Exploración Dirigida por el Objetivo en Aprendizaje por Refuerzo Basado en Modelo para Ambientes No estacionarios

Marcelo Luis Errecalde Alfredo Rubén Muchut

LIDIC *
Departamento de Informática
Universidad Nacional de San Luis(UNSL)
Ejército de los Andes 950 - Local 106
5700 - San Luis. Argentina
e-mail:{merreca,amuchut}@unsl.edu.ar

Resumen

El Aprendizaje por Refuerzo Basado en Modelo (ARBM) es una extensión al Aprendizaje por Refuerzo tradicional en la que el agente aprende una política (comportamiento), y en forma simultánea aprende un modelo de su ambiente. Distintos estudios han mostrado la superioridad de los métodos de ARBM sobre los métodos libres de Modelo en ambientes estacionarios. Sin embargo, existen serias dificultades para adaptar los métodos de ARBM a ambientes no estacionarios, existiendo actualmente un único método -la arquitectura Dyna con Bono de exploración- que empíricamente ha demostrado su adaptabilidad a los cambios ambientales. En este sentido, este paper presenta una extensión a esta arquitectura, manteniendo información relativa al estado objetivo, y definiendo una nueva heurística de exploración que, en base a esta información, permite concentrar la actividad del agente en las zonas más relevantes del problema. Los resultados experimentales obtenidos con distintas instancias de ambientes estacionarios y no estacionarios sustentan la factibilidad de nuestra propuesta observándose una mejora de performance significativa con respecto a la arquitectura Dyna original.

Palabras Claves: Agentes Inteligentes, Aprendizaje por refuerzo basado en modelo, Técnicas de exploración, Ambientes no estacionarios.

1 Introducción

El Aprendizaje por Refuerzo Basado en Modelo (ARBM) ha demostrado una mejor performance que los métodos de AR sin modelo cuando aplicados a ambientes estacionarios [2, 5, 6, 8, 9]. Los ambientes no estacionarios (aquellos que cambian con el tiempo) crean dificultades

*El laboratorio es dirigido por el Dr. Raúl Gallard y subvencionado por la UNSL y la ANPCYT (Agencia Nacional para la Promoción de la Ciencia y la Tecnología)

adicionales en el ARBM, surgidas de las potenciales inconsistencias del modelo aprendido con el mundo real. Algunos trabajos plantean soluciones a este problema asumiendo conocimiento previo del agente sobre el ambiente y la ubicación del objetivo [1] y en otros casos [8] el problema es resuelto utilizando heurísticas de exploración muy elementales. Este trabajo plantea un esquema de exploración dirigido por el objetivo para ARBM en el que no se asume ningún tipo de conocimiento previo del agente sobre el ambiente y que utiliza una estrategia de exploración más inteligente que la utilizada en [8]. Mediante resultados experimentales probamos que nuestra propuesta de exploración se concentra sobre las partes más relevantes del ambiente superando la performance del único método que, de acuerdo a nuestro conocimiento, ha resultado efectivo en ambientes no estacionarios.

Este trabajo se organiza de la siguiente manera: las secciones 2 a 4 tienen un objetivo tutorial, introduciendo en forma muy resumida algunos de los conceptos principales utilizados en el resto del trabajo. Estas secciones podrán ser omitidas por los lectores familiarizados en el tema recomendándose a [3, 4, 10] para una introducción al tema o una explicación más detallada. La sección 5 presenta nuestra propuesta de exploración dirigida por el objetivo para el ARBM. La sección 6 describe distintos experimentos involucrando ambientes estacionarios y no estacionarios con resultados comparativos de las performances obtenidas. La sección 6 finaliza este trabajo con algunas conclusiones sobre los resultados obtenidos

2 El problema del Aprendizaje por Refuerzo

El Aprendizaje por Refuerzo (AR) ataca el problema de controlar agentes autónomos en ambientes desconocidos. El objetivo de un agente de AR es aprender una política (óptima) que le permita mapear situaciones en acciones, basándose únicamente en la información (*recompensas o refuerzos*) que recibe durante la interacción con su ambiente.

