

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS
SUPERIORES DE MONTERREY



**Representación y Aprendizaje de
Procesos de Decisión de Markov
Cualitativos**

Autor:

Alberto Reyes Ballesteros

Sometido al Programa de Graduados en Informática y
Computación en cumplimiento parcial con los
requerimientos para obtener el grado de:

Doctor en Ciencias Computacionales

Asesores:

Dr. Luis Enrique Sucar Succar

Dr. Eduardo Morales Manzanares


Dr. Pablo Ibargüengoytia González

Cuernavaca, Morelos. Noviembre 2006


Representación y Aprendizaje de Procesos de Decisión de
Markov Cualitativos

Presentada por:
Alberto Reyes Ballesteros

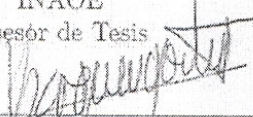
Aprobada por:



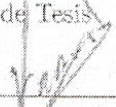
Dr. Luis Enrique Sucar Succar
Investigador de la Coordinación de Ciencias Computacionales
INAOE
Asesor de Tesis



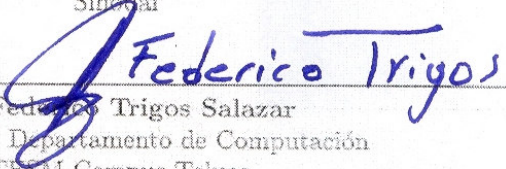
Dr. Eduardo Morales Manzanares
Investigador de la Coordinación de Ciencias Computacionales
INAOE
Asesor de Tesis



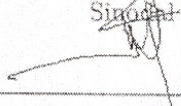
Dr. Pablo Ibarquiengoytia González
Investigador de la Gerencia de Control e Instrumentación
IIE
Asesor de Tesis



Dr. Juan Frausto Solís
Profesor del Departamento de Computación
ITESM Campus Cuernavaca
Sinodal



Dr. Federico Trigos Salazar
Profesor del Departamento de Computación
ITESM Campus Toluca
Sinodal



Dr. Jaime Mora Vargas
Profesor del Departamento de Computación
ITESM Campus Estado de México
Sinodal

A mi madre

Profra. Ruth Ballesteros Huerta

1936-2000

*quién nunca dudó que las metas que se persiguen
con el corazón siempre se pueden alcanzar*

A mi esposa

LAE. Maricela Romero Andrade

*por acompañarme en esta aventura
con gran paciencia y comprensión*

A mi hijo

Sebastián A. Reyes Romero

*quién con su gran amor a la vida
ha sido mi mayor fuente de inspiración*

Agradecimientos

Antes que nada, deseo darle las gracias y hacer un reconocimiento muy especial a mis asesores de tesis en el Instituto Nacional de Astrofísica Óptica y Electrónica (INAOE), Dr. Luis Enrique Sucar y Dr. Eduardo Morales, por su paciencia, confianza y, sobre todo, por los conocimientos y experiencias transmitida durante mi formación. De igual manera, a mis profesores en la Universidad de la Columbia Británica en Canadá (UBC), Prof. Martin L. Puterman y Prof. David Poole quienes con sus acertadas discusiones ayudaron a estructurar este trabajo. También deseo agradecer a mis amigos Pantelis Elinas (UBC) y Jesse Hoey (Universidad de Toronto) por su ayuda incondicional en el manejo de la herramienta SPUDD.

Deseo agradecer al Instituto de Investigaciones Eléctricas (IIE) el apoyo brindado para la realización de este trabajo a través de los proyectos 12504 y 12941. Particularmente al Dr. Salvador González Castro, Director de la División de Sistemas de Control y Coordinador del Programa de Doctorado en Casa IIE-ITESM. Asimismo, deseo hacer evidente mi reconocimiento al grupo de asesores técnicos del IIE, compañeros, y personal administrativo del programa Doctorado en Casa IIE-ITESM, especialmente al Dr. Pablo Ibarguengoytia González, y al M.I. Miguel Ángel Delgadillo Valencia por sus consejos y discusiones interesantes. Igualmente, deseo agradecer al Ing. Rafael Chávez Trujillo, Gerente de Control e Instrumentación, como facilitador de toda la infraestructura necesaria para la realización de pruebas experimentales, y a todos mis compañeros del IIE por su gran apoyo moral y confianza.

En el ITESM Cuernavaca deseo agradecer al programa de graduados en informática y computación por las facilidades prestadas, particularmente a mi profesor y director de programa Dr. Fernando Ramos Quintana. Igualmente, a mis examinadores Dr. Juan Frausto (ITESM Cuernavaca), Dr. Federico Trigos (ITESM Toluca), y Dr. Jaime Mora por sus valiosos comentarios para el enriquecimiento de este trabajo. Finalmente, quiero agradecer el apoyo de mis compañeros profesores, personal de apoyo informático y administrativo en el ITESM Cuernavaca, quienes también vivieron de cerca el desarrollo de esta investigación.

Resumen

La planificación automática en problemas del mundo real se ha convertido en una disciplina de interés para la comunidad científica ya que permite establecer computacionalmente rumbos de acción en casos que, por la complejidad del problema, un humano no puede abordar adecuadamente.

En particular, la planificación con incertidumbre permite generar estrategias de control en ambientes inciertos con metas multivaluadas, ponderando el costo de generación de un plan contra la ganancia de utilidad en el tiempo por ejecución del mismo (planes útiles). Gracias a los recientes adelantos tecnológicos en procesamiento de información y al perfeccionamiento de algoritmos en materia de teoría de decisiones y razonamiento con incertidumbre, han resurgido las técnicas basadas en los Procesos de Decisión de Markov (MDPs por sus siglas en inglés) como marco estándar para la planificación con incertidumbre. Una limitación de los MDPs es que ante problemas altamente dimensionales, con grandes espacios de acciones, y la existencia de variables continuas, se producen espacios de solución no manejables con algoritmos estándar.

En este trabajo se propone una técnica de representación de MDPs abstractos para simplificar espacios de estados muy grandes, que puede resolverse con métodos estándar de programación dinámica. Dado que esta técnica está basada en restricciones cualitativas impuestas por características (ó factores) propias del mismo problema de decisión, la hemos llamado *MDPs cualitativos*. Aun cuando el método de representación resulta novedoso y fácil de implementar, la especificación manual del modelo de decisión abstracto y sus parámetros puede volverse impráctica. En este trabajo, tal modelo se aproxima usando algoritmos de aprendizaje automático donde, a partir de un conjunto de datos de muestreo, se aprende una abstracción inicial del espacio de estados, y un modelo de transición sobre esta abstracción. La solución de este MDP abstracto inicial es una política de acción que en general es satisfactoria, sin embargo, para los casos donde ésta resulte insuficiente, se puede aplicar una segunda fase donde la solución es detallada o refinada.

La calidad del método se demuestra empíricamente usando problemas simples de planificación de movimiento en robótica, y un problema de control de procesos industriales con diferentes dimensiones y de los espacios de estados y de acciones. Los resultados muestran buenas soluciones con ahorros en el tamaño del espacio de estados, y reducciones en el tiempo de aprendizaje e inferencia al compararse con discretizaciones uniformes finas.

Índice general

1. Introducción	7
1.1. La planificación automática	7
1.2. El problema de la planificación	8
1.3. Motivación	9
1.4. Contribuciones	10
1.5. Organización del documento	11
2. Planificación con incertidumbre	13
2.1. Planificación clásica	13
2.1.1. Planificadores clásicos	14
2.2. Planificación con incertidumbre	16
2.2.1. No determinismo	16
2.2.2. Observabilidad parcial	18
2.2.3. Metas extendidas	18
2.3. Planificación basada en teoría de decisiones	19
2.3.1. Extensiones a planificación clásica	19
2.3.2. Procesos de decisión de Markov	21
2.3.3. POMDPs	21
2.4. Resumen del capítulo	22
3. Procesos de decisión de Markov	24
3.1. Definición del problema	24
3.2. Formalización	27
3.3. Métodos de solución	27
3.3.1. Programación dinámica	28
3.3.2. Programación lineal	29
3.4. Limitaciones y problemas	30
3.5. Representaciones factorizadas	31
3.5.1. Espacios de estados factorizados	31
3.5.2. Acciones factorizadas	32
3.5.3. Recompensas factorizadas	33
3.6. Abstracción, agregación y descomposición	34
3.6.1. Abstracción y agregación	34
3.6.2. Métodos de descomposición	35

3.7. Procesos de decisión de Markov continuos	38
3.8. Resumen del capítulo	40
4. MDPs cualitativos	41
4.1. Representación	41
4.1.1. Restricciones cualitativas	41
4.1.2. Arbol de decisión de recompensa	42
4.1.3. Estados cualitativos	43
4.2. Formalización	45
4.3. Aprendizaje	47
4.3.1. Recolección de datos (exploración)	48
4.3.2. Abstracción de datos	50
4.3.3. Inducción de un MDP cualitativo	52
4.4. Refinamiento de estados cualitativos	55
4.5. Resumen del capítulo	58
5. Resultados experimentales	59
5.1. Navegación robótica	59
5.1.1. Aprendizaje de MDPs factorizados	61
5.1.2. Aprendizaje de MDPs cualitativos	65
5.2. Generación de energía eléctrica	73
5.2.1. Especificación de un MDP	74
5.2.2. Solución usando MDPs cualitativos	76
5.3. Resumen del capítulo	79
6. Conclusiones y trabajo futuro	80
6.1. Conclusiones	80
6.2. Trabajo futuro	82
A. Problema 1.	90
A.1. Datos de exploración	90
A.1.1. Archivo de atributos continuos	90
A.1.2. Archivo de atributos híbridos	90
A.1.3. Muestra archivo de ejemplos	90
A.2. Función de recompensa	92
A.2.1. Muestra archivo de ejemplos de Weka	92
A.2.2. Resultados de J4.8	93
A.3. Modelos de transición (RBD).	94
A.3.1. Muestra archivo de ejemplos de Elvira para la acción 0: abrir válvula de agua de alimentación (<i>+fwv</i>)	94
A.3.2. Resultados de aprendizaje con K2	97
A.4. Solución del MDP usando SPUDD	103
A.4.1. Archivo de entrada	103
A.4.2. Archivo de salida	106

B. Problema 2.	108
B.1. Función de recompensa	108
B.2. Solución del MDP usando SPUDD	110
B.2.1. Archivo de entrada	110
B.2.2. Archivo de salida	115
C. Algoritmos de refinamiento de estados cualitativos.	117

Índice de figuras

1.1. Ejemplo de planificación en el dominio de las plantas eléctricas. . .	8
2.1. Tres versiones del operador (<i>cerrar interruptorPpal</i>) con pre- condiciones y efectos.	17
2.2. Otros operadores con precondiciones y efectos determinísticos del ejemplo de la planta eléctrica.	17
3.1. Ejemplo de un proceso de decisión de Markov.	25
3.2. Ejemplo de política óptima para el problema del tanque de agua.	26
3.3. Algoritmo de iteración de política.	28
3.4. Algoritmo de iteración de valor.	29
3.5. Representación estructurada de las funciones de transición y rec- ompensa.	33
3.6. Ejemplo del problema de navegación robótica en [22].	37
3.7. Partición del espacio de estados en regiones empleado por Dean y Lin.	38
4.1. Ejemplo del contenedor de gas de Suc y Bratko	42
4.2. Diagrama Temperatura-Volumen para el problema del gas ideal y su sistema de recompensa.	43
4.3. Transformación de un árbol de decisión de recompensa en un árbol-Q.	44
4.4. Arbol Q y su representación en 2 dimensiones.	45
4.5. Modelo de transición cualitativo y función de recompensa cuali- tativa.	46
4.6. Diagramas de influencia representando un MDP cualitativo.	47
4.7. Rastro de exploración para el ejemplo del gas ideal.	50
4.8. Posible política y valores de utilidad esperada para el ejemplo del gas ideal.	54
4.9. Proceso de refinamiento cualitativo para el espacio de estados del problema del gas ideal.	57
5.1. Problemas de navegación robótica libre de obstáculos con regiones recompensadas.	60

5.2.	Caso de prueba con rastro de exploración para problema con 16 estados discretos, 2 zonas con recompensa, y acciones rotacionales (izquierda), y su solución (derecha).	61
5.3.	Árbol de decisión aprendido con J4.8 de Weka para el ejemplo del robot rotatorio.	62
5.4.	Red Bayesiana dinámica aprendida por K2 de Elvira para el ejemplo del robot rotatorio.	63
5.5.	Funciones de política y de valor para el problema de la sección 5.1.	64
5.6.	Funciones de política y de valor para un problema tipo laberinto.	64
5.7.	Problema de planificación de movimiento y su proceso de solución.	66
5.8.	Gráficas de exploración mostrando la precisión de la política y el error de utilidad vs número de ejemplos para el ejemplo 3 de la tabla 5.2	70
5.9.	Refinamiento de la solución de un laberinto típico.	71
5.10.	Aproximación incremental de la función de valor de un laberinto típico por refinamiento	71
5.11.	Refinamiento de la solución de un problema con la meta oculta .	72
5.12.	Aproximación incremental de la función de valor para el problema de la meta oculta.	72
5.13.	Proceso de generación de vapor en una central de ciclo combinado. Por simplicidad, la conexión con la turbina de gas ha sido omitida.	73
5.14.	Efectos de las acciones sobre el sistema de generación de vapor. .	75
5.15.	Función de recompensa para el problema del generador de vapor	76
5.16.	Estructura del árbol de decisión de 2D (Fms y Pd) obtenido para el problema de generación de vapor.	77
5.17.	Función de transición de estados aprendida por $K2$ para la acción <i>abrir válvula de alimentación de agua, (+fwv)</i> para el problema de generación de vapor	78
5.18.	Política generada por el sistema SPUDD y partición de estados para el problema de generación de vapor.	78
A.1.	Red Bayesiana Dinámica para la acción <i>abrir válvula de agua de alimentación (+fwv)</i> del problema de generación de vapor.	98
A.2.	Red Bayesiana Dinámica para la acción <i>cerrar válvula de agua de alimentación (-fwv)</i> del problema de generación de vapor.	99
A.3.	Red Bayesiana Dinámica para la acción <i>abrir válvula ed vapor principal (+msv)</i> del problema de generación de vapor.	100
A.4.	Red Bayesiana Dinámica para la acción <i>cerrar válvula de vapor principal (-msv)</i> del problema de generación de vapor.	101
A.5.	Red Bayesiana Dinámica para la <i>acción nula</i> del problema de generación de vapor.	102
C.1.	Algoritmo de refinamiento.	117
C.2.	Algoritmo de selección de estado a biseccionar.	118
C.3.	Algoritmo de bisección de estados.	118

Índice de tablas

4.1. Datos de exploración para el problema del gas ideal.	48
4.2. Datos resultantes del proceso de abstracción para el ejemplo del gas ideal.	51
4.3. Arreglo de datos cualitativos en secuencia temporal para el ejemplo del gas ideal.	53
4.4. Subconjuntos de datos abstractos por acción para el problema del gas ideal	54
5.1. Resultados experimentales de aprendizaje de MDPs factorizados.	65
5.2. Casos de prueba comparando nuestra abstracción respecto a una discretización fina “convencional”.	68
5.3. Resultados comparativos entre la abstracción inicial y la abstracción refinada.	68
5.4. Resultados experimentales con SPUDD.	69
5.5. Conjuntos de acciones para el problema de control del <i>HRSG</i>	74

Capítulo 1

Introducción

1.1. La planificación automática

La planificación es la forma racional de actuar. Es un proceso abstracto, de deliberación explícita que escoge y organiza las acciones de un sistema anticipando sus posibles efectos en el mundo. Esta deliberación tiene por objeto alcanzar en lo posible algunos objetivos preestablecidos.

Una conducta con determinación requiere de la planificación cuando conduce a nuevas situaciones, tareas u objetivos complejos, o cuando se usan acciones nuevas o poco familiares. La deliberación también es útil cuando la organización de las acciones esta sujeta a restricciones establecidas, por ejemplo, por un ambiente crítico que involucre alto riesgo o alto costo, actividad conjunta con alguien más, o por una actividad que debe de sincronizarse con un sistema dinámico. Dado que la planificación es un proceso muy complicado y costoso con altos consumos de tiempo, recurrimos a ella solo cuando es estrictamente necesario o cuando la relación costo-beneficio nos obliga a planear. Además, generalmente buscamos planes factibles suficientemente buenos y no necesariamente planes óptimos [69].

Algunos profesionales se enfrentan con tareas complejas y cambiantes que demandan altos requerimientos de seguridad y/o eficiencia. Imagínese por ejemplo, una operación de rescate despues de un accidente en una planta nuclear. Tal operación puede involucrar a muchos actores y requerir el despliegue de mucha infraestructura de comunicación y transporte. Se requiere de una planificación cuidadosa y el estudio de diversos planes alternativos. También está restringido por el tiempo y demanda decisiones inmediatas que deben apoyarse con alguna herramienta computacional para resolver el problema.

La *planificación automática* es el área de la Inteligencia Artificial (IA) que estudia computacionalmente el proceso de deliberación para actuar. Desde el punto de vista práctico, su objetivo es construir herramientas de procesamiento de información que ofrezcan acceso a recursos de planificación eficientes para resolver problemas complejos para un humano. Actualmente, un problema com-

plejo es del orden de 10,000 estados, y que se puede resolver usando una herramienta sofisticada de planificación en aproximadamente dos días. Desde el punto de vista teórico, la planificación es un componente importante de la conducta racional. Si uno de los propósitos de la IA es comprender los aspectos computacionales de la inteligencia, entonces la planificación, como el lado racional de actuar, es un elemento clave para tal propósito.

El reto aquí es estudiar a la planificación no sólo como un proceso abstracto independiente sino como un componente totalmente integrado de la conducta deliberativa que permita construir sistemas inteligentes robustos.

1.2. El problema de la planificación

Un problema de planificación, desde el punto de vista de la inteligencia artificial, se especifica normalmente como sigue: dada una descripción del estado actual de un sistema, un conjunto de acciones que pueden realizarse sobre el sistema y una descripción de un conjunto de estados meta para el sistema, encontrar una secuencia de acciones para transformar el estado actual en uno de los estados meta. Considere, por ejemplo, el dominio de las plantas eléctricas donde un sistema de generación consta de un conjunto de equipos de proceso como turbinas de gas (*TG*), recuperador de calor (*HRS*), quemadores posteriores (*QP*), y turbina de vapor (*TV*), cada uno de los cuales puede mantener un estado de operación coherente con el resto de los demás. Las acciones posibles permiten arrancar un equipo para ponerlo en operación, y la meta es una configuración particular de cada componente, como por ejemplo todos los equipos encendidos. El problema consiste en determinar cual es la secuencia de acciones necesaria, incluyendo aquellas de naturaleza paralela, para llevar una planta apagada a su estado de operación estable (meta). En la figura 1.1 se representa esta situación mediante un conjunto de listas describiendo el estado actual de los equipos de proceso, las operaciones posibles para transformar este estado (por ejemplo, arrancar un equipo *x*), y un conjunto de listas describiendo el estado deseado de la planta (meta).

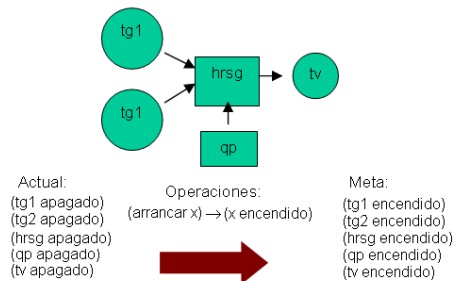


Figura 1.1: Ejemplo de planificación en el dominio de las plantas eléctricas.

Para resolver este tipo de problemas se usan los algoritmos de planificación.

El tipo de problemas que estos algoritmos abordan define sus alcances y las suposiciones que pueden hacerse sobre el mundo. Por ejemplo, existen casos muy sencillos que asumen mundos estáticos y espacios de estados muy pequeños donde la simple selección de un plan típico puede dar resultados satisfactorios [64]. En otro tipo de sistemas, ligeramente más complejos, las acciones sólo se aplican bajo ciertas condiciones y producen efectos sobre el mundo. Las metas en estos sistemas son descripciones lógicas de un estado, de modo que la medida de éxito de un plan se vuelve binaria (verdadero o falso). Los algoritmos de planificación clásica han intentado resolver esta problemática mediante la composición de planes como un proceso de búsqueda, satisfacción de restricciones, representaciones gráficas, etc.

1.3. Motivación

La motivación en este trabajo radica en dos aspectos de la planificación automática, la contribución científica y su aplicación en la práctica. La motivación científica radica en extender los resultados de científicos en planificación automática para mejorar la eficiencia de los sistemas actuales. La motivación práctica consiste en aprovechar los avances científicos más recientes para extender su radio de aplicación a problemas de planificación del mundo real. Particularmente, problemas de gran escala, de naturaleza dinámica, y en donde la incertidumbre juega un papel importante a considerar durante la búsqueda de soluciones.

Uno de los retos más importantes de la comunidad científica es la tratabilidad de problemas de planificación con incertidumbre en dominios altamente dimensionales y de naturaleza continua. Científicos como Craig Boutilier y Tomas Dean, han logrado avances importantes en representar en forma compacta un problema de planificación mediante el uso de modelos basados en Inteligencia Artificial. Otros, como Daphne Koller, Ronald Parr y Jesse Hoey han logrado mejorar la solución de representaciones compactas de problemas del mundo real mediante técnicas de abstracción y agregación. Por su parte, Steward Russell y J. Tash han logrado resolver problemas de planificación muy complejos a través del uso de la descomposición de problemas bajo la estrategia “divide y vencerás”. En conjunto, todos han contribuido desde distintas perspectivas a resolver el problema de tratabilidad de la planificación con incertidumbre. Debido a la dificultad de integrar las ventajas de sus descubrimientos en un solo sistema, y a la necesidad de resolver un problema práctico con características muy particulares, creemos que a través de un estudio global del estado del arte y mediante la aportación de nuevas ideas es posible lograr en este trabajo nuevos rumbos que contribuyan con esta labor científica.

Uno de los problemas prácticos, motivo del presente trabajo, es diseñar una herramienta computacional para asistir operadores de plantas eléctricas en la toma de decisiones. En este dominio existen muchas variables que cambian dinámicamente por la operación de dispositivos como válvulas, bombas o actuadores, e incluso por la ocurrencia de algún evento exógeno no controlado.

Uno de los aspectos que ignoran los sistemas de control tradicionales de una planta de procesos es el hecho de que estos dispositivos pueden fallar y por tanto la acción de control puede no tener los efectos esperados. Adicionalmente, un sistema de control tradicional nunca toma en cuenta la utilidad de una secuencia de acciones en el futuro. No es lo mismo operar un equipo en sus límites cuando es nuevo que cuando es obsoleto. Si a todo esto agregamos la dificultad de adquirir, codificar y enriquecer correctamente el conocimiento experto sobre la operación de un proceso, el problema es aun mayormente complicado. Imagínese capturar la experiencia de operación de cuarenta años de vida útil de una planta de generación de electricidad con todos los deterioros ocasionados por el paso del tiempo. Aun en el caso de que existiera disponible tal conocimiento experto, cuál sería la mejor manera de codificarlo para ser usado y actualizado por un sistema computacional. Consideramos que el uso de la planificación automática con extensiones para abordar este dominio permitirá complementar los sistemas de control actuales con intervenciones humanas basadas en planes generados por un sistema computacional.

1.4. Contribuciones

Los Procesos de Decisión de Markov (*MDPs*) ofrecen plataformas de solución a las deficiencias de los sistemas de control tradicionales. Sin embargo, éstas fueron originalmente orientadas a sistemas discretos hipotéticos muy simples cuyo propósito era demostrar académicamente su potencial. En un proceso físico real muchas variables son continuas y su discretización puede traducirse en una pérdida importante de precisión y gran ganancia de complejidad al representarse computacionalmente. La técnica propuesta en este trabajo ofrece una solución al problema de control y otros problemas del mundo real a través del diseño de una representación compacta de un proceso de decisión de Markov para abordar tanto dominios discretos como continuos. Adicionalmente, atiende el problema de la adquisición y codificación del conocimiento experto mediante el uso de técnicas de muestreo y algoritmos de aprendizaje automático para simplificar este proceso. Con estas dos mejoras se busca robustecer el formalismo de los MDPs para adquirir conocimiento experto en forma automática, y resolver problemas complejos en tiempos razonablemente cortos.

Las características técnicas comentadas sobre este trabajo representan las aportaciones a la comunidad de planificación automática. Estas contribuciones se detallan a continuación:

1. El sistema planificador puede manejar espacios continuos. Se hace uso de una técnica de agregación de estados basada en *restricciones cualitativas* [70] para agrupar estados con el mismo valor de recompensa. De acuerdo con el sistema de recompensa, el espacio de estados se divide en un conjunto de estados *abstractos* a los que denominamos *estados cualitativos*. Otras técnicas [55, 12] inician con una discretización uniforme exhaustiva de todas las variables del dominio que iterativamente se va abstractando.

Nuestra representación es particularmente útil para estructurar dominios con variables continuas (y discretas), y permite aproximar una solución inicial que posteriormente puede refinarse.

2. El conocimiento experto es adquirido artificialmente mediante técnicas de aprendizaje automático. Mediante la exploración del ambiente, se colecta información sobre las recompensas y la dinámica del sistema. Esta información se usa para construir un árbol de decisión representando los estados cualitativos, y luego para aprender la función de transición de estados con un algoritmo de aprendizaje bayesiano. El resultado del sistema de aprendizaje es un modelo aproximado híbrido continuo-discreto listo para resolverse con técnicas de programación dinámica estándar. Una ventaja importante del uso de algoritmos de aprendizaje es que todo el proceso, desde la construcción del modelo abstracto hasta su solución, es completamente automático.
3. Como una consecuencia del punto anterior, y dado que un MDP cualitativo utiliza una representación basada en características (factores), es posible el aprendizaje de MDPs factorizados discretos. Hasta el inicio de esta investigación no existían trabajos que lograran esto. Fue muy recientemente cuando Degris [24] aprende un MDP factorizado basado en algoritmos de planificación incremental en el contexto de aprendizaje por refuerzo [71].
4. El modelo aproximado es refinado para mejorar la solución final. La abstracción obtenida en la etapa de aprendizaje se refina bisectando iterativamente una dimensión de los estados donde mejor se contribuya a la solución global. La selección de estados candidatos a división se basa en un criterio de variabilidad de la función de valor respecto a su vecindad inmediata. Por su parte, la dimensión del estado candidato se selecciona mediante un criterio de gradiente de utilidad. Aun cuando en el pasado otros trabajos hacían uso del refinamiento de una solución [35], no existe reportado en la literatura que el criterio de refinamiento sea selectivo. Este método permite refinar una abstracción donde mejor se contribuye a la solución final.

Consideramos que en conjunto, estas aportaciones permitirán resolver problemas prácticos de planificación con incertidumbre con espacios de estados continuos y discretos, y de alta dimensionalidad, sin necesidad de la codificación explícita de un modelo de decisión por parte de un experto.

1.5. Organización del documento

Este documento se organiza de la siguiente manera: En el capítulo 2 se describen en forma general las técnicas de planificación con incertidumbre que tratan de dar solución a los problemas del mundo real. Entre éstas se explican inicialmente las técnicas que bajo suposiciones muy restrictivas sirvieron de

inspiración para sistemas más sofisticados. Posteriormente se describen los retos y suposiciones de la planificación con incertidumbre mostrando algunos trabajos que abordan estos retos bajo enfoques a veces diferentes.

El capítulo 3 profundiza en uno de los paradigmas más importante de la planificación basada en teoría de decisiones, los procesos de decisión de Markov. Aquí se hace un breve análisis del estado del arte enfatizando los aspectos más relevantes que motivaron la realización del presente trabajo. Particularmente, se comentan los avances más recientes en representaciones factorizadas, abstracción y agregación, métodos de descomposición y trabajos sobre refinamiento de abstracciones.

En el capítulo 4 se exponen los detalles de la técnica propuesta en este trabajo para resolver problemas de planificación con variables híbridas continuas-discretas, con descubrimiento automático de conocimiento y refinamiento iterativo. Se inicia explicando los detalles de representación de la abstracción llamada *estados cualitativos*. Igualmente se expone un metodología basada en técnicas estándar de aprendizaje automático para aproximar un MDP abstracto, y se ilustra un algoritmo para el mejoramiento iterativo del modelo de conocimiento basado en varianza local de utilidad sobre los estados abstractos.

En el capítulo 5 se describen los resultados experimentales obtenidos en los dominios de control de procesos y de planificación de movimiento. Aquí se desarrollan los métodos de evaluación y se hace un análisis comparativo para demostrar su calidad.

Finalmente, en el capítulo 6 se hace un breve resumen de este trabajo y sus resultados, comentado el trabajo en proceso y direcciones futuras hacia nuevas aplicaciones.

Capítulo 2

Planificación con incertidumbre

En este capítulo se exponen los fundamentos de la planificación con incertidumbre, algunas de sus variantes, y los trabajos más representativos en esta área. Inicialmente, se define a la planificación clásica en términos de las suposiciones restrictivas del mundo que la han caracterizado, y se ilustran aquellos planificadores clásicos que sentaron las bases de la planificación con incertidumbre. Posteriormente, se define la planificación con incertidumbre en función de las restricciones que elimina de su contraparte clásica, y se describen aquellos trabajos que bajo diferentes enfoques abordan sus principales retos.

2.1. Planificación clásica

STRIPS (*The Stanford Research Institute Problem Solver*)[31] es considerado el primer sistema planificador clásico. Fue diseñado como elemento de planificación en 1969 para el robot Shakey del Instituto de Investigaciones de Stanford (SRI por sus siglas en inglés). En STRIPS, un problema de planificación se representa mediante fórmulas bien formadas (*wff*) de la lógica de primer orden. STRIPS pertenece a la familia de planificadores que buscan sobre un espacio de modelos del mundo para hallar uno donde se alcance una meta dada. Para cualquier modelo del mundo, se asume que existe un conjunto de operadores aplicables capaces de transformar el modelo actual en otro. La tarea particular de STRIPS es hallar alguna composición de operadores que transforme el modelo inicial en uno que satisfaga la condición meta establecida. Su estructura de control para extraer diferencias entre el estado actual y el estado meta, y para identificar operadores relevantes se basaba en el sistema GPS (General Problem Solver) de Newell y Simon [57].

La planificación clásica, también referida en la literatura como planificación tipo STRIPS, se refiere a la planificación para sistemas de transición de estados restringidos que reúnen las siguientes suposiciones restrictivas:

1. Sistema finito. El sistema de transición de estados tiene un conjunto finito de estados.
2. Completamente observable. El ambiente es completamente observable, i.e., se tiene un conocimiento completo del estado del sistema de transición de estados.
3. Determinista. El sistema es determinista cuando la aplicación de una acción o la presencia de un evento externo tenga un efecto totalmente predecible en el mundo.
4. Estático. El sistema carece de una dinámica interna; permanece en el mismo estado hasta que el controlador aplique alguna acción.
5. Metas restringidas. El planificador maneja sólo metas restringidas que se especifican en forma de un estado meta explícito s_g o un conjunto de estados meta S_g . El objetivo es obtener cualquier secuencia de transiciones de estado que termine en uno de los estados meta. Las metas extendidas tales como los estados por evitar y las restricciones sobre las trayectorias de estados, o funciones de utilidad no pueden manejarse bajo esta suposición.
6. Planes secuenciales. Un plan solución para un problema de planificación es una secuencia de acciones finitas ordenadas linealmente.
7. Tiempo implícito. Las acciones y eventos no tienen duración; son transiciones de estados instantáneas. Esta suposición está implícita en los sistemas de transición de estados, un modelo que no representa el tiempo explícitamente.
8. Planificación fuera de línea. El planificador no considera ningún cambio que pueda ocurrir mientras está elaborando un plan; planea para el estado inicial y estados meta sin importarle, en caso de existir, la dinámica actual.

En resumen, la planificación clásica se refiere a los casos que combinan todas estas suposiciones restrictivas para el sistema de transición de estados: conocimiento completo del mundo, determinismo, procesos estáticos, sistemas finitos con metas restringidas y tiempo implícito.

2.1.1. Planificadores clásicos

A raíz de que las suposiciones restringidas mencionadas empezaron a limitar la aplicabilidad de los planificadores puramente basados en STRIPS, se inició la construcción de sistemas que introducían satisfacción de restricciones, orden parcial o manejo del tiempo. Varios de estos planificadores clásicos han servido de inspiración para la construcción de los sistemas de planificación con incertidumbre que se explicarán en la sección 2.3.1. Entre estos sistemas se encuentran: SNLP, GRAPHPLAN, SATPLAN y PRODIGY.

SNLP (*Systematic Non-Linear Planner*) [52] es un planificador no lineal que representa un plan como un conjunto de acciones parciales con restricciones

de orden entre pasos. El plan contiene variables instanciables y una estructura de datos explícita llamada *ligas causales*, las cuales representan los supuestos efectos. En este sistema, una liga causal registra una submeta, el paso que la contiene como precondition (*consumidor*), y el paso que la contiene como efecto (*proveedor*). Al igual que SNLP, UCPOP [59] realiza un tipo de planificación no lineal usando la técnica conocida como planificación de mínimo compromiso. En ésta, un plan se construye bajo el principio de que un planificador (o cualquier algoritmo de búsqueda) debe evitar tomar una decisión, como la inserción de algún paso, hasta que no tenga un buen motivo para ello.

GRAPHPLAN [6] construye una estructura de datos llamada grafo de planificación sobre la cual posteriormente realiza la búsqueda sistemática de un plan válido. Los elementos de un problema para GRAPHPLAN son: un conjunto de operadores, un conjunto de objetos, un conjunto de proposiciones llamadas condiciones iniciales, y un conjunto de metas que deben hacerse verdaderas al final de la ejecución del plan. Bajo esta formalización, una acción es un operador completamente instanciado como, por ejemplo, (*abrir ?V ?O*) que puede instanciarse como (*abrir valvula52 20%*). Un grafo de planificación es un grafo dirigido etiquetado en el cual hay dos tipos de nodos y tres tipos de arcos. El tipo de nodos corresponde a acciones y proposiciones, mientras que los arcos pueden ser operadores sin efecto (*no-op*), efectos agregados (*add-effects*) y efectos eliminados (*delete-effects*). Esta representación hace que los grafos de planificación no correspondan directamente a un espacio basado en estados. Un plan válido en GRAPHPLAN es un conjunto de acciones especificando en qué tiempo se deben ejecutar. Debe haber una o varias acciones para cada instante de tiempo, que pueden ejecutarse en forma simultánea siempre y cuando no interfieran entre sí. Para garantizar el plan más corto, normalmente se utiliza una estrategia de búsqueda con encadenamiento hacia atrás.

SATPLAN [42] es un sistema que, a diferencia de sus antecesores que hacen deducción lógica, trata un problema de planificación como un problema de satisfacción de restricciones. En este enfoque, un plan corresponde a cualquier modelo que satisface un conjunto de restricciones lógicas representando el estado inicial, el estado meta, y los axiomas del dominio. En SATPLAN, el tiempo consta de un número de instancias discretas fijas, y una situación se representa con una proposición variable en el tiempo (*fluente*). Las restricciones generales sobre hechos y acciones se representan como esquemas axiomáticos, los cuales serán instanciados por los objetos y el tiempo para un problema en particular. La máxima longitud de un plan se fija como un punto en el tiempo que si se desconoce debe buscarse entre un conjunto de instancias de varios tamaños hasta encontrar la solución de menor longitud. Los problemas de satisfacibilidad planteados por SATPLAN pueden resolverse directamente usando algoritmos estándar de SAT como el método de Davis-Putnam [50]. Una ventaja de SATPLAN es que puede expresar restricciones arbitrarias sobre estados intermedios y la estructura del plan mismo. Otra ventaja es que todos los modelos del conjunto de axiomas corresponden a planes válidos.

PRODIGY [49] es un sistema integral que no sólo incluye algoritmos de planificación sino procedimientos de aprendizaje basado en explicaciones (*EBL*)

para incrementar considerablemente su eficiencia. PRODIGY es capaz de aprender reglas de control, conducir experimentos para adquirir nuevo conocimiento, generar jerarquías de abstracción, y usar razonamiento analógico para reconocer y explotar las similitudes entre distintos problemas de planificación. PRODIGY consta de un simulador de la ejecución del plan y un algoritmo de búsqueda por encadenamiento hacia atrás. El algoritmo de búsqueda es el responsable del razonamiento orientado a metas mientras que el simulador realiza, en tiempo de ejecución, una búsqueda guiada por los datos con elementos de encadenamiento hacia adelante.

2.2. Planificación con incertidumbre

La mayoría de los problemas del mundo real son problemas de mayor complejidad que los problemas que abordan los planificadores clásicos. En éstos se debe tomar en cuenta que el mundo cambia dinámicamente, que las acciones pueden tener diferentes efectos, o que el conocimiento de un estado puede ser incierto. Además, también deben balancear el potencial de algún plan para alcanzar una meta contra el riesgo de producir estados indeseables o planes demasiado costosos de elaborar.

La planificación con incertidumbre elimina tres suposiciones restrictivas de la planificación clásica como el determinismo, la observabilidad total, y las metas restringidas a éxito o fracaso. Con esto se introducen los conceptos de no determinismo, observabilidad parcial, y metas extendidas.

2.2.1. No determinismo

Para generar planes con conductas condicionales, los algoritmos de planificación requieren formas eficientes para analizar y razonar sobre los efectos de una acción. La suposición de que los efectos de una acción son deterministas es irreal e impráctica ya que como bien es sabido nada es totalmente predecible. El no determinismo toma una postura más realista y útil modelando el hecho de que después de la aplicación de una acción o la aparición de un evento externo algunos efectos son más factibles de suceder que otros. El no determinismo puede modelarse asociando probabilidades a los resultados de una acción o evento.

