Procesos de Decisión de Markov rápidos mediante Reglas de Asociación

Ma.de Guadalupe García-Hernández[1], José Ruiz-Pinales[1], Alberto Reyes[2], Eva Onaindía[3], J. Gabriel Aviña-Cervantes[1], Donato Hernández-Fusilier[1]

[1]Universidad de Guanajuato, Comunidad de Palo Blanco S/N, C.P. 36701, Salamanca, Guanajuato, México {garciag, pinales, avina, donato}@salamanca.ugto.mx
[2]Instituto de Investigaciones Eléctricas, Reforma 113, C.P.62490, Temixco, Morelos, México, areyes@iie.org.mx
[3]Universidad Politécnica de Valencia, Departamento de Sistemas Informáticos y de Computación, Camino de Vera s/n, 46022, Valencia, España, onaindia@dsic.upv.es

Paper received on 24/07/08, accepted on 11/09/08.

Abstract. La planificación basada en teoría de decisiones ha permitido solucionar problemas del mundo real, donde la incertidumbre juega un papel muy importante. Esta metodología ha incrementado su desarrollo debido a los avances en representación e inferencia Bayesianas, así como al relativo éxito de los procesos Markovianos en problemas de control de procesos, análisis de decisión y economía. Sin embargo, ante problemas complejos estos procesos han presentado intratabilidad computacional, debido a que el espacio de búsqueda crece exponencialmente con el número de variables. Con el objeto de reducir el tiempo de solución en los procesos de decisión de Markov, en este documento proponemos representar, aprender y aplicar las acciones que realmente operan en cada estado como un conjunto de reglas de asociación. También aquí proponemos un nuevo enfoque del algoritmo de iteración de valor basado en estas reglas. Nuestros resultados experimentales, obtenidos sobre una tarea de planificación de movimientos de un robot, indican que la complejidad y el tiempo de solución son considerablemente reducidos, pues resultan lineales en el número de estados.

Palabras claves: procesos de decisión de Markov, acciones relacionales, reglas de asociación.

1 Introducción

Para que la planificación en Inteligencia Artificial pueda construir rumbos de acción en ambientes reales, al menos debe considerar que las acciones pueden tener efectos distintos en el mundo (no determinismo) y debe de ponderar el potencial de algún plan alternativo para alcanzar las metas del problema, considerando sus costos y recompensas (metas extendidas). Al respecto, la planificación basada en teoría de decisiones [7] ha permitido evaluar fortalezas y debilidades de los planes alternativos. Por ejemplo, en un problema de control de procesos, muchas variables cambian dinámicamente debido a la operación de dispositivos tales como válvulas, e interruptores de equipo, y por la presencia de eventos exógenos (no controlables). Pero si el sistema no considera la posibilidad de fallo de dichos dispositivos, entonces no será capaz de tomar una decisión inteligente ante dicho fallo. Este ejemplo viene a ser un

Advances in Intelligent and Information Technologies.
Research in Computing Science 38, 2008, pp. 117-127

