., N. Cesa-Bianchi, C. Gentile, S. Mannor, Y. Mansour y O. Shamir (2017), «Nonstochastic multi-armed bandits with graph-structured feedback», *SIAM Journal on Computing*, **46**(6), págs. 1785–1826.

Auer, D., S. Hinterholzer, J. Kubovy y J. Küng (2014), «Business process management for knowledge Work: Considerations on current needs, basic concepts and models», en *Novel methods and technologies for enterprise information systems*, Springer, págs. 79–95.

Ávila Cartes, J. E. (2018), *Diseño y análisis de algoritmos para encaminamiento en redes de comunicación con fallos*, Tesis Doctoral, Universidad de Concepción. Facultad de Ciencias Físicas y Matemáticas. Departamento de Ingeniería Matemática.

Brandstädt, A., V. B. Le y J. P. Spinrad (1999), *Graph classes: a survey*, SIAM.

Breitenmoser, R. y T. Keller (2015), «Case management model and notation-a showcase», *European Scientific Journal*, **11**(25).

Frece, A., G. Srdić y M. B. Jurič (2012), «BPM and iBPMS in the Cloud», en *The 1st International Conference on CLoud Assisted ServiceS*, pág. 54.

Garimella, K., M. Lees y B. Williams (2008), «Introducción a BPM para Dummies», .

Gokcesu, K. y S. S. Kozat (2018), «An online minimax optimal algorithm for adversarial multiarmed bandit problem», *IEEE transactions on neural networks and learning systems*, **29**(11), págs. 5565–5580.

Group, O. M. (2013), *Case Management Model and Notation (CMMN)*, Object Management Group, URL `https://www.omg.org/spec/CMMN/1.0/Beta1/PDF`.

Group, O. M. (2016), *Case Management Model and Notation (CMMN) Version 1.1*, URL `https://www.omg.org/spec/CMMN/1.1/PDF`.

Hauder, M., S. Pigat y F. Matthes (2014), «Research challenges in adaptive case management: a literature review», en *2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations*, IEEE, págs. 98–107.

Herbst, J. (2000), «A machine learning approach to workflow management», en *European conference on machine learning*, Springer, págs. 183–194.

Hernández Sampieri, R., C. Fernández Collado, P. Baptista Lucio *et al.* (2010), «Metodología de la investigación», .

Hitpass, B. (2017), *BPM: Business Process Management: Fundamentos y Conceptos de Implementación 4a Edición actualizada y ampliada*, Dr. Bernhard Hitpass.

Hitpass, B., J. Freund, B. Rucker y B. Hitpass (2017), *BPMN Manual de Referencia y Guía Práctica 5a Edición: Con una introducción a CMMN y DMN*, Dr. Bernhard Hitpass.

IBM Knowledge Center (2017), «Gestión de procesos de negocio y gestión de casos», *Informe técnico*.

IBM Sala de Prensa (2017), «Inteligencia Artificial de IBM alcanzará a 100 millones de personas en América Latina en 2017», URL `http://www-03.ibm.com/press/mx/es/pressrelease/52757.wss`.

Khoshafian, S. y C. Rostetter (2015), «Digital Prescriptive Maintenance», *Internet of Things, Process of Everything, BPM Everywhere*, págs. 1–20.

Kim, Y. M., A. Kols, A. Martin, D. Silva, W. Rinehart, S. Prammawat, S. Johnson y K. Church (2005), «Promoting informed choice: evaluating a decision-making tool for family planning clients and providers in Mexico», *International family planning perspectives*, págs. 162–171.

Lakshmanan, G. T., S. Duan, P. T. Keyser, F. Curbera y R. Khalaf (2010), «Predictive analytics for semi-structured case oriented business processes», en *International Conference on Business Process Management*, Springer, págs. 640–651.

Lee, S. y T. Chung (2005), «A reinforcement learning algorithm using temporal difference error in ant model», en *International Work-Conference on Artificial Neural Networks*, Springer, págs. 217–224.

Liu, K. y Q. Zhao (2011), «Multi-armed bandit problems with heavy-tailed reward distributions», en *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, págs. 485–492.

Liu, K. y Q. Zhao (2012), «Adaptive shortest-path routing under unknown and stochastically varying link states», en *2012 10th International Symposium on Mo-*

deling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt), IEEE, págs. 232–237.

López Guarín, C. E. *et al.* (2013), *Data mining model to predict academic performance at the Universidad Nacional de Colombia*, Tesis Doctoral, Universidad Nacional de Colombia.

Marin, M. A. (2016), «Introduction to the case management model and notation (CMMN)», *arXiv preprint arXiv:1608.05011*.

Omg, O., R. Parida y S. Mahapatra (2011), «Business process model and notation (bpmn) version 2.0», *Object Management Group*.

Parnell, G. S., P. J. Driscoll y D. L. Henderson (2011), *Decision making in systems engineering and management*, tomo 81, Wiley Online Library.

Pega (2013), «Hoja de datos corporativa de Pegasystems», URL `https://www1.pega.com/es/insights/resources/hoja-de-datos-corporativa-de-pegasystems`.