Considere como ejemplo el problema de generar energía eléctrica mediante una turbina de vapor de una central de ciclo combinado. La meta es llevar el generador de la turbina de vapor al estado de generación a plena carga (mayor o igual al 90% de su capacidad total). Esta meta puede representarse mediante los hechos tipo lista (*estado tv generando*) y (*< carga 90*), donde el primer término es el nombre del predicado o función y los restantes sus argumentos. La figura 2.1 muestra tres descripciones con diferente detalle de la acción (*cerrar interruptorPpal*) para alcanzar la meta de generar a plena carga. La primera es determinista y podría usarse con alguno de los operadores extras de la figura 2.2 (izquierda) para formar el plan de dos pasos que alcanza la meta con toda certeza: (*rodar tv*), (*cerrar interruptorPpal*). El segundo

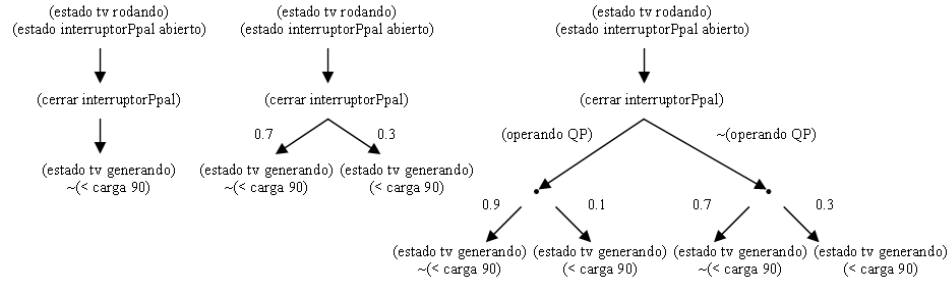


Figura 2.1: Tres versiones del operador (*cerrar interruptorPpal*) con precondiciones y efectos. Izquierda: Determinístico, Centro: Probabilista Simple, Izquierda: Probabilístico Condicional.

operador (*cerrar interruptorPpal*) de la figura 2.1 (centro) tiene dos conjuntos de efectos alternos, modelando dos estados diferentes del mundo que pudieran darse al aplicar el operador. Los números sobre los arcos son las probabilidades de llegar a cada conjunto de estados. Bajo este modelo, el plan de dos pasos del caso anterior solo alcanzaría la meta con una probabilidad de 0.7. En la tercera formulación del operador (*cerrar interruptorPpal*) de la figura 2.1 (derecha), son posibles diferentes conjuntos de resultados, los cuales están basados en otras condiciones existentes al aplicar el operador, y los arcos ahora muestran probabilidades condicionales. El plan de dos pasos aun tiene probabilidad de éxito de 0.7. Sin embargo, el plan de tres pasos: (*encender QP*), (*rodar tv*), (*cerrar interruptorPpal*) tiene 0.9 de probabilidad de éxito. Encender QP significa encender los quemadores posteriores del recuperador de calor con la idea de soportar variaciones en el flujo de gases calientes saliendo de la o las turbinas de gas que permiten la generación de vapor.

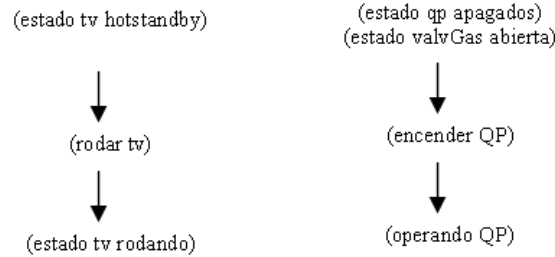


Figura 2.2: Otros operadores con precondiciones y efectos determinísticos del ejemplo de la planta eléctrica.

2.2.2. Observabilidad parcial

En varias aplicaciones y durante tiempo de ejecución, el estado de un sistema puede ser parcialmente accesible, y por tanto dar lugar a estados indistinguibles. En algunos casos algunas variables nunca son observables, en otros las variables son observables en ciertos estados, y en otros más las variables sólo son observables después de una rutina de sensado. Tanto teórica como experimentalmente, se ha demostrado que la planificación con observabilidad parcial es un problema muy difícil de resolver. Dado que las observaciones devuelven conjuntos de estados, la consecuencia principal de la observabilidad parcial es que una observación puede ser exponencialmente mayor que el conjunto de estados del dominio. Peor aún, en el caso de la planificación con incertidumbre, las observaciones devuelven distribuciones de probabilidades sobre conjuntos de estados que pueden hacer infinito el espacio de búsqueda. Por ejemplo, si las precondiciones de los operadores del ejemplo de las figuras 2.1 y 2.2 fueran no deterministas, y existiera una distribución de probabilidades sobre los estados posibles del sistema, el espacio de búsqueda para encontrar un plan con alta probabilidad de éxito sería muy grande.

2.2.3. Metas extendidas

En dominios no determinísticos, las metas necesitan especificar implícitamente consideraciones para casos de posibles fallas, de tal suerte que cuando un sistema intente alcanzar una meta pueda garantizar que, si no lo logra, al menos se mantendrá en un estado seguro. Retomando el ejemplo anterior, suponga que se requiere que una planta industrial alcance un nivel de operación dado, pero también que garantice evitar estados peligrosos durante su camino al estado meta. Las metas de esta clase se conocen como metas extendidas y pueden representarse de varias formas.

Una forma de lograr que las metas especifiquen las preferencias de un agente es mediante el uso de funciones de utilidad (costos y recompensas). En este caso, la planificación consiste en buscar un plan que maximice una función de utilidad. Por ejemplo, suponga que el estado en el cual el generador alcanza el 100% de generación tiene una utilidad de 10, y uno en el cual se alcanzó el 95% de generación tiene una utilidad de 6, cualquier estado en el cual la generación sea menor al 90% tiene una utilidad de -100. Si el operador (*encender QP*) logra encender los quemadores posteriores con la mitad de su eficiencia, entonces el plan de dos pasos (*rodar tv*), (*cerrar interruptor Ppal*) tiene una utilidad esperada de -23, y el plan de 3 pasos (*encender QP*), (*rodar tv*), (*cerrar interruptor Ppal*) tiene una utilidad de -4.6. En este caso, el plan con encendido de quemadores aun sin ser muy eficiente, es mejor que el plan de dos pasos. Una técnica alternativa es representar las metas extendidas mediante fórmulas de lógica temporal. En este caso, la planificación consiste en generar un plan cuyas conductas del dominio satisfagan la fórmula temporal. En ambos casos, la planificación con incertidumbre con metas extendidas es un problema difícil ya que las metas extendidas agregan complejidad adicional a un problema en sí complicado.

2.3. Planificación basada en teoría de decisiones

Los algoritmos de planificación permiten la construcción de rumbos de acción en ambientes reales, que al menos consideran que las acciones pueden tener distintos efectos en el mundo (no determinismo), y que un plan tenga el potencial para alcanzar un estado meta (metas extendidas).

El marco de la teoría de decisiones [48], por su parte, permite evaluar las fortalezas y debilidades de planes alternativos con base en teoría de probabilidad y teoría de utilidad. Dada una distribución de probabilidades sobre los posibles efectos de una acción en un estado, y una función razonable de preferencia sobre los resultados de una decisión, es posible definir una función de utilidad tal que, cada vez que un agente prefiera un plan sobre otro, el plan preferido tendrá la mayor utilidad esperada. De este modo, la tarea de un planificador es encontrar un plan con la máxima utilidad esperada.

Debido a lo anterior, la planificación automática y la teoría de decisiones se aprecian como técnicas complementarias que desde hace mucho tiempo ha existido el interés por combinarlas [29]. Esta combinación ha permitido abordar problemas con cientos o miles de estados que consideran a las metas como una función de utilidad, y a la incertidumbre como parte del ambiente. El reciente incremento en investigación en *planificación basada en teoría de decisiones* (como se le ha llamado a esta combinación) se debe tanto a los nuevos avances en representaciones e inferencia bayesiana, como al éxito de las técnicas Markovianas en áreas como reconocimiento de voz, y sus muy cercanas técnicas de aprendizaje por refuerzo [71]. Las computadoras actuales, con mayor poder de cómputo, han permitido contruir planificadores basados en teoría de decisiones que en general tienen espacios de búsqueda más complicados que sus contrapartes clásicas.

En esta sección se hace una breve reseña del trabajo más importante en planificación basada en teoría de decisiones enfatizando los avances y problemas técnicos más importantes. Por claridad, la exposición se ha dividido en dos partes: los planificadores clásicos extendidos, y las técnicas Markovianas (MDPS y POMDPS).

2.3.1. Extensiones a planificación clásica

Existe una creciente cantidad de trabajos que extienden a los planificadores clásicos para el manejo de incertidumbre, sin embargo en esta sección solo se describirán los más representativos. Entre éstos se encuentran los sistemas basado en SNLP como BURIDAN, CNLP y C-BURIDAN, los sistemas de planes abstractos con refinamiento: DRIPS y PYRRHUS, el sistema para manejo de eventos exógenos basado en Prodigy WEAVER, y el planificador por traducción a satisfacibilidad, MAXPLAN.

BURIDAN [46] es una versión modificada del algoritmo de planificación SNLP el cual puede crear planes en dominios no observables que cumplen con un umbral de probabilidad de éxito ante acciones no deterministas. BURIDAN difiere de SNLP en que puede tener más de una liga causal por condición. Bajo diferentes escenarios de ejecución, diferentes acciones pueden hacer verdadera la

condición, de manera que las ligas se combinen para incrementar el soporte a la condición. En estos casos, el criterio de terminación de SNLP donde todas las condiciones tienen exactamente un solo enlace ya no es aplicable. En su lugar, BURIDAN computa explícitamente la probabilidad de éxito del plan y termina si se supera el umbral dado.

CNLP [60] fue el primer planificador basado en SNLP en representar planes condicionales para dominios parcialmente observables. Cada paso en el plan tiene un conjunto de etiquetas de contexto asociadas que denotan las condiciones bajo las cuales se puede ejecutar el paso. C-BURIDAN [26] refina el planteamiento de CNLP manteniendo distintas observaciones de los efectos y permitiendo resultados diferentes de una acción de sentido para tener el mismo nivel de observación, modelando así la observabilidad parcial. En C-BURIDAN se añaden nuevos contextos en respuesta a las amenazas entre las acciones que pueden ponerse en ramas separadas de un plan condicional.

DRIPS [35] y PYRRHUS [75] son sistemas de planificación con refinamiento de abstracciones que consideran la utilidad de un plan en forma multivaluada. Estos sistemas asumen que, dada una distribución de probabilidades sobre los posibles resultados de una acción dado un estado, y una función de preferencia razonable, es posible definir una función de utilidad tal que, cuando un agente prefiera un plan es por que éste tiene la mayor utilidad esperada. En DRIPS las acciones se abstraen colapsando sus posibles efectos (representados por redes) y manteniendo intervalos probabilísticos sobre efectos disjuntos. Las acciones se combinan en una jerarquía de descomposición muy parecida a las redes de tareas jerárquicas. Debido a que en estas redes la búsqueda se hace sobre horizontes finitos, es posible codificar conocimiento heurístico *a priori* sobre la estructura de buenos planes.

WEAVER [7] es un planificador probabilista basado en PRODIGY que, al igual que C-BURIDAN, no tiene una representación de las utilidades. WEAVER tampoco tiene un modelo de observabilidad explícito por lo que asume que el mundo es completamente observable en tiempo de ejecución del plan. Sin embargo, sí tiene una representación explícita de los eventos exógenos inciertos. En otras palabras, considera eventos que tienen lugar fuera del alcance del agente planificador y que pueden modelarse como si ocurrieran con alguna probabilidad condicional dado el estado del mundo.

MAXPLAN [51] es un planificador inspirado en el algoritmo de planificación clásica SATPLAN, el cual aborda dominios probabilistas complejos en cuanto al número de estados posibles y a la observabilidad de cada uno de ellos. Al igual que SATPLAN, MAXPLAN compila un problema de planificación como un problema de satisfacibilidad lógica, el cual se resuelve con algoritmos estándar de SAT. MAXPLAN asume dominios totalmente no observables, que son especializaciones de un POMDPs (ver sección 2.3.3) vistos como modelos ocultos de Markov.

2.3.2. Procesos de decisión de Markov

Las técnicas basadas en los Procesos de Decisión de Markov (MDP) [2, 40, 3, 62] modelan un *problema de decisión secuencial*, en el cual un sistema evoluciona en el tiempo y es controlado por un agente. La dinámica del sistema es gobernada por una función de transición probabilista que asocia estados y acciones con nuevos estados, y que satisfacen la propiedad de Markov (poder de predicción del efecto de una acción sobre un estado). En otras palabras, existe la posibilidad de que una acción realizada en un estado produzca una distribución de probabilidades sobre los efectos posibles. En cada paso, el agente recibe una recompensa numérica que en general depende del estado actual y de la acción aplicada. Así, el problema principal es encontrar una estrategia reactiva de control o *política de acción* que maximice la recompensa esperada en el tiempo. Un MDP normal asume que el agente siempre conocerá el estado en que se encuentra al momento de iniciar la ejecución de sus acciones (observabilidad total).

Los algoritmos estándar para MDPs, en esencia, resuelven el mismo problema de planificación con incertidumbre que las técnicas clásicas extendidas. Sin embargo, lo hacen de formas distintas. Por ejemplo, mientras un MDP encuentra políticas, los planificadores clásicos encuentran secuencias de acciones. La mayoría de los planificadores clásicos describen el estado de un sistema como un conjunto de cláusulas lógicas. Los MDPs en general lo hacen a través de instancias de las variables relevantes del dominio. Los métodos de solución de un MDP enfatizan el uso de la programación dinámica, mientras que los métodos de solución de la planificación clásica realizan búsqueda sobre espacios de estados y de acciones estructuradas.

Otra de las diferencias entre la planificación clásica y los MDPs es la especificación de las metas. Los MDPs asocian una recompensa a una acción en un estado, y los planificadores clásicos toman una descripción lógica de un conjunto de estados como meta. Algunos planificadores clásicos extendidos agregan utilidades a las metas, sin embargo, los algoritmos para la resolución de MDPs usan la estructura del dominio en forma limitada, y sus funciones objetivo se vuelven más del tipo meta.

Al igual que los MDPs, algunos planificadores de la sección 2.3.1 buscan una política con la máxima utilidad esperada (planes óptimos) o planes que superen cierto umbral de probabilidad de éxito (basados en umbrales). Lo interesante es que un planificador basado en umbrales puede usarse como base para la construcción de un planificador de optimización mediante el incremento iterativo del umbral hasta que no haya más planes posibles. En forma inversa, si en un planificador de optimización detiene la iteración de valores al llegar a un valor de umbral dado, entonces el resultado es un planificador basado en umbrales.

2.3.3. POMDPs

Los MDPs consideran observabilidad completa del ambiente. Sin embargo, algunas extensiones consideran que la observabilidad de un estado es limita-

da y que puede ser incierta. Estas otras técnicas se conocen como Procesos de Decisión de Markov Parcialmente Observables o POMDPs. En un POMDP el espacio de estados está formado por el conjunto de distribuciones de probabilidad sobre el conjunto de estados determinísticos S . Estos estados probabilistas se conocen como *estados de creencia*. Los POMDPs han sido ampliamente estudiados en Investigación de Operaciones y Teoría de Control [1], y han generado mucho interés en los círculos de IA [15, 58]. Los diagramas de influencia [41] son un medio de representación de POMDPs.

En un POMDP, hay un conjunto de etiquetas de observación O y un conjunto de probabilidades condicionales $P(o|a, s)$, $o \in O$, $a \in A$, $s \in S$, tal que si el sistema realiza una transición al estado s mediante la acción a , este recibe la etiqueta de observación o con probabilidad $P(o|a, s)$. Cassandra, Kaelbling, y Littman [15] presentan el algoritmo Witness para la resolución de POMDPs. Una técnica estándar para encontrar la política óptima en un POMDP es construir el MDP cuyo conjunto de estados son los estados de creencia del POMDP original; esto es, cada estado es una distribución de probabilidades sobre los estados del POMDP, donde las creencias mantenidas se basan en las etiquetas de observación determinadas por la regla de Bayes. En este espacio de estados se puede usar una variante del método de iteración de valores [62] asumiendo que cada política de horizonte finito será convexa y lineal.

Los POMDPs se destacan por su dificultad computacional. El problema de encontrar una política óptima con el objetivo de maximizar la recompensa total esperada o la recompensa descontada total esperada sobre un horizonte finito T (ver sección 3.2) ha demostrado ser duro exponencialmente tanto respecto a $|S|$ como a T . El problema de encontrar una política que maximice la recompensa total descontada esperada sobre un horizonte infinito simplemente es indecidible.

Aunque el trabajo en POMDPs es prometedor, sólo se pueden usar para resolver problemas pequeños en cuanto al número de estados y observaciones posibles [13].

2.4. Resumen del capítulo

En este capítulo se describieron algunos trabajos que extienden las suposiciones básicas de los planificadores clásicos. Todas las técnicas hacen uso de distintos estilos y atacan diferentes partes de las extensiones al problema básico de la planificación expuestas aquí. Entre estos se encuentran enfoques que introducen representaciones no deterministas de las acciones, representaciones de planes condicionales, técnicas para computar o estimar la máxima utilidad del plan, representaciones explícitas de los eventos exógenos inciertos, y planificación probabilista. Como se puede notar, cada uno resuelve distintas partes del problema mediante representaciones compactas del mundo, pero dado que cada uno visualiza el problema desde un ángulo distinto, su integración parece difícil. Por su parte, el formalismo de los MDPs parece ser más adecuado dado que cubre mayor número de suposiciones básicas de la planificación con incertidumbre, aunque con la desventaja de que solo pueden ser aplicados en su forma

natural sobre espacios de estados y acciones pequeños.

La siguiente sección profundiza en las técnicas Markovianas para la planificación con incertidumbre donde el reto a vencer es el crecimiento exponencial de combinaciones de estados y acciones para alcanzar una solución tratable. Igualmente se exponen los trabajos más representativos alrededor de estas corrientes buscando que las funciones de utilidad y recompensa también aprovechen la estructura del dominio.

Capítulo 3

Procesos de decisión de Markov

A pesar de que por separado cada una de las técnicas de la sección 2.3.1 satisfacen diferentes suposiciones de la planificación con incertidumbre, las técnicas Markovianas se han convertido en un estándar para la planificación basada en teoría de decisiones. Una de las razones es que simultáneamente el formalismo asume no determinismo en las acciones y distintos grados de preferencia en las metas. Otra razón ha sido el éxito de las técnicas Markovianas en áreas como reconocimiento de voz, y el de sus muy cercanas técnicas de aprendizaje por refuerzo [71] en control automático. Quizá, lo que mayormente ha motivado su uso en IA han sido los avances recientes en modelos gráficos probabilistas para representar problemas complejos en forma compacta.

En este capítulo se profundiza en el estudio de los Procesos de Decisión de Markov como uno de los paradigmas más importante de la planificación basada en teoría de decisiones. Inicialmente, se define el problema básico de planificación junto con algunos términos de uso común. Igualmente se formaliza el método y se explican sus algoritmos de solución. Finalmente, se hace un análisis del estado del arte en MDPs enfatizando nuevas formas para su representación, y técnicas para asegurar su tratabilidad al codificar problemas complejos.

3.1. Definición del problema

Un *problema de decisión secuencial* es un proceso en el cual un sistema evoluciona en el tiempo y es controlado por un agente. La dinámica del sistema es gobernada por una función de transición probabilista que asocia estados y acciones con nuevos estados. Esto es, la dinámica del sistema se representa mediante una función que devuelve una distribución de probabilidades de transición a otros estados, dado un estado actual y una acción. En cada paso, el agente recibe una recompensa numérica que en general depende del estado actual y de la acción aplicada. En este contexto, una acción es un tipo de evento instigado

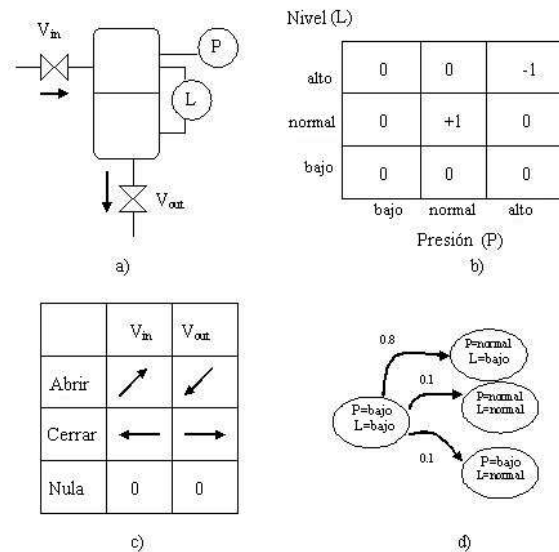


Figura 3.1: Ejemplo de un proceso de decisión de Markov. a) Ejemplo del tanque de agua. b) Espacio de estados 2-D y valores de recompensa. c) Espacio de acciones y sus efectos esperados sobre las variables del sistema (denotado por flechas). d) Cadena de Markov representando una función de transición para el estado $L = bajo$ y $P = bajo$ dada la acción cerrar V_{in} .

por un agente con la intención de cambiar el estado de un sistema. Nosotros asumimos que el agente tiene control sobre qué acciones se van a ejecutar y cuándo, aun cuando los efectos de tomar una acción pueden no ser perfectamente predecibles. En contraste, los eventos exógenos no se encuentran bajo el control del agente, y su ocurrencia puede ser sólo parcialmente predecible.

Al problema de encontrar una estrategia reactiva de control o *política de acción* en un problema de decisión secuencial que maximice la recompensa esperada en el tiempo se le conoce como *proceso de decisión de Markov*, en honor al estadístico ruso Andrei A. Markov. El trabajo de Markov está estrechamente ligado a las suposiciones de que el agente siempre conocerá el estado en que se encuentra al momento de iniciar la ejecución de sus acciones (observabilidad total), y que la probabilidad de transición de un estado depende sólo del estado y no de la historia (*propiedad de Markov*). Los MDPs fueron introducidos originalmente por Bellman [2] y han sido estudiados a profundidad en los campos de análisis de decisiones e investigación de operaciones desde los 60's iniciando con el trabajo seminal de Howard [40]. Entre los textos más importantes en el área de MDPs se encuentran los trabajos de Bertsekas [3] y Puterman [62].

Para entender mejor la idea de un MDP, considere un tanque con dos válvulas conteniendo agua, una para cargar, V_{in} , y otra para descargar líquido, V_{out} (ver figura 3.1 a). Las variables del sistema pueden representarse mediante

Nivel (L)				
alto	→	↙	-1	
normal	→	+1	←	
bajo	↗	↗	←	
		bajo normal alto		Presión (P)

Figura 3.2: Ejemplo de política óptima para el problema del tanque de agua. Si el sistema se encuentra en el estado $P = bajo$, $L = normal$, ó $P = alto$, $L = normal$, puede llegar a la meta (+1) mediante la aplicación respectiva de las acciones $cerrar V_{out} \rightarrow$ ó $cerrar V_{in} \leftarrow$.

dos propiedades discretas, nivel L y presión P , las cuales pueden tomar los valores $\{alto, normal, bajo\}$. El espacio de estados está conformado por todas las combinaciones posibles de valores para estas dos variables, por ejemplo: $P = bajo$ y $L = bajo$, $P = bajo$ y $L = normal$, $P = alto$ y $L = bajo$, etc., de modo que la dimensión del espacio de estados $|S|$ es $3^2 = 9$. Las acciones de control posibles en el sistema son abrir/cerrar V_{out} o V_{in} , y la acción nula. El espacio de acciones y los efectos deseados después de aplicar cada acción se muestran en la figura 3.1 c. Note que las acciones de apertura y cierre de válvulas cambian idealmente el estado del sistema de acuerdo a las direcciones de las flechas. La acción nula no tiene efectos en el proceso. Una función probabilista de transición de estados se puede especificar de acuerdo a lo siguiente: el sistema cambia al estado indicado por las direcciones de cambio ideal con probabilidad de 0.8, y a los estados cercanos con probabilidad de 0.2. Por ejemplo, si el estado del proceso es $P=bajo$ y $L=bajo$, la probabilidad de transición al estado $P=normal$ y $L=bajo$, dada la acción cerrar V_{out} , es 0.8. Con probabilidad de 0.2 esta misma acción lleva al sistema a los estados $L=normal$ y $P=normal$, o $L=normal$ y $P=bajo$ (ver cadena de Markov en la figura 3.1 d). Finalmente, la asignación de recompensa a cada estado se hace en función de los estados de proceso más seguros, premiando a los estados centrales y penalizando los estados con valores máximos de las variables. En este caso, se asigna un valor de recompensa inmediata de +1 al estado donde $P = normal$ y $L = normal$, -1 al estado inseguro $P = alto$ y $L = bajo$, y 0 a cualquier otro estado. Los valores de recompensa también se muestran en la figura 3.1 b. Considerando que los estados del sistema son accesibles, esto es que se pueden determinar sin errores con instrumentos de presión y de nivel, el problema es encontrar una asociación estado-acción que maximice la recompensa en el futuro. Una probable política óptima para el problema del tanque de agua se muestra en la figura 3.2. Nótese que si el sistema se encuentra en el estado $P = bajo$, $L = normal$, ó $P = alto$,

$L = normal$, puede llegar a la meta (+1) mediante la aplicación respectiva de las acciones $cerrar V_{out} \rightarrow$ ó $cerrar V_{in} \leftarrow$.

3.2. Formalización

Formalmente, un MDP es una tupla $M = \langle S, A, \Phi, R \rangle$, donde S es un conjunto finito de estados del sistema $\{s_1, \dots, s_n\}$. A es un conjunto finito de acciones $\{a_1, \dots, a_m\}$. $\Phi : A \times S \rightarrow \Pi(S)$ es la función de transición de estados, la cual asocia un conjunto de posibles estados resultantes dada una acción en el estado actual. La probabilidad de alcanzar el estado s' realizando la acción a en el estado s se escribe $\Phi(a, s, s')$. El proceso de transición se considera como *estacionario*, es decir que sólo depende del estado y no del tiempo. $R : S \times A \rightarrow \mathfrak{R}$ es la función de recompensa. $R(s, a)$ es la recompensa que el sistema recibe si lleva a cabo la acción a en el estado s .

Una política para un MDP es una asociación $\pi : S \rightarrow A$ que selecciona una acción por cada estado. Para evaluar una política o curso de acción particular, se necesita especificar su longitud de ejecución en etapas. Esto se conoce como el *horizonte* del problema. En problemas de *horizonte finito*, la eficiencia del agente se evalúa sobre un número finito de etapas T . El objetivo en este caso es maximizar la recompensa esperada total. Por otra parte, los problemas de *horizonte infinito* requieren que la eficiencia del agente sea evaluada sobre una trayectoria infinita. En este caso la recompensa total puede estar no limitada, implicando que cualquier política puede ser arbitrariamente buena o mala si se ejecuta durante mucho tiempo.

Dada una política, es posible definir su función de valor en horizonte finito $V_n^\pi : S \rightarrow \mathfrak{R}$, donde $V_n^\pi(s)$ es el valor esperado de aplicar la política π durante n pasos a partir del estado s . La función de valor se define inductivamente con $V_0^\pi(s) = R(s, \pi(s))$ y $V_m^\pi(s) = R(s, \pi(s)) + \sum_{s' \in S} \Phi(\pi(s), s, s') V_{m-1}^\pi(s')$. Sobre un horizonte infinito se acostumbra usar un modelo de descuento para asegurar que las políticas tengan un valor esperado limitado, donde el parámetro $0 \leq \gamma < 1$ es el *factor de descuento*, usado para descontar recompensas futuras en proporciones geométricas. Así, si $V^\pi(s)$ es el valor esperado descontado del estado s siguiendo indefinidamente la política π , debemos tener que $V^\pi(s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} \Phi(\pi(s), s, s') V_{m-1}^\pi(s')$, lo cual produce un conjunto de ecuaciones lineales en los valores de $V^\pi()$.

3.3. Métodos de solución

La solución a un MDP es una política que maximice su valor esperado. Para el caso del horizonte infinito descontado con cualquier factor dado $\gamma \in [0, 1]$, existe una política π^* que es óptima sin importar el estado de inicio y que satisface la ecuación de *Bellman* [2]:

$$V^*(s) = \max_a \{R(s, a) + \gamma \sum_{s' \in S} \Phi(a, s, s') V^*(s')\} \quad (3.1)$$

Figura 3.3: Algoritmo de iteración de política.

IteraciondePolitica(S, A, Φ, R, γ):

1. Para cada $s \in S$, $\pi(s) = \text{ElementoAleatorio}(A)$
2. Computar $V_\pi(\cdot)$
3. Para cada $s \in S$ {
 - a) Encontrar una acción a tal que $R(s, a) + \gamma \sum_{u \in S} \phi(a, s, u) V^\pi(u) > V^\pi(s)$;
 - b) Hacer $\pi'(s) = a$ si tal a existe;
 - c) de otra forma, hacer $\pi'(s) = \pi(s)$.
- }
4. Si $\pi'(s) \neq \pi(s)$ para algún $s \in S$, ir a 2.
5. Devolver π

Las técnicas para resolver esta ecuación y determinar la política óptima para un MDP son: programación dinámica y programación lineal.

3.3.1. Programación dinámica

Los métodos más populares de programación dinámica son: (a) iteración de valor e (b) iteración de política [62].

En la iteración de política, la política actual es repetidamente mejorada al encontrar alguna acción en cada estado que tenga un valor más alto que el de la acción escogida en la política actual para el estado. La política inicial es aleatoria, y el proceso termina cuando ya no es posible realizar más mejoras. Este proceso, que converge a una política óptima [62], se muestra en el la figura 3.3.

En iteración de valor, las políticas óptimas son producidas por horizontes finitos sucesivamente más largos hasta que convergen. Es relativamente sencillo encontrar una política óptima en n pasos $\pi_n^*(\cdot)$, con la función de valor $V_n^*(\cdot)$ usando la relación de recurrencia:

$$\pi_n^*(s) = \arg \max_a \{R(s, a) + \gamma \sum_{s' \in S} \Phi(a, s, s') V_{n-1}^*(s')\}, \quad (3.2)$$

con la condición inicial $V_0^*(\cdot) = 0 \forall s \in S$, donde V_m^* es derivado de la política π_m^* como se describió anteriormente.

El algoritmo toma un MDP, un valor de descuento, y un parámetro de convergencia y produce políticas óptimas de horizonte finito sucesivas, terminando cuando el máximo cambio en los valores entre la función del valor actual y el del

Figura 3.4: Algoritmo de iteración de valor.

IteraciondeValor($S, A, \Phi, R, \gamma, \epsilon$):

1. Para cada $s \in S$, $V_0(s) = 0$
2. $t = 0$
3. $t = t + 1$
4. Para cada $s \in S$ {
 - a) Para cada $a \in A$ hacer

$$Q_t(s, a) = R(s, a) + \gamma \sum_{u \in S} \phi(a, s, u) V_{t-1}(u);$$
 - b) $\pi'(s) = \operatorname{argmax}_a Q_t(s, a)$
 - c) $V_t(s) = Q_t(s, \pi_t(s))$
5. Si $(\max_s |V_t(s) - V_{t-1}(s)| \geq \epsilon)$, ir a 3.
6. Devolver π_t

previo es menor que ϵ . El algoritmo converge a la política óptima y se ilustra a detalle en la figura 3.4.

3.3.2. Programación lineal

Otro método muy utilizado para la resolución de MDPs es formular un problema de decisión como un problema de *programación lineal (PL)*. Un problema de programación lineal es un caso especial de *programación matemática*, el cual trata de identificar un punto extremo (máximo o mínimo) de una función $f(x_1, x_2, \dots, x_n)$, que además satisface un conjunto de restricciones $g(x_1, x_2, \dots, x_n)$. La programación lineal especializa la programación matemática en el caso donde la función f es llamada la función objetivo, y donde las restricciones g del problema son lineales.

Se puede demostrar (ver [62] sección 6.2) que si V satisface $V \geq R(s, a) + \gamma \phi(a, s, s')V$ entonces V es el límite superior para el valor óptimo V^* . Con base en este teorema, se plantea el *programa lineal primal* para resolver un problema de decisión de Markov descontado, el cual se expresa de la siguiente manera:

$$\text{Minimizar } \sum_{s \in S} V(s)$$

sujeto a

$$V(s) - \gamma \sum_{s' \in S} \Phi(a, s, s') V_{s'} - R(s, a) \geq 0, \forall s, a$$

donde los valores de $V(s)$ por cada estado s son tratados como variables.

Normalmente, se prefiere la formulación dual sobre la primal ya que tiene $|S|$ filas y $\sum_{s \in S} |A_s|$ columnas (la primal tiene $|S|$ columnas y $\sum_{s \in S} |A_s|$ filas). Entre los métodos eficientes de solución de programas lineales destacan aquellos basados en *Simplex Coseno* [74, 73, 18]. Aun cuando la década pasada la programación lineal tradicional se limitaba a la solución de MDPs descontados con horizontes finitos, muy recientemente, esta área emerge como una alternativa prometedora para la solución de MDPs altamente dimensionales y con espacios continuos [34, 68, 36]. De Farias y Van Roy [28] incluso proponen una técnica para aproximar la programación dinámica con la idea de resolver problemas de control estocástico de gran escala.

3.4. Limitaciones y problemas

Un MDP completamente observable puede resolverse con los algoritmos de programación dinámica de la sección 3.3.1 en tiempo polinomial respecto al tamaño del espacio de estados $|S|$, el número de acciones $|A|$, y la máxima precisión en bits para representar valores de recompensa $\log \max_{s,a} |R(s, a)|$. Tanto en problemas de horizonte finito como de horizonte infinito descontado la cantidad de cómputo requerida por iteración es polinomial ($O(|S|^2|A|)$ y $O(|S|^2|A| + |S|^3)$, respectivamente), y convergen en un número polinomial de iteraciones. El espacio requerido para almacenar la política para un problema de horizonte finito de n etapas es $O(|S|n)$.

Debido a que el espacio de estados de un dominio crece exponencialmente al número de variables $|X|$, en problemas con alta dimensionalidad en variables (y acciones), o donde existen variables continuas, los MDPs pueden volverse imprácticos, ineficientes y, lo que es peor, no computables. Por lo anterior, hemos clasificado éstos problemas de acuerdo a los retos que éstos imponen: representación y solución.

1. Representación. Cómo representar en forma estructurada y compacta los estados y acciones de un dominio altamente dimensional.
2. Solución. Aun en el caso de que tal representación existe, como asegurarse de que su solución es tratable.

La siguiente sección muestra como las representaciones factorizadas, se pueden utilizar para representar en forma muy compacta espacios de estados exponencialmente grandes. La sección 3.6.1 por su parte expone algunos de los trabajos más recientes en métodos de abstracción, agregación y descomposición para

asistir a las representaciones factorizadas en la construcción de políticas. Finalmente, se comentan los avances, desde el punto de vista representacional y de inferencia (solución), para abordar dominios con variables continuas y combinaciones continuas-discretas.

3.5. Representaciones factorizadas

Hasta este punto de nuestra discusión, los MDPs han usado una representación *extensiva* de los estados y las acciones en la cual los estados se enumeran directamente. En general, al menos desde el punto de vista computacional, es más conveniente y compacto describir conjuntos de estados basados en ciertas propiedades o características que enumerarlas explícitamente. Las representaciones en las que las descripciones de los objetos sustituyen a los objetos mismos se llaman *intencionales o factorizadas* [11, 9] y son una alternativa para representar grandes espacios de estados y acciones en forma compacta.

Una representación intencional en un problema de planificación se construye normalmente mediante la definición de un conjunto de características, las cuales deben ser suficientes para describir el estado del sistema dinámico de interés.

El hecho de que sea posible describir el estado de un sistema mediante un conjunto de características permite adoptar la representación factorizada de las acciones, recompensas, y otros componentes de un MDP. En la representación factorizada de las acciones, por ejemplo, se describe el efecto de una acción sobre características específicas de un estado y no sobre estados enteros. Frecuentemente esto ofrece representaciones más compactas. Por ejemplo, en la representación de acciones tipo STRIPS, las transiciones de estado inducidas por las acciones son representadas implícitamente al describir los efectos de las acciones sobre aquellas características que cambian de valor cuando la acción es ejecutada. Las representaciones factorizadas pueden ser muy compactas cuando las acciones individuales afectan relativamente pocas características, o cuando sus efectos exhiben ciertas regularidades. Estas notas aplican igualmente a las funciones de recompensa, modelos de observación, modelos de transición, etc. Las regularidades que hacen que las representaciones factorizadas sean útiles en algunos problemas de planificación pueden explotarse por algoritmos de planificación y de toma de decisiones. Distintos autores [14, 25, 27, 33, 37, 44, 65] han usado estas representaciones en dominios completamente observables.

3.5.1. Espacios de estados factorizados

Boutilier [9] denomina *estados estructurados* a aquellos sistemas cuyos estados pueden describirse mediante un conjunto finito de variables de estado cuyos valores cambian con el tiempo. Suponga que el estado de una máquina de combustión es descrito por un conjunto de tres características: cantidad de combustible, cantidad de aire, y temperatura de gases producidos. Cada una de las tres características puede tomar uno de tres valores (alta, media, baja). Una asignación de valores a las tres características define completamente un

estado; el espacio de estados comprende así todas las posibles combinaciones de los valores característicos, con $|S| = 3^3 = 27$. A cada característica o factor, se le asigna un nombre simbólico único tal como: *aire(alto)*, *temp(media)*, o *gases(bajo)*. Las representaciones extensionales visualizan un estado como una sola variable que puede tomar 27 valores posibles, mientras que la factorizada ve a un estado como el producto cruz de tres variables, cada uno de los cuales toma sustancialmente menos valores. Generalmente, el espacio de estados crece exponencialmente con el número de características requeridas para describir un sistema, y no con el número total de literales del dominio que normalmente es mucho mayor.

Más formalmente, el conjunto de estados se describe mediante un conjunto de variables aleatorias $\mathbf{X} = \{X_1, \dots, X_n\}$, donde cada X_i toma valores de algún dominio finito $Dom(X_i)$. Un estado \mathbf{s} define un valor $x_i \in Dom(X_i)$ por cada variable X_i . Se dice que el espacio de estados es *plano* si se especifica usando una sola variable de estado, y *factorizado* si hay más de una variable de estado. El espacio de estados es el producto cruz del espacio de valores para las variables de estado individuales; esto es $S = \times_{i=1}^M x_i$. Al igual que S^t denota el estado de un proceso en el tiempo t , x_i^t representa el valor de la i -ésima variable aleatoria en el tiempo t .

En ambientes parcialmente observables, la especificación extensional de las probabilidades de un estado es aún más complicada, por lo que con mayor razón se recomienda el uso de representaciones factorizadas.