problema muy complejo, donde la incertidumbre juega un papel importante a considerar durante la búsqueda de soluciones. Para tratar de abordarlos, se han extendido las capacidades de planificadores clásicos, pero la búsqueda heurística ha presentado limitaciones en los casos de datos no enteros y de grafos aditivos en la solución de problemas del mundo real [6], por lo que se han resuelto por representación e inferencia Bayesianas [10] y por los procesos de decisión de Markov (MDPs) [17]. Últimamente ha habido mucho avance en la aplicación de técnicas Markovianas en áreas como el reconocimiento de voz, control de procesos, análisis de decisión y economía. Sin embargo, estos procesos han resultado intratables ante problemas complejos (que contienen variables continuas o de grandes dimensiones). Al respecto existen algunos esfuerzos en el desarrollo de diferentes formas de representación, así como métodos computacionales para disminuir el espacio de búsqueda. Boutilier *et al.* [8] agruparon estados similares, pero no lograron la automatización debido a que tuvieron que realizar algunos pasos manualmente. En este sentido, Reyes *et al.* [19] propusieron un simulador de planificación de movimientos robóticos, que segmenta el espacio de búsqueda basándose en la función recompensa. También Bonet *et al.* [6] combinaron los beneficios de la programación dinámica con la potencia de la búsqueda heurística acompañada con aprendizaje. Sin embargo, no han tenido éxito con los grafos AND/OR aditivos. Por otro lado, Hauskrecht *et al.* [12] probaron la jerarquía con macro-acciones, pero no pudieron acoplarlas adecuadamente, obteniendo una complejidad adicional por la presencia de las macro-acciones. También Hoey *et al.* [13] impusieron una discretización *a priori* del espacio de observación, que resultó subóptima para la toma de decisiones, mediante partición dinámica. Aunque solo abordaron estados discretos y no han logrado la integración total de programación dinámica con la regresión basada en puntos. Por otro lado, Zhang *et al.* [22] restringieron subconjuntos o estados de creencia, aunque esto le adicionó complejidad al proceso. Muy recientemente Chenggang *et al.* [9] usaron diagramas de decisión de primer orden, grafos acíclicos *booleanos* que usan operadores al estilo de ReBell [20], aunque no lograron la cuantificación universal en parámetros y funciones. Como se puede observar, la escalabilidad ha sido el principal problema en la solución de los MDPs, resultando intratabilidad computacional [21].

Por otro lado, con el objeto de reducir el tiempo de solución de los MDPs, en este documento proponemos un nuevo enfoque donde las acciones que realmente operan en cada estado son representadas, aprendidas y aplicadas como un conjunto de reglas de asociación. Estas reglas implican ciertas relaciones entre objetos en una base de datos. También presentamos un nuevo enfoque del algoritmo de iteración de valor (AIV) [17] basado en reglas de asociación (RA), así como los resultados experimentales obtenidos, que indican alta viabilidad de nuestro enfoque. En la Sección 2, describimos brevemente a los MDPs. En la Sección 3, estudiamos al enfoque de acciones relacionales y su representación mediante RA. En la Sección 4, presentamos nuestra contribución, que se basa en la representación, aprendizaje y aplicación de RA en la solución de los MDPs. En la Sección 5, discutimos los resultados experimentales. Por último, en la Sección 6 presentamos las conclusiones y trabajos relacionados.

2 Procesos de Decisión de Markov

Estos procesos -llamados así en honor del ruso Andrei A. Markov, estudioso de la Estadística- fueron introducidos originalmente por Bellman [2] y han sido estudiados a profundidad en análisis de decisiones e investigación de operaciones desde la década de los 60's [17]. Los MDPs son un formalismo matemático e intuitivo, cuyo objetivo principal es encontrar una estrategia reactiva de control o política de acción que maximice la recompensa esperada por el agente. Las técnicas basadas en MDPs modelan un problema de decisión secuencial, donde un sistema evoluciona en el tiempo y es controlado por un agente. Sus dominios se modelan como sistemas estocásticos, sus metas se definen por sus funciones de utilidad y costo, por lo que el problema de planificación se transforma en un problema de optimización. De ahí que la solución de los MDPs sea la política óptima que asigna la mejor acción de cada estado. Por otro lado, en los MDPs se supone que el agente siempre conoce 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 solamente del estado en evaluación y no de su historia (propiedad de Markov). Estos procesos están basados en la ecuación de Bellman (vea ecuación (1)) que hace un cuidadoso balance entre riesgo y recompensa. Para ello, en cada estado se plantea una ecuación, cuya incógnita es su utilidad dada por:

$$U(s) = R(s) + \gamma \max_{a} \sum_{s'} T(s,a,s')U(s') \qquad (1)$$

donde $U(s)$ es la utilidad del estado en evaluación s, $U(s')$ es la utilidad del estado alcanzado s', $R(s)$ es la recompensa inmediata del estado en evaluación s, $T(s,a,s')$ es el modelo de transición que entrega la probabilidad de alcanzar al estado s' cuando la acción a es aplicada en el estado s, y finalmente γ es el factor de descuento, donde $0 < g < 1$. Al respecto, cuando la recompensa futura es insignificante comparada con la del estado en evaluación, entonces $g = 0$. En contraste, cuando la recompensa futura es idéntica a la del estado en evaluación, entonces $g = 1$. Por otro lado, el modelo de transición de estados puede ser representado por una red Bayesiana dinámica [10]. Pero de dicho planteamiento se obtiene un sistema de ecuaciones que no puede resolverse simultáneamente, debido a que esas ecuaciones no son lineales, pues tratan de maximizar la utilidad futura de visitar cada estado. Dentro de las técnicas existentes para resolver a este sistema de ecuaciones se encuentra la programación dinámica, y uno de sus métodos más populares es el algoritmo de iteración de valor (AIV) [17] o de utilidad. Solo que este algoritmo ha presentado la siguiente limitación: la medida del modelo de transición y, por consiguiente, del espacio de búsqueda aumenta exponencialmente con el número de variables. Por tal motivo, al tratar de resolver problemas complejos (con variables continuas o de grandes dimensiones) los MDPs han presentado tiempos de solución intratables computacionalmente [19].

3 Acciones Relacionales mediante Reglas de Asociación.

Con el objeto de disminuir el espacio de búsqueda y, por ende, el tiempo de solución de los MDPs, en el siguiente segmento proponemos una modificación al AIV, basado en RA Para ello se estudia, en este segmento, la forma de obtener estas reglas, las cuales entregan relaciones entre los atributos habidos en una base de datos. Cada RA contiene una acción relacional (*r*-acción), que está basada en el enfoque de Morales [15], quien propuso el uso de acciones restringidas a los estados durante el proceso de inferencia. En este sentido, ya Benson [3] había propuesto la necesidad de una jerarquía de acciones en planificación bajo incertidumbre, aunque solo pudo probarla en problemas simples. Continuando con las *r*-acciones, éstas requieren de

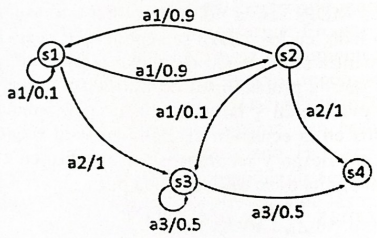

Fig. 1 Un modelo de Markov.

menos esfuerzo computacional, pues son conjuntos de acciones de menor tamaño que todas las acciones del dominio. Para obtener las *r*-acciones, se deben determinar aquellas que pueden aplicarse en cada estado. Consideremos un ejemplo sencillo representado en el modelo de Markov de la Figura 1, que contiene tres acciones (a1,a2,a3) y cuatro estados(s1,s2,s3,s4)para el movimiento de un robot. En esta figura se puede observar que cuando el robot opera a1 encontrándose en s1 (estado inicial), existe el 90% de probabilidad de que alcance s2, así como el 10% de probabilidad de que permanezca en s1. Por otro lado, cuando el robot se encuentra en s3 y ejecuta a3, tendrá un 50% de probabilidad de alcanzar s4 (estado final) y otro 50% de probabilidad de permanecer en s3. Siguiendo este diagrama es fácil obtener las *r*-acciones para cada estado (vea la Tabla 1).

Tabla 1. Conjuntos de r-acciones aplicables en cada estado.

Estado	*r-acciones*
s1	a1,a2
s2	a1,a2
s3	a3

Entonces una *r*-acción es un operador tipo: Si<estado(i)>entonces<*r*-acción(k)>, cuyo significado es que la k-ésima-*r*-acción es aplicable en el i-ésimo-estado en

evaluación. De esta manera, el uso de las *r*-acciones promete ahorros en el esfuerzo computacional [15] requerido durante la solución de los MDPs. Sin embargo, la escalabilidad de problemas permanece como un reto en estos procesos debido a que han presentado intratabilidad [21].