El AR permite resolver problemas en los cuales se asume que si bien existe incertidumbre en cuanto al efecto de las acciones del agente, el resultado de estas acciones es *completamente observable*. Este tipo de problemas, puede ser formalizado mediante un modelo matemático conocido como *Proceso de Decisión Markov*.

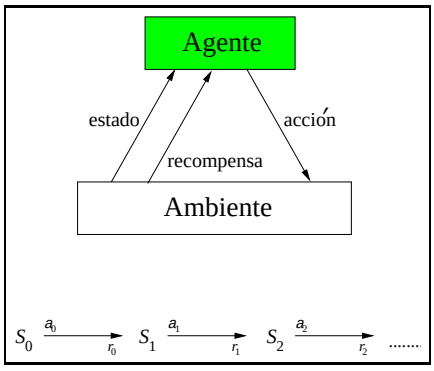


Figura 1: Interacción entre agente y ambiente

La idea general intuitiva dentro de este modelo, es tener al agente conectado con el ambiente vía percepción y acción. En cada paso del tiempo t, el agente recibe como entrada desde el ambiente una indicación del estado actual s del ambiente, y selecciona una acción a disponible en el estado s. La acción generada por el agente cambia el estado del ambiente, el cual responde un paso del tiempo después con una indicación del nuevo estado y una señal de recompensa inmediata r, por la acción tomada previamente. Esta interacción se puede visualizar gráficamente en la **Figura 1**.

Si los objetivos del agente están definidos por la función de recompensa inmediata, la tarea del agente se reduce a encontrar un comportamiento que le permita decidir en cada estado, qué acciones tomar para maximizar las recompensas acumuladas a

lo largo del tiempo.

Cuando las respuestas del ambiente a una acción del agente, dependen sólo del estado en que se encontraba y de la acción tomada, sin influencias de los estados del ambiente y las acciones del agente previas, el ambiente y la tarea en su conjunto se dice que satisfacen la *propiedad Markov*, y pueden ser definidas formalmente como un MDP que consiste de: un conjunto de estados posibles S, un conjunto de acciones posibles A, las dinámicas de un paso del ambiente, dadas por

- las *probabilidades de transición*, $P(s,a,s') = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\}$
- Las *recompensas esperadas*, $R(s,a,s') = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\}$

definidos para todo s, $s' \in S$ y $a \in A(s)$, donde $A(s)$ es el conjunto de acciones válidas en s.

La *solución* de este MDP se reduce a encontrar una política $\pi^* : S \to A$, que en cada paso del tiempo t en el estado s_t, elige la acción a_t que maximiza el valor esperado de la suma de las recompensas descontadas que recibe en el futuro. Esta suma, se suele denominar *retorno descontado* y lo denotaremos R_t, por lo que el agente se puede decir que maximiza el retorno descontado esperado:

$$\begin{aligned} E\{R_t\} &= E\left\{r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots\right\} \\ &= E\left\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1}\right\} \end{aligned}$$

donde $0 \leq \gamma < 1$, recibe el nombre de *factor de descuento*.

2.1 Métodos para la resolución de MDP's

La resolución de un MDP, se basa normalmente en la estimación de *funciones de valor óptimas*, definidas sobre el conjunto de estados o de acciones, y que denotaremos como V^* y Q^* respectivamente.

V^*, la *función de valor-acción óptima*, se define sobre el conjunto de estados, tal que $V^*(s)$ indica cual es el retorno esperado máximo que se puede obtener desde un estado s, sobre todas las políticas posibles, es decir cual es el retorno esperado que se obtiene siguiendo una política óptima a partir de s.