3.5.2. Acciones factorizadas

Cuando el número de variables de estado M es muy grande, el conjunto de estados $S = Dom(X_i)$ puede ser exponencialmente grande, haciendo impráctico representar explícitamente el modelo de transición de estados bajo la influencia de una acción en forma de matrices de $N \times N$ o por diagramas de transición de estados. Afortunadamente, el marco de las redes Bayesianas dinámicas (RBD) [21, 19] ofrecen herramientas para describir el modelo de transición en forma concisa.

Un modelo de transición Markoviano Φ define una distribución de probabilidades sobre el siguiente estado dado el estado actual y una acción a . Hagamos que X_i denote la variable X_i en el tiempo actual y X'_i la variable en el siguiente instante. El *gráfico de transición* de una RBD es un gráfico acíclico dirigido de dos capas G_T cuyos nodos son $\{X_1, \dots, X_n, X'_1, \dots, X'_n\}$. En esta representación, los padres de X'_i son denotados como $Parents(X'_i)$. Cada nodo X'_i está asociado con una distribución de probabilidades condicionales (CPD) $P_\Phi(X'_i | Parents(X'_i))$, la cual normalmente se representa con una tabla (*conditional probability table*). La probabilidad de transición $\Phi(a, s_i, s'_i)$ se define como $\Pi_i P_\Phi(x'_i | \mathbf{u}_i)$ donde \mathbf{u}_i es el valor en \mathbf{s} de las variables en $Parents(X'_i)$. El lado izquierdo de la figura 3.5 muestra un modelo de transición arbitrario para una acción en particular. En la figura, los nodos del lado izquierdo representan las variables del dominio en el tiempo t y los de la derecha en el tiempo siguiente. Los arcos muestran las relaciones causales entre los nodos del tiempo t (*nodos de*

acción) y el tiempo $t+1$ (nodos de post-acción). Nótese que existe una estructura de red dinámica (2 etapas) con parámetros por cada acción del dominio.

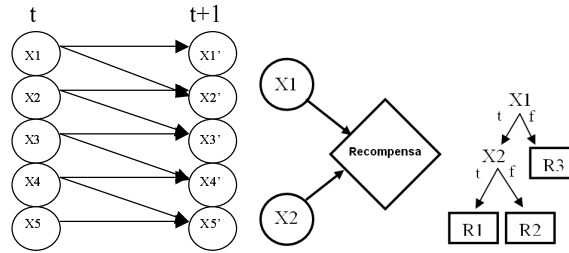


Figura 3.5: Representación estructurada de las funciones de transición y recompensa. (izquierda) Representación de la transición de estados por la aplicación de una acción mediante una RBD simple con cinco variables de estado. (centro) Diagrama de influencia denotando una función de recompensa en términos de las variables X_1 y X_2 , y su recompensa condicional estructurada (CR) representada como un árbol de decisión binario (derecha).

3.5.3. Recompensas factorizadas

La función de recompensa se representa extensionalmente como un vector de tamaño $|S|$ que se vuelve infactible ante grandes espacios de estados (S). Las representaciones factorizadas de recompensas y costos de las acciones pueden aliviar la necesidad de enumerar explícitamente los parámetros de los estados y las acciones.

Al igual que en un modelo de transición, la recompensa asociada con un estado depende sólo de los valores de ciertas características del dominio. Esta recompensa (o penalización) es independiente de otras variables, y las recompensas individuales pueden asociarse con los grupos de estados que difieren en los valores de las variables relevantes. La relación entre recompensas y variables de estado se representa mediante nodos de utilidad en un diagrama de influencia figura 3.5 (centro). Por su parte, las tablas de recompensa condicional (CRT) de cada nodo son tablas que asocian una recompensa con cada combinación de valores de sus padres en la gráfica. Esta tabla es localmente exponencial de acuerdo al número de variables relevantes. Aun cuando en los peores casos se requieran espacios exponenciales para almacenar los CRTs, en muchos casos la función de recompensa exhibe cierta estructura, lo cual permite que se represente en forma compacta usando árboles de decisión o gráficas (ver lado derecho de la figura 3.5).

3.6. Abstracción, agregación y descomposición

Las representaciones factorizadas normalmente se usan para describir un problema en forma compacta. Sin embargo, no garantizan que el modelo se pueda resolver efectivamente al abordar dominios continuos o altamente dimensionales. Los métodos de abstracción y agregación, así como los métodos de descomposición de problemas permiten contruir planes y políticas sobre representaciones factorizadas [9]. De esta manera, ambas técnicas combinan sus capacidades para mejorar la tratabilidad de un problema complejo.

En esta sección se comentan estas técnicas ya que constituyen las bases de las líneas de investigación actuales de la comunidad científica para la solución de problemas de planificación con MDPs.

3.6.1. Abstracción y agregación

Las técnicas de *abstracción y agregación* permiten agrupar estados implícita o explícitamente respecto a ciertas características como por ejemplo la dinámica de transición de estados, el valor de utilidad o la política. A estas agrupaciones se les conoce como *estados abstractos*, *estados agregados*, o simplemente *agrupaciones* ya que asumen que el conjunto de estados abstractos constituyen una partición del espacio de estados. Estas técnicas pueden usarse por aproximación, si la característica de los elementos de un estado abstracto coincide, con cierto umbral de error, con la característica de agrupación; o por exactitud cuando la característica de agrupación y la característica del estado abstracto coinciden exactamente. Varios autores usan estas ideas para encontrar métodos computacionalmente factibles que permitan la construcción de políticas óptimas ó, aproximadamente óptimas y satisfactorias.

Ciertas técnicas de abstracción conocidas como “minimización del modelo” buscan reducir el modelo de transición representado por una cadena de Markov a un autómata finito llamado “modelo mínimo”. Dean y Givan [20] describen un algoritmo que particiona el espacio de estados en un conjunto de bloques tal que cada bloque sea *estable*; esto es, que conserve las mismas probabilidades de transición que el modelo original. Aunque el algoritmo produce una partición exacta, ésta aun podría seguir siendo muy compleja. En muchas aplicaciones podría ser suficiente construir un modelo aproximado para obtener políticas cercánamente óptimas.

Koller y Parr [44] aprovechan los casos donde algunas acciones tienen dinámicas de transición similares difiriendo en su efectos sobre un pequeño grupo de variables o en las probabilidades condicionales de las variables de post-acción. En particular, hay muchos casos donde una variable tiene un modelo de evolución por defecto, el cual cambia si una acción la afecta directamente. Por lo tanto es posible abstraer el espacio de acciones usando el concepto de *modelo de transición por defecto* $\Phi_d = \langle G_d, P_d \rangle$. Para cada acción a , ellos definen $Effects[a] \subseteq X'$ como las variables del siguiente estado cuyo modelo local de probabilidad es diferente de Φ_d , i.e., aquellas variables X'_i tales que $P_a(X'_i \mid Parents_a(X'_i)) \neq P_d(X'_i \mid Parents_d(X'_i))$. Si se definen 5 acciones

a_1, \dots, a_5 , y el modelo de transición por defecto Φ_d de la figura 3.5, y suponiendo que la acción a_2 sólo cambia la CPT de la variable X'_2 , entonces $Effects[a_2] = X'_2$.

Otra forma de abstracción puede obtenerse representando las funciones de valor y de política en forma de diagramas de decisión algebraicos (ADDs por sus siglas en inglés) [37]. En esta representación una función de valor (o de política) se describe en términos de las variables del dominio en vez de hacerlo en la forma tabular clásica. Esta función del dominio se representa como una gráfica de decisión que con frecuencia es extremadamente compacta y que implícitamente agrupa estados coincidentes por valor en diferentes etapas del proceso de programación dinámica. Por lo tanto, el número de enumeraciones del valor esperado y de maximizaciones requeridas por el proceso de programación dinámica se reduce en forma importante. En términos generales, la solución de estas representaciones estructuradas se realiza mediante técnicas tradicionales de programación dinámica estocástica.

Una de las dificultades con los esquemas de abstracción adaptivos como el anterior es que en cada etapa se debe construir una abstracción, lo cual implica gastos computacionales importantes. Una forma de reducir este gasto es adoptando esquemas de abstracción fija [67, 43]. En estos trabajos, las variables (en este caso proposicionales) son ordenadas por *criticalidad*, esto es, qué tan importantes son para la solución del problema de planificación, y la abstracción se construye eliminando las proposiciones de baja criticalidad. Estas intuiciones son aplicadas por Boutilier y Dearden [10] quienes representan las acciones en forma similar al planificador BURIDAN [46] y la función de utilidad en términos de las literales del dominio. Su idea se basa en seleccionar el subconjunto de literales del dominio que produzca la mayor variación en la utilidad del estado. Estos atributos forman las bases de un MDP abstracto por proyección de los estados originales. Dado que el tamaño del espacio de estados es exponencial respecto al conjunto de literales, esta reducción genera ahorros en tiempo de cómputo. Boutilier y Dearden muestran las diferencias entre el valor de la política abstracta y la política óptima en el MDP original para verificar el ahorro computacional a favor de su método. Más recientemente, Pineau [61] usa el concepto de agregación con base en técnicas de agrupamiento jerárquico para construir MDPs jerárquicos.

3.6.2. Métodos de descomposición

Otro tipo de técnicas consideran como alternativa la descomposición del problema, en la cual un MDP se particiona en diferentes *sub-problemas* que se resuelven en forma independiente y luego se arman mediante alguna estructura de comunicación entre partes. Estas técnicas pueden restringir la atención a regiones del espacio de estados reelevantes. En conjunto, se puede usar la relación de regiones relevantes y la estructura de comunicación para producir descomposiciones seriales o paralelas. El resultado de una descomposición serial es una partición del espacio de estados en bloques (subprocesos) más o menos independientes. Una descomposición paralela se parece más a las representaciones abstractas. En esta variante, un MDP se divide en “sub-mdps paralelos”

tales que cada decisión o acción permita un cambio de estado dentro de cada sub-mdp.

Dean *et al.* [22] proponen un método para aproximar la solución de un MDP en un espacio de estados restringido llamado *envolvente*. Suponiendo que es posible generar rápidamente un camino entre un estado inicial y una región meta, la envolvente queda conformada por los estados dentro de tal camino y sus transiciones al exterior (*estados frontera*). La idea general del método es seleccionar un subconjunto del espacio de estados, reemplazando cada transición al exterior por una transición a un nuevo “estado Salida”, el cual tendrá recompensa 0. Este “estado Salida” no tiene transiciones a otros estados. Dean desarrolló un algoritmo donde aplica iteración de política inicialmente sobre una envolvente muy pequeña (envolvente inicial). Bajo criterios dependientes del dominio iterativamente expande, y resuelve otro MDP. El método termina cuando la envolvente alcanza el tamaño del espacio de estados original del problema o hasta cumplir con un plazo de tiempo máximo. La figura 3.6 muestra el ejemplo de navegación robótica utilizado por Dean para ilustrar el método de la envolvente y el cálculo progresivo de la política en particiones del espacio de estados. En la figura 3.6(a) se puede observar que la envolvente inicial incluye la meta (punto negro sobre la parte central izquierda). Igualmente se puede apreciar la existencia del conjunto de estados abstracto que no forma parte de la envolvente inicial, el cual es denotado con la etiqueta “no en la envolvente”, y que para el cálculo de la política óptima, se considera con recompensa 0 (estado Salida). El algoritmo en este caso resuelve tres MDPs, primero el espacio de estados de (a), luego (b), y finalmente (c) que es donde la envolvente contiene el espacio de estados del problema original con su política óptima.

El algoritmo converge a una política óptima mucho más rápido que la versión estándar de iteración de política sobre todo el espacio de estados. Sin embargo, las suposiciones de independencia que hace sobre la influencia de las soluciones locales en la solución global limitan su aplicabilidad a ciertos dominios. Tash y Russell [72] extienden la idea de la envolvente con un estimado inicial a la meta por estado y un modelo que toma en cuenta el tiempo de cómputo. Además usan una función heurística que les permite ignorar partes del espacio de estados.

Dean y Lin [23] combinan esta idea de descomponer el problema y resolver varios MDPs factorizados, con la intención de generar abstracciones diferentes en cada sub-MDP. Según Dean y Lin este es un caso especial de “divide y vencerás” donde, dado un MDP y una partición del espacio de estados se aplica el siguiente algoritmo: i) plantear el problema en términos de pequeños MDPs, ii) resolver cada uno de estos sub-problemas, y luego iii) combinar las soluciones locales para obtener la solución global del problema. El criterio de agregación usado por Dean y Lin se basa en agrupar estados que difieren en los valores de las variables irrelevantes dentro de sus subespacios. En la figura 3.7 es posible observar que un espacio tridimensional puede expresarse como la unión de sub-espacios abstractos bi-dimensiones. Los estados marcados en negro, indican el uso de criterios de abstracción distintos respecto a los otros. Para este caso, se asume que sólo existen dos criterios distintos. Este enfoque puede dar una mejor aproximación puesto que varias literales pueden ser relevantes en difer-

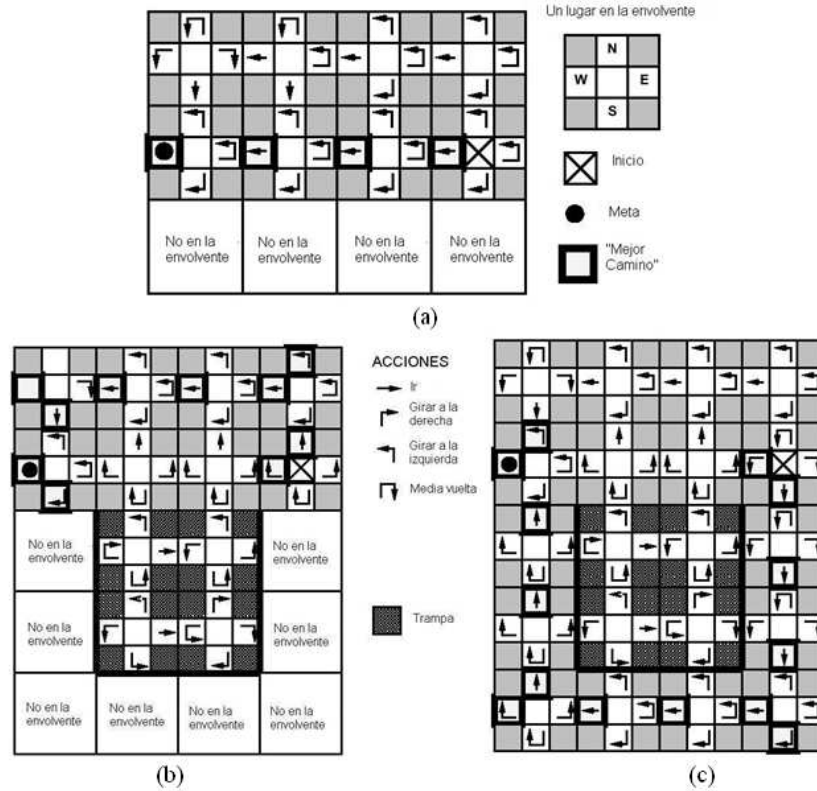


Figura 3.6: Ejemplo del problema de navegación robótica en [22]. Se muestran descripciones de los estados inicial y meta, las acciones y los estados no deseados. Igualmente, se esquematizan las etapas de la solución con la evolución de la política óptima.

entes partes del espacio de estados. Sin embargo, hay un costo adicional por recombinación de las piezas separadas. Lin y Dean asumen que la partición del estado esta dada por un oráculo externo.

Elinas *et al.* [27] presentan un método para la coordinación de robots sociales basado en el concepto de MDPs multi-seccionados (MS-MDP). Un MS-MDP es un conjunto de N MDPs, los cuales comparten la misma meta y el mismo espacio de estados. Suponiendo que las acciones de cada MDP no entran en conflicto, la política óptima de cada proceso se ejecuta concurrentemente con los demás. En forma coordinada por un vector de estados común, cada MDP resuelve un aspecto de la tarea global y de esta forma la meta común. Aún cuando no se obtiene una solución global óptima, la ejecución simultánea y coordinada de varias tareas agrega naturalidad a una solución satisfactoria. El concepto fue probado usando el robot mensajero “HOMER”, el cual esta basado

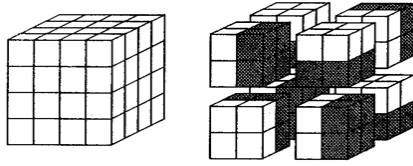


Figura 3.7: Partición del espacio de estados en regiones empleado por Dean y Lin. En este ejemplo arbitrario cada región utiliza dos posibles criterios de partición (blanco o negro).

en una plataforma robótica B-14 con una cámara de visión estéreo, una pantalla de cristal líquido con expresiones faciales, un sistema de síntesis de voz, y un sistema de navegación híbrido reactivo-deliberativo. Las tareas de planificación de HOMER en esta aplicación son: diálogo, navegación y gestos. Estas técnicas donde varias políticas se ejecutan concurrentemente se conocen como métodos de *descomposición paralela*, mientras que cuando la solución global es de naturaleza secuencial, se conocen como métodos de *descomposición serial*.

3.7. Procesos de decisión de Markov continuos

Muchos procesos estocásticos controlados se definen más naturalmente usando directamente variables de estado continuas. Estos procesos se resuelven mediante lo que llamamos *Procesos de Decisión de Markov Continuos* (CMDPs) o su generalización *MDPs de espacio general de estados*, por analogía con las cadenas de Markov de espacio general de estados. En esta técnica la función de valor óptima satisface la ecuación de Bellman de punto fijo:

$$V(x) = \max_a [R(x, a) + \gamma \int_{x'} p(x'|x, a) V(x') dx'] \quad (3.3)$$

Las soluciones existentes intentan reemplazar la función de valor o la política óptima con aproximaciones finitas. Los métodos recientes para resolver CMDPs se conocen como *discretización de MDPs basada en rejillas*, y *aproximaciones paramétricas*.

La idea de la discretización de MDPs basados en rejillas es aproximar el espacio de estados con conjuntos de celdas, y aproximar la función de valor en tales conjuntos. Definamos $G = \{x^1, x^2, \dots, x^N\}$ como un conjunto de celdas puntuales sobre el espacio de estados de $[0, 1]^n$. El operador de Bellman H puede aproximarse con un operador H_G restringido a las celdas G . Tal operador ha sido estudiado por Rust [66] y es definido como:

$$V_G(x^i) = \max_a [R(x^i, a) + \gamma \sum_{j=1}^N P_G(x^j|x^i, a) V_G(x^j)] \quad (3.4)$$

donde $P_G(x^j|x^i, a) = \psi_a(x^i) p(x^j|x^i, a)$ define una probabilidad de transición normalizada tal que $\psi(x^i)$ es una constante de normalización. La ecuación an-

terior aplicada a los puntos tipo celda, G , define un MDP de estados finito con $|G|$ estados. La solución, $V_G = H_G V_G$, aproxima el MDP de estados continuos original.

Desafortunadamente, el problema con los CMDPs es que si se discretiza el espacio para hallar la solución, la discretización puede producir un nivel de explosión exponencial. Este problema de dimensionalidad ha limitado el uso de los MDPs en dominios continuos, por lo su superación ha motivado el interés científico.

Una alternativa para resolver MDPs con estados continuos es aproximar la función de valor óptima $V(x)$ con un modelo paramétrico [5]. Los parámetros del modelo se van ajustando iterativamente aplicando un respaldo de Bellman de un paso a un conjunto finito de puntos de estado arreglados sobre una malla fija u obtenidos mediante muestreo Monte Carlo. Para ajustar los parámetros del modelo se usa el criterio de mínimos cuadrados. Además de las actualizaciones paralelas y optimizaciones del modelo, los esquemas de actualización en línea basados en descenso de gradiente [5, 71] son muy populares y pueden usarse para optimizar parámetros. La desventaja de los métodos es su inestabilidad y posible divergencia [4].

Feng *et al.* [30] proponen una técnica de agregación de estados donde se aprovecha la estructura de los dominios continuos para dinámicamente particionar el espacio de estados en regiones con la misma función de valor. La técnica proviene del área de POMDPs para representar y razonar sobre superficies lineales en forma efectiva. Li y Littman [47] proponen una técnica libre de discretizaciones llamada *aproximación floja* para manejar espacios híbridos continuos-discretos. En contraste con los métodos tradicionales de discretización que de antemano aproximan un modelo de MDP, su técnica conserva un modelo continuo y aplaza la aproximación hasta que sea necesario. En este trabajo se comparan con el método de Feng *et al.* con buenos resultados.

Hauskrecht [36] demuestra que la programación lineal aproximada permite resolver tanto MDPs con estados finitos como MDPs factorizados continuos. En sus experimentos, se usaron problemas con 25 variables de estado continuas. En forma similar, Guestrin [33] presenta una técnica que aprovecha la estructura del dominio para modelar y resolver MDPs factorizados. Sin embargo, en este trabajo se extiende el alcance para tratar con variables discretas y continuas en un ambiente colaborativo. Este grupo de investigación probó sus enfoque con espacios de estado continuos hasta de 28 dimensiones.

Más recientemente, Hoey y Poupart proponen en [38] un algoritmo que va más allá y trata de resolver POMDPs basados en modelos con espacios de observación discretos muy grandes o continuos. Ellos demuestran que es posible reducir en forma importante la complejidad de los espacios de observación sin afectar la calidad de la solución. Intuitivamente, las observaciones brindan información a un agente para escoger un rumbo de acción en el futuro. Cuando se elige un mismo rumbo de acción para dos observaciones diferentes, estas observaciones se vuelven iguales desde el punto de vista del agente, y por tanto pueden ser agrupadas (*aggregated*). Cuando una política se compone de un conjunto pequeño de planes condicionales (a las observaciones), es posible particionar la

observación en un pequeño número de regiones correspondientes a las características relevantes del espacio de observaciones del agente. Básicamente, ellos demuestran que el problema de decisión se puede usar para definir un conjunto de variables relevantes suficientes para hallar una solución. Una limitación con este procedimiento es que, aunque tratan con variables de observación continuas, asumen espacios de estados discretos.

3.8. Resumen del capítulo

En este capítulo se presentó un breve resumen del formalismo de los MDPs desde el enfoque tradicional discreto hasta las técnicas del estado del arte que lo extienden. Se explicó la formulación básica del modelo, sus métodos de solución, y algunas de sus limitaciones cuando se usa en dominios complejos. Posteriormente, se expuso la importancia de las representaciones factorizadas en la especificación compacta de problemas de planificación, y las técnicas de abstracción y agregación, y los métodos de descomposición como ayudas para mejorar la tratabilidad de problemas complejos. Finalmente, se ilustró el marco de los MDPs continuos como uno de los tópicos de investigación más retadores en IA.

Aunque varios de los trabajos anteriores resuelven bien algunos aspectos de complejidad al usar MDPs, al llevarlos a la práctica se percatan de dificultades tales como el trato con variables continuas y discretas (sistemas híbridos), múltiples variables relevantes, y la adquisición de conocimiento para la construcción de los modelos de decisión.

En el siguiente capítulo, se describirá un nuevo enfoque para la solución de MDPs continuos basado en agregación por recompensa y con aproximación automática de modelos. El método, además de ser muy simple y fácil de implementar, está inspirado en los modelos cualitativos, técnicas de muestreo simples, y aprendizaje automático. La especificación final del problema es un MDP factorizado que se resuelve con métodos tradicionales de programación dinámica.

Capítulo 4

MDPs cualitativos

En este capítulo se presenta una técnica novedosa y práctica, basada en aprendizaje automático, para resolver MDPs continuos e híbridos continuos-discretos. Aunque muy relacionada con las técnicas de agregación y abstracción expuestas anteriormente, difiere en varios aspectos. Primeramente, esta basada en *modelos cualitativos* [45], que en lo particular son muy útiles en dominios continuos. También difiere en la forma de construir la abstracción. Otras técnicas [55, 12] inician con una discretización uniforme exhaustiva de todas las variables del dominio que iterativamente se va abstrayendo. En ésta propuesta, inicialmente se conforma una partición con la recompensa como criterio de agrupación que posteriormente se refina.

Dado que la especificación de un modelo abstracto a partir de un *experto* no es una tarea fácil, se prefiere estructurar el conocimiento del dominio mediante algoritmos de aprendizaje automático. En el método propuesto, mediante la exploración del ambiente se colecta información acerca del sistema de recompensa y de la dinámica del sistema. Esta información primero se usa para construir un árbol de decisión [63] que representa un pequeño conjunto de estados abstractos con recompensas equivalentes (llamado *partición cualitativa*), y luego para aprender una función de transición de estados usando el algoritmo de aprendizaje Bayesiano K2 [17]. El modelo aproximado resultante puede resolverse mediante algoritmos tradicionales de programación dinámica.

Con la intención de hacer mas clara la presentación de esta técnica, los temas se han organizado de la siguiente manera: representación, formalización, aprendizaje y refinamiento de estados cualitativos.

4.1. Representación

4.1.1. Restricciones cualitativas

Muchos sistemas complejos se representan más naturalmente usando alguna forma de representación cualitativa. Los modelos cualitativos [32, 45, 70] se han usado para representar y simular sistemas dinámicos. Bonet y Pearl [8] usan este

tipo de representación para MDPs y POMDPs a través de una teoría cualitativa especial en la cual se propone aproximar cualitativamente las probabilidades y utilidades de un modelo.

La representación mostrada en esta sección está inspirada en el trabajo de Suc y Bratko en [70] para expresar el estado de un sistema en forma de cambios cualitativos y vectores de cambio cualitativo. Un *cambio cualitativo* ch_j es el signo de cambio de la variable continua x_j , donde $ch_j \in \{pos, neg, cero\}$ corresponde a un cambio positivo, negativo o nulo. Un *vector de cambio cualitativo* $ch = (ch_1, ch_2, \dots, ch_n)$ es un vector de cambios cualitativos de las n variables del dominio.

Como ejemplo considere el problema de la figura 4.1 (citado en [70]) donde un recipiente cerrado contiene un gas ideal. Las variables continuas del sistema son la temperatura y volumen del gas, y la presión que puede calcularse mediante la función $Pres = 2 Temp/Vol$. Una forma de representar el estado del sistema mediante vectores de cambio cualitativo para el conjunto de puntos en el espacio de atributos de la figura respecto al punto $(Temp=315, Vol=56, Pres=11.25)$ daría como resultado $ch_1 : (cero, pos, neg)$, $ch_2 : (pos, neg, pos)$, $ch_3 : (neg, neg, pos)$, y $ch_4 : (neg, neg, neg)$.

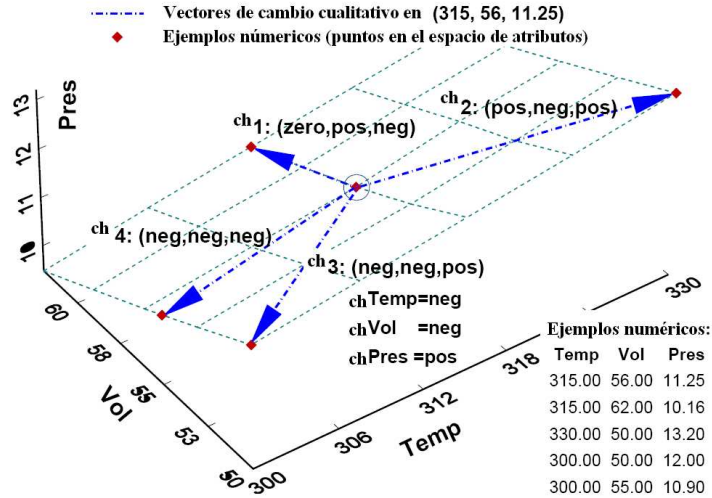


Figura 4.1: Ejemplo del contenedor de gas de Suc y Bratko [70] donde la temperatura, volumen y presión se representan como puntos en el espacio de variables. Las flechas denotan vectores de cambio cualitativo respecto al punto dentro del círculo.

4.1.2. Arbol de decisión de recompensa

En forma similar a como se representa la función de recompensa en un MDP factorizado, las restricciones cualitativas o discretas de un dominio que dis-

tinguen zonas con valores de recompensa distintos pueden representarse bien mediante árboles de decisión (*ADR*). Los nodos internos en éstos árboles representan variables continuas o discretas, mientras que los arcos determinan su valor. En el caso de variables continuas, los arcos evalúan si el valor de la variable es menor o igual (*neg*) o mayor (*pos*) a un límite particular, mientras que con variables discretas los arcos evalúan directamente su valor discreto. Las hojas del árbol de decisión identifican el valor de la recompensa de la región restringida cualitativa o discretamente.

Para ilustrar como representar la función de recompensa con un *ADR* retomemos el ejemplo ilustrado anteriormente y supongamos ahora que, en virtud de que la presión es una función de la temperatura y el volúmen, el sistema del gas ideal se reduce a éstas dos últimas variables. Usemos la variable presión sólo para recompensar con 300 (R_0) puntos a aquellas regiones cuyo punto central (centroide) tenga un valor moderado, castigar con -200 (R_1) y -300 (R_2) puntos a las regiones con un valor alto, y castigar también pero con -150 (R_3) a las regiones de bajo valor. A la izquierda de la figura 4.2 se muestra un posible esquema de recompensa para regiones no uniformes en el sistema del gas ideal, y a la derecha el árbol de decisión de recompensa (*ADR*) representando el problema mediante restricciones cualitativas.

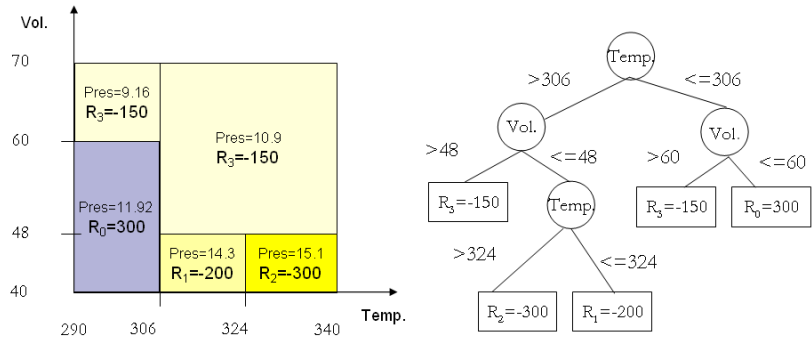


Figura 4.2: Diagrama Temperatura-Volumén para el problema del gas ideal y su sistema de recompensa. (Izquierda) Sistema de recompensa que premia a regiones cuyo centroide tienen una presión moderada, y que castiga a regiones cuyos centroides tienen presiones altas o bajas. (Derecha) Arbol de decisión de recompensa representando el problema con restricciones cualitativas.

4.1.3. Estados cualitativos

Un estado cualitativo, o *estado-q*, es un conjunto de puntos en el espacio n -dimensional con recompensas inmediatas similares. Un espacio de estados cualitativos Q , también llamado *partición cualitativa*, es un conjunto de estados- q : q_1, q_2, \dots, q_n . La partición cualitativa Q se representa mediante un árbol

de decisión llamado *árbol-Q*, el cual divide el espacio de estados de acuerdo con un conjunto de restricciones para las variables relevantes de la función de recompensa.

Similarmente a un *ADR*, cada rama del *árbol-Q* denota un conjunto de restricciones que limita una región en el espacio de las variables para conformar cada estado- q , q_i . El lado derecho de la figura 4.3 ilustra como las restricciones cualitativas $Temp > 306$ y $Vol > 48$ producen el estado- q q_0 que, en el dominio del gas ideal comentado anteriormente, abstrae el conjunto de estados que comparte el valor de recompensa $R_3 = -150$.

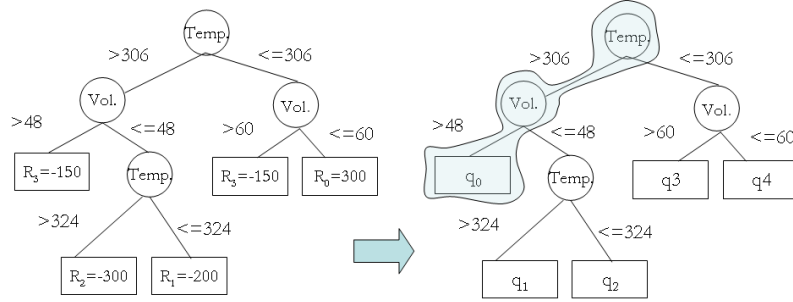


Figura 4.3: Transformación de un árbol de decisión de recompensa, *ADR*, en un árbol- Q . (izquierda) Arbol de decisión de recompensa, *ADR*. (derecha) Arbol- Q . Los nodos hoja en el *ADR* representan recompensas, y en el árbol- Q los estados- q .

Un árbol de decisión de la recompensa, *ADR*, se puede transformar en un árbol- Q simplemente renombrando sus nodos de recompensa con etiquetas de cada estado- q . Cada hoja en el árbol- Q se etiqueta con un nuevo estado cualitativo, por lo que aún en hojas con el mismo valor de recompensa se debe asignar un identificador de estado cualitativo diferente. Aunque esto produce un mayor número de estados, a la vez crea mayor guía para producir políticas más adecuadas. La figura 4.3 ilustra esta transformación para el caso bidimensional del gas ideal. La parte izquierda muestra el árbol de decisión de recompensa *ADR* el cual, después de reetiquetar sus hojas, se transforma en el *árbol-Q* de la derecha. Los nuevos estados cualitativos son q_0, q_1, q_2, q_3 y q_4 , de los cuales, por ejemplo, q_0 , y q_3 tienen el mismo valor de recompensa $R_3 = -150$ aunque bajo diferentes regiones restringidas cualitativamente.

Desde el punto de vista sintáctico, un *árbol Q* es un tipo de dato abstracto, Q , que puede tomar los valores q_1, q_2, \dots, q_n , y los cuales se encuentran identificados por las hojas del árbol. Para su análisis y mejor comprensión, la partición cualitativa Q puede expresarse a su vez mediante hipercubos n -dimensionales denotando regiones con valores de recompensa inmediata posiblemente diferente. Para fines prácticos, se asume que cuando alguna variable no tiene restricciones (el límite inferior o superior de una dimensión es ∞) los límites en el hipercubo

se establecen de acuerdo a los valores de frontera del dominio. La figura 4.4 muestra un ejemplo de un árbol Q y su representación plana en 2D. Dado que el límite superior de ambas variables es ∞ , los límites son establecidos por los rangos máximos de las variables de estado.

Las mayores ventajas de esta representación radican en que la partición Q permite reducir la dimensionalidad de un dominio al tiempo que trata a las variables continuas usando restricciones cualitativas. Si no existen variables continuas en un problema, sólo se reduce la dimensionalidad, sin embargo siempre se obtiene una reducción en el espacio de estados respecto a la especificación original.

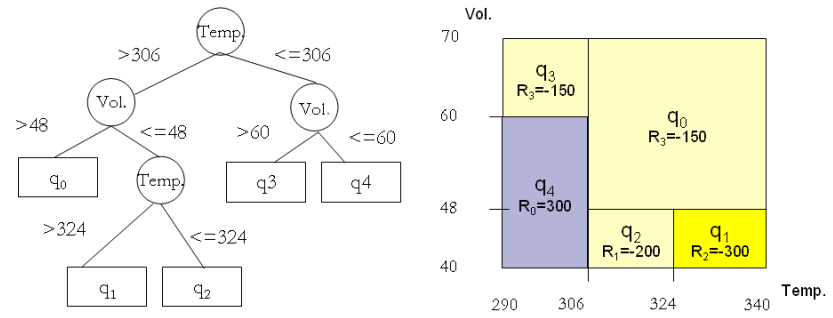


Figura 4.4: Arbol Q y su representación en 2 dimensiones.

4.2. Formalización

Un MDP cualitativo se puede definir como un MDP factorizado conteniendo, entre sus factores, a las variables discretas $X_0, X_1, \dots \in X$, y a la partición de estados cualitativos Q (ver sección 4.1.3). La idea es substituir las variables de la función de recompensa (continuas o híbridas continuas–discretas) por el factor Q resultante de la abstracción, y así reducir la dimensionalidad del problema. Formalmente, en un MDP cualitativo el conjunto de estados es descrito por un conjunto de variables aleatorias $\mathbf{S}_h = \{X_1, \dots, X_n, Q\}$, donde X_1, \dots, X_n son las variables discretas del dominio que no pertenecen a la función de recompensa, y Q es una abstracción que resume las dimensiones relevantes de la función de recompensa. Un estado \mathbf{s}_h define un valor $\mathbf{s}_i \in Dom(X_i, Q)$ por cada variable discreta X_i y el factor Q .

Aun cuando la partición de espacio de estados, Q , abstrae algunas dimensiones de un dominio, el conjunto de estados $\mathbf{S}_h = Dom(X_i, Q)$ puede ser todavía suficientemente grande como para representar explícitamente todas las posibles transiciones entre estados cualitativos debidas a una acción. Por esta razón, y al igual que en un MDP factorizado discreto, la función de transición

de un MDP cualitativo se representa mediante $|A|$ redes Bayesianas dinámicas de dos etapas, donde A es el espacio de acciones del dominio a_1, a_2, \dots, a_m (figura 4.5 izquierda).

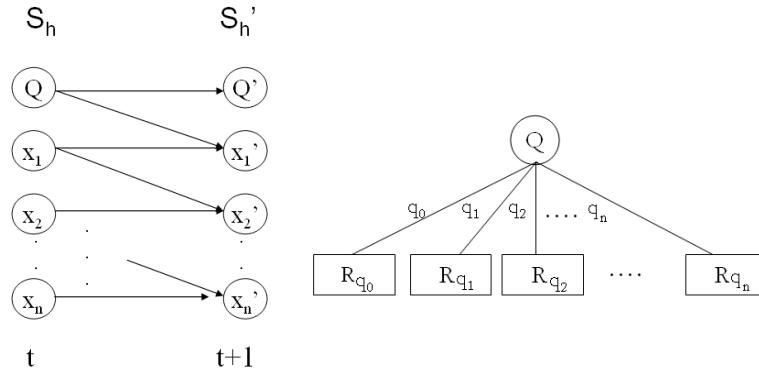


Figura 4.5: Modelo de transición cualitativo y función de recompensa cualitativa. (Izquierda) Red Bayesiana dinámica de 2 etapas representando el modelo de transición cualitativo para la acción a . (Derecha) Función de recompensa estructurada en forma de un árbol de decisión de dos niveles.