Por otro lado, las *r*-acciones se pueden conocer a partir de un árbol de decisión, que está formado por nodos (atributos), ramas (posibles valores del atributo) y hojas (*r*-acciones). El árbol de decisión se puede generar mediante el algoritmo C4.5 [18], que efectúa particiones recursivas sobre los datos, aplicando la estrategia "primero-en-profundidad".

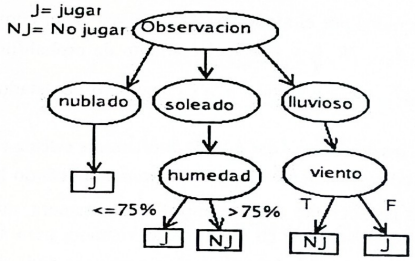

Fig 2. Un árbol de decisión.

Por ejemplo, en la Figura 2 se puede apreciar un árbol de decisión derivado del problema Jugar Golf [18] de acuerdo con la observación del estado del tiempo. Si la rama derivada de "Observación" contiene a los atributos "lluvioso" y "viento", entonces la acción que hace verdadera (*True*) a esa expresión debe ser "NJ" (No Jugar) y se puede establecer como:

$$\text{Si} < \text{lluvioso} \wedge \text{viento} > \text{ entonces } < \text{NJ} >$$

En consecuencia, cada rama del árbol -desde raíz hasta hoja- representa una RA, la cual contiene a la *r*-acción (hoja) correspondiente a dicho estado de atributos. Por otro lado, las RA pueden ser generadas mediante el algoritmo Apriori [1], que se basa en el cálculo y selección del conjunto de atributos que presentan el soporte (combinación bien predicha) y la confianza (porcentaje de aciertos de la regla) máximos, para la aplicación de cada *r*-acción. Por ejemplo, para la Figura *2*, cuatro de las mejores RA que entrega Apriori son:

observacion=nublado 4 ==> jugar=si 4 conf:(1)
humedad=normal, viento=FALSO 4 ==> jugar=si 4 conf:(1)
observacion=soleado, humedad=alta 3 ==> jugar=no 3 conf:(1)
observacion=lluvia, viento=FALSO 3 ==> jugar=si 3 conf:(1)

donde los números 3 y 4 son el soporte de la regla y el número 1 se refiere al 100% de confidencia. Por último, se usó Apriori debido a que este algoritmo entrega las RA en el formato requerido por el AIV, las cuales deben contener tres atributos: es-

tado inicial del autómata, estado final del mismo y la r-acción requerida para que el autómata alcance a este último estado (vea Figura 1).

4 Inserción de Reglas de Asociación en los MDPs

Para reducir el tiempo de solución de los MDPs con problemas complejos, aquí proponemos modificar al AIV para que solo itere sobre RA (vea Algoritmo 1), que contienen r-acciones. Sea L el conjunto de RA $L_k = (s_k, s_k', a_k)$ con soporte y confidencia máximos obtenidos por el algoritmo Apriori, sea R el conjunto de recompensas de estado $\{R_1, R_2, \ldots, R_n\}$ y sea T el conjunto de probabilidades de transición de las reglas $\{T_1, T_2, \ldots, T_n\}$, donde n es el número de estados. El algoritmo resultante es:

En nuestro algoritmo se puede observar que el AIV directamente calcula sobre la *k-ésima*-RA $L_k = (s_k, s_k', a_k)$ correspondiente al *k-ésimo-estado*, aplicando la *k-th-ésima-r-acción*, no sobre todas las acciones del dominio. De esta manera, nuestro algoritmo entrega la política óptima (r-action) en cada estado visitado, para alcanzar la meta con menos esfuerzo computacional. Además, para un número dado de estado n_s, un número de acciones n_a, y un número de iteraciones n_{it}, la complejidad de nuestro algoritmo esta dada por $O(n_s, n_a, n_{it}) = n_s n_a n_{it}$. El máximo número de iteraciones [14] se calcula de la siguiente manera:

$$n_{it} \leq \frac{B + \log(\frac{1}{e}) + \log(\frac{1}{1-\gamma}) + 1}{1 - g} \qquad (1)$$

donde B es el número de *bits* usado en codificar los costos instantáneos y las probabilidades de transición, e es el margen de error del residuo de Bellman y por último γ es el factor de descuento. Para grandes valores de n_s, el número de iteraciones es menor que el número de estados, lo que significa que la complejidad es una función lineal del número de estados. De aquí se concluye que nuestro algoritmo puede resolver rápidamente a los MDPs. Para aplicar nuestro algoritmo a un problema complejo del mundo real, primero creamos un archivo de instancias generadas mediante un paseo aleatorio sobre todo el espacio de estados y con todas las acciones del dominio. Después, el archivo de instancias es usado para generar al conjunto de RA, aplicando A priori como ya se estudió. Finalmente, las probabilidades de transición de estados son estimadas usando redes Bayesianas dinámicas [10].

Algoritmo 1 RulesMDP

```
function RulesMDP(R, L, T, g, e, NumIt)
U_i^0 = R_i for i = 1,2,...,n
t = 1
J(s,a) = 0 for s = 1,2,...,n and a = 1,2,...,m
do
    for k = 1 to |L|
        a = action(L_k)
        s = initialState(L_k)
        s' = finalState(L_k)
        J(s,a) = J(s,a) + T_k U_s'^{t-1}
    end
    for s = 1 to n
        U_s^t = R_s + g max_a J(s,a)
        p_s = arg max_a J(s,a)
    end
    t = t + 1
while t £ NumIt and [illegible] > e
return p
```

5 Resultados Experimentales

Probamos nuestro algoritmo RulesMDP en un simulador de planificación de movimientos robóticos (SPRM [19]). Para cada experimento actualizamos un espacio bidimensional, donde un agente autónomo se puede mover con 5 acciones. Las pruebas se hicieron variando el número de estados desde 25 hasta 400, éstos asociados con diferentes valores de recompensas (vea Tabla 2). Para todos nuestros experimentos usamos $\gamma = 0.9, \varepsilon = 10^{-6}, B = 32$ y $n_{it} = 500$ (este último valor resulta de aplicar estos parámetros en la ecuación (2)). Repetimos 120 veces cada combinación de parámetros para tener mayor certeza en nuestros resultados, dado que los tiempos de solución obtenidos son del orden de milisegundos. Por otro lado, nuestro algoritmo fue implementado en *Java* dentro del simulador SPRM y compa-

ramos la ejecución de ambos algoritmos: nuestro algoritmo (RulesMDP) y el algoritmo AIV previo (PREV). Todos los experimentos fueron ejecutados en una computadora *Pentium D* con 2.66 GHz y 1 GB de RAM. Los resultados obtenidos de las diferentes combinaciones se presentan en la Tabla 2. Las políticas o rutas de acción obtenidas por ambos algoritmos fueron muy similares (difieren en algunos estados visitados, pero con el mismo estado meta alcanzado y con el mismo número de acciones en la trayectoria).

Tabla 2. Resultados obtenidos en SPRM: PREV vs. RulesMDP.

recompensas	estados	Tiempo(ms). PREV	Número de operaciones. PREV	Tiempo (ms), RulesMDP	Número de operaciones RulesMDP
2	25	47	375000	12	14625
4	25	47	384375	12	14750
10	100	94	6000000	12	60000
26	100	94	6200000	15	60500
6	225	389	30261000	21	128625
10	225	396	30832000	21	128900
6	400	1195	96000000	31	224000
10	400	1266	102400000	31	254000
12	400	2969	242187500	266	1512500