Q^*, la *función de valor-acción óptima*, se define sobre cada acción aplicable en cada estado, tal que $Q^*(s,a)$ da el retorno esperado de tomar la acción a en el estado s y en lo sucesivo seguir una política óptima.

Una vez obtenidas V^* o Q^* es fácil determinar una política óptima π^*. Cualquier política que es *greedy* con respecto a las funciones de valor óptimas es una política óptima.

Si bien existe una gran variedad de técnicas para resolver problemas modelizados como MDP's, podemos en general clasificarlas dentro de 2 grandes grupos: las técnicas *basadas en planning* y las *basadas en aprendizaje*.

En el primer caso, al hablar de planning queremos resaltar el hecho de que estas técnicas requieren, al igual que en planning clásico, un modelo del ambiente. En el contexto de los MDP's este modelo está dado por las dinámicas de un paso del ambiente. Como ejemplo de este tipo de técnicas podemos mencionar los métodos de *Programación Dinámica* (ver [4, 10] para un survey de estas técnicas).

Las técnicas de aprendizaje por su parte, asumen en general que este modelo no es conocido y que las estrategias (o el controlador) deberán ser derivados "on-line" en base a las experiencias que el agente obtiene a partir de su interacción con el ambiente. Como ejemplo de estas técnicas podemos citar los métodos de AR((ver [4, 10] para un survey de estas técnicas).

Dado que en el AR, no se dispone de un modelo del ambiente, existen en general 2 maneras de proceder que determinan dos grupos de métodos de AR:

- **AR sin modelo**: consiste en aprender una política sin aprender un modelo del ambiente.
- **AR basado en modelo**: Aprender un modelo y usarlo para derivar un controlador

No está claro que enfoque es mejor y este aspecto es aún tema de debate dentro de la comunidad de AR. No obstante esto, el AR basado en modelo ha demostrado mejoras de performance significaticas con respecto al AR sin modelo en experimentos realizados en ambientes estacionarios [2, 5, 6, 8]. Las ventajas del AR basado en modelo no son tan claras en ambientes no estacionarios, donde los cambios ambientales pueden hacer inconsistente el modelo aprendido por el agente. En este sentido este trabajo demuestra no sólo cómo el ARBM es factible en ambientes no estacionarios, sino además que utilizando una política de exploración adecuada puede resultar en una técnica relativamente eficiente.

3 Aprendizaje por Refuerzo Basado en Modelo

El Aprendizaje por refuerzo basado en modelo (de ahora en más ARBM) surge de la integración del AR sin modelo simple con técnicas de planning basadas en *Programación dinámica*(PD).

Al igual que en el AR simple el agente aprende a partir de la interacción con el ambiente, pero incorpora además un modelo del ambiente (las dinámicas de un paso) el cual utiliza para realizar actualizaciones de las funciones de valor de manera similar a la realizada en PD. La diferencia con PD es que este modelo no es provisto a priori, sino que es aprendido on-line.

En ARBM, la información que el agente obtiene en su interacción con el ambiente juega dos roles principales:

1. Permite actualizar las funciones de valor en forma similar a la realizada en los métodos de aprendizaje de AR estándar.
2. Permite actualizar el modelo del ambiente.

Este esquema es complementado con un proceso de planning que permite realizar múltiples actualizaciones de las funciones de valor utilizando la experiencia simulada obtenida a partir del modelo.

Existen distintos métodos de ARBM [8, 9, 5, 6]. A continuación presentamos el algoritmo *Dyna-Q* el cual es tomado como base en nuestro trabajo.