Bajo esta representación y debido a que la partición de estados es una función directa de la recompensa $Q = f(R)$, la única variable ó factor de la función de recompensa es el factor Q . Por lo tanto, en un MDP cualitativo, la función de recompensa se representa mediante un árbol de decisión de dos niveles semejante al ilustrado en la figura 4.5 (derecha). En la figura se puede observar que la raíz del árbol corresponde al factor Q , los arcos denotan sus posibles valores q_0, q_1, \dots, q_n , y las hojas identifican los valores de recompensa $R_{q_0}, R_{q_1}, \dots, R_{q_n}$.

Dependiendo del conjunto de variables del dominio que conformen la partición cualitativa Q , un MDP puede ser cualitativo puro o híbrido cualitativo–discreto. Un MDP *cualitativo puro* es aquel donde todas las variables del dominio son términos de la función de recompensa, y por lo tanto todas aparecerán en el árbol Q . De esta forma, en el modelo de transición sólo existirá un factor Q afectado por la recompensa, e influenciado por las n acciones en el tiempo $t + 1$. Un MDP cualitativo puro se puede describir gráficamente y en forma más concisa con un diagrama de influencia de dos etapas, tal y como se ilustra a la izquierda de la figura 4.6. Un ejemplo de un MDP cualitativo puro es el caso de un problema de navegación en un ambiente bidimensional (x, y) donde un agente, que puede desplazarse mediante movimientos sobre las coordenadas x o y , debe alcanzar ciertas localidades (metas) a lo largo de diferentes partes del ambiente. En este problema, el sistema de recompensa depende de la localización en el plano $x - y$ y por tanto todas las variables del dominio, (x, y) , forman parte de la función de recompensa.

Un MDP *híbrido cualitativo–discreto* es aquel donde sólo un subconjunto de las literales del dominio son términos de la función de recompensa. La figura 4.6

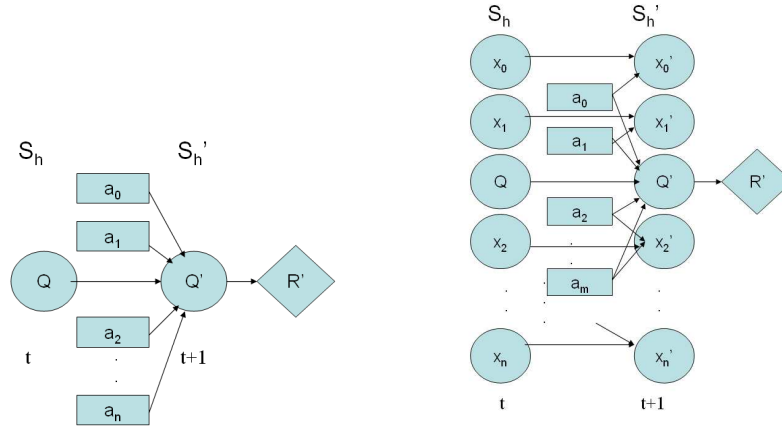


Figura 4.6: Diagramas de influencia representando un MDP cualitativo. (Izquierda) MDP cualitativo puro. (Derecha) MDP cualitativo híbrido.

(derecha) muestra con un diagrama de influencia las variables del dominio, la transición de estados, la influencia de las m acciones sobre los nodos de post-acción (en el tiempo $t + 1$), y la relación de la recompensa con el factor Q en un MDP cualitativo híbrido. Un ejemplo de un MDP cualitativo híbrido sería que el mismo agente del problema de navegación anterior contara con una variable extra, θ , denotando su orientación. Sin embargo, si conservamos el mismo esquema de recompensa anterior, el valor de recompensa inmediata resulta independiente de la orientación. Al generar la abstracción las nuevas variables serían Q y θ .

Al igual que un MDP factorizado discreto, la solución de un MDP cualitativo híbrido cualitativo–discreto es una política sobre los estados formados por la instanciación del conjunto de variables $\mathbf{S}_h = \{X_1, \dots, X_n, Q\}$. La solución de un MDP cualitativo puro es una política sobre los estados abstractos q . En ambos casos, esta solución puede obtenerse mediante algoritmos estándar de programación dinámica como iteración de valor o iteración de política.

En la siguiente sección describiremos una técnica práctica y relativamente simple de generar la abstracción y automatizar la construcción de estos modelos con base en técnicas de aprendizaje automático. Por su parte, en la sección 4.4 mostraremos como un MDP *cualitativo puro* se puede refinar para mejorar la aproximación de la función de valor y la función de política.

4.3. Aprendizaje

En la práctica, y particularmente en dominios complejos, la especificación manual de estados abstractos, y sus funciones cualitativas de recompensa y transición se puede tornar impráctica e imprecisa. Afortunadamente, los algoritmos de aprendizaje permiten la especificación automática de éstos modelos a partir

de la observación de un sistema. En esta sección, se presenta un método de inducción de MDPs cualitativos basado en aprendizaje de árboles de decisión y redes Bayesianas para conformar un modelo de decisión cualitativo que se puede resolver con técnicas estándar de programación dinámica. El método consta de las etapas siguientes: recolección de datos mediante exploración, abstracción de datos, e inducción de los modelos de transición y recompensa.

4.3.1. Recolección de datos (exploración)

La primera fase del proceso de inducción de un MDP cualitativo consiste en coleccionar datos del sistema asumiendo que un agente puede explorar su espacio de estados, y que el sistema lo retroalimenta con alguna recompensa inmediata positiva (premio) o negativa (penalización). Por ejemplo, mientras un robot navega en el mundo puede registrar simultáneamente datos históricos sobre su posición, orientación, acciones efectuadas y estados con distintos grados de preferencia. En aplicaciones de control industrial, un agente puede usar datos históricos de operación, o explorar el proceso con un simulador. En cualquier caso, lo que se desea es registrar los valores de las variables, acciones o comandos posibles, y la recompensa en cada instante del proceso. Más formalmente, lo que se busca es obtener mediante una exploración aleatoria no exhaustiva un conjunto de observaciones del sistema en n instantes de la forma $O^t = \{\mathbf{X}_h^t, a, R^t\}$ donde el estado $\mathbf{X}_h^t = \mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_m$ es una instancia del conjunto de variables discretas o continuas (atributos) $X_h = \{X_1, \dots, X_m\}$ en el instante t , a es la acción ejecutada en el tiempo t , y R^t es el valor de recompensa inmediata (o costo) asociada a cada observación.

Tabla 4.1: Datos de exploración para el problema del gas ideal.

Temp	Vol.	Acción	Recompensa
290	40	T+	300
305	40	V+	300
305	49	null	300
305	49	V+	300
305	56	T+	300
322	56	T+	-150
339	56	null	-150
339	56	V+	-150
339	63	T-	-150
320	63	T-	-150
303	63	null	-150
303	63	T+	-150
321	63	V-	-150
321	56	V-	-150
321	47	T+	-200
338	47	null	-300
338	47	V+	-300
338	57	V-	-150
338	47	T-	-300
322	47	T-	-200

El proceso de recolección de datos mediante exploración contempla la selección de la acción y su magnitud, la selección del número de muestras, y como

distribuir la exploración a lo largo del ambiente.

Para poder conformar un modelo de transición completo (para todo el espacio de acciones), y lograr estructurar la función de recompensa se debe probar cada acción del espacio de acciones en una parte representativa del espacio de estados. La selección de la acción debe ser lo suficientemente aleatoria para asegurar tener, por cada acción, conjuntos de aproximadamente el mismo tamaño.

La selección inadecuada de la magnitud o intensidad de las acciones, puede dar lugar tanto a oscilaciones en una misma región recompensada, como a pasar por alto pequeñas zonas con el mismo valor de recompensa. Al conformar los modelos de transición de estados con estos datos, la capacidad de predictibilidad de los efectos de una acción se pierde y por tanto estos modelos carecen de fidelidad. Una heurística que ofrece buenos resultados es calcular la magnitud de la acción exploratoria en función del tamaño de la región recompensada más pequeña. La idea es que la intensidad de la acción haga transitar suficientemente al agente a lo largo de todas sus dimensiones.

Una mala selección del número de muestras puede traducirse en modelos poco confiables. Por una parte, se pueden pasar por alto ciertas zonas de recompensa, y por otra se pueden omitir los efectos de las acciones en ciertos estados. Una medida segura es realizar varios experimentos *a priori* con distinto número de ejemplos y generar una curva de aprendizaje que muestre la calidad de la solución y la variabilidad del número de estados conforme se incrementa el número de ejemplos. Una buena heurística para la selección del mejor modelo (más estable) es usar aquél cuya varianza en número de estados y política sea inferior al 10 %.

En muchos problemas, una exploración aleatoria suficiente asegura la cobertura del espacio de pares estado-acción del dominio. Sin embargo, en donde la dinámica de la exploración conduce a estados inexistentes, peligrosos o imposibles se prefiere la exploración por zonas. La idea es garantizar la utilidad y usabilidad de los modelos resultantes. Por ejemplo, la recolección de demasiados casos a los que difícilmente el agente se enfrentará podría omitir el registro de datos realmente representativos del dominio. La selección inadecuada de la distribución de muestras puede traer como consecuencia sobreajuste (*overfitting*) local y sesgos en los resultados. Por su parte, cuando los datos de entrenamiento son incompletos, el algoritmo de aprendizaje Bayesiano puede ser incapaz de relacionar causalmente las variables, y por tanto no encontrar ninguna estructura de RBD para formar el modelo de transición de estados.

Para ilustrar la forma en que los datos de exploración son registrados, considere nuevamente el ejemplo del gas ideal pero suponiendo ahora que el estado del sistema puede ser alterado mediante la adición de calor y compresión para, con cierta intensidad, controlar respectivamente los valores de la temperatura y el volumen. De esta manera, las posibles acciones podrían ser aumentar y decrementar la temperatura en 16 unidades ($T + / -$), incrementar y decrementar el volumen en 8 unidades ($V + / -$), y la acción nula (*null*). Los valores de la intensidad son seleccionados de acuerdo con la estructura de la función de recompensa, la cual se ilustra en la sección 4.1.2. La tabla 4.1 corresponde a un pequeño conjunto de datos para el ejemplo del gas ideal que muestra la

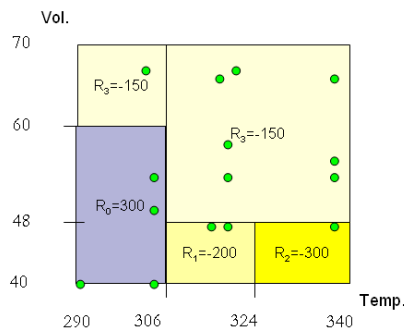


Figura 4.7: Rastro de exploración para el ejemplo del gas ideal.

recompensa de cada estado registrado, y los efectos no deterministas de las acciones. Por su parte, la figura 4.7, muestra gráficamente el rastro trazado por la estrategia de exploración para este conjunto de datos. En el capítulo 5 se presentan casos de como la exploración puede afectar el aprendizaje de los modelos cualitativos de decisión.

4.3.2. Abstracción de datos

La idea de la abstracción de datos es producir un conjunto de datos abstractos con los cuales aprender posteriormente los modelos cualitativos de transición y recompensa. El procedimiento inicia usando los datos de exploración para construir la función de recompensa en forma de un *ADR*¹. Posteriormente, el *ADR* se transforma para conformar el árbol *Q*, el cual a su vez permite clasificar los datos originales y así generar un nuevo conjunto de datos de menor dimensionalidad.

Inicialmente, con los datos de exploración crudos, el algoritmo de aprendizaje de árboles de decisión *C4.5* [63] induce una estructura, *ADR*, que aproxima la función de recompensa. Este algoritmo está diseñado para: (i) manejar conjuntos de datos con valores continuos y/o discretos, (ii) tratar con conjuntos de datos incompletos, y (iii) simplificar estructuras mediante la poda del árbol. El problema del aprendizaje de árboles de decisión con *C4.5* es encontrar una estructura tal que, con base en respuestas a preguntas sobre sus atributos, pueda predecir correctamente el valor de una clase. Para iniciar este proceso, se requiere de un conjunto representativo de ejemplos, $D = \{X_h, R\}$, tal que las variables de estado X_h sean los atributos y el valor de recompensa inmediata R la clase. Excluyendo la columna de acciones, una fracción del conjunto de datos de entrenamiento para formar el *ADR* en el dominio del gas ideal se encuentra en la tabla 4.1. La estructura resultante de *ADR* es similar al mostrado en la

¹Es posible aproximar una función de recompensa con una Red Neural Artificial [53, 39] usando los datos de exploración. Sin embargo, para los fines de este trabajo, las restricciones cualitativas del problema se extraen en forma más natural a partir de un árbol de decisión.

Tabla 4.2: Datos resultantes del proceso de abstracción para el ejemplo del gas ideal.

Q	Acción	Recompensa
q_4	T+	300
q_4	V+	300
q_4	null	300
q_4	V+	300
q_4	T+	300
q_0	T+	-150
q_0	null	-150
q_0	V+	-150
q_0	T-	-150
q_0	T-	-150
q_3	null	-150
q_3	T+	-150
q_0	V-	-150
q_0	V-	-150
q_2	T+	-200
q_1	null	-300
q_1	V+	-300
q_0	V-	-150
q_1	T-	-300
q_2	T-	-200

figura 4.2.

Posteriormente, el árbol ADR se transforma en una partición del espacio de datos o *árbol* Q , de modo tal que cada región $q_i \in Q$ tenga la misma recompensa inmediata. Finalmente, los datos de entrenamiento originales son filtrados a través de las restricciones impuestas por el *árbol* Q para generar un tipo de datos abstractos Q con valores finitos q_1, q_2, \dots, q_n . Para terminar de conformar el nuevo conjunto de datos abstractos, las variables continuas del dominio no incluidas en el árbol Q se discretizan de acuerdo al problema. La figura 4.3 muestra la transformación del ADR para el problema del gas ideal, y la tabla 4.2 los datos abstractos generados al clasificar los datos crudos originales con el *árbol* Q .

En resumen, el algoritmo de abstracción procede de la siguiente manera: Dado un conjunto de observaciones del sistema en n instantes t de la forma $O^t = \{\mathbf{X}_h^t, a, R^t\}$, hacer:

1. A partir de un conjunto de datos $D \subset O$ de la forma $\{X_h, R\}$ inducir con el algoritmo C4.5, un árbol de decisión ADR que prediga la función de recompensa R_h .
2. Reetiquetar las hojas del árbol de decisión de recompensa ADR con un identificador consecutivo q_i desde $i = 1$ hasta N =número de hojas para conformar el *árbol* Q .
3. Utilizar el *árbol* Q para cualificar los datos del conjunto original O de tal manera que el nuevo conjunto de atributos O_q sean: la variable abstracta Q , las variables no incluidas en el *árbol* Q , la acción a y el valor de recom-

pensa inmediata R . Las variables continuas sobrantes son discretizadas en intervalos uniformes de acuerdo al conocimiento del dominio.

La siguiente etapa consiste en procesar los datos cualificados, O_q , con el algoritmo de inducción de MDPs factorizados para obtener la especificación de un modelo de MDP cualitativo.

4.3.3. Inducción de un MDP cualitativo

En esta fase la idea es, a partir del conjunto de datos cualificados, construir en forma automática los modelos cualitativos de recompensa y transición; concentrar en una especificación estos modelos y los parámetros del MDP (como factor de descuento y error de convergencia); y resolver el MDP abstracto con métodos de programación dinámica estándar. Para inducir los modelos cualitativos de recompensa y transición de estados se usan procedimientos de aprendizaje automático tales como $C4.5$ y $K2$, y el procedimiento de iteración de valor como método de inferencia.

El proceso inicia aproximando, con el algoritmo $C4.5$ (explicado en la sección 4.3.2) y los datos cualificados O_q , un árbol de decisión de recompensa de dos niveles semejante al mostrado en la figura 4.5 (derecha). Después, se induce el modelo de transición mediante el algoritmo para el aprendizaje de redes Bayesianas, $K2$ [17]. $K2$ es un algoritmo de búsqueda ambiciosa basado en un sistema de puntuación Bayesiana para ordenar diferentes estructuras de red (Bs) resultantes de un conjunto de datos completos, D . Su método de búsqueda trata de maximizar la expresión $P(Bs|D)$, esto es la probabilidad de la estructura (Bs) dados los datos (D). La entrada a $K2$ está conformada por: (i) un conjunto de nodos (atributos), (ii) un orden de los nodos, (iii) un límite superior u sobre el número de padres que un nodo puede tener, y (iv) una base de datos D conteniendo m casos de datos de entrenamiento. El orden de los nodos se refiere a la forma en que un experto arregla secuencialmente las variables de acuerdo con su conocimiento del dominio, y los requisitos del algoritmo de aprendizaje. El límite superior u de padres de un nodo también permite controlar la complejidad de la inferencia probabilística de acuerdo a la interconexión entre nodos. $K2$ devuelve por cada nodo, sus padres más probables dado los datos de entrenamiento D . Aun cuando originalmente fue diseñado para inducir redes Bayesianas estáticas, es posible habilitarlo para aprender modelos dinámicos de al menos dos etapas arreglando los datos de modo que las variables de estado en el tiempo t antecedan las variables en el tiempo $t + 1$, y formando subconjuntos de datos por acción. Un ejemplo de este arreglo para el ejemplo del gas ideal se muestra en la tabla 4.3. La tabla 4.4, por su parte, ilustra los subconjuntos de transiciones separados por acción para el mismo problema.

Más formalmente, el método general basado en $K2$ para la inducción del modelo de transición cualitativo procede como sigue:

Dado un conjunto de datos O_q denotando las transiciones entre estados cualitativos en cierto período de tiempo, hacer lo siguiente:

1. Arreglar el conjunto de datos O_q de tal manera que los atributos sigan un orden causal temporal y que expresen transiciones de la forma $T = \{X^t, X^{t+1}, a\}$. Por ejemplo, la variable X_0^t debe estar antes que X_0^{t+1} , X_1^t antes de X_1^{t+1} , etc. El total de atributos deben ser las variables del sistema en el tiempo t , las variables del sistema en el tiempo $t + 1$, y la acción $a \in A$.
2. Preparar los datos para la inducción de una red Bayesiana de 2 etapas por cada acción $a \in A$. Para tal efecto, formar $dimAcc = |A|$ subconjuntos de datos $\subset T$ para los casos donde se aplicó cada acción, y eliminar a A como atributo.
3. Mediante el algoritmo *K2* inducir, a partir de cada subconjunto de datos $T_1, T_2, \dots, T_{dimAcc}$, la estructura y parámetros de una RBD por cada acción. Este conjunto de modelos probabilísticos representa la función de transición de estados del sistema.

Tabla 4.3: Arreglo de datos cualitativos en secuencia temporal para el ejemplo del gas ideal.

Q	Q'	Acción
q_4	q_4	T+
q_4	q_4	V+
q_4	q_4	null
q_4	q_4	V+
q_4	q_0	T+
q_0	q_0	T+
q_0	q_0	null
q_0	q_0	V+
q_0	q_3	T-
q_0	q_3	T-
q_3	q_0	null
q_3	q_0	T+
q_0	q_2	V-
q_0	q_1	V-
q_2	q_1	T+
q_1	q_0	null
q_1	q_4	V+
q_0	q_1	V-
q_1	q_2	T-
q_2	q	T-

Después de aproximar las funciones de transición y recompensa, la especificación de MDP se resume en el siguiente orden: (i) variables de estado, (ii) tablas de probabilidades condicionales (*CPTs*) de los nodos de post-acción de cada RBD (una por acción), (iii) árbol de decisión representando la función de recompensa, (iv) tolerancia del error ϵ , y (v) factor de descuento γ . Dicha especificación puede resolverse con una técnica de programación dinámica estándar, tal como iteración de valor, para obtener la política óptima.

La figura 4.8 muestra una posible política óptima y los valores de utilidad obtenidos para el ejemplo del gas ideal después de aplicar iteración de valor

Tabla 4.4: Subconjuntos de datos abstractos por acción para el problema del gas ideal. A partir del conjunto de datos abstractos se conforman tantos subconjuntos como número de acciones del dominio. Hay que aclarar que, para fines de entrenamiento con $K2$, el atributo A se debe eliminar.

A	Q	Q'
$T+$	q_4	q_4
	q_4	q_0
	q_3	q_0
	q_2	q_1
$T-$	q_0	q_0
	q_0	q_3
	q_1	q_2
$V+$	q_4	q_4
	q_4	q_4
	q_0	q_0
	q_1	q_0
$V-$	q_0	q_0
	q_0	q_2
	q_0	q_1
$nula$	q_4	q_4
	q_0	q_0
	q_3	q_3
	q_1	q_1

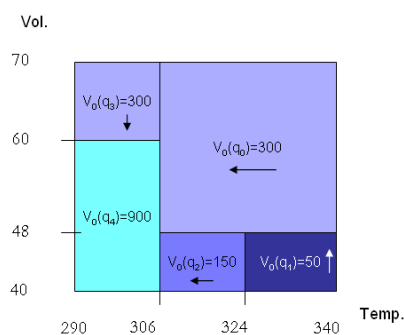


Figura 4.8: Posible política y valores de utilidad esperada para el ejemplo del gas ideal.

sobre una especificación automática basada en aprendizaje. En la figura se puede apreciar que las acciones en cada estado cualitativo buscan transitar a través de las zonas con mayor utilidad para colocar al sistema en la región meta con la presión más estable (q_4).

Una ventaja con esta metodología es que, omitiendo la etapa de abstracción de datos, puede usarse directamente para inducir MDPs factorizados discretos convencionales. Esta característica se debe a que un MDP abstracto es en cierta forma un MDP con una discretización no uniforme de los estados, producto de un proceso de agregación, de acuerdo a los valores de recompensa del problema. De alguna manera, esto ofrece una alternativa para la solución de problemas

complejos donde las técnicas de aprendizaje por refuerzo [71] requieren un gran número de iteraciones antes de converger a una política razonable. Algunos resultados de aplicar esta técnica en dominios discretos se comentan en la sección 5.1.1.

4.4. Refinamiento de estados cualitativos

En cierto tipo de dominios, la abstracción inicial puede ser suficiente para resolver problemas de planificación, sin embargo en otros puede carecer de detalles relevantes y consecuentemente producir políticas poco satisfactorias. Una forma de superar este problema es refinar la partición inicial, dividiendo estados donde mejor se contribuya a la solución. En esta sección, presentamos un procedimiento basado en utilidad para el refinamiento de estados cualitativos puros.

Entre las nociones más importantes del método se encuentran los conceptos de vecindad y región; y las funciones de varianza, gradiente y tamaño un estado. La *frontera* de un estado q se define como el conjunto de *estados- q* colindantes con una o varias de sus d dimensiones. De aquí, una *región* es el conjunto de *estados- q* conformado por q y sus n *estados- q* frontera.

La varianza de la utilidad en una región se define como:

$$S_{region}^2 = \frac{1}{n+1} \sum_{i=1}^{n+1} (v_{q_i} - \bar{v}_{region})^2 \quad (4.1)$$

donde: n es el número de estados frontera, v_{q_i} es la utilidad en cada estado i de la región, y \bar{v}_{region} es la utilidad media en la región.

El gradiente de utilidad es la diferencia entre la utilidad de un estado- q respecto a la utilidad de un vecino q_j , y se expresa formalmente con la siguiente ecuación:

$$\delta v = |v_q - v_{q_j}| \quad (4.2)$$

El tamaño o espacio que ocupa un *estado- q* se conoce como hipervolumen. El hipervolumen es definido como el producto del valor de sus d dimensiones mediante la siguiente expresión:

$$hv_q = \prod_{k=1}^d x_k \quad (4.3)$$

donde x_k es el valor de cada dimensión k de un estado q .

El términos generales, el refinamiento de estados cualitativos es un método ambicioso (*greedy*) para MDPs cualitativos puros que selecciona y particiona recursivamente estados abstractos q con alta varianza de utilidad esperada respecto a sus estados vecinos. El método, cuyo objetivo es obtener mejores aproximaciones de las funciones de valor y de política, procede como sigue:

Dada la partición Q_0 , la aproximación inicial de la función de valor $V_0(Q_0)$, la aproximación de la función de política $\pi_0(Q_0)$ de un MDP cualitativo, y mientras exista un estado cualitativo $q \in Q$ no marcado mayor que el tamaño mínimo ² hacer lo siguiente:

1. Guardar una copia del MDP previo (antes de la partición) y de su solución.
2. Seleccionar el estado cualitativo con la varianza más alta en su valor de utilidad respecto a sus vecinos.
3. Del conjunto de variables $\{X_0, X_1, \dots, X_n\}$, seleccionar una dimensión continua del estado q del paso anterior sobre la cual efectuar la división. Seleccionar aquella con mayor diferencia en utilidad de los estados frontera.
4. Bisectar (dividir en dos partes iguales) el estado q sobre la dimensión seleccionada.
5. Resolver el nuevo MDP.
6. Si la política del MDP nuevo se mantiene igual en los nuevos estados producto de la bisección, regresar al MDP previo, y marcar el estado q original antes de la partición para evitar dividirlo otra vez. De otra manera, aceptar el refinamiento y continuar.

El tamaño mínimo de un estado- q normalmente es una función de la región más pequeña ocupada por grupos de puntos con la misma recompensa, y la aproximación de un hipervolumen basado en la magnitud de la acción usada durante la etapa de exploración. El pseudo-código del algoritmo general para el refinamiento de estados cualitativos, selección del estado a partir, y de bisección del estado se ilustra en el apéndice C.

Como ejemplo, suponga que en la partición inicial del espacio de estados del problema del gas ideal de la figura 4.9 (esquina superior izquierda), la región formada por q_0 y sus vecinos q_1, q_2, q_3, q_4 tiene la máxima varianza de utilidad de todo el espacio. Si la variable *Volumen* es la dimensión con la mayor diferencia en utilidad en la frontera de q_0 , entonces el estado q_0 se bisecta sobre esta dimensión para producir los nuevos estados q_0 y q_1 (ver paso 1). Los estados restantes deben re-etiquetarse para conservar una numeración progresiva, de modo que el q_1 anterior se vuelve q_2 , el q_2 anterior cambia a q_3 , y así sucesivamente. Después de resolver el nuevo MDP y suponiendo que cambia la política en los nuevos estados q_0 y q_1 , entonces la bisección es aceptada y el proceso se repite sobre esta nueva partición. Asumamos que, ahora el nuevo candidato a partición es el nuevo estado q_1 sobre la variable *Temperatura* produciendo la partición del paso 2. Al resolver con iteración de valor se verifica que no hay cambios en la política y por tanto se regresa a la partición anterior (paso 3), se marca el estado q_1 como evaluado, y se busca otro estado con máxima varianza en la utilidad

²El tamaño mínimo de un estado se define por el usuario y es dependiente del dominio. Por ejemplo, en un dominio cartesiano $x - y - z$, el estado mínimo podría medir $10cm^3$.

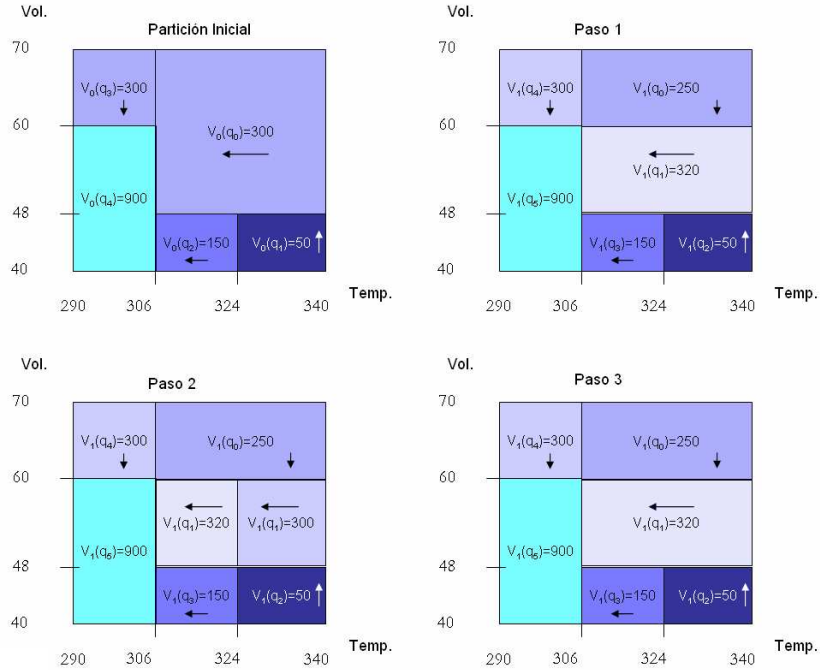


Figura 4.9: Proceso de refinamiento cualitativo para el espacio de estados del problema del gas ideal. En esta etapa, la abstracción inicial (izquierda-arriba) se refina para aproximar mejor la política. Cuando se presentan mejoras en la política (paso 1) la nueva partición se acepta. En casos donde la partición de un estado no aporte a la solución (paso 2), se regresa a la partición anterior inmediata (paso 3).

esperada sobre la partición. En caso de no encontrarlo, el algoritmo termina. La figura 4.9 ilustra estos tres pasos del proceso de refinamiento donde se puede apreciar como la adición del primer corte *suaviza* la trayectoria hacia el estado meta q_2 .

Una ventaja del método es que, dado que evalúa cada solución global para buscar mejoras locales donde se necesitan, minimiza el número de cortes para no complicar nuevamente el problema. Sin embargo, y dado que busca en forma ambiciosa, podría evitar la realización de cortes necesarios para mejorar la solución después de varias etapas. Una forma de resolver este problema es usar algoritmos más sofisticados para permitir profundizar en la búsqueda, o intercalar con cierta periodicidad criterios de búsqueda ambiciosa y no ambiciosa. Por su parte, creemos que si en los primeros ciclos del procedimiento de refinamiento se logran poner en juego las variables relevantes no relacionadas con la recompensa, entonces podremos resolver MDPs cualitativos híbridos. El involucramiento de estas variables en el árbol Q puede realizarse mediante alguna discretización inicial y una medida para la selección de atributos.

4.5. Resumen del capítulo

En este capítulo se describió una estrategia de abstracción alternativa de un MDP para contender con espacios de estados continuos basada en modelos cualitativos. Se definieron los conceptos de estado cualitativo, partición cualitativa, y MDPs cualitativos. Igualmente, se mostró como obtener una partición cualitativa de estados inicial y su solución usando técnicas de aprendizaje automático. Finalmente propusimos una estrategia de refinamiento de la partición inicial a través de la cual lograr mejores aproximaciones tanto de la función de valor como de la política.

Las ventajas representacionales de los MDPs cualitativos radican en que la partición Q permite reducir la dimensionalidad de un dominio al tiempo que trata a las variables continuas usando restricciones cualitativas. Si no existen variables continuas en un problema, sólo se reduce la dimensionalidad, sin embargo, siempre se presentan ahorros computacionales respecto a la especificación original del problema.

Una ventaja del método de aprendizaje es que, a partir de datos puramente discretos y sin la necesidad de la etapa de abstracción de datos, se pueden inducir MDPs factorizados convencionales. Esto de alguna manera ofrece una alternativa para la solución de problemas complejos donde las técnicas de aprendizaje por refuerzo[71] requieren un gran número de iteraciones antes de converger a una política razonable. Un riesgo al tratar de aprender una representación abstracta es que, como consecuencia de una mala exploración, la propiedad de Markov o predictibilidad en el sistema puede llegar a perderse. Las heurísticas aquí propuestas se basan en la selección adecuada de la magnitud de la acción y de la distribución de la muestras en el ambiente.

La mayor ventaja del método de refinamiento es que, dado que evalúa cada solución global para buscar mejoras locales donde se necesitan, minimiza el número de cortes para no complicar nuevamente el problema. Sin embargo, y dado que el algoritmo realiza búsqueda ambiciosa, eventualmente puede abstenerse de partir zonas sin mejora inmediata. Otra desventaja radica en que la etapa de refinamiento, hasta el momento, esta limitada a MDPs cualitativos puros. Para hacer más eficiente este procedimiento, se sugiere usar algoritmos de búsqueda más sofisticados, e involucrar a todas las variables del dominio durante sus etapas iniciales.

En el próximo capítulo, se mostrarán algunos resultados experimentales realizados en dominios diferentes para evaluar, desde distintos aspectos, las bondades del método de abstracción-descomposición aquí expuesto.

Capítulo 5

Resultados experimentales

Para evaluar la técnica propuesta en esta tesis, se realizaron experimentos en dos dominios de aplicación diferentes. Primero, se resolvió un problema de navegación robótica relativamente simple con estados continuos e híbridos continuos-discretos en 2 y 3 dimensiones. El objetivo de las pruebas en este dominio es comparar la solución aproximada de los MDPs cualitativos, en términos de calidad y complejidad, con una discretización muy fina emulando una solución exacta. Posteriormente, se consideró el problema de control de una planta de generación de electricidad con más dimensiones y más acciones para demostrar que el método permite resolver problemas más complejos.

Las experimentos se ejecutaron con tecnología Java [56] utilizando equipo de cómputo con procesadores Pentium IV, y simuladores basados en comunicaciones tipo cliente-servidor, JDBC, y OPC. Con la intención de implementar acciones imperfectas en las simulaciones, la magnitud de los comandos en los dominios de prueba fueron influenciadas por ruido Gaussiano. Esto, además de agregar una dosis de realidad, demuestra la robustez y efectividad del formalismo de los MDPs para generar planes en ambientes inciertos.

Las funciones de recompensa (árboles de decisión) fueron aprendidas usando J48, una implementación en Java del algoritmo C4.5 en la herramienta Weka [76], mientras que los modelos de transición (redes Bayesianas dinámicas) fueron aprendidas usando el algoritmo K2 de *Elvira* [16]. Los modelos generados fueron resueltos usando una herramienta propietaria para la planificación con incertidumbre llamada *SPI*, y el sistema de planificación *SPUDD* [37].

5.1. Navegación robótica

La técnica propuesta en este trabajo fue probada en un dominio de navegación robótica usando un ambiente simulado. Para esto, se dotó a un robot virtual con sensores de posición x-y, orientación angular (θ) y detección de los límites del ambiente de navegación. Los primeros experimentos incluyeron como acciones movimientos ortogonales discretos a la derecha (r), a la izquierda (l),

hacia arriba (u), y hacia abajo (d). En una segunda variante de experimentos, simulamos un robot con las acciones: avanzar (go), rotar en sentido horario (rt), y rotar en sentido anti-horario (lt). Ambas variantes consideraban también la acción nula.

Las metas del problema tienen recompensa inmediata positiva y están representadas como regiones cuadradas en tonos de azul. Las regiones no deseables tienen recompensa negativa y se representan con cuadrados en tonos de amarillo. Las regiones sin recompensa (o recompensa nula) se representan en color negro. Los tonos de azul o amarillo indican que las recompensas y penalizaciones pueden ser multivaluadas. Con la idea de no limitar la función de recompensa a la posición en el plano x-y, en algunos experimentos el robot puede obtener recompensas (o penalizaciones) extras dependiendo de su orientación. En general, el problema de planificación en este dominio consiste en obtener automáticamente una política óptima de movimiento para que el robot alcance sus metas evitando las regiones negativas. A la izquierda de la figura 5.1 se muestra un ejemplo de problema de navegación libre de obstáculos con 26 regiones recompensadas. A la derecha se ilustra un problema donde la meta está rodeada de zonas penalizadas (emulando obstáculos). En ambos ejemplos, la función de recompensa se encuentra en términos de la posición x-y y el robot puede obtener uno de seis posibles valores.

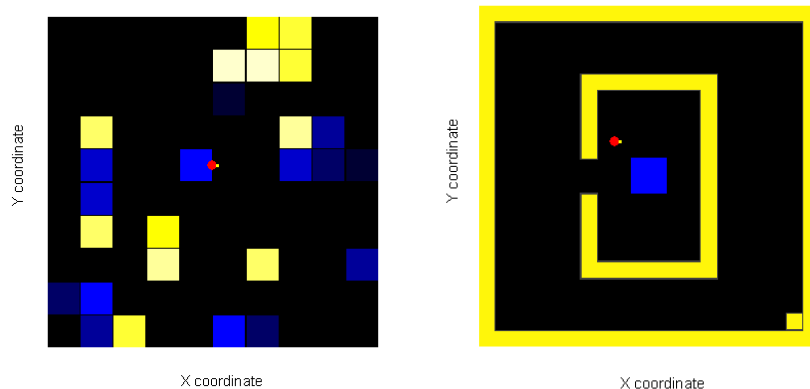


Figura 5.1: (Izquierda) Problema libre de obstáculos con regiones coloreadas que denotan regiones de recompensas positivas y negativas distribuidas aleatoriamente. (Derecha) Problema con regiones de alta recompensa negativa denotando obstáculos, y con una region meta (recompensa positiva) al centro

Para evaluar la técnica de aprendizaje de MDPs factorizados, que es un componente fundamental de esta investigación, primero se resolvieron distintos problemas de navegación robótica con estados discretos en 2 y 3 dimensiones. El énfasis de este conjunto de experimentos está en la descripción de los modelos aprendidos. Posteriormente, y para demostrar la calidad y beneficios de los

métodos de abstracción y refinamiento de MDPs cualitativos, se compararon las soluciones aproximadas de nuestro método contra soluciones finas de mayor complejidad. Durante estas pruebas se usaron problemas con variables continuas e híbridas, continuas-discretas.

5.1.1. Aprendizaje de MDPs factorizados

Para validar el sistema de aprendizaje de modelos factorizados, se realizaron pruebas sobre espacios discretos con distintas configuraciones; esto es, problemas con diferente número de estados y diferente tipo de acciones (rotacionales u ortogonales).

La figura 5.2 muestra uno de los casos de prueba con 16 estados discretos usando un robot rotatorio y una función de recompensa en términos del plano $x-y$. La parte izquierda de la figura muestra el problema con una zona premiada y una penalizada, así como el rastro del robot durante su etapa de exploración. Los modelos de recompensa y transición de estados fueron aproximados usando los datos recolectados y el procedimiento de aprendizaje de MDPs factorizados para datos discretos de la sección 4.3.3. En la parte derecha de la figura 5.2 se ilustra la política resultante y su terminología. Como ejemplo suponga que el agente se encuentra en la posición $x = s1$ y $y = s1$, o simplemente $(s1, s1)$, con un ángulo discreto $s0$ correspondiente a un rango de 0 a 90 grados. En este estado, la política sugiere avanzar al frente con lo cual muy probablemente se situará, con la misma orientación, en la celda $(s2, s2)$. En este nuevo estado la política indica rotar a la derecha hasta alcanzar la orientación $s3$ correspondiente a un rango de 0 a -90 grados. En este nuevo estado la acción óptima es avanzar para que con alta probabilidad el agente se sitúe, con esta misma orientación, en la meta. Para detener al agente, la política indica rotar a la izquierda hasta alcanzar la orientación $s1$ en la meta.