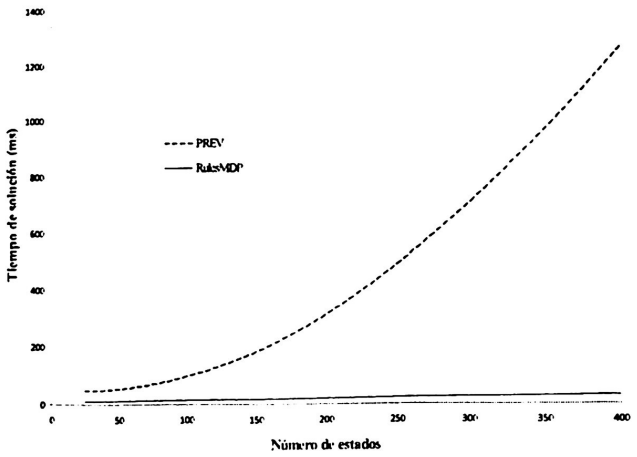

Fig. 3. Tiempo de solución vs. Número de estados del algoritmo previo (PREV) y de nuestro algoritmo (RulesMDP) en el SPRM.

En la Tabla 2, podemos observar que cuando el número de estados es 25, el tiempo de solución requerido por el algoritmo previo es de 47 milisegundos, mientras que es de 12 ms por nuestro algoritmo. Por otro lado, cuando el número de estados es 400, el tiempo de solución del algoritmo previo es de 1266 ms, mientras que es de 31 ms por nuestro algoritmo. En ambos casos, nuestro algoritmo (RulesMDP) es más rápido que el previo. Ahí también podemos observar que el tiempo de solución se incrementa suavemente al aumentar el número de recompensas, para un mismo número de estados. Los resultados obtenidos en la Tabla 2 son presentados en la Figura 3, que muestra a la función ajustada de cada método. Esta figura muestra el incremento polinomial al aumentar el número de estados, mientras que el tiempo de solución requerido por nuestro algoritmo se incrementa linealmente (con una pendiente muy suave) con el aumento del número de estados. Por otro lado, en la misma figura se observa que con el aumento en el número de recompensas, para un mismo número de estados, el tiempo de solución prácticamente permanece constante. En conclusión, al aplicar reducción sobre el espacio de búsqueda de los MDPs mediante RA se obtiene una considerable reducción en el tiempo de solución, resultando la complejidad como una función lineal en el número de estados.

6 Conclusiones

La extensión de capacidades de un planificador clásico mediante búsqueda heurística, ha presentado limitaciones ante datos no enteros y grafos aditivos [6], por lo que se ha buscado resolverlos mediante representación e inferencia Bayesianas [10] y mediante los procesos de decisión de Markov [17]. Estos últimos han resuelto exitosamente problemas de control de procesos, pero ante problemas con variables continuas o de grandes dimensiones han resultado intratables [21]. Lo anterior es debido a que en estos procesos el espacio de búsqueda crece exponencialmente con el número de variables. Con el objeto de reducir el tiempo de solución de estos procesos estocásticos, hemos propuesto un novedoso enfoque que genera automáticamente RA. Estas reglas contienen r-acciones, que son aquellas acciones con cierta probabilidad de operar en cada estado. También propusimos un nuevo AIV basado en estas reglas. La técnica propuesta utiliza el algoritmo Apriori para generar el grupo de RA más frecuentes. Al menos en nuestros resultados experimentales, encontramos que el tiempo de solución fue lineal en función del número de estados. Las políticas (rutas de acción) obtenidas con ambos algoritmos fueron muy similares, alcanzando el mismo estado meta y teniendo el mismo número de acciones en ellas, difiriendo solamente en algunos estados visitados. Como trabajos relacionados se encuentran dos métodos que abordan la solución de procesos estocásticos con variables de estado continuas: la discretización basada en rejillas y las aproximaciones paramétricas. Pero los algoritmos de rejilla clásicos crecen exponencialmente con el número de variables de estado [5] y las aproximaciones paramétricas con base en técnicas de agrupamiento jerárquico han reducido en poca medida el costo computacional [16].