Algoritmo Dyna-Q

Inicializar: Tabla $Q(s,a)$ y $Model(s,a)$ para todo $s \in S$ y para todo $a \in A(s)$.
$s \leftarrow$ estado inicial.
Repetir por siempre

1. $a \leftarrow \epsilon - greedy(s,Q)$ (selección de una acción mediante una política).
2. Aplicar la acción a al mundo real y observar el estado s' resultante y la recompensa r.
3. Actualización de la Tabla Q:

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

4. $Model(s,a) \leftarrow s', r$ (Actualización del Modelo)
5. Repetir N veces
   (a) Elegir un estado y una acción hipotética.
   $s \leftarrow$ estado random observado previamente
   $a \leftarrow$ acción random tomada previamente en s
   (b) Predecir s', el estado al cual arrivo desde s tomando a, y reward r resultante usando el modelo.
   (c) Modificar la tabla Q con esta información empleando la siguiente formula.

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

6. $s \leftarrow s'$

El término Dyna referencia en realidad a una propuesta de arquitectura general para ARBM y Dyna-Q es una instancia particular de la misma. En este caso, se asume que el ambiente es determinístico y que el método de aprendizaje base utilizado es *Q-Learning*(paso 3). $Model(s,a)$ denota los contenidos del modelo (predicción del estado siguiente y recompensa para cada par (s,a)). El AR directo, el aprendizaje del modelo y la etapa de planning son implementados por los pasos 3,4 y 5 respectivamente.

4 Exploración

Las técnicas de AR son generalmente utilizadas en problemas donde el desafío surge de la ignorancia que el agente tiene acerca del ambiente. Cuando un agente inteligente aprende a controlar estos ambientes desconocidos debe, en cualquier caso, balancear dos aspectos que usualmente entran en conflicto.

Por un lado, el agente debe maximizar las recompensas esperadas, utilizando para ello su conocimiento actual representado en sus estimaciones de las funciones de valor. De acuerdo a esta información, la acción que se presenta como mejor se denomina acción *greedy*, y cuando el agente la elige se dice que está *explotando* su conocimiento corriente de los valores de las acciones.

Por otra parte, la elección sistemática de la acción greedy (denominada *política greedy*) sólo sería adecuada si la estimación de las funciones de valor óptimas se correspondieran directamente con los verdaderos valores de las funciones de valor óptima. Esta estimación será precisa en la medida que el agente explore nuevos estados y acciones y no sólo los sugeridos por los valores

corrientes de las funciones de valor. En estos casos, cuando el agente elige una acción no greedy se dice que está *explorando* el ambiente, de manera tal de mejorar sus estimaciones de los valores de la acciones no greedy. A diferencia de la explotación, cuando el agente explora no busca maximizar sus recompensas acumuladas, sino maximizar la adquisición de nuevo conocimiento que permitirá obtener una recompensa total mayor a largo plazo.

Dado que es imposible explorar y explotar en forma simultánea con una única selección de acción, esta situación es a menudo denominada como el *conflicto entre exploración y explotación* [10] o *problema de control dual* [1].

Siguiendo a [11, 12] podemos clasificar los métodos de exploración en dos grandes grupos: métodos de exploración *no dirigida* y *dirigida*.

4.1 Exploración No Dirigida

Este tipo de métodos exploran el ambiente basándose en una selección estocástica de las acciones.

La idea principal en este enfoque es que el agente elige entre sus acciones disponibles, tomando como base alguna distribución de probabilidad que asigna una probabilidad positiva a cada par estado-acción.

La denominación de estas técnicas como "no dirigidas" responde al hecho de que si bien cierta información sobre la utilidad de las acciones puede influenciar las distribuciones de probabilidad, existe aún una componente estocástica en el proceso de selección de las acciones.