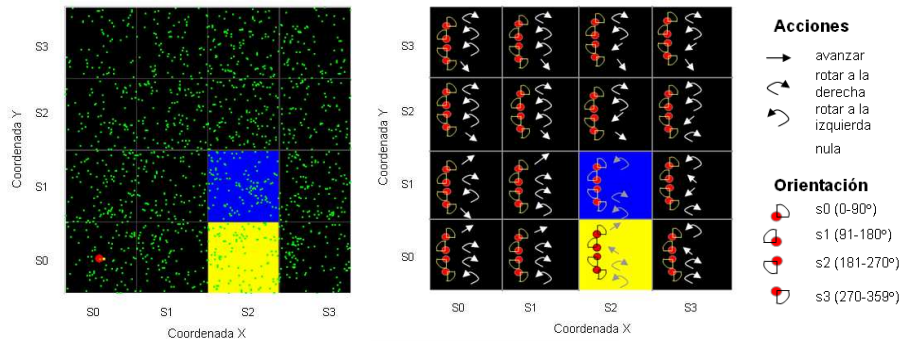


Figura 5.2: Caso de prueba con rastro de exploración para problema con 16 estados discretos, 2 zonas con recompensa, y acciones rotacionales (izquierda), y su solución (derecha).

La figura 5.3 muestra el árbol de decisión resultante de explorar el ambiente

para recopilar 40,000 muestras, y con el que se estructura la función de recompensa del problema anterior. Nótese que la estructura de árbol se expresa en términos de las variables x y y , por lo que si el robot llega a la localidad $(s2, s0)$ entonces recibe una penalización de -300, pero si alcanza la localidad $(s2, s1)$ recibe una recompensa de 300. El resto de las localidades no reciben recompensa y por eso se les asigna un valor de 0 a las hojas.

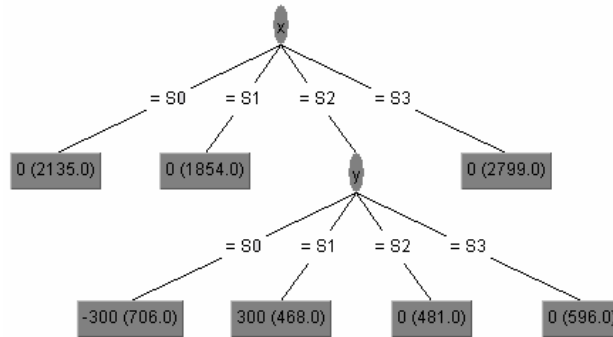


Figura 5.3: Árbol de decisión aprendido con J4.8 de Weka para el ejemplo del robot rotatorio.

La figura 5.4 muestra la red Bayesiana dinámica de la función de transición de estados cuando el robot aplica la acción *avanzar*. Los tres nodos de la izquierda representan las variables de estado al tiempo t , mientras que los de la derecha al tiempo $t + 1$. En este caso, el estado del robot corresponde a la situación en que se encuentre en la localidad $(s1, s0)$ con un ángulo de orientación $s0$ (valor entre 0 y 90 grados). De acuerdo a la figura 5.4, el robot tiene una probabilidad del 53% de alcanzar el estado $(s2, s0)$, y una probabilidad del 47% de terminar en el estado $(s2, s1)$ cuando aplica la acción *avanzar*. Como se puede notar la distribución de probabilidades para la variable ángulo es la misma antes y después de aplicar la acción, lo que significa que el ángulo del agente no cambia en la ejecución de esta acción.

La figura 5.5 por su parte muestra gráficamente las funciones de política y de valor para un caso de prueba con acciones ortogonales (figura 5.1 izquierda). A la izquierda de la figura 5.5 se ilustra la política de acción sobre un mapa de recompensa inmediata. A la derecha se muestra el mapa de la función de utilidad donde las regiones más oscuras representan bajas utilidades y las claras valores más altos. Como se puede apreciar, aun cuando no es totalmente perfecta, la política es satisfactoriamente buena. Por ejemplo, en la columna $x = s1$ y el eje y en el intervalo $[s3, s6]$ se puede ver una región con recompensa positiva rodeado de localidades penalizadas. En este caso, la política guía al sistema hacia las regiones premiadas tratando de evitar las zonas penalizadas.

La figura 5.6 ilustra las funciones de política y de valor para un problema tipo laberinto con acciones ortogonales. A la izquierda se ilustra la política de

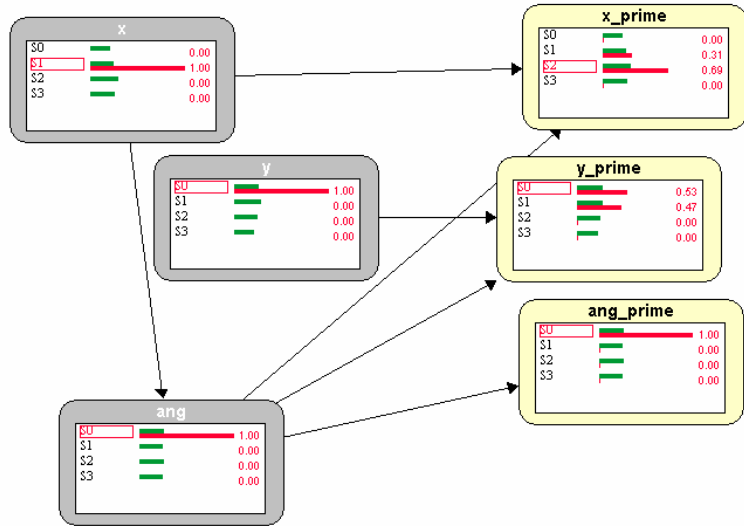


Figura 5.4: Red Bayesiana dinámica aprendida por K2 de Elvira para el ejemplo del robot rotatorio.

acción que busca optimizar el camino a la meta localizada en la parte inferior derecha. A la derecha se muestra el mapa de la función de utilidad donde las regiones blancas, amarillo tenue, y azul oscuro representan gradualmente bajas utilidades. Las zonas en azul denotan valores gradualmente más altos. Con excepción de un par de celdas, la política permite resolver satisfactoriamente problemas con una discretización más fina.

Evaluación del aprendizaje y la inferencia

La tabla 5.1 muestra algunos datos relativos a la evaluación de los procesos de aprendizaje e inferencia de MDPs factorizados para un conjunto seleccionado de experimentos de distintas complejidades. Particularmente, se presenta el número de muestras colectadas en la exploración, el tiempo de aprendizaje e inferencia, y el tamaño del espacio de estados. Los estados enumerados durante la inferencia resultan de multiplicar el número de estados discretos por el número de iteraciones. La primera fila de la tabla muestra los resultados de los experimentos con el robot rotatorio. Las siguientes filas corresponden a los experimentos con movimientos ortogonales. En los experimentos utilizamos un factor de descuento cercano a 1.0 buscando tomar en cuenta la recompensa esperada en el futuro. La literatura sobre MDPs reporta recurrentemente el uso de un valor de 0.9.

Los tiempos de aprendizaje de un modelo factorizado varían desde 2.033 seg. para un problema con 64 estados, hasta 11.697 seg. para casos con 900 estados. Lo cual demuestra que el método de aprendizaje es un proceso razonablemente

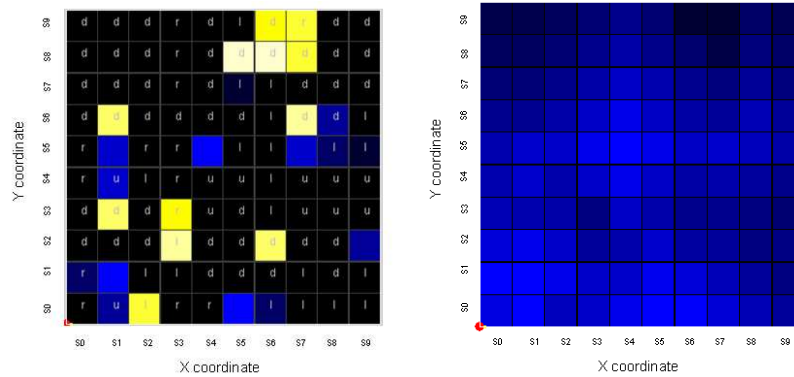


Figura 5.5: Funciones de política y de valor para el problema de la figura 5.1 izquierda. (Izquierda) Política encontrada por el algoritmo de iteración de valor en términos de movimientos ortogonales hacia arriba (u), abajo (d), izquierda (l), derecha (r), y la acción nula $.$ (Derecha) Aproximación de la función de utilidad esperada.

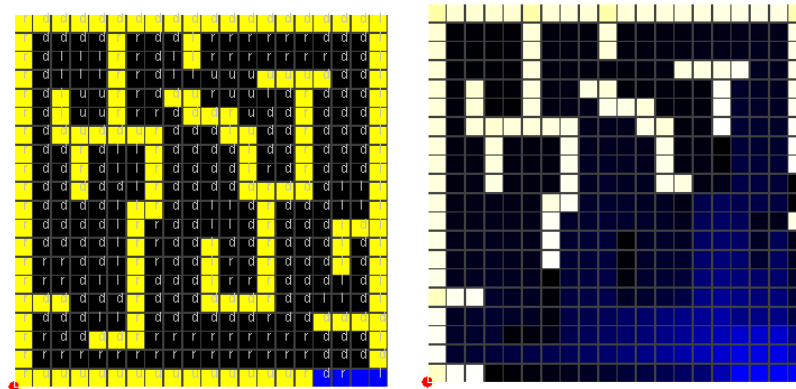


Figura 5.6: Funciones de política y de valor para un problema tipo laberinto. Política encontrada por el algoritmo de iteración de valor en términos de movimientos ortogonales hacia arriba (u), abajo (d), izquierda (l), derecha (r), y la acción nula (izquierda). Aproximación de la función de utilidad esperada (derecha).

Tabla 5.1: Resultados experimentales de aprendizaje de MDPs factorizados.

Recompensas			Aprendizaje			Inferencia		
Num. zonas	tamaño zona (% dim)	Num. Valores	Num. estados	Tiempo (ms)	Num. Muestras	Num. Iteraciones	Estados enumerados	Tiempo (ms)
2	25	3	64	2,033	9,039	120	7,680	81
2	20	3	25	6,940	40,000	120	3,000	40
4	20	5	25	2,945	40,000	123	3,075	30
10	10	3	100	5,017	40,000	120	12,000	100
26	10	11	100	8,152	40,000	124	12,400	120
6	5	3	400	5,418	40,000	120	48,000	1,602
10	5	5	400	4,546	28,868	128	51,200	1,712
12	5	4	400	7,420	29,250	124	49,600	4,016
98	5	8	400	6,239	40,000	116	46,400	1,572
14	3.3	9	900	11,697	50,000	117	105,300	9,734

rápido que se le puede considerar como tratable. Por otra parte, se pudo observar que conforme se usaba un mayor número de muestras, las aproximaciones de la política óptima y la función de valor también mejoraban. Los tiempos de solución del modelo con iteración de valor son estándar y dependen tanto del número de estados enumerados como el procesador utilizado en la prueba. Estos tiempos varían entre 30 mseg. y 9.73 seg.

5.1.2. Aprendizaje de MDPs cualitativos

Para demostrar la calidad y beneficios de los métodos de abstracción y refinamiento de MDPs cualitativos, se compararon las soluciones aproximadas de nuestro método contra soluciones finas de mayor complejidad. Durante estas pruebas se usaron problemas con variables continuas e híbridas continuas-discretas.

La figura 5.7 muestra el proceso de aprendizaje de la partición inicial y su posterior refinamiento en uno de los casos de prueba. La parte superior izquierda de la figura muestra un ambiente de navegación continuo con 12 regiones recompensadas, un robot virtual con acciones ortogonales, y un sistema de recompensa tetra valente. En la parte superior derecha se puede apreciar el rastro aleatorio impreso por el sistema de muestreo. Para simular movimientos más realistas del agente y disociar mejor la exploración, se indujo a los efectos de las acciones un 10% de ruido Gaussiano. La parte inferior izquierda de la figura muestra la partición inicial de estados cualitativos y sus correspondientes políticas, mientras que la parte inferior derecha la forma en que el proceso de refinamiento produce una mejor aproximación de la función de política.

Precisión de la política y error de la utilidad esperada

Como métrica de evaluación se han definido dos funciones empíricas que permiten comparar la política y utilidad de los modelos cualitativos respecto a

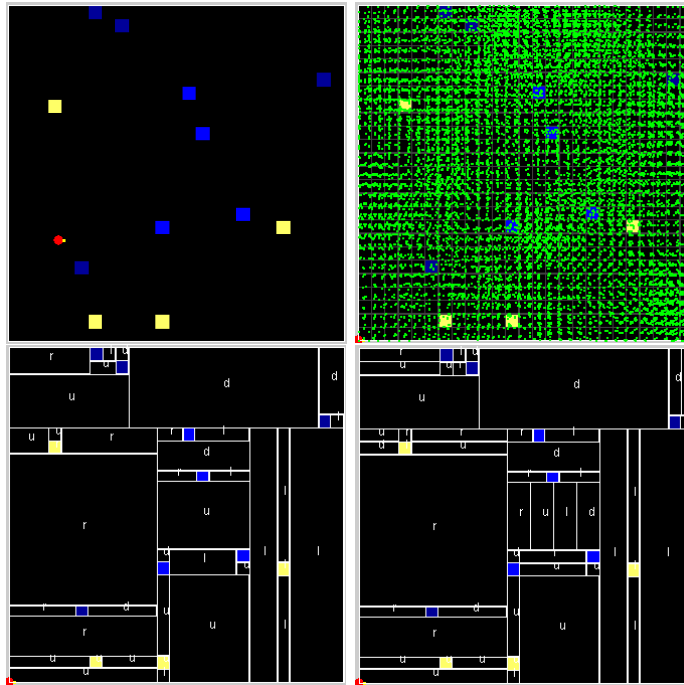


Figura 5.7: Problema de planificación de movimiento y su proceso de solución. Superior izquierda: Área de navegación mostrando la distribución de metas y trampas. El robot simulado se localiza en la esquina inferior izquierda. Superior derecha: Rastro de exploración con acciones ruidosas (no perfectas). Inferior izquierda: Partición inicial. Inferior derecha: Partición refinada.

la solución discreta más fina posible con los recursos de cómputo actualmente disponibles. Estas métricas se conocen como precisión de la política y error de la utilidad.

La *precisión de la política* se puede obtener comparando las políticas generadas con nuestros métodos, contra las políticas obtenidas usando discretizaciones más finas. Para esto, obtenemos la razón del número de celdas *finas* pertenecientes a un estado abstracto donde las políticas son iguales a las de dicho estado, respecto al número total de celdas finas. La medida se puede expresar en forma porcentual de acuerdo a la siguiente ecuación:

$$PP = (NEC/NTC) \times 100,$$

donde PP es el porcentaje de precisión de la política, NEC es el número de celdas finas con la misma política y NTC es el número total de celdas finas. Esta medida puede ser pesimista ya que, aunque pueda haber varias acciones óptimas por cada estado fino ó abstracto con un valor de utilidad muy parecido, para fines de cálculo sólo se considera una de ellas.

El error de utilidad se calcula de la siguiente manera: los valores de utilidad de todos los estados en cada representación se normalizan primeramente. La suma de las diferencias absolutas de los valores de utilidad de los estados correspondientes se evalúan y se promedian sobre todas las diferencias. La siguiente expresión denota el error medio de utilidad en un *estado-q*:

$$EU = \frac{1}{m} \sum_{j=1}^m |v - v_{fine_j}| \quad (5.1)$$

donde m es el número de celdas finas coincidentes con el *estado-q*, v es el valor normalizado de la utilidad en el *estado-q*, y v_{fine} es el valor normalizado de la utilidad en una celda fina coincidente con el *estado-q*.

Evaluación del método de representación

La representación cualitativa y el proceso de refinamiento descrito en el capítulo 4 fue probado con problemas de distintas complejidades en cuanto a tamaño, cantidad y distribución de las recompensas en el ambiente. En los experimentos se usaron casos con acciones rotacionales y ortogonales, y funciones de recompensa de 2D $x-y$, y 3D x, y y θ .

La evaluación consiste primero en comparar la complejidad en espacio y tiempo de los modelos abstractos respecto a versiones discretas finas de los mismos problemas. Asimismo, se califica la calidad de la solución abstracta en términos de política y utilidad esperada. Posteriormente, se evalúa el mejoramiento de la solución abstracta logrado por la etapa de refinamiento. Finalmente, se comparan los tiempos de ejecución de nuestro método contra los tiempos de ejecución del sistema SPUDD.

La tabla 5.2 presenta una comparación entre el comportamiento de siete problemas resueltos con el método de discretización fina y el método de MDPs

Tabla 5.2: Casos de prueba comparando nuestra abstracción respecto a una discretización fina “convencional”.

Recompensas				Discreto				Qualitativo				
id problema	no. zonas	tamaño (% dim)	no. valores	no. muestras	Aprend.		Inferencia		Aprend.		Inferencia	
					no. estados	tiempo (ms)	iteraciones	tiempo (ms)	no. estados	tiempo (ms)	no. iteraciones	tiempo (ms)
1	2	20	3	40,000	25	7,671	120	20	8	2,634	120	20
2	4	20	5	40,000	25	1,763	123	20	13	2,423	122	20
3	10	10	3	40,000	100	4,026	120	80	26	2,503	120	20
4	6	5	3	40,000	400	5,418	120	1,602	24	4,527	120	40
5	10	5	5	28,868	400	3,595	128	2,774	29	2,203	127	60
6	12	5	4	29,250	400	7,351	124	7,921	46	2,163	124	30
7	14	3.3	9	50,000	900	9,223	117	16,784	60	4,296	117	241

qualitativos. Los problemas se encuentran identificados con el número consecutivo de la primera columna. Las siguientes 4 columnas describen las características de cada problema. Por ejemplo, el problema 1 tiene 2 celdas recompensadas con valores de -10 y 10, y un tamaño expresado como una fracción de la dimensión, x o y más grande. Así, si se tiene un área de navegación de 10×10 m., y la celda de recompensa tiene una arista de 20% del ancho o alto, el área de la celda será de $2 \times 2 = 4$ m². Otra característica de este problema es que el modelo de decisión se construyó a partir de 40,000 muestras. La siguiente columna compara los modelos discretos y cualitativos en término del número de estados, del tiempo requerido en el aprendizaje del modelo, y del tiempo y número de iteraciones requeridas para resolver el problema usando el método de inferencia de iteración de valor. Como se puede apreciar, usando nuestro método de abstracción existe una reducción significativa en la complejidad tanto en número de estados como en tiempo de inferencia. Esto es importante, puesto que en dominios donde puede ser difícil o hasta imposible encontrar una solución, se pueden obtener políticas satisfactorias a partir de abstracciones adecuadas.

Tabla 5.3: Resultados comparativos entre la abstracción inicial y la abstracción refinada.

id	Qualitativo		Refinado		
	error utilidad (%)	precisión política (%)	error utilidad (%)	precisión política (%)	tiempo refin. (min)
1	7.38	80	7.38	80	0.33
2	9.03	64	9.03	64	0.247
3	10.68	64	9.45	74	6.3
4	12.65	52	8.82	54.5	6.13
5	7.13	35	5.79	36	1.23
6	11.56	47.2	11.32	46.72	10.31
7	5.78	44.78	5.45	43.89	23.34

La tabla 5.3 presenta una comparación entre las representaciones cualitativas y refinadas para los mismos problemas de la tabla 5.2. Las columnas dos y tres describen las características del modelo cualitativo y las siguientes describen las características después del proceso de refinamiento. Todos los valores se calcularon respecto a una discretización fina del problema. En esta tabla comparativa se demuestra que la abstracción propuesta produce en promedio un 9.17% de error en el valor de utilidad en relación a una versión discreta fina del problema. Lo interesante es que para este conjunto de casos de prueba la representación abstracta es, en número de estados promedio, 11 veces más compacta que la representación fina (tabla 5.2). El proceso de refinamiento por su parte es capaz de mantener o mejorar los valores de utilidad en todos los casos con un promedio de error de utilidad de 8.17%. El porcentaje de precisión de la política con respecto al modelo cualitativo inicial, puede algunas veces decrementarse con el proceso de refinamiento por la medida pesimista de comparación de políticas explicada con anterioridad, sin embargo, la política resultante siempre mantiene en el sistema una conducta orientada a metas.

Tabla 5.4: Resultados experimentales con SPUDD.

SPUDD Repres. fina			Iterac. Valor + Repres. Q			SPUDD + Repres. Q		
Ident.	No. iteraciones	No. nodos ADD accion	Tiempo (seg)	No. iteraciones	Tiempo (seg)	No. Iteraciones	No. nodos ADD accion	Tiempo (seg)
4	209	101	10.11	120	7.393	209	31	1.1
5	218	174	12.27	127	5.191	218	24	0.75
6	213	199	18.84	124	6.235	213	44	1.78
7	209	201	25.91	117	10.07	209	43	1.83

Para verificar la eficiencia con la que diferentes métodos resuelven el mismo problema, se realizó una comparación (tabla 5.4), en tiempo de ejecución, de nuestro método contra el sistema SPUDD [37] (sección 3.6.1). La complejidad del modelo producido por SPUDD se expresa en número de nodos de la función de política, y el tiempo de solución del método en todos los casos se expresa en segundos. Los casos de prueba corresponden a los problemas 4,5,6 y 7 de la tabla 5.3, los cuales fueron seleccionados por contar con el mayor número de estados (400 y 900 estados). Para hacer más justa la comparación, los experimentos se realizaron en la misma computadora y bajo el mismo sistema operativo. Primeramente, se resolvió, con la herramienta SPUDD, una especificación fina de los casos de prueba, obteniéndose un tiempo promedio de 16.78 segundos y un número de nodos promedio de 168. Al ejecutar el algoritmo de iteración de valor usando nuestro sistema de aprendizaje y representación cualitativa (o simplemente Q), éste logro resolver los problemas con un tiempo promedio de 7.22 segundos. Finalmente, el sistema SPUDD se ejecutó con nuestra representación reduciendo el tiempo de ejecución a 1.36 segundos, y la complejidad del modelo a 35 nodos.

Evaluación de la exploración vs calidad de la solución

Con la intención de demostrar la influencia de la exploración en la solución del modelo abstracto, se experimentó 10 veces con el problema 3 de la tabla 5.2 usando conjuntos de datos desde 5,000 a 70,000 ejemplos. Tanto el número de casos como el número de ejemplos por caso fue seleccionado después de dos ciclos de prueba y error. La figura 5.8 muestra los resultados obtenidos. A la izquierda se ilustra una gráfica donde la precisión de la política se incrementa ligeramente conforme se aumenta el número de ejemplos, sin embargo no es sino en el rango de 60,000 a 70,000 ejemplos donde tanto el número de estados y la precisión media de la política se hace estable. A la derecha, el error de utilidad se decrementa gradualmente al aumentar el número de ejemplos. Al igual que la gráfica de la izquierda, la varianza del número de estados y el error de la utilidad para los distintos casos de prueba se estabiliza en el mismo rango de ejemplos.

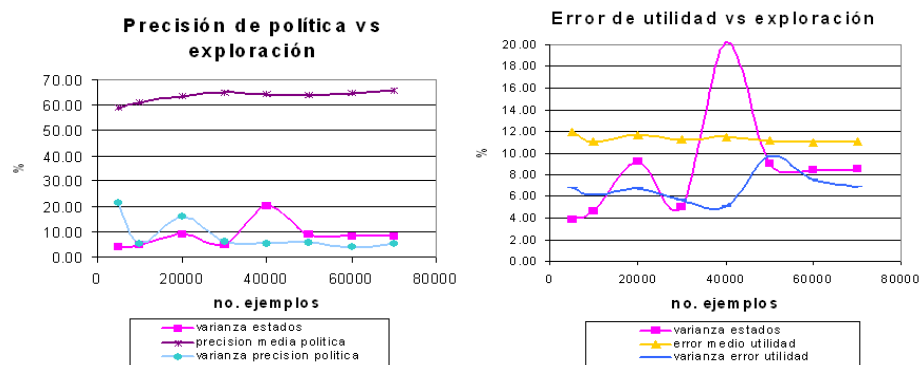


Figura 5.8: Gráficas de exploración mostrando la precisión de la política y el error de utilidad vs número de ejemplos para el problema 3 de la tabla 5.2. (Izquierda) Gráfica mostrando la precisión media de la política, la varianza de la precisión de la política, y la varianza del número de estados cualitativos. (Derecha) Gráfica mostrando el error medio de la utilidad, la varianza del error de la utilidad, y la varianza del número de estados cualitativos.

Evaluación del método de refinamiento

Para demostrar las ventajas del uso de la etapa de refinamiento durante la solución de problemas difíciles se resolvieron dos casos de navegación donde la meta está oculta entre zonas de recompensa negativa. En esta prueba, primeramente resolvimos un laberinto y un problema donde un region de alta recompensa se encuentra encerrada entre paredes de recompensa negativa. Por simplicidad y para mayor objetividad, en ambos problemas se usaron las acciones ortogonales arriba (u), abajo (d), a la derecha (r) y a la izquierda (l). La acción nula no tiene identificador.

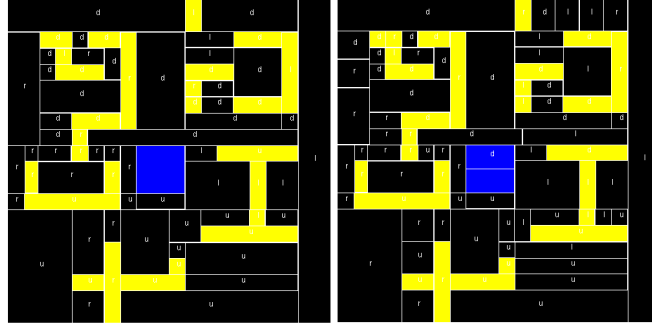


Figura 5.9: Refinamiento de la solución de un laberinto típico. (Izquierda) Política abstracta para una representación cualitativa del problema. (Derecha) Política resultante al aplicar el algoritmo de refinamiento.

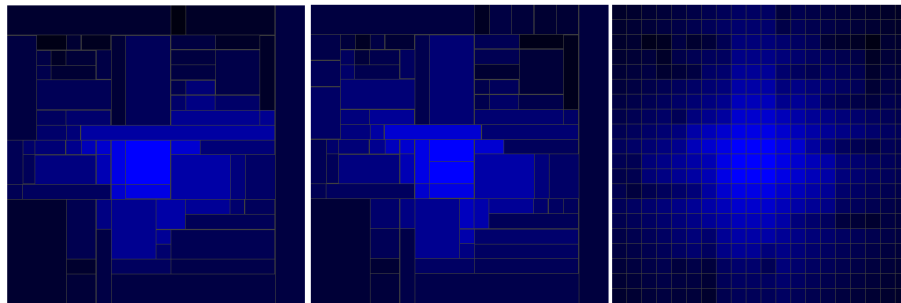


Figura 5.10: Aproximación incremental de la función de valor de un laberinto típico por refinamiento. (Izquierda) Función de valor abstracta gráfica para el problema del laberinto. (Centro) Función de valor refinada. (Derecha) Aproximación discreta fina de la función de valor.

El primer problema se ilustra en la figura 5.9 (izquierda) y en éste se puede apreciar que en varios estados abstractos la política sugiere avanzar sobre las paredes (color amarillo). Este efecto se debe a que durante la abstracción se calcula la mejor acción en términos de la utilidad esperada abstracta que rodea a un estado y por tanto estas sugerencias pueden despreciar zonas castigadas localmente. Sin embargo mediante el uso del refinamiento se logra superar algunos de estos problemas. Por ejemplo, a través de la inclusión de nuevos cortes en la parte inferior del cuarto cuadrante y central de la figura se logra evitar zonas negativas y lograr llegar a la meta sin problemas (figura derecha). Aún cuando la ganancia de precisión de la política no es altamente significativa en este problema, es posible ver claramente en la figura 5.10 como la función de utilidad se aproxima mejor a la versión más exacta (derecha). En esta problema el error de la función de utilidad se reduce de 20.11 % a 16.95 %, mientras que la precisión de la política mejora de 40.25 a 43.75 %.

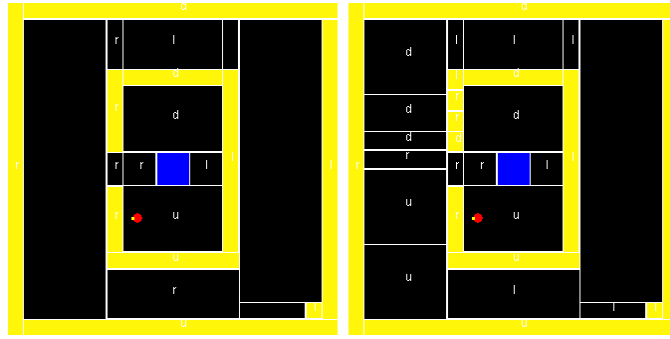


Figura 5.11: Refinamiento de la solución de un problema con la meta oculta. (Izquierda) Política abstracta para una representación cualitativa del problema. (Derecha) Política resultante al aplicar el algoritmo de refinamiento.

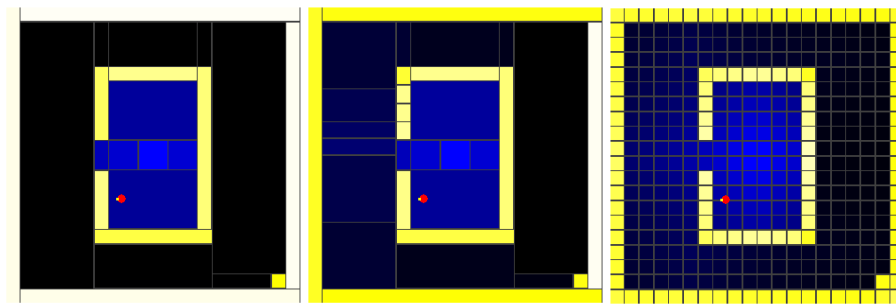


Figura 5.12: Aproximación incremental de la función de valor por refinamiento para el problema de la meta escondida. (Izquierda) Función de valor abstracta gráfica. (Centro) Función de valor refinada. (Derecha) Aproximación discreta fina de la función de valor.

El segundo problema se ilustra en la figura 5.1 (derecha), y su solución en la figura 5.11. En el lado izquierdo de ésta última, se puede apreciar que la forma de la función de recompensa dificulta generar una abstracción suficiente como para resolver el problema. Sin embargo, el uso del refinamiento permite asignar acciones donde el agente se queda inmóvil, y generar caminos mejor estructurados hacia la meta. Aún cuando solo se ejecutó el algoritmo para un número de etapas de refinamiento limitado, la ganancia de precisión de la política si es significativa. En la figura 5.12 se ilustra como gradualmente la función de utilidad se va aproximando mejor a una versión más exacta (derecha). En esta problema el error de la función de utilidad se reduce de 26.7% a 24.26%, mientras que la precisión de la política mejora de 44.25 a 69.5%.

5.2. Generación de energía eléctrica

Los problemas de planificación de movimiento nos permiten evaluar la técnica de MDP cualitativos de una manera objetiva al compararla con una discretización fina. Sin embargo, también comentamos nuestros hallazgos rumbo a la solución de problemas de planificación con fines de toma de decisiones de operación en un dominio industrial. Para tal efecto, utilizamos un simulador del proceso de generación de vapor en una planta de ciclo combinado, a partir del cual analizamos la conducta del sistema después de aplicar nuestros procedimientos de aprendizaje y solución con MDPs cualitativos.

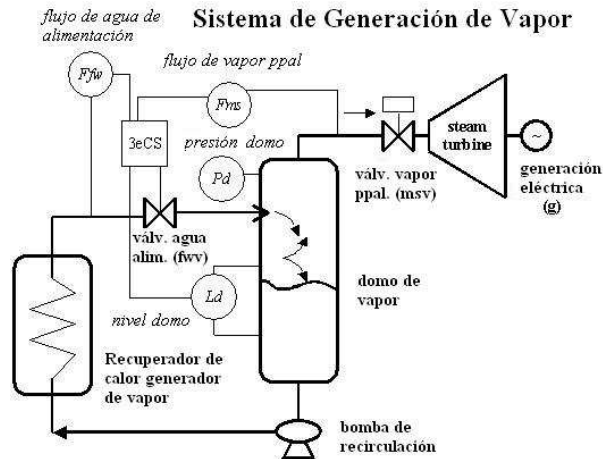


Figura 5.13: Proceso de generación de vapor en una central de ciclo combinado. Por simplicidad, la conexión con la turbina de gas ha sido omitida.

Una central de ciclo combinado es una planta que aprovecha el poder calorífico residual de una turbina de gas para producir energía eléctrica en una turbina de vapor. El equipo de proceso que conecta los ciclos de gas y vapor en este tipo de centrales se conoce como *recuperador de calor*. Un recuperador de calor generador de vapor (*HRSG* por sus siglas en Inglés) es un equipo capaz de recuperar energía de los gases de salida de una turbina de gas para generar vapor en un depósito especial llamado *domo de vapor*. El flujo de agua (F_{fw}) entre el domo y el HRSG se mantiene por medio de la *bomba de re-circulación de agua*, la cual extrae residuos de agua del domo y la envía a calentar al HRSG para su posterior transformación en vapor. El resultado de este proceso es un flujo de vapor de alta presión hacia la *turbina* que, acoplada a un *generador eléctrico*, produce electricidad (g). Los principales elementos asociados al control de este proceso son la válvula de agua de alimentación (fvr) y la válvula de vapor principal (msv). Un disturbio en la carga eléctrica (d) es un evento exógeno que puede producir una transición importante en el estado del sistema, similar al

producido por una mala operación de éstas válvulas. Este fenómeno puede ser ocasionado por una desconexión súbita de los consumidores de la planta generadora (*rechazo de carga*) o por un exceso de usuarios conectados a la misma (*alta demanda de carga*), y puede deberse principalmente a fallas en los sistemas de transmisión o distribución de carga a los centros de consumo. El proceso de generación de vapor se muestra en la figura 5.13.

Durante operación normal, el control de tres-elementos del sistema de generación de vapor (*3eCS*) manipula una válvula (*fvv*) para regular el nivel (*dl*) y presión (*pd*) del domo. Por su parte, el sistema de control de carga hace lo mismo con las válvula de vapor principal (*msv*) para mantener un equilibrio entre la potencia eléctrica demandada y la energía de presión suministrada por el domo. Un problema es que estos controladores no consideran la posibilidad de que las válvulas del ciclo de control, instrumentos o cualquier otro dispositivo del proceso no ejecuten con precisión sus órdenes. En consecuencia, la conducta del sistema de control puede afectar en el futuro la vida útil de los equipos, la eficiencia del proceso o la seguridad del mismo. Una posible solución sería diseñar una estrategia que no sólo considere errores en las operaciones de control, sino que tenga la capacidad de recomendar el uso de válvulas u otros dispositivos de acuerdo a una función de preferencia para optimizar la vida útil de los equipos de proceso, minimizar el riesgo de accidentes, o mejorar la generación de electricidad.

5.2.1. Especificación de un MDP

Con la idea de evaluar la solución al problema de control del recuperador de calor usando nuestra técnica cualitativa, definamos un MDP cuyas variables de estado sean la presión del domo Pd , el flujo de vapor principal Fms , el flujo de agua de alimentación Ffw , la posición de la válvula de agua de alimentación fvv , la posición de la válvula de vapor principal msv , la generación eléctrica g , y un disturbio d que en este caso corresponde a un rechazo de carga. La instanciación de estas variables en un momento dado caracterizan el estado operativo del generador de vapor.

Tabla 5.5: Conjuntos de acciones para el problema de control del *HRSG*. (Izquierda) Conjunto de acciones mutuamente exclusivas. (Derecha) Conjunto de acciones mutuamente exclusivas y combinadas.

Id. Acción	Comando	
	<i>fvv</i>	<i>msv</i>
0	+	0
1	-	0
2	0	+
3	0	-
4	0	0

Id. Acción	Comando	
	<i>fvv</i>	<i>msv</i>
0	+	0
1	-	0
2	0	+
3	0	-
4	+	+
5	+	-
6	-	+
7	-	-
8	0	0

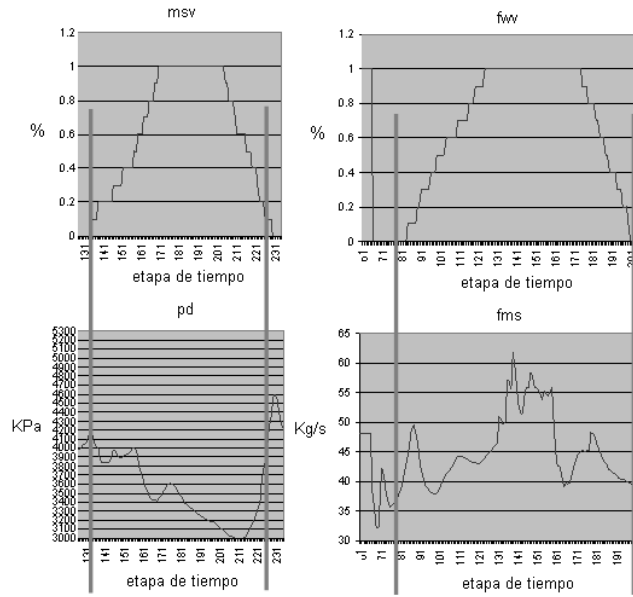


Figura 5.14: Efectos de las acciones sobre el sistema de generación de vapor. (Columna izquierda) Comportamiento de la presión de domo (Pd) debido a cambios en la posición de la válvula de vapor msv . (Columna derecha) Comportamiento del flujo de vapor principal (Fms) debido a cambios en la posición de la válvula de agua de alimentación fww .