Referencias

1. Agrawal, R., Srikant, R., Fast Algorithms for Mining Association Rules, Proceedings of the 20th VLDB Conference, IBM Almaden Research Center (1994).
2. Bellman, R.E., Dynamic Programming, Princeton United Press, Princeton, USA (1957).
3. Benson, S.S., Learning action models for reactive autonomous agents, PhD Thesis, University of Stanford (1996).
4. Bertsekas, D.P., Dynamic Programming, Prentice Hall, Eaglewood Cliffs, MA, USA (1987).
5. Bonet, B., Pearl, J., Qualitative MDP and POMDP: An order-of-magnitude approach, Proceedings of the 18th Conference on Uncertainty in AI, Edmonton, Canada, pages 61-68 (2002).
6. Bonet, B., Geffner, H., Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings and its application to MDP, Proceedings of ICAPS (2006).
7. Boutilier, Craig. Dean, T., Hanks, S., Decision-theoretic planning: structural assumptions and computational leverage, Journal of AI Research, 11, pages 1-94 (1999).
8. Boutilier, C., Dearden, R., Goldszmidt, M., Stochastic Dynamic Programming with factored representations, Artificial Intelligence, 121(1-2) pages 49-107 (2000).
9. Chenggang, W., Saket, J., Khardon, R., First-Order Decision Diagrams for Relational MDP's, Proceedings of IJCAI-07 (2007).
10. Darwiche, A., Goldszmidt M., Action networks: A framework for reasoning about actions and change under understanding, Proceedings of the 10th Conference on Uncertainty in AI, UAI-94, pages 136-144, Seattle, USA (1994).
11. Dean, T., Givan, R., Model minimization in Markov Decision Processes, Proceedings of the 14th National Conference on AI, pages 106-111 (1997).
12. Hauskrecht, M., Kveton, B., Linear program approximations for factored continuous-states Markov Decision Processes, Proceedings of the NIPS (2003).
13. Hoey, J., Poupart, P., Solving POMDP's with Continuous or Large Discrete Observation Spaces, Proc. Of International Joint Conference on Artificial Intelligence, IJCAI-05, Edinburgh, July (2005).
14. Littman, M. L., Dean, T. L. and Kaelbling, L. P., On the Complexity of Solving Markov Decision Problems, Proc. of the Eleventh International Conference on Uncertainty in Artificial Intelligence, pages 394-402 (1995)
15. Morales, E., Scaling up reinforcement learning with a Relational Representation, Proceedings of the Workshop on Adaptability in MultiAgent Systems, AORC, Sydney, Australia (2003).
16. Pineau, J., Gordon, G., Thrun, S.,Policy-contingent abstraction for robust control, Proceedings of the 19th Conference on Uncertainty in AI, pages 477-484 (2003).
17. Puterman, M.L., Markov Decision Processes, Wiley Editors, New York, USA (1994).
18. Quinlan, J.R., C4.5: Programs for machine learning, Morgan Kaufmann, San Francisco, CA, USA (1993).
19. Reyes, A., Ibarguengoytia, P., Sucar, L.E., Morales, E., Abstraction and Refinement for Solving Continuous Markov Decision Processes, 3rd European Workshop on Probabilistic Graphical Models, Czech Republic, pages 263-270 (2006).
20. Van Otterlo, M., Kersting, K., De Raedt, L., Bellman goes Relational, Proc.of the 21st International Conference on Machine Learning, Banff, Canada, 2004
21. Van Otterlo, M., A Survey of Reinforcement Learning in Relational Domains, Technical Report Series CTIT-05-31, ISSN 1381-3625, July (2005).

22. Zhang, W., Zhang, N., Restricted Value Iteration: Theory and Algorithms, Journal of Artificial Intelligence Research, JAIR, 23, pages 123-165 (2006).