4.2 Exploración Dirigida

A diferencia de las técnicas no dirigidas, la exploración dirigida utiliza conocimiento específico del proceso de exploración para determinar de que manera el agente explora mejor el ambiente. Obviamente, la mejor forma de explorar el ambiente para optimizar el controlador a lo largo del tiempo es imposible dado que el ambiente es parcial o totalmente desconocido por lo que estos métodos son *heurísticos* por naturaleza [11]. La idea entonces consiste en definir una *función de recompensa de exploración heurística*(R^e) que determina las *recompensas de exploración inmediata*, similar a la función de recompensa R de un MDP que definía las recompensas en base al problema. Existen distintas técnicas de exploración dirigida, como por ejemplo las que ejecutan la acción seleccionada *menos frecuentemente*, *menos recientemente*, etc. En esta sección sólo describiremos la *exploración basada en el tiempo transcurrido (recency-based exploration)* por su importancia para nuestro trabajo, ya que ha demostrado ser efectiva en ambientes no estacionarios. Esta técnica selecciona la acción que ha sido ejecutada *menos recientemente*.

Para su implementación es necesario mantener por cada acción a válida en cada estado s una estimación $t(s,a)$ del tiempo transcurrido desde la última vez que la acción a ha sido ejecutada. La función de recompensa de exploración inmediata en este caso puede ser definida como :

$$R^e(s,a,*) = \sqrt{t(s,a)}$$

donde el "*" denota el símbolo "don't care".

Sin lugar a dudas el ejemplo más conocido de este tipo de exploración es el *bono de exploración* introducido por Sutton en una de las instancias de la arquitectura Dyna [8]. En este caso, Sutton modificó el paso 5.c del algoritmo presentado previamente, reemplazando la recompensa simulada r por

$$r + \epsilon\sqrt{t(s,a)}$$

donde ϵ es una constante pequeña. En su trabajo, Sutton mostró que esta instancia de Dyna (a la que llamaremos *Dyna con Bono de exploración*) fue la única que logró producir un comportamiento de exploración exitoso en distintos ambientes no estacionarios.

5 Exploración dirigida por el objetivo en ARBM

En la arquitectura Dyna con bono de exploración, cada par estado-acción se torna más atractivo de acuerdo al lapso de tiempo transcurrido desde que la acción fue ejecutada. Esta heurística sencilla funciona adecuadamente en ambientes no estacionarios ya que permite que el agente "escape" de trayectorias subóptimas y encuetre pasos óptimos generados dinámicamente. Sin embargo esta exploración incluye estados y acciones que muy probablemente no pertenecen a los pasos más cortos desde el estado inicial al objetivo.

Nuestra hipótesis de trabajo plantea que es posible mejorar esta heurística de exploración si el agente incorpora información que permite "orientarlo" hacia la zona del objetivo. El problema es que en el AR el agente tiene un desconocimiento completo del ambiente incluyendo la ubicación del estado objetivo. Esto significa que una técnica de exploración dirigida por el objetivo para AR, debe resolver 3 problemas principales:

1. Determinar la ubicación del objetivo.
2. Definir una heurística de exploración basada en el punto anterior
3. Combinar esta heurística con técnicas de exploración dirigidas adaptadas a ambientes no estacionarios.

Nuestra solución para el primer problema es considerar al espacio de estados (laberinto) como un plano cartesiano cuyo origen es el estado de comienzo (start) y cada estado del laberinto es un punto en este plano. Si al estado de comienzo le asignamos la coordenada (0,0), el agente puede ser "conciente" en todo momento de su ubicación relativa a esta posición simplemente actualizando *su posición* de acuerdo a la acción que tome (incrementando en 1 su coordenada x si la acción es *arriba*, y actuando en forma consistente con las acciones restantes). Cuando el agente arriba por primera vez al objetivo, podrá registrar su ubicación relativa respecto al estado de comienzo y esta información será utilizada en el cálculo de la heurística del paso siguiente.