El conjunto de acciones está conformado por los comandos de apertura ($+\delta$) y cierre ($-\delta$) de la válvula de agua de alimentación (fww) y de la válvula de vapor principal (msv). Una acción nula en este caso indica que no hay ningún cambio en la posición de dichas válvulas. La tabla 5.5 muestra dos conjuntos de acciones para el problema de control del *HRSG*, donde el primer conjunto (izquierda) corresponde a cuatro acciones mutuamente exclusivas y la acción nula, y el segundo (derecha) contiene acciones mutuamente exclusivas, conjuntas y la acción nula. En una acción mutuamente exclusiva sólo un dispositivo se opera a la vez, mientras que en una conjunta existe la posibilidad de operar los dos dispositivos de control en forma simultánea. La dinámica de la presión del domo Pd y el flujo de vapor principal Fms debida a la ejecución de estos comandos se muestra en la figura 5.14. Nótese que la respuesta de la presión del domo (Pd) es inversamente proporcional a la posición de la msv , mientras que la respuesta del flujo de vapor principal (Fms) es directamente proporcional a la posición de la fww .

La figura 5.15 muestra las regiones de seguridad establecidas por el fabricante del domo en base a los valores de Pd y Fms . Las zonas azules representan las condiciones de operación deseables, y las zonas negras estados de operación neutrales (recompensa nula). En una función de recompensa más sofisticada (3D)

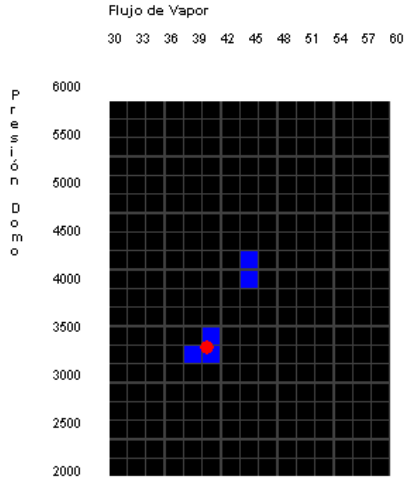


Figura 5.15: Función de recompensa para el problema de generación de vapor expresada como regiones de seguridad en términos de Pd y Fms . Las regiones objetivo (azul) se obtuvieron experimentalmente usando un sistema de control clásico.

se podría asignar un bono de recompensa extra cuando la generación eléctrica g esté por arriba de la mitad del valor máximo de generación. Aunque en la realidad la función de recompensa puede ser multi-valuada, en esta aplicación usaremos una función de recompensa binaria.

El problema de planificación consiste en determinar las recomendaciones de control manual (política) que maximicen la seguridad del domo, y eventualmente la generación eléctrica, durante operación normal o condiciones de rechazo de carga.

5.2.2. Solución usando MDPs cualitativos

Dada la cantidad de dimensiones del problema y la forma de la función de recompensa, la solución se planteó en forma de un MDP híbrido cualitativo-discreto con 2 variables continuas (Fms y Pd), 5 variables discretas (g , msv , fwv , Ffw y d), y un conjunto de 5 acciones mutuamente exclusivas.

Para el aprendizaje del modelo híbrido, se capturaron datos del simulador de ciclo combinado operando a carga media utilizando una estrategia de exploración no exhaustiva. La exploración se llevó a cabo seleccionando aleatoriamente una acción de dimensión tal que el cambio de valor de la función de recompensa fuera el mínimo suficiente para garantizar una transición de estados. Una muestra de los datos de exploración para el aprendizaje del modelo con Weka [76] y Elvira [16] se encuentran disponibles a detalle en la sección A.1 del apéndice A.

El árbol de decisión de la figura 5.16 representa la función de recompensa

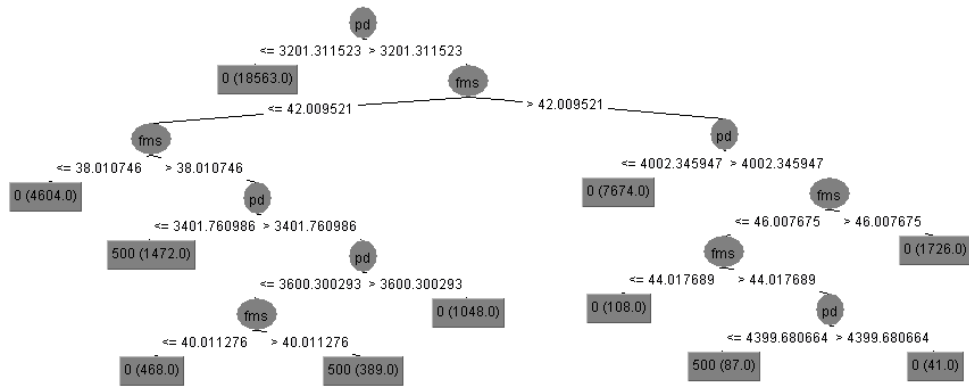


Figura 5.16: Estructura del árbol de decisión de 2D (Fms y Pd) obtenido para el problema de generación de vapor. Los nodos son variables continuas del problema, y las hojas el valor de recompensa con el número de ejemplos cubiertos por dicho valor.

aprendida por $C4.5$ para este problema, y su documentación se encuentra en la sección A.2 del apéndice A. Por su parte, la red Bayesiana dinámica de dos etapas de la figura 5.17 representa la función de transición de estados aprendida por $K2$ para la acción *abrir válvula de alimentación de agua*, ($+fww$). Durante el proceso de aprendizaje la variable $+Ffw$ resultó no tener relevancia en el proceso y por tanto no aparece en la red. En la figura, las variables $g, msv, d, yfww$ son las variables discretas, y Q concentra a Pd y Fms . Las variables de la izquierda son los factores relevantes en el instante t , y las de la derecha los mismos factores pero un instante después (*prime*). Toda la información relativa al aprendizaje del modelo de transición para este problema se ilustra con mayor detalle en la sección A.3 del apéndice A.

La figura 5.18 (izquierda) muestra la política encontrada por el sistema de planificación estocástica SPUDD [37] en forma de un diagrama de decisión algebraico. La raíz del diagrama corresponde a la variable abstracta Q que resume Fms y Pd , los arcos representan sus posibles valores q_0, \dots, q_9 , y las hojas denotan la política. Por ejemplo, en $Q = q_0$ y $Q = q_7$ la acción recomendada es *cerrar la válvula de vapor principal msv* (acción 03 o a3). A la derecha se ilustra la partición de estados, equivalente al diagrama de decisión, en un mapa bidimensional con los parámetros de seguridad Fms y Pd . Después de analizar los resultados, algunos expertos en operación de plantas opinan que en la mayoría de los estados cualitativos la estrategia de control es aproximadamente óptima. Por ejemplo, si la generación de vapor se encuentra en el estado abstracto q_0 , entonces la acción sugerida es *cerrar msv* para lograr un incremento en la presión del domo. No importa en cual de los puntos continuos del estado q_0 se encuentre el proceso, la política siempre sugerirá mover el proceso al estado cualitativo q_1

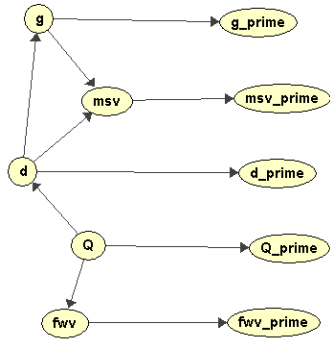


Figura 5.17: Función de transición de estados aprendida por *K2* para la acción *abrir válvula de alimentación de agua, (+fww)* para el problema de generación de vapor

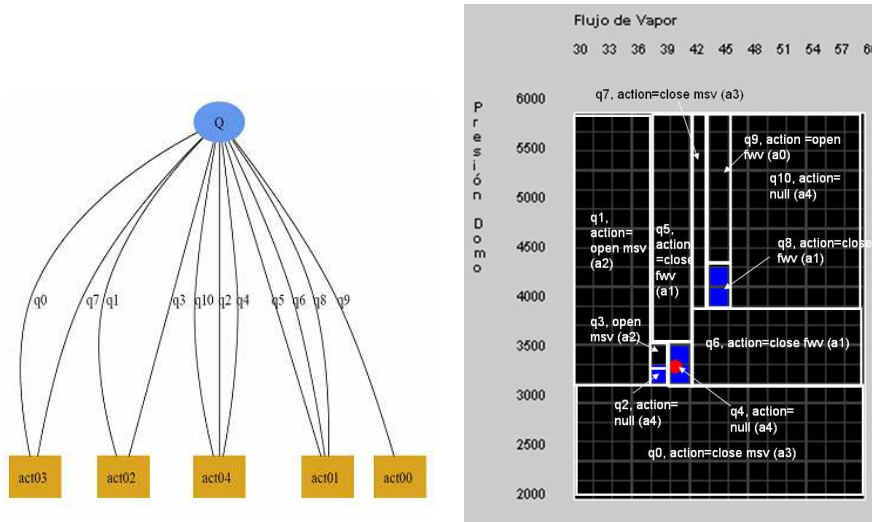


Figura 5.18: Política generada por el sistema SPUDD y partición de estados para el problema de generación de vapor. (Izquierda) Diagrama de decisión algebraico representando la política. (Derecha) Partición de estados cualitativos.

o $q6$, o en el mejor de los casos, moverse a $q2$ o $q4$ que son estados absorbentes (meta). Por otro lado, si el sistema se encontrara en $q6$ como resultado de *cerrar msv*, la nueva acción recomendable será entonces *cerrar fww* para decrementar el flujo de vapor principal Fms y llevar al sistema, con una alta probabilidad, al estado abstracto $q4$. Otro ejemplo de acciones cercanas al óptimo se presenta cuando el sistema se encuentra en $q3$ y se sugiere *abrir msv*. El efecto de esta acción será el decremento en la presión del domo y así, el sistema transitará al

estado terminal q_4 . En caso de un disturbio, esta política puede ser lo suficientemente buena como para hacer que el proceso se reestablezca hacia zonas de operación seguras.

Como experimento adicional para validar la técnica, resolvimos el mismo problema pero usando un esquema de 9 acciones (tabla 5.5 derecha). Esta configuración produjo un MDP híbrido cualitativo-discreto de 7 dimensiones y 9 acciones que, a pesar de su complejidad, el sistema SPUDD resolvió después de 182 iteraciones, en 9.4 seg. Asimismo, se pudo observar que la política obtenida resultó más uniforme que en la versión compacta con 5 acciones presentada anteriormente. La documentación completa de este experimento se encuentra en el apéndice B.

5.3. Resumen del capítulo

A lo largo de este capítulo, hemos ilustrado algunos resultados que demuestran las ventajas de uso de los MDPs cualitativos para resolver problemas complejos. Particularmente se enfatizó el ahorro computacional producido por la abstracción, el mejoramiento de la calidad de la solución al usar el algoritmo de refinamiento, la construcción de modelos factorizados a partir de datos de exploración, y el uso del algoritmo de aprendizaje de MDPs cualitativos tanto durante la abstracción como durante el refinamiento de la misma. Aún cuando para ciertos problemas el error producido por la abstracción genera imprecisiones en la solución, el costo computacional en espacio y tiempo para resolver modelos abstractos es mucho menor que el necesario para resolver problemas con espacios continuos basados en discretizaciones finas. Asimismo, y aún cuando las abstracciones producidas para ciertos problemas fueron insuficientes para resolverlos, el algoritmo de refinamiento demostró mejorar las particiones iniciales y así generar soluciones satisfactorias.

Para evaluar la técnica propuesta, se realizaron experimentos en distintos tipos de problemas de planificación de movimiento con estados discretos, continuos, e híbridos cualitativos-discretos. Las soluciones abstractas se compararon en términos de calidad y complejidad, contra una discretización fina emulando una solución exacta. Igualmente, se demostró que el uso de SPUDD con una representación cualitativa de este dominio produce soluciones en tiempos de ejecución muy buenos. Posteriormente, se mostraron resultados preliminares de un problema de control de una planta eléctrica con espacios de estados y acciones más grandes, para demostrar que el método puede generar soluciones satisfactorias aun ante problemas complejos.

En el siguiente capítulo se concluye la realización de este trabajo mediante una discusión sobre sus aspectos más importantes, ventajas y desventajas, y aplicabilidad en otros dominios. Finalmente se establece una guía para dar continuidad al trabajo futuro.

Capítulo 6

Conclusiones y trabajo futuro

6.1. Conclusiones

En este trabajo se presentó una representación abstracta basada en restricciones cualitativas de las variables para resolver MDPs con espacios continuos, y una metodología para aprender automáticamente el modelo con esta representación que se resuelve con técnicas convencionales. Basado en una estructura de recompensa inmediata se particiona el espacio de estados para formar un conjunto de estados abstractos, a los que denominamos *estados cualitativos*. Esta partición se obtiene automáticamente usando técnicas de aprendizaje y transformación de árboles de decisión. Posteriormente, mediante datos de exploración del ambiente con acciones aleatorias se induce, con un algoritmo de aprendizaje de redes Bayesianas, un modelo de transición de estados probabilista sobre los estados cualitativos. De esta manera, se ha aprendido un modelo de decisión de Markov con una representación estructurada de las funciones de recompensa y transición. Este modelo abstracto inicial, finalmente se detalla mediante un procedimiento que realiza selectivamente mejoras a la función de valor y consecuentemente a la política. Una de las ventajas del método es que el MDP abstracto se resuelve usando técnicas estándar de programación dinámica.

La motivación científica para la realización de este trabajo fue la oportunidad de contribuir en la mejora de los métodos de planificación automática actuales basados en MDPs, particularmente en la búsqueda de nuevas técnicas de representación y aprendizaje para poder resolver problemas complejos. Entre algunos de estos problemas destacan el problema de la navegación robótica y el control de procesos de generación eléctrica, donde muchas variables, a veces de naturaleza continua, cambian dinámicamente por sí mismas, por la operación de los dispositivos de control (motores, válvulas, bombas, actuadores), o por la aparición de algún evento exógeno. Las contribuciones a la comunidad de planificación automática basada en MDPs se resume como sigue:

1. Un método de abstracción que reduce el espacio de estados basado en recompensa inmediata.
2. Un procedimiento para construir y resolver un modelo abstracto en forma totalmente automática.
3. Un método de aprendizaje de MDPs factorizados basado en aprendizaje de árboles de decisión y de redes Bayesianas dinámicas.
4. Un método de refinamiento que iterativamente detalla la abstracción donde mejor se contribuye a la solución final.

Las ventajas representacionales de los MDPs cualitativos radican en que la abstracción llamada partición Q , permite reducir la dimensionalidad de un dominio al tiempo que trata a las variables continuas usando restricciones cualitativas. Si no existen variables continuas en un problema, sólo se reduce la dimensionalidad, sin embargo siempre se presentan ahorros representacionales respecto a la especificación original.

La técnica de abstracción-descomposición presentada difiere del trabajo relacionado en distintas formas. Por ejemplo, mientras otros trabajos inician su proceso suponiendo la existencia de un estado abstracto único o una partición discreta uniforme arbitraria, nosotros usamos a la función de recompensa como heurística para la construcción de una abstracción inicial. En ocasiones ésta puede ser tan buena, que puede resolver bien el problema. En aquellos casos donde no es así, nuestro método cuenta con una etapa de refinamiento que mejora gradualmente la solución. Una característica del método de refinamiento es que, dado que evalúa cada solución global para buscar mejoras locales donde se necesitan, minimiza el número de cortes para no complicar nuevamente el problema. Otra ventaja de nuestro método es que el proceso de representación es totalmente automático, esto es, construye sus modelos a partir de datos de exploración fuera de línea. Aun cuando las técnicas basadas en aprendizaje por refuerzo resuelven MDPs en tiempo de ejecución, ante problemas complejos pueden presentar problemas de convergencia que este método no tiene.

Para evaluar la técnica propuesta, se realizaron experimentos en distintos tipos de problemas de planificación de movimiento con estados discretos, continuos, e híbridos continuos-discretos. Las soluciones abstractas se compararon en términos de calidad y complejidad, contra una discretización fina emulando la solución exacta, y contra soluciones generadas por el sistema SPUDD. En cada caso, se obtuvieron políticas *cuasi-óptimas* en aproximadamente dos órdenes menos de magnitud en tiempo. Por su parte, los valores de utilidad en nuestras políticas abstractas se aproximan a las soluciones exactas con un error promedio del 10%. La etapa de refinamiento igualmente demostró mejorar hasta en un 57% la precisión de la política sobre la abstracción inicial. Aún en casos donde las mejoras fueron menores ($< 10\%$), el comportamiento orientado a metas fue en general satisfactorio. SPUDD demostró, por su parte, alta eficiencia al resolver problemas especificados bajo una representación cualitativa.

Por otra parte, se mostraron algunos resultados preliminares obtenidos mediante experimentación con un simulador de una planta de ciclo combinado para demostrar, que el método también puede generar soluciones satisfactorias en problemas con espacios de estados y acciones más grandes, y ante la presencia de eventos exógenos (en este caso, modelados como variables de estado).

De acuerdo con los resultados arrojados por los experimentos, creemos que el uso de la planificación con incertidumbre basada en la estrategia abstracción-refinamiento permite generar incrementalmente buenas políticas a partir de representaciones sencillas de problemas complejos. Además, ya que la función de recompensa es la base del método, y que ésta se construye con algoritmos de aprendizaje que estructuran un dominio, esta técnica puede aplicarse bien en ambientes reales, usualmente no estructurados.

6.2. Trabajo futuro

Los resultados experimentales realizados soportan la aplicabilidad de los MDPs cualitativos en dominios complejos. Sin embargo, aún hay trabajo por realizar para robustecer su representación e inferencia. Entre algunas de las mejoras se encuentran:

1. Medir la ganancia de precisión en el problema de la generación eléctrica usando el algoritmo de refinamiento.
2. Reducir la enumeración del espacio de acciones durante el proceso de búsqueda de la política óptima (programación dinámica).
3. Evaluar el poder de generalización de un MDP cualitativo. Creemos que además de resolver problemas particulares, nuestra técnica podría resolver tipos de problemas. Por ejemplo, problemas con la misma función de recompensa.
4. Combinar el método de abstracción-refinamiento con las técnicas de descomposición de problemas.
5. Extender el concepto de estados cualitativos para su utilización en procesos de decisión de Markov parcialmente observables (*POMDPs*).
6. Evaluar los MDPs cualitativos en otros dominios.

Debido al grado de dificultad del problema de plantas eléctricas mostrado, la evaluación de este dominio en este trabajo estuvo restringida a verificar el poder de solución del problema usando una abstracción inicial. Aunque desde el punto de vista de un experto, la conducta del sistema usando la política resultante fue satisfactoria, en un futuro próximo extenderemos nuestro método para poder también refinar su solución.

Como complemento al concepto de estados cualitativos, se propone reducir la complejidad del problema mediante la especificación de una clase de operadores

tipo STRIPS llamados *acciones relacionales* [54]. Estos operadores podrían restringir la enumeración explícita del espacio de acciones durante el proceso de programación dinámica haciendo a las acciones una función de los estados. El espacio de acciones A en un MDP cualitativo podrá entonces ser reemplazado por este nuevo conjunto de acciones relacionales $A-r$. La efectividad de esta idea se probará en problemas con espacios de acciones muy grandes.

Consideramos que, debido a las capacidades de generalización de los modelos cualitativos, las soluciones particulares generadas para ciertos problemas se podrían reutilizar en problemas con el mismo sistema de recompensa. De esta forma se podría usar una misma política para, por ejemplo, dos o más sistemas de generación de vapor de diferentes fabricantes. Dado la infraestructura de simulación disponible, esta evaluación no sería complicada y sí muy útil.

Debido al éxito de las técnicas de descomposición paralela, los *MDPs* multiseccionados [27] ahora podrían basarse en estados cualitativos refinados, y así abordar problemas de mayor complejidad. Por ejemplo, se pueden resolver diferentes MDPs cualitativos para varios lazos de control y luego combinar sus políticas mediante algún mecanismo de arbitraje, tipo supresión de conductas, para la resolución de conflictos.

Consideramos que la reducción en la complejidad ofrecida por los MDPs cualitativos puede contribuir a superar el problema de tratabilidad de los *MDPs* en ambientes parcialmente observables (*POMDPs*). Consecuentemente, la solución de un POMDP cualitativo permitiría agregar una dosis de realidad extra a las políticas generadas ya que consideraría no solo la incertidumbre sobre los efectos de las acciones, sino también la incertidumbre de los estados. Por ejemplo, en el dominio de los sistemas de control, es bien conocido que los sensores pueden no ser siempre los adecuados, o simplemente ser insuficientes para determinar con efectividad el estado de un proceso en un momento dado. Como trabajo futuro, realizaremos pruebas con simuladores de plantas usando modelos de fallas de sensores, y evaluaremos que tan factible es la solución de un POMDP en este dominio.

Actualmente, estamos aplicando los resultados de la presente investigación en el desarrollo de un proyecto donde se combinan la generación de recomendaciones y explicaciones de operación de plantas. Entre las posibles aplicaciones prometedoras de la planificación con incertidumbre usando MDPs cualitativos, se encuentran la planificación del sensado, planificación de operaciones de alto nivel en centros de control de energía, planificación del mantenimiento industrial, elaboración de planes de emergencia en zonas de alto riesgo, entre muchas otras.

Bibliografía

- [1] K. J. Astrom. Optimal control of markov processes with incomplete state information. *J. Math. Anal. App.*, 10:174–205, 1965.
- [2] R.E. Bellman. *Dynamic Programming*. Princeton U. Press, Princeton, N.J., 1957.
- [3] D. P. Bertsekas. *Dynamic Programming*. Prentice Hall, Eaglewood Cliffs, NJ. MA, 1987.
- [4] D. P. Bertsekas. *A counter-example to temporal difference learning*. Neural Computation, 1994.
- [5] D. P. Bertsekas and J.N. Tsitsiklis. *Neuro-dynamic programming*. Athena Sciences, 1996.
- [6] A. Blum and M. Furst. Fast planning through planning graph analysis. *Artificial Intelligence*, 90(1–2):281–300, 1997.
- [7] J. Blythe. *Planning under Uncertainty in Dynamic Domains*. PhD thesis, Computer Science Department, Carnegie Mellon University, 1998.
- [8] B. Bonet and J. Pearl. Qualitative mdps and pomdps: An order-of-magnitude approach. In *Proceedings of the 18th Conf. on Uncertainty in AI, UAI-02*, pages 61–68, Edmonton, Canada, 2002.
- [9] C. Boutilier, T. Dean, and S. Hanks. Decision-theoretic planning: structural assumptions and computational leverage. *Journal of AI Research*, 11:1–94, 1999.
- [10] Craig Boutilier and Richard Dearden. Using abstractions for decision-theoretic planning with time constraints. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1016–1022, 1994.
- [11] Craig Boutilier, Richard Dearden, and Moises Goldszmidt. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1104–1111, Montreal, 1995.

- [12] Craig Boutilier, Moisés Goldszmidt, and Bikash Sabata. Continuous value function approximation for sequential bidding policies. In Kathryn Laskey and Henri Prade, editors, *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence (UAI-99)*, pages 81–90. Morgan Kaufmann Publishers, San Francisco, California, USA, 1999.
- [13] R. Brafman. A heuristic variable grid solution method of pomdps. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 727–733, Menlo Park, CA, 1997. AAAI press.
- [14] Daphne Koller Carlos Guestrin and Ronald Parr. Multiagent planning with factored mdps. In *Advances in Neural Information Processing Systems (NIPS 2001)*, Vancouver, Canada, December 2001.
- [15] Anthony R. Cassandra, Leslie Pack Kaelbling, and Michael L. Littman. Acting optimally in partially observable stochastic domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, 1994.
- [16] Elvira Consortium. Elvira: an environment for creating and using probabilistic graphical models. Technical report, U. de Granada, Spain, 2002.
- [17] G. F. Cooper and E. Herskovits. A bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 1992.
- [18] H. W. Corley, J. M. Rosenberg, W-C. Yeh, and T. K. Sung. The cosine simplex algorithm. *International Journal of Advanced Manufacturing Technology*, 27:1047–1050, 2006.
- [19] A. Darwiche and Goldszmidt M. Action networks: A framework for reasoning about actions and change under understanding. In *Proceedings of the Tenth Conf. on Uncertainty in AI, UAI-94*, pages 136–144, Seattle, WA, USA, 1994.
- [20] T. Dean and R. Givan. Model minimization in markov decision processes. In *Proc. of the 14th National Conf. on AI*, pages 106–111. AAAI, 1997.
- [21] T. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5:142–150, 1989.
- [22] Thomas Dean, Leslie Pack Kaelbling, Jak Kirman, and Ann Nicholson. Planning with deadlines in stochastic domains. In *Proc. of the Eleventh National Conference on AI*, pages 574–579, Menlo Park, CA, 1993. AAAI.
- [23] Thomas Dean and Shieu-Hong Lin. Decomposition techniques for planning in stochastic domains. In *Proceedings IJCAI-95*, pages 1121–1127, San Francisco, California, 1995. Morgan Kaufmann Publishers.

- [24] Thomas Degris, Olivier Sigaud, and Pierre-Henri Wuillemin. Learning the structure of factored markov decision processes in reinforcement learning problems. In *Proceedings of the 23rd International Conference on Machine Learning*, Pittsburg, PA, 2006.
- [25] T. G. Dietterich and N. S. Flann. Explanation-based learning and reinforcement learning: A unified view. In *Proceedings of the 12th International Conference on Machine Learning*, pages 176–184, Tahoe City, CA. San Francisco, 1995. Morgan Kaufmann.
- [26] S.; Draper, D.; Hanks and D. Weld. Probabilistic planning with information gathering and contingent execution. In ed. K. Hammond, editor, *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, pages 31–37, Menlo Park, CA, 1994. AAAI Press.
- [27] P. Elinas, L.E. Sucar, A. Reyes, and J. Hoey. A decision theoretic approach for task coordination in social robots. In *Proc. of the IEEE International Workshop on Robot and Human Interactive Communication (RO-MAN)*, pages 679–684, Japan, 2004.
- [28] D. P. De Farias and B. Van Roy. The linear programming approach to approximate dynamic programming. *Operations Research*, 51(6):850–865, November-December 2003.
- [29] J. A. Feldman and R. F. Sproull. Decision theory and artificial intelligence ii: The hungry monkey. *Cognitive Science*, 1:158–192, 1977.
- [30] Z. Feng, R. Dearden, N. Meuleau, and R. Washington. Dynamic programming for structured continuous markov decision problems. In *Proc. of the 20th Conf. on Uncertainty in AI (UAI-2004)*. Banff, Canada, 2004.
- [31] R. Fikes and N. Nilsson. Strips: a new approach to the application of theorem proving to problem solving. *AI*, 2:189–208, 1971.
- [32] K. Forbus. Qualitative process theory. *AI*, 24:85–168, 1984.
- [33] C. Guestrin, M. Hauskrecht, and B. Kveton. Solving factored mdps with continuous and discrete variables. In *Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004)*, Banff, Canada, 2004.
- [34] C. Guestrin, D. Koller, and R. Parr. Max-norm projections for factored MDPs. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, pages 673–682, 2001.
- [35] P. Haddawy and M. Suwandy. Decision-theoretic refinement planning using inheritance abstraction. In ed K. Hammond, editor, *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, Menlo Park, CA, 1994. AAAI Press.

- [36] M. Hauskrecht and B. Kveton. Linear program approximation for factored continuous-state markov decision process. In *In Advances in Neural Information Processing Systems NIPS(03)*, pages 895–902, 2003.
- [37] J. Hoey, R. St-Aubin, A. Hu, and C. Boutilier. Spudd: Stochastic planning using decision diagrams. In *Proc. of the 15th Conf. on Uncertainty in AI, UAI-99*, pages 279–288, 1999.
- [38] Jesse Hoey and Pascal Poupart. Solving pomdps with continuous or large discrete observation spaces. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*, Edinburgh, 2005.
- [39] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the national academy of sciences*, volume 79, pages 2554–2558, USA, 1982.
- [40] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, Massachusetts, 1960.
- [41] R. A. Howard and J. E. Matheson. *Influence Diagrams*. Principles and Applications of Decision Analysis. Howard, R. A. and Matheson J. E. Editors, Menlo Park, CA, 1984.
- [42] H. A. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, Menlo Park, CA, 1996. American Association for Artificial Intelligence.
- [43] Craig A. Knoblock. *Automatically generating abstractions for problem solving*. PhD thesis, School of Computer Science, Carnegie Mellon University, 1991.
- [44] D. Koller and R. Parr. Policy iteration for factored MDPs. In *Proc. of the Sixteenth Conf. on Uncertainty in AI, UAI-00*, pages 326–334, Stanford, CA, USA, 2000.
- [45] B. Kuipers. Qualitative simulation. *AI*, 29:289–338, 1986.
- [46] N. Kushmerick, S. Hanks S, and D. Weld. An algorithm for probabilistic least-commitment planning. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 1073–1078, Menlo Park, CA, 1995. AAAI Press.
- [47] Lihong Li and Michael L. Littman. Lazy approximation for solving continuous finite-horizon mdps. In *AAAI-05*, pages 1175–1180, Pittsburgh, PA, 2005.
- [48] R. D. Luce and H. Raiffa. *Games and Decisions: Introduction and Critical Survey*. Wiley, 1957.

- [49] Veloso M., Carbonel J., Perez A., Borrajo D., Fink E., and Blythe J. Integrating planning and learning: The prodigy architecture. *Journal of Experimental and Theoretical AI*, 1:81–120, 1995.
- [50] D. Johnson M. Trick. Challenge on satisfiability testing. In *Proc. DIMACS 93*, DIMACS Series on disc math., Piscataway, NJ, 1993.
- [51] S. M. Majercik and M.L. Littman. Maxplan: A new approach to probabilistic planning. In eds. R. Simmons and S. Smith, editors, *Proceedings of the Fourth International Conference on Artificial Intelligence Planning Systems*, pages 86–93, Menlo Park, Calif., 1998. AAAI Press.
- [52] D. McAllester and D. Rosenblitt. Systematic non-linear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, pages 634–639, Anaheim, CA, 1991.
- [53] W. S. McCulloch and W. Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943.
- [54] E. Morales. Scaling up reinforcement learning with a relation representation. pages 15–26. Proc. of the Workshop on Adaptability in Multi-agent Systems (AORC-2003), 2003.
- [55] Remi Munos and Andrew Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. In Thomas Dean, editor, *Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI-99)*, pages 1348–1355. Morgan Kaufmann Publishers, San Francisco, California, USA, August 1999.
- [56] Patrick Naughton and Herbert Schildt. *Java 2: The Complete Reference*. Osborne/Mc Graw Hill, 3rd Edition, London, Great Britain, 2000.
- [57] A. Newell and H. Simon. *GPS: A Program That Simulates Human Thought*. A. Feigenbaum and J. Feldman McGraw-Hill, New York, 1963.
- [58] R. Parr and S. Russell. Approximating optimal policies for partially observable stochastic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence IJCAI 95*, pages 1088–1094, Menlo Park, Calif., 1995.
- [59] J. S. Penberthy and D. S. Weld. Ucpop: a sound, complete, partial order planner for adl. In *Proc. of Knowledge Representation and Reasoning*, pages 103–114, Cambridge, MA, 1992.
- [60] M. A. Peot and D. E. Smith. Conditional nonlinear planning. In ed. J. Hendler, editor, *Proceedings of the First International Conference on Artificial Intelligence Planning Systems*, pages 189–197, San Francisco, CA, 1992. Morgan Kaufmann.

- [61] J. Pineau, G. Gordon, and S. Thrun. Policy-contingent abstraction for robust control. In *Proc. of the 19th Conf. on Uncertainty in AI, UAI-03*, pages 477–484, 2003.
- [62] M.L. Puterman. *Markov Decision Processes*. Wiley, New York, 1994.
- [63] J.R. Quinlan. *C4.5: Programs for machine learning*. Morgan Kaufmann, San Francisco, Calif., USA., 1993.
- [64] A. Reyes. Asistente inteligente para la resolución de fallas en una unidad turbogas de generación eléctrica. Technical report, ITESM Cuernavaca, Reforma 185A Col. Palmira Cuernavaca Morelos México, 2002.
- [65] A. Reyes, P. H. Ibarquengoytia, and L. E. Sucar. Power plant operator assistant: An industrial application of factored mdps. In *MICAI*, 2004.
- [66] J. Rush. *Using randomizing to break the course of dimensionality*, volume 65. *Econometrica*, 1997.
- [67] Earl D. Sacerdoti. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5:115–135, 1974.
- [68] D. Schuurmans and R. Patrascu. Direct value-approximations for factored MDPs. In *Advances in Neural Information Processing Systems*, 2002.
- [69] H. Simon. *The Sciences of the Artificial*. MIT Press, 1996.
- [70] D. Suc and I. Bratko. Qualitative reverse engineering. In *Proc. of the 19th International Conf. on Machine Learning*, 2000.
- [71] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
- [72] J. Tash and S. Russell. Control strategies for a stochastic planner. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, Seattle, WA, 1994.
- [73] Federico Trigos-Salazar and Juan Frausto-Solis. Experimental evaluation of the theta heuristic for starting the cosine simplex method. *Computational Science and Its Applications-ICCSA 2005*, May 2005.
- [74] Federico Trigos-Salazar, Juan Frausto-Solis, and Rafael Rivera-Lopez. A simplex cosine method for solving the klee-minty cube. *Advances in Simulation, System Theory and Systems Engineering*, pages 27–32, 2002.
- [75] M. Williamson and S. Hanks. Optimal planning with a goal-directed utility model. In ed K. Hammond, editor, *Proceedings of the Second International Conference on Artificial Intelligence Planning Systems*, pages 176–181, Menlo Park, CA, 1994. AAAI Press.
- [76] I.H. Witten. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations, 2nd Ed.* Morgan Kaufmann, USA, 2005.

Apéndice A

Problema 1.

Problema de la caldera (HRSG) con la siguiente configuración: dos variables de estado continuas, 5 variables lógicas, 5 acciones, y función de recompensa con 2 dimensiones (presión y flujo de vapor).