Potrzeba (2016), «Machine Learning Contributing to BPM Innovation», URL `https://www.appian.com/blog/bpm/machine-learning-contributing-to-bpm-innovation`.

Pucher, M. J. (2010), «The elements of adaptive case management», *Mastering the Unpredictable*, págs. 89–134.

Russell, S. J. y P. Norvig (2004), *Inteligencia Artificial: un enfoque moderno*, 04; Q335, R8y 2004.

Silver, D., A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.* (2016), «Mastering the game of Go with deep neural networks and tree search», *nature*, **529**(7587), pág. 484.

Silver, D., J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton *et al.* (2017), «Mastering the game of go without human knowledge», *Nature*, **550**(7676), págs. 354–359.

Smith, H. y P. Fingar (2003), *Business process management: the third wave*, tomo 1, Meghan-Kiffer Press Tampa.

Sutton, R. S., A. G. Barto y R. J. Williams (1992), «Reinforcement learning is direct adaptive optimal control», *IEEE Control Systems Magazine*, **12**(2), págs. 19–22.

Sutton, R. S., A. G. Barto *et al.* (1998), *Introduction to reinforcement learning*, tomo 135, MIT press Cambridge.

Thompson, W. R. (1933), «On the likelihood that one unknown probability exceeds another in view of the evidence of two samples», *Biometrika*, **25**(3/4), págs. 285–294.

Tossou, A. C., C. Dimitrakakis y D. Dubhashi (2017), «Thompson sampling for stochastic bandits with graph feedback», en *Thirty-First AAAI Conference on Artificial Intelligence*.

Tremblay, J. P. y W. K. Grassmann (1996), *Matemática discreta y lógica*, Prentice Hall.

Underdahl, B. (2013), «Gestión de procesos de negocio para dummies. Segunda edición limitada de IBM. Nueva Jersey», .

Valko, M. (2016), *Bandits on graphs and structures*, Tesis Doctoral.

Van der Aalst, W. M., M. Weske y D. Grünbauer (2005), «Case handling: a new paradigm for business process support», *Data & Knowledge Engineering*, **53**(2), págs. 129–162.

website, E. B. (2016), «The Softmax function and its derivative», "https://eli.thegreenplace.net/2016/the-softmax-function-and-its-derivative/".

Xu, X., S. Vakili, Q. Zhao y A. Swami (2017), «Online learning with side information», en *MILCOM 2017 - 2017 IEEE Military Communications Conference (MILCOM)*, págs. 303–308.

Xu, X., S. Vakili, Q. Zhao y A. Swami (2017), «Online learning with side information», en *MILCOM 2017-2017 IEEE Military Communications Conference (MILCOM)*, IEEE, págs. 303–308.

Zhou, P., J. Xu, W. Wang, Y. Hu, D. O. Wu y S. Ji (2019), «Toward Optimal Adaptive Online Shortest Path Routing With Acceleration Under Jamming Attack», *IEEE/ACM Transactions on Networking*, **27**(5), págs. 1815–1829.

Resumen autobiográfico

Carmen Constanza Uribe Sandoval

Candidato para obtener el grado de
Doctorado en Ingeniería
con especialidad en Ingeniería de Sistemas

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Tesis:
Aprendizaje por Refuerzo para la toma de decisiones en problemas con alta incertidumbre que se puedan modelar por etapas

Nacida en la ciudad de Tunja, capital del departamento de Boyacá en Colombia. Hija de Laurentino Uribe López (qepd) y de María Antonia Sandoval Garavito. Cursó estudios de primaria en la Normal femenina de Tunja y en el Instituto Integrado Guillermo León Valencia de Duitama, ciudad cercana a Tunja, donde culminó su bachillerato en el Colegio Salesiano.

Sus formación profesional de pregrado fue como Ingeniera de Sistemas en la Universidad Industrial de Santander, en la ciudad de Bucaramanga (Colombia) y de maestría fue como Magister en Ingeniería de Sistemas de la Universidad Nacional de

Colombia, en Bogotá (Colombia).

Desde su graduación como profesional, se ha desempeñado como docente de la Universidad de Boyacá en su tierra natal, donde ha sido representante docente ante los consejos de facultad y académico, Secretaria del Consejo de Facultad de Ciencias e Ingeniería y Directora del programa de Ingeniería de Sistemas en la misma universidad. También se ha desempeñó como docente ocasional en universidades de Bogotá como la Universidad Nacional de Colombia, la Universidad Autónoma, la Universidad Católica de Colombia, la Universidad América, la Universidad Javeriana y la Universidad Agraria.

Su gusto por la Matemática le ha permitido orientar algunas de las asignaturas de matemática para ingeniería, como lógica, matemática discreta, teoría de la computación, métodos numéricos e investigación operacional, así como la matemática básica, que buscar nivelar los conocimientos que los estudiantes de primer semestre traen de sus diferentes centros educativos de secundaria, siendo esta su asignatura de mayor éxito, por el amor a la matemática que siembra en sus estudiantes.