A.1. Datos de exploración

A.1.1. Archivo de atributos continuos

```
fms:real, 0, 204
ffw:real, 0,345000
d:real, 0, 1
pd:real,0,5510
g:real, 0,26110
msv:real, 0, 1
fwv:real, 0, 1
r:0,500
```

A.1.2. Archivo de atributos híbridos

```
fms:real, 0, 204
ffw:S0,S1,S2,S3
d: S0,S1
pd:real,0,5510
g:S0,S1,S2,S3
msv:S0,S1,S2,S3,S4,S5,S6,S7,S8
fwv:S0,S1,S2,S3,S4,S5,S6,S7,S8
r:0,500
```

A.1.3. Muestra archivo de ejemplos

```
fms ffw d pd g msv fwv a r
34.38868713 0.001 1 2731.455566 0 0 0 3 0
34.3821373 0.001 1 2730.403564 0 0 0 1 0
34.37568283 0.001 1 2729.411377 0 0 0 2 0
38.46545029 0.001 1 2680.400635 0 0.100000001 0 2 0
41.61096954 0.001 1 2602.460205 0 0.100000001 0 3 0
34.25814056 0.001 1 2690.789551 0 1.49E-09 0 4 0
34.15793228 0.001 1 2729.582031 0 1.49E-09 0 0 0
35.02843475 3.913500547 1 2760.381836 0 1.49E-09 0.100000001 3 0
34.98811722 3.829193354 1 2783.210693 0 0 0.100000001 1 0
34.03686905 0.001 1 2794.869629 0 0 1.49E-09 3 0
```

33.94315338 0.001 1 2799.833496 0 0 1.49E-09 4 0
33.93058777 0.001 1 2799.619141 0 0 1.49E-09 0 0
34.85762024 4.007261276 1 2804.489502 0 0 0.100000001 4 0
34.85087967 3.811034679 1 2809.828613 0 0 0.100000001 3 0
34.87799835 3.946823597 1 2813.472168 0 0 0.100000001 1 0
34.0292778 0.001 1 2811.803467 0 0 1.49E-09 2 0
39.50028229 0.001 1 2756.916992 0 0.100000001 1.49E-09 1 0
39.12135696 0.001 1 2729.129639 0 0.100000001 0 1 0
38.68962479 0.001 1 2704.292725 0 0.100000001 0 0 0
38.07309723 3.981935024 1 2691.62207 0 0.100000001 0.100000001 3 0
34.83923721 3.768754482 1 2749.374268 0 1.49E-09 0.100000001 3 0
34.99512482 3.902376652 1 2772.144531 0 0 0.100000001 0 0
35.78328705 7.210760593 1 2807.20459 0 0 0.200000003 2 0
39.5157547 7.687091351 1 2763.180176 0 0.100000001 0.200000003 2 0
43.80437088 8.170621872 1 2700.664795 0 0.200000003 0.200000003 4 0
42.19573212 8.684866905 1 2671.05542 0 0.200000003 0.200000003 1 0
40.09329987 6.422456741 1 2648.017822 0 0.200000003 0.100000001 0 0
38.26039124 9.424733162 1 2645.599121 0 0.200000003 0.200000003 0 0

44.84001541 16.26805496 0 3459.601563 0 0.1172176 0.332392156 3 0
36.63386536 16.31977272 0 3478.508545 0 1.54E-04 0.332392156 1 0
36.00759506 12.38331699 0 3469.276123 0 1.54E-04 0.225043848 1 0
35.73466492 8.478414536 0 3455.922852 0 1.54E-04 0.118302785 3 0
35.22135925 7.967651367 0 3445.411133 0 0 0.118302785 4 0
34.94552994 7.494360924 0 3435.030029 0 0 0.118302785 2 0
42.74835587 6.932432652 0 3370.871826 0 0.111947581 0.118302785 1 0
42.02416229 4.567979336 0 3327.767334 0 0.111947581 0.02999199 1 0
38.74968719 0.001 0 3300.930664 0 0.111947581 0 2 500
42.14693069 0.001 0 3243.055908 0 0.215623334 0 0 0
38.3735466 5.046450138 0 3218.74707 0 0.215623334 0.110433646 1 500
37.2047081 2.132188082 0 3174.284424 0 0.215623334 0.022872679 0 0
35.71730804 4.961258888 0 3166.423096 0 0.215623334 0.132030785 1 0
33.53073502 0.747078419 0 3157.087646 0 0.215623334 0.04131696 4 0
34.04434586 2.242835999 0 3151.124268 0 0.215623334 0.04131696 2 0
35.1950531 0.469926536 0 3129.695068 0 0.304234654 0.04131696 1 0
34.92549896 0.001 0 3118.239258 0 0.304234654 0 0 0
33.97914505 4.569783688 0 3119.752441 0 0.304234654 0.099571243 3 0
33.06609726 4.278151989 0 3208.949219 0 0.229894102 0.099571243 1 0
32.16609573 0.001 0 3256.53125 0 0.229894102 0 3 0
32.09698486 0.001 0 3338.444336 0 0.127616987 0 1 0
32.02378845 0.001 0 3418.624756 0 0.127616987 0 0 0
33.08149719 4.432181835 0 3479.593262 0 0.127616987 0.096970335 0 0
33.73783875 6.805000782 0 3527.163574 0 0.127616987 0.189742804 2 0
36.17166138 6.595819473 0 3480.866699 0 0.230076164 0.189742804 1 0
36.96544647 3.765723705 0 3463.949463 0 0.230076164 0.094943255 3 0
32.68418503 4.055000305 0 3496.736084 0 0.142857343 0.094943255 3 0
32.76668167 4.25833559 0 3573.068115 0 0.030215498 0.094943255 1 0
31.43440628 0.001 0 3643.442627 0 0.030215498 0 2 0
35.40282059 0.001 0 3596.427734 0 0.129999667 0 4 0
35.03035736 0.001 0 3582.087646 0 0.129999667 0 2 0
40.94315338 0.001 0 3528.938965 0 0.218315572 0 4 500
40.72145462 0.001 0 3494.811279 0 0.218315572 0 0 500
40.35005188 4.01375103 0 3468.361084 0 0.218315572 0.094634183 0 500
39.64906311 5.391868114 0 3443.409912 0 0.218315572 0.156004623 1 0
39.11143494 2.874258518 0 3391.869873 0 0.218315572 0.058834996 0 500
38.88205338 5.804513454 0 3372.716309 0 0.218315572 0.159268886 2 500
42.80253983 6.089603424 0 3325.355469 0 0.313492358 0.159268886 2 0
45.15245056 6.327824593 0 3274.232178 0 0.40735805 0.159268886 2 0
45.42948914 6.594819546 0 3226.775391 0 0.501772702 0.159268886 1 0
41.75751877 3.179507017 0 3193.62207 0 0.501772702 0.05882803 1 0
39.71514893 0.001 0 3166.956787 0 0.501772702 0 0 0
37.89009094 4.926984787 0 3155.458984 0 0.501772702 0.095399171 3 0

A.2. Función de recompensa

A.2.1. Muestra archivo de ejemplos de Weka

```
@relation dTreeCont

@attribute fms real
@attribute ffw {S0, S1, S2, S3}
@attribute d {S0, S1}
@attribute pd real
@attribute g {S0, S1, S2, S3}
@attribute msv {S0, S1, S2, S3, S4, S5, S6, S7, S8}
@attribute fwv {S0, S1, S2, S3, S4, S5, S6, S7, S8}
@attribute r {0, 500}

@data

34.38868713,S0,S1,2731.455566,S0,S0,S0,0
34.3821373,S0,S1,2730.403564,S0,S0,S0,0
34.37568283,S0,S1,2729.411377,S0,S0,S0,0
38.46545029,S0,S1,2680.400635,S0,S0,S0,0
41.61096954,S0,S1,2602.460205,S0,S0,S0,0
34.25814056,S0,S1,2690.789551,S0,S0,S0,0
34.15793228,S0,S1,2729.582031,S0,S0,S0,0
35.02843475,S0,S1,2760.381836,S0,S0,S0,0
34.98811722,S0,S1,2783.210693,S0,S0,S0,0
34.03686905,S0,S1,2794.869629,S0,S0,S0,0
33.94315338,S0,S1,2799.833496,S0,S0,S0,0
33.93058777,S0,S1,2799.619141,S0,S0,S0,0
34.85762024,S0,S1,2804.489502,S0,S0,S0,0
34.85087967,S0,S1,2809.828613,S0,S0,S0,0
34.87799835,S0,S1,2813.472168,S0,S0,S0,0
34.0292778,S0,S1,2811.803467,S0,S0,S0,0
39.50028229,S0,S1,2756.916992,S0,S0,S0,0
39.12135696,S0,S1,2729.129639,S0,S0,S0,0
38.68962479,S0,S1,2704.292725,S0,S0,S0,0
38.07309723,S0,S1,2691.62207,S0,S0,S0,0
34.83923721,S0,S1,2749.374268,S0,S0,S0,0
34.99512482,S0,S1,2772.144531,S0,S0,S0,0
35.78328705,S0,S1,2807.20459,S0,S0,S1,0
39.5157547,S0,S1,2763.180176,S0,S0,S1,0
43.80437088,S0,S1,2700.664795,S0,S1,S1,0
42.19573212,S0,S1,2671.05542,S0,S1,S1,0
40.09329987,S0,S1,2648.017822,S0,S1,S0,0
38.26039124,S0,S1,2645.599121,S0,S1,S1,0
37.18085861,S0,S1,2651.695557,S0,S1,S2,0
36.44587326,S0,S1,2664.06665,S0,S1,S1,0
36.47600937,S0,S1,2665.377197,S0,S1,S1,0

51.87585449,S0,S1,3615.537598,S1,S0,S7,0
49.47713089,S0,S1,3698.910889,S1,S0,S6,0
48.93881989,S0,S1,3713.950439,S1,S0,S6,0
48.02793884,S0,S1,3712.421631,S1,S0,S6,0
47.72353363,S0,S1,3721.634766,S1,S0,S5,0
41.44467163,S0,S1,3579.585205,S1,S0,S4,500
42.98546982,S0,S1,3512.280518,S0,S0,S4,0
40.34567261,S0,S1,3462.095703,S0,S0,S4,500
36.84727097,S0,S1,3432.523926,S0,S0,S4,0
34.31417084,S0,S1,3415.443359,S0,S0,S4,0
32.50988388,S0,S1,3407.558838,S0,S0,S3,0
31.08898926,S0,S1,3408.404297,S0,S0,S4,0
31.06973076,S0,S1,3427.020996,S0,S0,S5,0
31.95132065,S0,S1,3500.914307,S0,S0,S4,0
33.14791489,S0,S1,3591.856689,S0,S0,S3,0
34.53631973,S0,S1,3638.711426,S0,S0,S3,0
35.59376526,S0,S1,3662.962158,S0,S0,S2,0
43.81890869,S0,S1,3580.518799,S0,S0,S2,0
43.46008682,S0,S1,3550.743164,S0,S0,S1,0
```

```

43.06192017,S0,S1,3526.490723,S0,S0,S1,0
42.45004654,S0,S1,3506.31665,S0,S0,S1,0
41.32126617,S0,S1,3497.342285,S0,S0,S2,500
39.46115112,S0,S1,3491.377686,S0,S0,S3,0
39.01203537,S0,S1,3488.942871,S0,S0,S4,0
44.46405792,S0,S1,3434.30127,S0,S1,S4,0
41.90384674,S0,S1,3409.533936,S0,S1,S3,500
39.92964554,S0,S1,3395.164551,S0,S1,S4,500
42.64624786,S0,S1,3348.588623,S0,S2,S4,0
43.69466782,S0,S1,3308.161621,S0,S2,S5,0
42.63303375,S0,S1,3276.046387,S0,S2,S6,0
40.90623093,S0,S1,3248.726807,S0,S2,S5,500
39.78145599,S0,S1,3226.668945,S0,S2,S6,500
38.17373657,S0,S1,3209.543213,S0,S2,S5,500
37.23124313,S0,S1,3198.50293,S0,S2,S6,0
38.19697189,S0,S1,3179.221191,S0,S3,S6,0
37.41596603,S0,S1,3171.404541,S0,S3,S7,0
36.38858795,S0,S1,3170.643066,S0,S3,S7,0
35.5459404,S0,S1,3174.780518,S0,S3,S6,0
35.83073425,S0,S1,3189.914063,S0,S3,S6,0
36.3717041,S0,S1,3202.859375,S0,S4,S6,0
36.93125153,S0,S1,3246.239746,S0,S3,S6,0
36.61388397,S0,S1,3303.019531,S0,S3,S5,0
37.56491089,S0,S1,3368.411621,S0,S3,S5,0

```

A.2.2. Resultados de J4.8

=== Run information ===

```

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    dTreeCont
Instances:    36180
Attributes:   8
              fms
              ffw
              d
              pd
              g
              msv
              fwv
              r
Test mode:    10-fold cross-validation

```

=== Classifier model (full training set) ===

J48 pruned tree

```

-----
pd <= 3201.311523: 0 (18563.0)
pd > 3201.311523
|  fms <= 42.009521
|  |  fms <= 38.010746: 0 (4604.0)
|  |  fms > 38.010746
|  |  |  pd <= 3401.760986: 500 (1472.0)
|  |  |  pd > 3401.760986
|  |  |  |  pd <= 3600.300293
|  |  |  |  |  fms <= 40.011276: 0 (468.0)
|  |  |  |  |  fms > 40.011276: 500 (389.0)
|  |  |  |  |  pd > 3600.300293: 0 (1048.0)
|  |  |  |  |  pd > 42.009521
|  |  |  |  |  |  pd <= 4002.345947: 0 (7674.0)
|  |  |  |  |  |  pd > 4002.345947
|  |  |  |  |  |  |  fms <= 46.007675
|  |  |  |  |  |  |  |  fms <= 44.017689: 0 (108.0)
|  |  |  |  |  |  |  |  fms > 44.017689
|  |  |  |  |  |  |  |  |  pd <= 4399.680664: 500 (87.0)
|  |  |  |  |  |  |  |  |  pd > 4399.680664: 0 (41.0)

```

```

| | | fms > 46.007675: 0 (1726.0)

Number of Leaves : 11

Size of the tree : 21

Time taken to build model: 0.99 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      36170          99.9724 %
Incorrectly Classified Instances     10            0.0276 %
Kappa statistic                     0.9973
Mean absolute error                  0.0003
Root mean squared error              0.0166
Relative absolute error              0.2712 %
Root relative squared error          7.3659 %
Total Number of Instances           36180

=== Detailed Accuracy By Class ===

TP Rate  FP Rate  Precision  Recall  F-Measure  Class
1         0.004     1          1       1          0
0.996    0         0.999     0.996  0.997     500

=== Confusion Matrix ===

      a      b  <-- classified as
34230  2 |   a = 0
      8 1940 |   b = 500

```

A.3. Modelos de transición (RBD).

A.3.1. Muestra archivo de ejemplos de Elvira para la acción 0: abrir válvula de agua de alimentación (*+fwv*)

```

data-base base_nodos{

number-of-cases = 7191;

// Network Variables

// Q
node Q (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 11;
states = (q0 q1 q2 q3 q4 q5 q6 q7 q8 q9 q10);
}

// ffw
node ffw (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 4;
states = (S0 S1 S2 S3);
}

// d
node d (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 2;
states = (S0 S1);
}

```

```

}

// g
node g (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 4;
states = (S0 S1 S2 S3);
}

// msv
node msv (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 9;
states = (S0 S1 S2 S3 S4 S5 S6 S7 S8);
}

// fwv
node fwv (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 9;
states = (S0 S1 S2 S3 S4 S5 S6 S7 S8);
}

// Q_prime
node Q_prime (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 11;
states = (q0 q1 q2 q3 q4 q5 q6 q7 q8 q9 q10);
}

// ffw_prime
node ffw_prime (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 4;
states = (S0 S1 S2 S3);
}

// d_prime
node d_prime (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 2;
states = (S0 S1);
}

// g_prime
node g_prime (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 4;
states = (S0 S1 S2 S3);
}

// msv_prime
node msv_prime (finite-states){
kind-of-node = chance;
type-of-variable = finite-states;
num-states = 9;
states = (S0 S1 S2 S3 S4 S5 S6 S7 S8);
}

// fwv_prime
node fwv_prime (finite-states){

```

```

kind-of-node = chance;
type-of-variable = finite-states;
num-states = 9;
states = (S0 S1 S2 S3 S4 S5 S6 S7 S8);
}

relation {

memory = true;

cases = (

[q0,S0,S1,S0,S0,S0,q0,S0,S1,S0,S0,S0]
[q0,S0,S1,S0,S0,S0,q0,S0,S1,S0,S0,S0]
[q0,S0,S1,S0,S0,S0,q0,S0,S1,S0,S0,S0]
[q0,S0,S1,S0,S0,S0,q0,S0,S1,S0,S0,S1]
[q0,S0,S1,S0,S1,S0,q0,S0,S1,S0,S1,S1]
[q0,S0,S1,S0,S1,S1,q0,S0,S1,S0,S1,S2]
[q0,S0,S1,S0,S1,S1,q0,S0,S1,S0,S1,S2]
[q0,S0,S1,S0,S1,S2,q0,S0,S1,S0,S1,S3]
[q0,S0,S1,S0,S2,S1,q0,S0,S1,S0,S2,S2]
[q0,S0,S1,S0,S3,S0,q0,S0,S1,S0,S3,S1]
[q0,S0,S1,S0,S4,S1,q0,S0,S1,S0,S4,S2]
[q0,S0,S1,S0,S4,S2,q0,S0,S1,S0,S4,S3]
[q0,S0,S1,S0,S6,S3,q0,S0,S1,S0,S6,S4]
[q0,S0,S1,S0,S7,S4,q0,S0,S1,S0,S7,S5]
[q0,S0,S1,S0,S7,S4,q0,S0,S1,S0,S7,S5]
[q0,S0,S1,S0,S8,S5,q0,S0,S1,S0,S8,S6]
[q0,S0,S1,S0,S8,S6,q0,S0,S1,S0,S8,S7]
[q0,S0,S1,S0,S8,S7,q0,S0,S1,S0,S8,S8]
[q0,S0,S1,S0,S8,S7,q0,S0,S1,S0,S8,S8]
[q0,S0,S1,S0,S8,S8,q0,S0,S1,S0,S8,S8]
[q0,S0,S1,S0,S6,S7,q6,S0,S1,S0,S6,S8]
[q0,S0,S1,S0,S8,S5,q0,S0,S1,S0,S8,S6]
[q0,S0,S1,S1,S8,S6,q0,S0,S1,S1,S8,S7]
[q0,S0,S1,S1,S8,S7,q0,S0,S1,S1,S8,S8]
[q0,S0,S1,S1,S8,S8,q0,S0,S1,S1,S8,S8]
[q0,S0,S1,S1,S6,S5,q0,S0,S1,S1,S6,S6]
[q0,S0,S1,S1,S6,S5,q0,S0,S1,S1,S6,S6]
[q0,S0,S1,S1,S8,S4,q0,S0,S1,S1,S8,S5]
[q0,S0,S1,S1,S8,S5,q0,S0,S1,S1,S8,S6]
[q6,S0,S1,S2,S8,S5,q6,S0,S1,S2,S8,S6]
[q2,S0,S1,S2,S8,S2,q2,S0,S1,S2,S8,S3]
[q2,S0,S1,S2,S8,S3,q2,S0,S1,S2,S8,S4]
[q2,S0,S1,S2,S8,S4,q2,S0,S1,S2,S8,S5]
[q2,S0,S1,S2,S8,S4,q2,S0,S1,S2,S8,S5]
[q2,S0,S1,S2,S8,S5,q2,S0,S1,S2,S8,S6]
[q2,S0,S1,S2,S8,S5,q2,S0,S1,S2,S8,S6]
[q2,S0,S1,S2,S8,S6,q6,S0,S1,S2,S8,S7]
[q6,S0,S1,S2,S8,S6,q6,S0,S1,S2,S8,S7]
[q6,S0,S1,S2,S8,S7,q6,S0,S1,S2,S8,S8]
[q6,S0,S1,S2,S7,S8,q6,S0,S1,S2,S7,S8]
[q6,S0,S1,S2,S8,S7,q6,S0,S1,S2,S8,S8]
[q6,S0,S1,S2,S8,S8,q6,S0,S1,S2,S8,S8]
[q6,S0,S1,S2,S8,S8,q6,S0,S1,S2,S8,S8]

[q0,S0,S0,S0,S8,S4,q0,S0,S0,S0,S8,S5]
[q0,S0,S0,S0,S7,S4,q0,S0,S0,S0,S7,S5]
[q0,S0,S0,S0,S6,S4,q0,S0,S0,S0,S6,S5]
[q0,S0,S0,S0,S5,S2,q0,S0,S0,S0,S5,S3]
[q0,S0,S0,S0,S5,S3,q0,S0,S0,S0,S5,S4]
[q0,S0,S0,S0,S5,S4,q0,S0,S0,S0,S5,S5]
[q1,S0,S0,S0,S1,S3,q2,S0,S0,S0,S1,S3]
[q2,S0,S0,S0,S1,S3,q2,S0,S0,S0,S1,S4]
[q0,S0,S0,S0,S2,S4,q0,S0,S0,S0,S2,S5]
[q0,S0,S0,S0,S3,S4,q0,S0,S0,S0,S3,S5]
[q0,S0,S0,S0,S4,S5,q0,S0,S0,S0,S4,S6]
[q0,S0,S0,S0,S6,S5,q0,S0,S0,S0,S6,S6]

```

```

[q0,S0,S0,S0,S6,S6,q0,S0,S0,S0,S6,S7]
[q0,S0,S0,S0,S5,S5,q0,S0,S0,S0,S5,S6]
[q6,S0,S0,S0,S5,S5,q6,S0,S0,S0,S5,S6]
[q2,S0,S0,S0,S4,S5,q2,S0,S0,S0,S4,S5]
[q2,S0,S0,S0,S4,S5,q6,S0,S0,S0,S4,S6]
[q6,S0,S0,S0,S5,S6,q6,S0,S0,S0,S5,S7]
[q6,S0,S0,S0,S5,S6,q6,S0,S0,S0,S5,S7]
[q6,S0,S0,S0,S4,S7,q6,S0,S0,S0,S4,S8]
[q6,S0,S0,S0,S3,S8,q6,S0,S0,S0,S3,S8]
[q6,S0,S0,S0,S3,S8,q6,S0,S0,S0,S3,S8]
[q6,S0,S0,S0,S4,S8,q6,S0,S0,S0,S4,S8]
[q6,S0,S0,S0,S4,S8,q6,S0,S0,S0,S4,S8]
[q10,S0,S0,S0,S1,S7,q10,S0,S0,S0,S1,S8]
[q6,S0,S0,S0,S0,S7,q6,S0,S0,S0,S0,S7]
[q6,S0,S0,S0,S0,S7,q6,S0,S0,S0,S0,S8]
[q6,S0,S0,S0,S0,S8,q6,S0,S0,S0,S0,S8]
[q6,S0,S0,S0,S0,S8,q6,S0,S0,S0,S0,S8]
[q6,S0,S0,S0,S0,S8,q6,S0,S0,S0,S0,S8]
[q6,S0,S0,S0,S1,S8,q6,S0,S0,S0,S1,S8]
[q0,S0,S0,S0,S1,S4,q0,S0,S0,S0,S1,S5]
[q0,S0,S0,S0,S1,S5,q0,S0,S0,S0,S1,S5]
[q1,S0,S0,S0,S1,S5,q1,S0,S0,S0,S1,S6]
[q1,S0,S0,S0,S1,S6,q1,S0,S0,S0,S1,S6]
[q1,S0,S0,S0,S0,S6,q1,S0,S0,S0,S0,S7]
[q6,S0,S0,S0,S2,S4,q6,S0,S0,S0,S2,S5]
[q2,S0,S0,S0,S2,S5,q2,S0,S0,S0,S2,S6]
[q0,S0,S0,S0,S4,S6,q0,S0,S0,S0,S4,S7]
[q1,S0,S0,S0,S3,S0,q1,S0,S0,S0,S3,S1]
[q1,S0,S0,S0,S2,S0,q1,S0,S0,S0,S2,S1]
[q1,S0,S0,S0,S3,S1,q1,S0,S0,S0,S3,S2]
[q1,S0,S0,S0,S3,S1,q1,S0,S0,S0,S3,S2]

```

```

);
}
}

```

A.3.2. Resultados de aprendizaje con K2

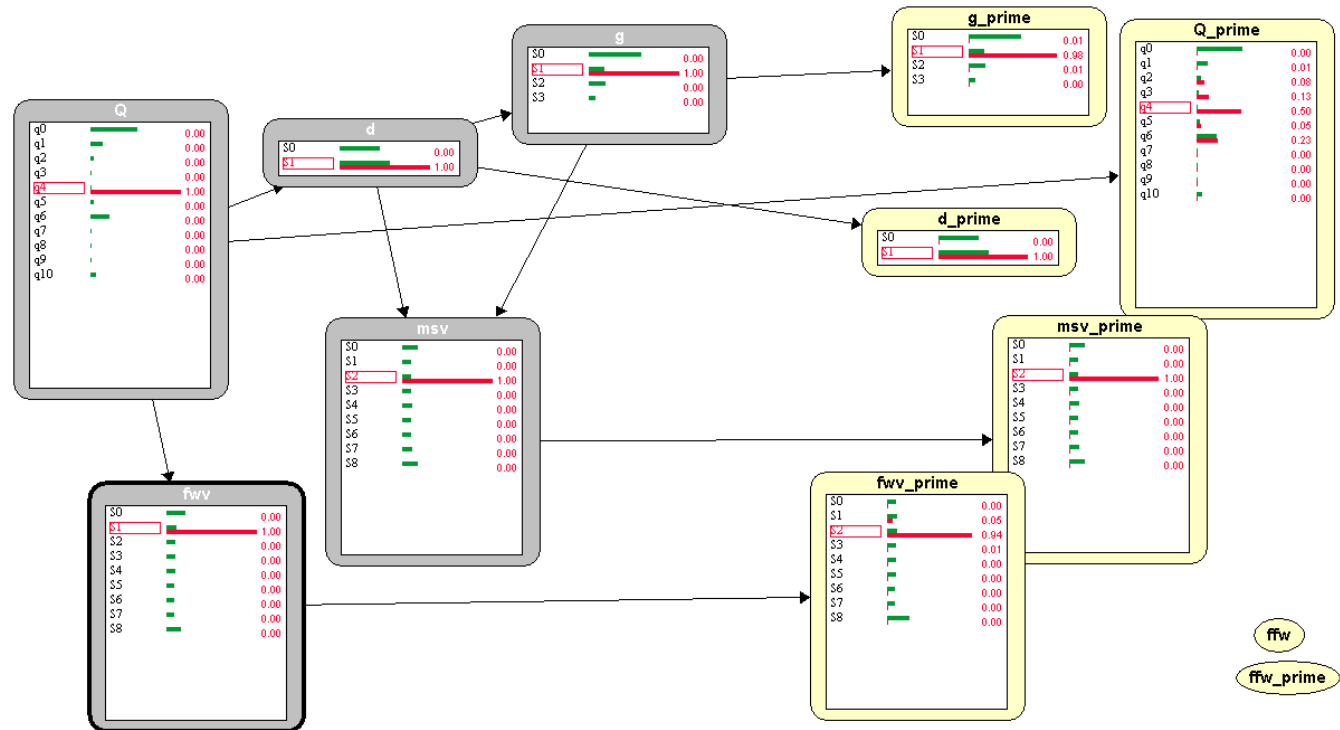


Figure A.1: Red Bayesiana Dinámica para la acción 0: *abrir válvula de agua de alimentación (+fww)* del problema de generación de vapor. La posición de la válvula *fww* cambia de *s1* a *s2* (abre) con probabilidad de 0.94. Existe probabilidad de cambios en el factor *Q*, el cual resume presión y flujo de vapor saliendo del domo. El resto de las variables sin cambios.

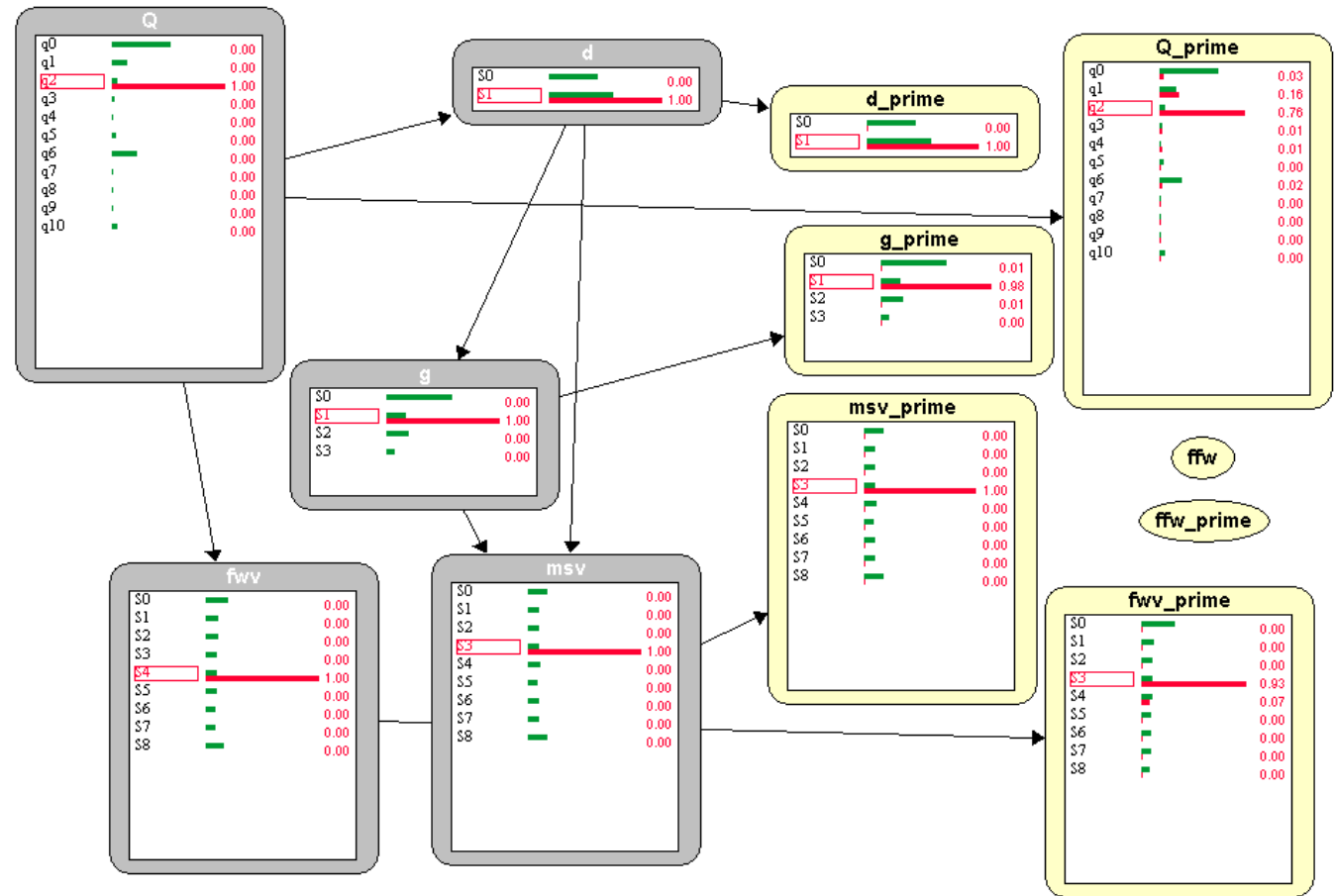


Figure A.2: Red Bayesiana Dinámica para la acción 1: *cerrar válvula de agua de alimentación (-fww)* del problema de generación de vapor. La posición de la válvula -fww cambia de s4 a s3 (cierra) con probabilidad de 0.93. Existen probabilidades de cambio en el factor Q y en la generación. El resto de las variables sin cambios.

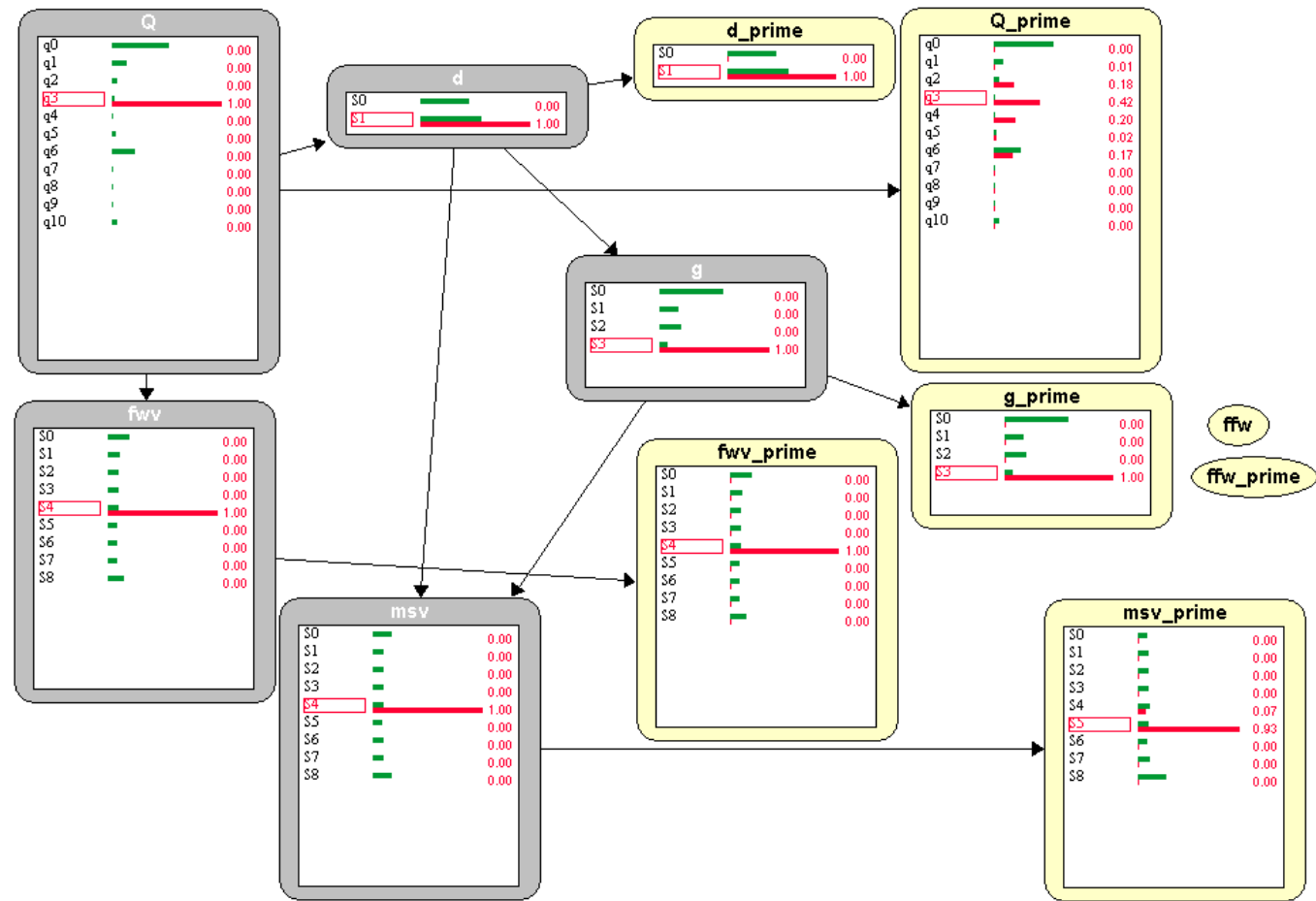


Figure A.3: Red Bayesiana Dinámica para la acción 2: *abrir válvula ed vapor principal (+msv)* del problema de generación de vapor. La posición de la válvula +msv cambia de s_4 a s_5 (abre) con probabilidad de 0.93. Si el proceso Q se encuentra en q_3 hay probabilidades de cambio a q_6 , q_2 o q_4 .

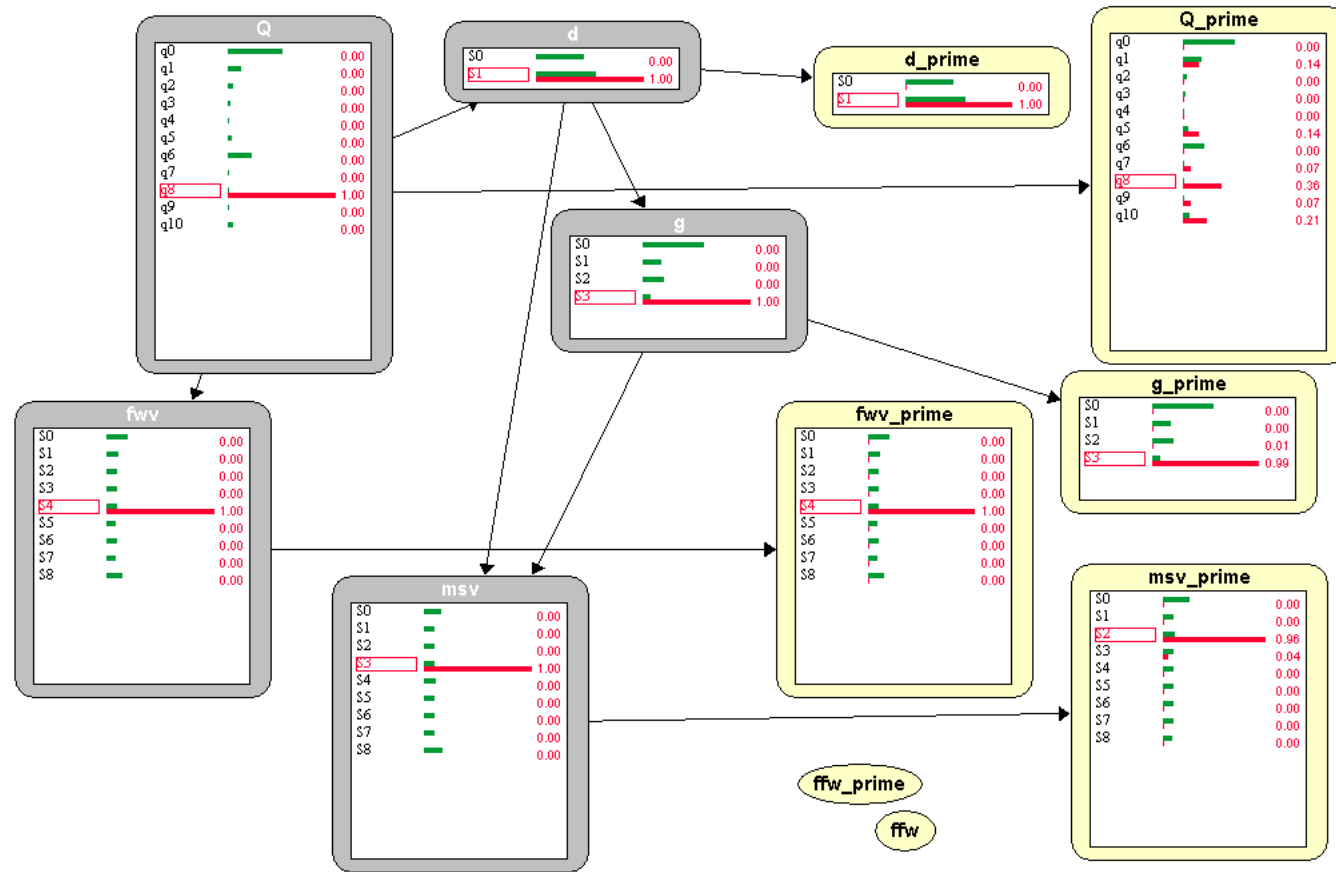


Figure A.4: Red Bayesiana Dinámica para la acción 3: Cerrar Main Steam Valve (-msv) del problema de generación de vapor. La posición de la válvula -msv cambia de s_3 a s_2 (cierra) con probabilidad de 0.96. Existe una distribución de probabilidades de cambio de los valores del factor Q a q_0 , q_5 , q_8 y q_9 .

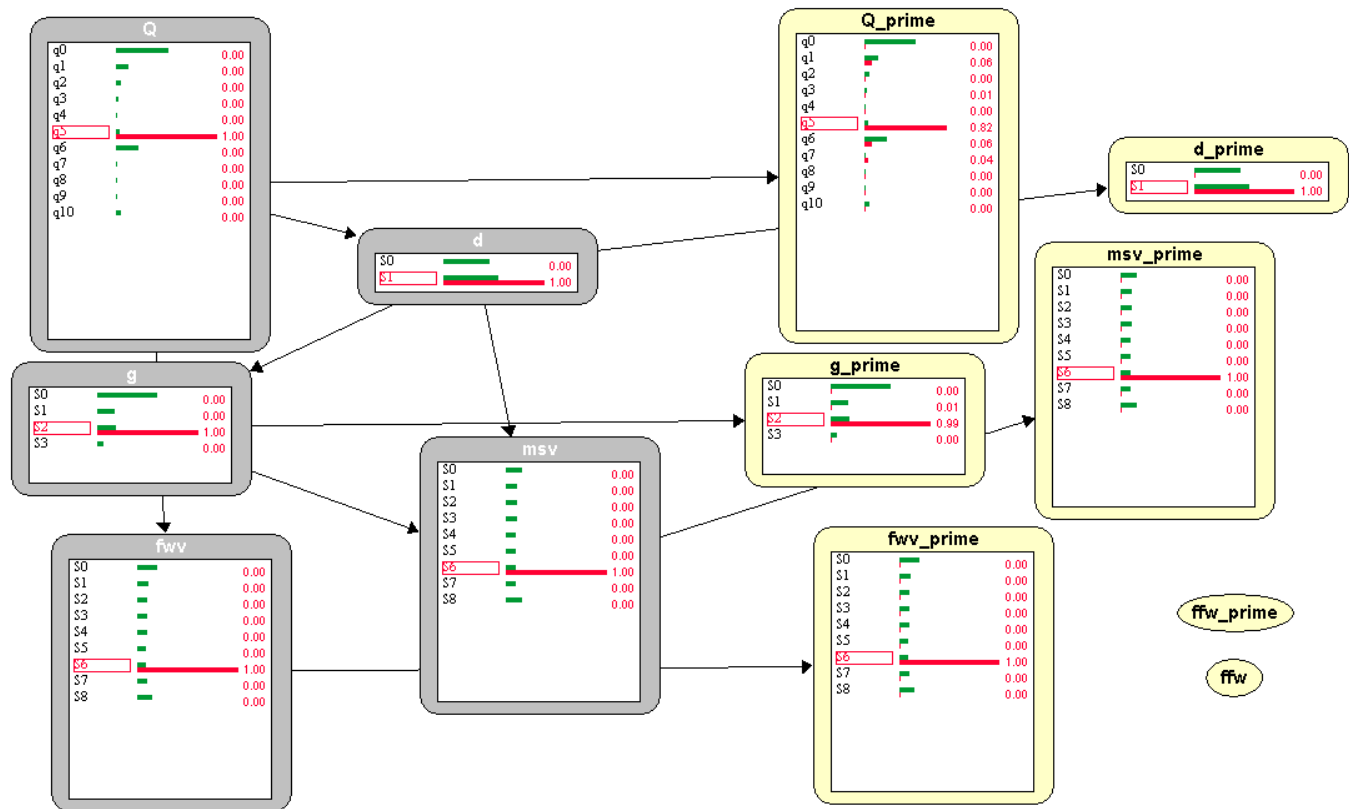


Figure A.5: Red Bayesiana Dinámica para la acción 4: Acción nula del problema de generación de vapor. Debido a la dinámica interna del proceso, aun cuando no hay acciones existe una distribución muy suave de probabilidades de cambio en el factor Q. El resto de las variables no se afectan.

A.4. Solución del MDP usando SPURD

A.4.1. Archivo de entrada

```
(variables (Q q0 q1 q2 q3 q4 q5 q6 q7 q8 q9 q10)
(d S0 S1)
(g S0 S1 S2 S3)
(msv S0 S1 S2 S3 S4 S5 S6 S7 S8)
(fwv S0 S1 S2 S3 S4 S5 S6 S7 S8 )
)
unnormalized
action act00
Q (Q (q0 (0.982 0.009 0.002 0.000 0.000 0.000 0.000 0.007 0.000 0.000 0.000 0.000 ))
(q1 (0.014 0.863 0.048 0.024 0.001 0.039 0.006 0.000 0.001 0.000 0.000 0.003 ))
(q2 (0.064 0.041 0.779 0.007 0.007 0.000 0.101 0.000 0.000 0.000 0.000 0.000 ))
(q3 (0.000 0.036 0.012 0.667 0.179 0.095 0.012 0.000 0.000 0.000 0.000 0.000 ))
(q4 (0.000 0.012 0.083 0.131 0.500 0.048 0.226 0.000 0.000 0.000 0.000 0.000 ))
(q5 (0.000 0.021 0.000 0.016 0.021 0.793 0.085 0.037 0.005 0.011 0.011 0.011 ))
(q6 (0.014 0.006 0.009 0.000 0.000 0.008 0.007 0.938 0.001 0.001 0.000 0.016 ))
(q7 (0.000 0.000 0.000 0.000 0.000 0.158 0.053 0.474 0.158 0.105 0.053 0.053 ))
(q8 (0.000 0.000 0.000 0.000 0.000 0.000 0.091 0.182 0.364 0.091 0.273 0.273 ))
(q9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.125 0.125 0.500 0.250 0.250 ))
(q10 (0.000 0.000 0.000 0.000 0.000 0.000 0.054 0.000 0.016 0.003 0.927 0.927 )))
d (d (S0 (1.000 0.000 ))
(S1 (0.000 1.000 )))
g (g (S0 (0.997 0.003 0.000 0.000 ))
(S1 (0.011 0.980 0.010 0.000 ))
(S2 (0.001 0.005 0.993 0.002 ))
(S3 (0.000 0.000 0.004 0.996 )))
msv (msv (S0 (1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.001 0.997 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.001 0.997 0.001 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.004 0.993 0.003 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.006 0.992 0.002 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.002 0.997 0.002 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.996 0.001 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 )))
fwv (fwv (S0 (0.488 0.512 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.000 0.053 0.941 0.005 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.061 0.937 0.002 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 0.074 0.926 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.000 0.079 0.920 0.001 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.078 0.922 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.062 0.938 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.070 0.930 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 )))
endaction

action act01
Q (Q (q0 (0.986 0.007 0.002 0.000 0.000 0.000 0.000 0.005 0.000 0.000 0.000 0.000 ))
(q1 (0.024 0.931 0.006 0.004 0.002 0.017 0.015 0.000 0.000 0.000 0.000 0.000 ))
(q2 (0.031 0.163 0.760 0.014 0.014 0.000 0.017 0.000 0.000 0.000 0.000 0.000 ))
(q3 (0.000 0.368 0.053 0.526 0.042 0.011 0.000 0.000 0.000 0.000 0.000 0.000 ))
(q4 (0.000 0.024 0.061 0.341 0.524 0.024 0.024 0.000 0.000 0.000 0.000 0.000 ))
(q5 (0.000 0.198 0.000 0.036 0.015 0.701 0.036 0.000 0.005 0.010 0.000 0.000 ))
(q6 (0.030 0.008 0.033 0.002 0.019 0.019 0.874 0.001 0.001 0.000 0.013 0.013 ))
(q7 (0.000 0.048 0.000 0.000 0.000 0.000 0.095 0.048 0.571 0.143 0.048 0.048 ))
(q8 (0.000 0.000 0.000 0.000 0.000 0.000 0.167 0.417 0.417 0.000 0.000 0.000 ))
(q9 (0.000 0.000 0.000 0.000 0.000 0.000 0.077 0.000 0.385 0.000 0.385 0.154 ))
(q10 (0.000 0.003 0.000 0.000 0.000 0.000 0.003 0.107 0.006 0.017 0.003 0.861 )))
d (d (S0 (1.000 0.000 ))
(S1 (0.000 1.000 )))
g (g (S0 (0.997 0.003 0.000 0.000 ))
(S1 (0.008 0.983 0.009 0.000 ))
(S2 (0.000 0.008 0.989 0.003 ))
(S3 (0.000 0.000 0.009 0.991 )))
msv (msv (S0 (1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
```

```

(S1 (0.000 0.997 0.003 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 0.997 0.003 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.003 0.996 0.001 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.998 0.002 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.002 0.997 0.002 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.001 0.996 0.003 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.001 0.999 )))
fwv (fwv (S0 (1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S1 (0.938 0.062 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.939 0.061 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.001 0.925 0.073 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.001 0.927 0.071 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.932 0.068 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.002 0.944 0.054 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.943 0.054 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.001 0.526 0.473 )))
endaction

action act02
Q (Q (q0 (0.991 0.003 0.001 0.000 0.000 0.000 0.000 0.004 0.000 0.000 0.000 )))
(q1 (0.066 0.565 0.109 0.044 0.021 0.070 0.106 0.005 0.003 0.001 0.009 ))
(q2 (0.139 0.016 0.631 0.010 0.003 0.000 0.201 0.000 0.000 0.000 0.000 ))
(q3 (0.000 0.010 0.180 0.420 0.200 0.020 0.170 0.000 0.000 0.000 0.000 ))
(q4 (0.000 0.015 0.046 0.015 0.508 0.000 0.415 0.000 0.000 0.000 0.000 ))
(q5 (0.000 0.014 0.000 0.010 0.033 0.405 0.395 0.033 0.024 0.005 0.081 ))
(q6 (0.048 0.009 0.002 0.000 0.001 0.001 0.929 0.000 0.001 0.000 0.010 ))
(q7 (0.000 0.000 0.000 0.000 0.000 0.000 0.048 0.190 0.095 0.000 0.667 ))
(q8 (0.000 0.000 0.000 0.000 0.000 0.000 0.429 0.048 0.143 0.000 0.381 ))
(q9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.286 0.714 ))
(q10 (0.000 0.003 0.000 0.000 0.000 0.000 0.000 0.142 0.000 0.000 0.855 )))
d (d (S0 (1.000 0.000 )))
(S1 (0.000 1.000 )))
g (g (S0 (0.997 0.003 0.000 0.000 )))
(S1 (0.004 0.991 0.005 0.000 ))
(S2 (0.000 0.004 0.994 0.002 ))
(S3 (0.000 0.000 0.000 1.000 )))
msv (msv (S0 (0.472 0.526 0.002 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S1 (0.000 0.089 0.911 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.000 0.063 0.934 0.003 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 0.050 0.950 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.069 0.930 0.001 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.000 0.067 0.932 0.002 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.056 0.942 0.001 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.087 0.913 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))
fwv (fwv (S0 (0.998 0.002 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S1 (0.001 0.999 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.003 0.997 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.002 0.997 0.002 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))
endaction

action act03
Q (Q (q0 (0.954 0.029 0.007 0.000 0.000 0.000 0.010 0.000 0.000 0.000 0.000 )))
(q1 (0.015 0.947 0.010 0.006 0.000 0.011 0.010 0.000 0.000 0.000 0.001 ))
(q2 (0.010 0.283 0.605 0.057 0.016 0.000 0.029 0.000 0.000 0.000 0.000 ))
(q3 (0.000 0.347 0.021 0.495 0.042 0.095 0.000 0.000 0.000 0.000 0.000 ))
(q4 (0.000 0.284 0.000 0.149 0.405 0.108 0.054 0.000 0.000 0.000 0.000 ))
(q5 (0.000 0.225 0.000 0.000 0.009 0.736 0.013 0.013 0.000 0.000 0.004 ))
(q6 (0.010 0.036 0.024 0.006 0.015 0.035 0.843 0.001 0.003 0.000 0.027 ))
(q7 (0.000 0.208 0.000 0.000 0.000 0.000 0.333 0.000 0.292 0.167 0.000 0.000 ))
(q8 (0.000 0.143 0.000 0.000 0.000 0.143 0.000 0.071 0.357 0.071 0.214 ))
(q9 (0.000 0.000 0.000 0.000 0.000 0.333 0.000 0.000 0.000 0.556 0.111 ))

```

```

(q10 (0.000 0.009 0.000 0.000 0.000 0.031 0.031 0.020 0.003 0.014 0.892 ))
d (d (S0 (1.000 0.000 ))
(S1 (0.000 1.000 )))
g (g (S0 (0.998 0.002 0.000 0.000 ))
(S1 (0.009 0.985 0.006 0.000 ))
(S2 (0.000 0.009 0.987 0.004 ))
(S3 (0.000 0.000 0.011 0.989 )))
msv (msv (S0 (1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.896 0.104 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.003 0.918 0.079 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.001 0.960 0.039 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.923 0.077 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.006 0.925 0.069 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.003 0.929 0.068 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.001 0.938 0.061 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.004 0.506 0.490 )))
fwv (fwv (S0 (0.999 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.001 0.999 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.002 0.998 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.001 0.997 0.001 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.000 0.998 0.002 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.998 0.001 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.002 0.998 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.001 0.999 )))
endaction

action act04
Q (Q (q0 (0.979 0.008 0.006 0.000 0.000 0.000 0.007 0.000 0.000 0.000 0.000 ))
(q1 (0.021 0.930 0.008 0.012 0.001 0.014 0.013 0.000 0.000 0.000 0.000 ))
(q2 (0.020 0.055 0.853 0.010 0.017 0.000 0.044 0.000 0.000 0.000 0.000 ))
(q3 (0.000 0.117 0.053 0.734 0.053 0.032 0.011 0.000 0.000 0.000 0.000 ))
(q4 (0.012 0.012 0.060 0.143 0.679 0.024 0.071 0.000 0.000 0.000 0.000 ))
(q5 (0.000 0.063 0.000 0.009 0.000 0.000 0.824 0.063 0.036 0.004 0.000 0.000 ))
(q6 (0.024 0.006 0.010 0.001 0.007 0.007 0.931 0.001 0.000 0.000 0.013 ))
(q7 (0.000 0.000 0.000 0.000 0.000 0.000 0.174 0.130 0.435 0.130 0.087 0.043 ))
(q8 (0.000 0.000 0.000 0.000 0.000 0.000 0.036 0.179 0.464 0.000 0.321 ))
(q9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.250 0.000 0.500 0.250 ))
(q10 (0.000 0.003 0.000 0.000 0.000 0.000 0.003 0.096 0.000 0.023 0.009 0.866 )))
d (d (S0 (1.000 0.000 ))
(S1 (0.000 1.000 )))
g (g (S0 (0.998 0.002 0.000 0.000 ))
(S1 (0.010 0.982 0.008 0.000 ))
(S2 (0.000 0.007 0.989 0.004 ))
(S3 (0.002 0.000 0.016 0.982 )))
msv (msv (S0 (0.999 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.001 0.999 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.999 0.001 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.003 0.997 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.001 0.998 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.001 0.000 0.000 0.000 0.000 0.999 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.999 0.001 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.002 0.998 )))
fwv (fwv (S0 (0.999 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.000 0.999 0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.997 0.003 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.001 0.997 0.001 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.001 0.999 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.000 0.998 0.002 0.000 0.000 ))
(S6 (0.000 0.000 0.002 0.000 0.000 0.000 0.998 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.998 0.002 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))
endaction

reward( Q ( q0 ( 0 ) ) ( q1 ( 0 ) ) ( q2 ( 500 ) ) ( q3 ( 0 ) ) ( q4 ( 500 ) ) ( q5 ( 0 ) ) (
q6 ( 0 ) ) ( q7 ( 0 ) ) ( q8 ( 500 ) ) ( q9 ( 0 ) ) ( q10 ( 0 ) ) )

```

discount 0.900000
tolerance 0.000001

A.4.2. Archivo de salida

----- Spudd stats -----

Discount factor is : 0.900000
Tolerance is: 0.000001
horizon is: -1.000000
The BIGADD limit is set to: 0
Target maximum memory use: 1086556160 bytes
Hard Limit on memory use 700000000 bytes
Iterations to convergence 177
Final execution time: 13.9300 seconds
Maximum memory usage: 14177012 bytes
Number of nodes in the action ADD: 52 internal nodes 6 leaves 58 total nodes
Number of nodes in the equivalent tree: 144 internal nodes 145 leaves 289 total nodes
Number of nodes in the value ADD: 233 internal nodes 27 leaves 260 total nodes
Number of nodes in the value tree: 685 internal nodes 686 leaves 1371 total nodes

----- other info from dd Manager -----

**** CUDD modifiable parameters ****
Hard limit for cache size: 14583333
Cache hit threshold for resizing: 30%
Garbage collection enabled: yes
Limit for fast unique table growth: 8750000
Maximum number of variables sifted per reordering: 1000
Maximum number of variable swaps per reordering: 2000000
Maximum growth while sifting a variable: 1.2
Dynamic reordering of BDDs enabled: no
Default BDD reordering method: 4
Dynamic reordering of ZDDs enabled: no
Default ZDD reordering method: 4
Realignment of ZDDs to BDDs enabled: no
Realignment of BDDs to ZDDs enabled: no
Dead nodes counted in triggering reordering: no
Group checking criterion: 7
Recombination threshold: 0
Symmetry violation threshold: 0
Arc violation threshold: 0
GA population size: 0
Number of crossovers for GA: 0
**** CUDD non-modifiable parameters ****
Memory in use: 14177012
Peak number of nodes: 91980
Peak number of live nodes: 5958
Number of BDD variables: 30
Number of ZDD variables: 0
Number of cache entries: 262144
Number of cache look-ups: 10967237
Number of cache hits: 1393651
Number of cache insertions: 9572472
Number of cache collisions: 1728408
Number of cache deletions: 7844047
Cache used slots = 100.00% (expected 0.01%)
Soft limit for cache size: 129024
Number of buckets in unique table: 32256
Used buckets in unique table: 6.98% (expected 7.01%)
Number of BDD and ADD nodes: 2446
Number of ZDD nodes: 0
Number of dead BDD and ADD nodes: 17
Number of dead ZDD nodes: 0

Total number of nodes allocated: 6900101
Total number of nodes reclaimed: 483505
Garbage collections so far: 96
Time for garbage collection: 4.18 sec
Reorderings so far: 0
Time for reordering: 0.00 sec
Next reordering threshold: 4004

Apéndice B

Problema 2.

Problema de la caldera (HRSG) con la siguiente configuración: seis variables de estado continuas, 1 variable lógica, 9 acciones, y función de recompensa con 3 dimensiones (presión y flujo de vapor, y generación). Solución híbrida con cuatro variables discretizadas y tres continuas.

B.1. Función de recompensa

```
=== Run information ===

Scheme:      weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:    dTreeCont
Instances:   27770
Attributes:  8
             fms
             ffw
             d
             pd
             g
             msv
             fwv
             r
Test mode:   10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
-----

g <= 14998.35449
|  g <= 9999.947266
|  |  pd <= 3201.180664: 0 (5903.0)
|  |  pd > 3201.180664
|  |  |  fms <= 42.005672
|  |  |  |  fms <= 38.020649: 0 (1780.0)
|  |  |  |  fms > 38.020649
|  |  |  |  |  pd <= 3401.084473: 500 (691.0)
|  |  |  |  |  pd > 3401.084473
|  |  |  |  |  |  fms <= 40.020737: 0 (345.0)
|  |  |  |  |  |  fms > 40.020737
|  |  |  |  |  |  |  pd <= 3602.523926: 500 (207.0)
|  |  |  |  |  |  |  pd > 3602.523926: 0 (164.0)
|  |  |  |  |  |  |  |  fms > 42.005672
```

```

| | | | | pd <= 4016.722412: 0 (2816.0)
| | | | | pd > 4016.722412
| | | | | | fms <= 46.052265
| | | | | | | fms <= 43.930325: 0 (24.0)
| | | | | | | fms > 43.930325: 500 (32.0/1.0)
| | | | | | fms > 46.052265: 0 (211.0)
| g > 9999.947266
| | fms <= 59.989357
| | | pd <= 3201.846436
| | | | fms <= 31.721706
| | | | | fms <= 29.849854: 0 (4.0)
| | | | | fms > 29.849854: 100 (21.0)
| | | | | fms > 31.721706: 100 (2342.0)
| | | | pd > 3201.846436
| | | | | fms <= 42.001286
| | | | | | fms <= 38.003387
| | | | | | | fms <= 29.974407: 0 (23.0)
| | | | | | | fms > 29.974407: 100 (751.0)
| | | | | | fms > 38.003387
| | | | | | | pd <= 3402.795654: 600 (227.0)
| | | | | | | pd > 3402.795654
| | | | | | | | pd <= 3598.309326
| | | | | | | | | fms <= 40.030392: 100 (73.0)
| | | | | | | | | fms > 40.030392: 600 (63.0)
| | | | | | | | | pd > 3598.309326: 100 (174.0)
| | | | | | fms > 42.001286
| | | | | | | pd <= 4003.959717: 100 (957.0)
| | | | | | | pd > 4003.959717
| | | | | | | | fms <= 45.635185
| | | | | | | | | g <= 14010.94922
| | | | | | | | | | pd <= 4023.323486: 600 (2.0)
| | | | | | | | | | pd > 4023.323486: 100 (9.0)
| | | | | | | | | | g > 14010.94922: 600 (8.0)
| | | | | | | | | | fms > 45.635185: 100 (34.0)
| | | | | | fms > 59.989357: 0 (278.0)
| g > 14998.35449
| | fms <= 60.02964
| | | pd <= 3201.044434
| | | | fms <= 31.445616
| | | | | fms <= 30.118378: 0 (4.0)
| | | | | fms > 30.118378: 250 (21.0)
| | | | | fms > 31.445616: 250 (4602.0)
| | | | pd > 3201.044434
| | | | | fms <= 42.009426
| | | | | | fms <= 38.00835
| | | | | | | fms <= 30.749098
| | | | | | | | fms <= 30.044147: 0 (4.0)
| | | | | | | | fms > 30.044147: 250 (13.0)
| | | | | | | | fms > 30.749098: 250 (984.0)
| | | | | | fms > 38.00835
| | | | | | | pd <= 3400.970947: 750 (585.0)
| | | | | | | pd > 3400.970947
| | | | | | | | pd <= 3600.820557
| | | | | | | | | fms <= 40.010345: 250 (162.0)
| | | | | | | | | fms > 40.010345: 750 (165.0)
| | | | | | | | | pd > 3600.820557: 250 (339.0)
| | | | | | fms > 42.009426
| | | | | | | pd <= 4000.881592: 250 (2942.0)
| | | | | | | pd > 4000.881592
| | | | | | | | fms <= 46.000904
| | | | | | | | | fms <= 44.066505: 250 (34.0)
| | | | | | | | | fms > 44.066505
| | | | | | | | | | pd <= 4388.857422: 750 (25.0)
| | | | | | | | | | pd > 4388.857422: 250 (4.0)
| | | | | | | | | | fms > 46.000904: 250 (163.0)
| | | | | | fms > 60.02964: 0 (584.0)

```

Number of Leaves : 41

Size of the tree : 81

Time taken to build model: 3.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances	27732	99.8632 %
Incorrectly Classified Instances	38	0.1368 %
Kappa statistic	0.998	
Mean absolute error	0.0005	
Root mean squared error	0.0212	
Relative absolute error	0.2157 %	
Root relative squared error	6.3427 %	
Total Number of Instances	27770	

=== Detailed Accuracy By Class ===

TP Rate	FP Rate	Precision	Recall	F-Measure	Class
0.999	0.001	0.999	0.999	0.999	0
0.999	0	0.997	0.999	0.998	100
1	0.001	0.999	1	0.999	250
0.994	0	0.997	0.994	0.995	500
0.98	0	0.987	0.98	0.983	600
0.994	0	0.997	0.994	0.995	750

=== Confusion Matrix ===

	a	b	c	d	e	f	<-- classified as
12129	5	4	3	0	0	0	a = 0
1	4355	1	0	4	0	0	b = 100
1	0	9261	0	0	2	0	c = 250
6	0	0	923	0	0	0	d = 500
0	6	0	0	294	0	0	e = 600
0	0	5	0	0	770	0	f = 750

B.2. Solución del MDP usando SPUD

B.2.1. Archivo de entrada

```
(variables (Q q0 q1 q2 q3 q4 q5 q6 q7 q8 q9 q10 q11 q12 q13 q14 q15 q16 q17 q18 q19 q20 q21 q22
q23 q24 q25 q26 q27 q28 q29 q30 q31 q32 q33 q34 q35 q36 q37 q38 q39 q40)
(d S0 S1)
(msv S0 S1 S2 S3 S4 S5 S6 S7 S8 S9)
(fwv S0 S1 S2 S3 S4 S5 S6 S7 S8 S9 )
)
unnormalized
action act00
Q (d (S0 (0.464 0.143 0.069 0.036 0.018 0.019 0.235 0.000 0.002 0.013 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.110 0.041 0.013 0.008 0.005 0.002 0.057 0.000 0.000 0.004 0.000 0.000 0.112 0.000 0.032
0.011 0.001 0.003 0.011 0.050 0.000 0.000 0.000 0.001 0.013 0.000 0.000 0.225 0.000 0.000 0.039
0.028 0.008 0.008 0.019 0.157 0.001 0.002 0.000 0.007 0.030 )))
d (d (S0 (1.000 0.000 ))
(S1 (0.000 1.000 )))
msv (msv (S0 (0.995 0.005 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.010 0.990 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.004 0.989 0.007 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.013 0.987 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.004 0.989 0.007 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.007 0.990 0.003 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.993 0.007 0.000 )))
```

```

(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.993 0.003 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.002 0.998 ))
fww (fww (S0 (0.327 0.635 0.038 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.000 0.145 0.821 0.034 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.086 0.879 0.035 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 0.084 0.875 0.041 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.000 0.071 0.900 0.029 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.097 0.897 0.006 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.077 0.892 0.031 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.077 0.871 0.052 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.096 0.903 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))
endaction

action act01
Q (d (S0 (0.511 0.146 0.042 0.027 0.009 0.015 0.218 0.007 0.006 0.017 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.001 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.118 0.049 0.009 0.007 0.005 0.003 0.050 0.000 0.000 0.004 0.000 0.112 0.002 0.035
0.011 0.004 0.004 0.011 0.045 0.000 0.000 0.001 0.003 0.010 0.000 0.002 0.235 0.000 0.001 0.052
0.026 0.007 0.007 0.015 0.139 0.000 0.000 0.001 0.005 0.024 )))
d (d (S0 (0.999 0.001 )))
(S1 (0.000 1.000 )))
msv (msv (S0 (0.997 0.000 0.000 0.002 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.006 0.994 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.003 0.996 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.004 0.007 0.989 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.008 0.979 0.013 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.004 0.996 0.000 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.993 0.007 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.004 0.996 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.009 0.984 0.006 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))
fww (fww (S0 (1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.913 0.086 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.019 0.906 0.075 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.031 0.888 0.081 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.029 0.903 0.068 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.032 0.896 0.071 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.003 0.000 0.027 0.929 0.041 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.048 0.858 0.093 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.035 0.872 0.094 0.000 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.037 0.662 0.301 )))
endaction

action act02
Q (d (S0 (0.544 0.081 0.057 0.011 0.018 0.017 0.255 0.001 0.002 0.012 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.127 0.018 0.011 0.004 0.002 0.002 0.063 0.000 0.000 0.003 0.000 0.001 0.121 0.000 0.020
0.010 0.004 0.005 0.006 0.053 0.000 0.000 0.000 0.001 0.020 0.000 0.000 0.240 0.000 0.001 0.035
0.030 0.006 0.007 0.011 0.150 0.001 0.002 0.000 0.010 0.035 )))
d (d (S0 (1.000 0.000 )))
(S1 (0.000 1.000 )))
msv (msv (S0 (0.321 0.653 0.025 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.000 0.104 0.847 0.049 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.084 0.883 0.032 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 0.072 0.902 0.026 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.089 0.900 0.011 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.000 0.080 0.901 0.019 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.058 0.917 0.026 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.063 0.890 0.044 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.000 0.097 0.900 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))
fww (fww (S0 (0.995 0.005 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.000 0.996 0.004 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.003 0.997 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 )))

```

```

(S4 (0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.984 0.013 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.996 0.004 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.002 0.998 ))
endaction

action act03
Q (d (S0 (0.450 0.196 0.063 0.029 0.012 0.021 0.200 0.005 0.001 0.023 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.095 0.058 0.011 0.013 0.005 0.004 0.051 0.000 0.000 0.004 0.000 0.002 0.098 0.002 0.053
0.009 0.004 0.003 0.009 0.047 0.000 0.000 0.000 0.001 0.011 0.001 0.002 0.219 0.001 0.001 0.066
0.023 0.009 0.007 0.020 0.133 0.003 0.001 0.000 0.009 0.023 )))
d (d (S0 (1.000 0.000 ))
(S1 (0.000 1.000 )))
msv (msv (S0 (0.995 0.005 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.906 0.094 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.022 0.885 0.085 0.007 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.017 0.896 0.087 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.004 0.000 0.039 0.886 0.071 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.011 0.908 0.081 0.000 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.035 0.896 0.069 0.000 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.017 0.908 0.075 0.000 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.034 0.853 0.113 0.000 0.000 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.049 0.643 0.308 )))
fwv (fwv (S0 (0.998 0.002 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.004 0.992 0.004 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.003 0.986 0.007 0.003 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.003 0.990 0.007 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.997 0.003 0.000 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000 0.000 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.996 0.004 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.002 0.997 )))
endaction

action act04
Q (d (S0 (0.499 0.075 0.069 0.026 0.014 0.014 0.277 0.004 0.003 0.019 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 ))
(S1 (0.130 0.016 0.013 0.006 0.003 0.001 0.066 0.000 0.000 0.005 0.000 0.000 0.130 0.000 0.020
0.013 0.004 0.002 0.003 0.051 0.000 0.000 0.000 0.002 0.016 0.000 0.000 0.228 0.000 0.000 0.026
0.029 0.006 0.008 0.013 0.166 0.001 0.000 0.000 0.005 0.035 )))
d (d (S0 (1.000 0.000 ))
(S1 (0.000 1.000 )))
msv (msv (S0 (0.338 0.635 0.027 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.003 0.108 0.864 0.024 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.104 0.866 0.030 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 0.110 0.863 0.027 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.061 0.913 0.027 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.081 0.899 0.020 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.077 0.886 0.037 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.101 0.865 0.034 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.101 0.899 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))
fwv (fwv (S0 (0.339 0.614 0.048 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S1 (0.000 0.142 0.810 0.048 0.000 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S2 (0.000 0.000 0.075 0.869 0.056 0.000 0.000 0.000 0.000 0.000 0.000 ))
(S3 (0.000 0.000 0.000 0.073 0.900 0.027 0.000 0.000 0.000 0.000 0.000 ))
(S4 (0.000 0.000 0.000 0.000 0.083 0.895 0.022 0.000 0.000 0.000 0.000 ))
(S5 (0.000 0.000 0.000 0.000 0.000 0.080 0.878 0.042 0.000 0.000 0.000 ))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.056 0.915 0.028 0.000 0.000 ))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.078 0.891 0.031 ))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.104 0.896 ))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 )))

```



```

(S1 (0.093 0.061 0.013 0.005 0.003 0.003 0.040 0.000 0.000 0.004 0.001 0.001 0.113 0.004 0.065
0.011 0.006 0.002 0.012 0.034 0.000 0.001 0.001 0.002 0.009 0.000 0.001 0.210 0.000 0.001 0.070
0.031 0.007 0.008 0.017 0.130 0.002 0.001 0.000 0.010 0.031 )))
d (d (S0 (1.000 0.000 )))
(S1 (0.000 1.000 )))
msv (msv (S0 (1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S1 (0.897 0.103 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S2 (0.016 0.905 0.078 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S3 (0.000 0.029 0.882 0.088 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S4 (0.000 0.000 0.032 0.875 0.092 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S5 (0.000 0.000 0.000 0.048 0.889 0.063 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S6 (0.000 0.000 0.000 0.000 0.030 0.868 0.102 0.000 0.000 0.000 0.000 0.000 )))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.027 0.884 0.089 0.000 0.000 0.000 )))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.042 0.873 0.085 0.000 0.000 )))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.027 0.615 0.357 )))
fww (fww (S0 (1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S1 (0.917 0.083 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S2 (0.041 0.891 0.068 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S3 (0.000 0.015 0.914 0.071 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S4 (0.000 0.000 0.033 0.907 0.060 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S5 (0.000 0.000 0.000 0.027 0.868 0.105 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S6 (0.000 0.000 0.000 0.000 0.000 0.026 0.898 0.075 0.000 0.000 0.000 0.000 )))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.038 0.892 0.069 0.000 0.000 0.000 )))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.036 0.822 0.141 0.000 0.000 )))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.036 0.599 0.365 )))
endaction

action act08
Q (d (S0 (0.486 0.118 0.058 0.032 0.025 0.010 0.244 0.004 0.001 0.022 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
0.000 0.000 0.000 0.000 )))
(S1 (0.105 0.040 0.008 0.009 0.005 0.003 0.052 0.000 0.001 0.006 0.000 0.002 0.117 0.000 0.044
0.013 0.002 0.005 0.010 0.048 0.000 0.000 0.001 0.001 0.015 0.000 0.003 0.222 0.000 0.000 0.046
0.027 0.011 0.010 0.016 0.140 0.002 0.001 0.000 0.004 0.032 )))
d (d (S0 (1.000 0.000 )))
(S1 (0.000 1.000 )))
msv (msv (S0 (0.995 0.005 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S1 (0.003 0.996 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S2 (0.000 0.000 0.996 0.004 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S3 (0.000 0.000 0.000 0.007 0.990 0.000 0.003 0.000 0.000 0.000 0.000 0.000 )))
(S4 (0.000 0.000 0.000 0.008 0.985 0.008 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S5 (0.000 0.000 0.000 0.000 0.000 0.000 0.992 0.007 0.000 0.000 0.000 0.000 )))
(S6 (0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.997 0.000 0.000 0.000 0.000 )))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.994 0.003 0.000 0.000 )))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.993 0.007 )))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.004 0.996 )))
fww (fww (S0 (0.997 0.003 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S1 (0.003 0.993 0.003 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S2 (0.000 0.000 1.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S3 (0.000 0.003 0.003 0.990 0.003 0.000 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S4 (0.000 0.000 0.000 0.000 0.003 0.994 0.003 0.000 0.000 0.000 0.000 0.000 )))
(S5 (0.000 0.000 0.000 0.000 0.004 0.996 0.000 0.000 0.000 0.000 0.000 0.000 )))
(S6 (0.000 0.000 0.000 0.000 0.000 0.007 0.987 0.007 0.000 0.000 0.000 0.000 )))
(S7 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.003 0.993 0.003 0.000 0.000 )))
(S8 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 0.000 )))
(S9 (0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 1.000 0.000 )))
endaction

reward( Q ( q0 ( 0 ) ) ( q1 ( 0 ) ) ( q2 ( 500 ) ) ( q3 ( 0 ) ) ( q4 ( 500 ) ) ( q5 ( 0 ) ) (
q6 ( 0 ) ) ( q7 ( 0 ) ) ( q8 ( 500 ) ) ( q9 ( 0 ) ) ( q10 ( 0 ) ) ( q11 ( 100 ) ) ( q12 ( 100
) ) ( q13 ( 0 ) ) ( q14 ( 100 ) ) ( q15 ( 600 ) ) ( q16 ( 100 ) ) ( q17 ( 600 ) ) ( q18 ( 100
) ) ( q19 ( 100 ) ) ( q20 ( 600 ) ) ( q21 ( 100 ) ) ( q22 ( 600 ) ) ( q23 ( 100 ) ) ( q24 ( 0
) ) ( q25 ( 0 ) ) ( q26 ( 250 ) ) ( q27 ( 250 ) ) ( q28 ( 0 ) ) ( q29 ( 250 ) ) ( q30 ( 250
) ) ( q31 ( 750 ) ) ( q32 ( 250 ) ) ( q33 ( 750 ) ) ( q34 ( 250 ) ) ( q35 ( 250 ) ) ( q36 ( 250
) ) ( q37 ( 750 ) ) ( q38 ( 250 ) ) ( q39 ( 250 ) ) ( q40 ( 0 ) ) )

discount 0.900000
tolerance 0.000001

```

B.2.2. Archivo de salida

```
----- Spudd stats -----
Discount factor is : 0.900000
Tolerance is: 0.000001
horizon is: -1.000000
The BIGADD limit is set to: 0
Target maximum memory use: 1086556160 bytes
Hard Limit on memory use 700000000 bytes
Iterations to convergence 182
Final execution time: 9.4000 seconds
Maximum memory usage: 13401524 bytes
Number of nodes in the action ADD: 12 internal nodes 2 leaves 14 total nodes
Number of nodes in the equivalent tree: 33 internal nodes 34 leaves 67 total nodes
Number of nodes in the value ADD: 163 internal nodes 22 leaves 185 total nodes
Number of nodes in the value tree: 1246 internal nodes 1247 leaves 2493 total nodes

----- other info from dd Manager -----

**** CUDD modifiable parameters ****
Hard limit for cache size: 14583333
Cache hit threshold for resizing: 30%
Garbage collection enabled: yes
Limit for fast unique table growth: 8750000
Maximum number of variables sifted per reordering: 1000
Maximum number of variable swaps per reordering: 2000000
Maximum growth while sifting a variable: 1.2
Dynamic reordering of BDDs enabled: no
Default BDD reordering method: 4
Dynamic reordering of ZDDs enabled: no
Default ZDD reordering method: 4
Realignment of ZDDs to BDDs enabled: no
Realignment of BDDs to ZDDs enabled: no
Dead nodes counted in triggering reordering: no
Group checking criterion: 7
Recombination threshold: 0
Symmetry violation threshold: 0
Arc violation threshold: 0
GA population size: 0
Number of crossovers for GA: 0
**** CUDD non-modifiable parameters ****
Memory in use: 13401524
Peak number of nodes: 47012
Peak number of live nodes: 5397
Number of BDD variables: 30
Number of ZDD variables: 0
Number of cache entries: 262144
Number of cache look-ups: 6263005
Number of cache hits: 879847
Number of cache insertions: 5381606
Number of cache collisions: 538090
Number of cache deletions: 4843491
Cache used slots = 100.00% (expected 0.01%)
Soft limit for cache size: 73728
Number of buckets in unique table: 18432
Used buckets in unique table: 18.22% (expected 18.16%)
Number of BDD and ADD nodes: 3916
Number of ZDD nodes: 0
Number of dead BDD and ADD nodes: 25
Number of dead ZDD nodes: 0
Total number of nodes allocated: 4147394
Total number of nodes reclaimed: 1151387
Garbage collections so far: 104
Time for garbage collection: 2.93 sec
```


Reorderings so far: 0
Time for reordering: 0.00 sec
Next reordering threshold: 4004

Apéndice C

Algoritmos de refinamiento de estados cualitativos.

Figura C.1: Algoritmo de refinamiento.

```
función ref-edos-cual(arbolQ, fmdp, atr)
devuelve nuevoMDP
entradas: arbolQ: árbol cualitativo original.
         fmdp: mdp cualitativo original.
         atr: información de atributos.
local: agenda, lista de estadosQ probados inicialmente vacía.
      MAX, máximo número de particiones.
      i=0;
repetir
  particion=particion(fmdp, arbolQ)
  candidato=selec-edo-a-partir(particion, atr, agenda)
  arbolQAux=arbolQ
  fmdpAux=fmdp
  arbolQ=parte-Edo(particion, candidato, arbolQ, atr)
  fmdp= aprende-y-resuelveMDP(arbolQ, fmdp)
  if cambia política en nueva partición
  then agenda ← candidato
  else
    arbolQ=arbolQAux
    fmdp=fmdpAux
hasta que candidato != null o i ≥ MAX
devolver arbolQ y fmdp
```

Figura C.2: Algoritmo de selección de estado a bisectar.

```
función selec-edo-a-partir(particion, atr, agenda)
devuelve candidato
entradas: particion: conjunto de estadosQ
        atr: información de atributos
        agenda: lista de estadosQ evaluados, inicialmente vacía.
local: candidato = null, maxVarianza=  $-\infty$ 
lhv = menor-hiperVolumen(particion, atributo)
por cada estadoQ q de la particion tal que
    q no esta en la agenda hacer
    r=region(q,particion)
    varianza_q=varianza(r)
    hv_q=hipervol(q, atributo)
    if varianza_q > maxVarianza and hv_q > 2 lhv then
        maxVarianza=varianza_q
        candidato=q
    end
devolver candidato
```

Figura C.3: Algoritmo de bisección de estados.

```
función parte-edo(particion, candidato, arbolQ, atr)
devuelve arbolQ
entradas: particion: conjunto de estados q
        candidato: estado q a particionar
        arbolQ: árbol cualitativo actual.
        atr: información de atributos
nodo=getNode(arbolQ, candidato)
frontera=inBounds(particion, candidato)
q_gug=q_greaterUtilityGradient(candidato, frontera)
dim=selec-dimension(nodo, q_gug)
arbolQ=biseciona-nodo(nodo, dim)
devuelve arbolQ
```