Una vez obtenida la ubicación del objetivo, ¿cuál sería una heurística adecuada para orientar las acciones del agente hacia el mismo?. En este caso, uno esperaría contar con algún tipo de *heurística admisible* [7] similar a la utilizada en algoritmos como A^*. En este caso particular, conociendo la ubicación del objetivo se podría utilizar la métrica denominada *distancia Manhattan* que cumple con estas características ya que nunca sobreestima el costo estimado de alcanzar el objetivo desde un estado particular. Sin embargo, la idea de heurística admisible debe ser adaptada al contexto del AR, ya que en búsqueda heurística uno intenta minimizar un costo, mientras que en AR uno debe maximizar las recompensas. Una forma de lograr esto, es calcular cual sería el mayor retorno que puedo obtener desde un estado. En este caso, un cálculo optimista estaría dado por el retorno que obtendría el agente si alcanzara el objetivo sin ser detenido por

ningún obstáculo. En este caso, si la distancia Manhattan desde el estado s al objetivo es dMh, y el agente sólo recibe una recompensa r^{obj} al llegar al objetivo, el retorno máximo sería:

$$0\gamma^1 + 0\gamma^2 + \ldots + r^{obj}\gamma^{dMh} = r^{obj}\gamma^{dMh}$$

Sólo falta definir como combinamos esta heurística con otro esquema que se adecúe a ambientes no estacionarios. En este caso, nuestra elección fue adaptar el bono de exploración de Sutton considerando que en la etapa de planning (y con $r^{obj} = 1$), la recompensa que el agente recibe en cada experiencia simulada es:

$$r + \lambda\epsilon\sqrt{t(s,a)} + (1-\lambda)\epsilon\gamma^{dMh}$$

con $0 < \lambda < 1$. Observar que esta ecuación surge de combinar el bono de exploración de Sutton con un peso de λ y la heurística basada en la distancia Manhattan al objetivo con un peso de $1 - \lambda$.

De esta manera, el paso 5.c del algoritmo Dyna-Q, quedará

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \lambda\epsilon\sqrt{t(s,a)} + (1-\lambda)\epsilon\gamma^{dMh} + \gamma max_{a'}Q(s',a') - Q(s,a)]$$

Nuestra extensión de la arquitectura Dyna, que implementa los 3 tareas citadas previamente, constituyen nuestra propuesta para realizar una exploración más inteligente que el método Dyna con bono de exploración. Lo denominaremos *Dyna con Exploración Dirigida por Objetivo* o en forma abreviada *Dyna-EDO*.

6 Experimentos

En la **Figura 2** se puede observar el laberinto con el cual experimentaremos. Cada celda oscura representa una barrera que el agente no puede atravesar.

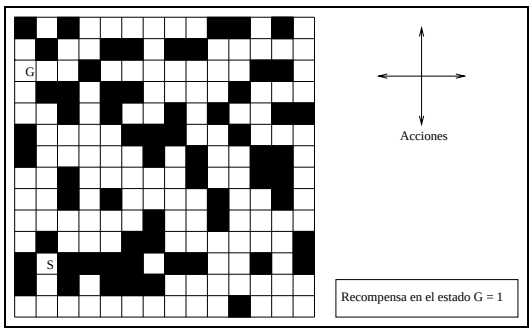


Figura 2: Laberinto

En cada estado (celda clara) el agente puede realizar una de cuatro acciones, *arriba, abajo, derecha e izquierda*, las que en forma determinística desplazan al agente en la dirección correspondiente. Las acciones bloqueadas (por una barrera o por los límites del laberinto) dejan inalterada la ubicación del agente. Todas las acciones tienen una recompensa de 0, a excepción de aquellas que conducen al estado objetivo (G) en cuyo caso la recompensa es 1. El aprendizaje del agente se produce por *episodios* donde cada episodio de entrenamiento comienza con el agente situado en el estado S (start) y finaliza cuando arriba al estado objetivo (G) en cuyo caso es reubicado en S.

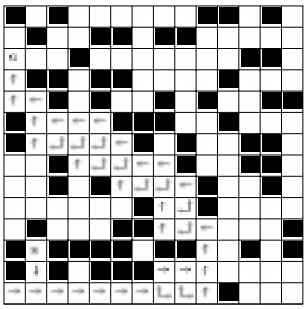


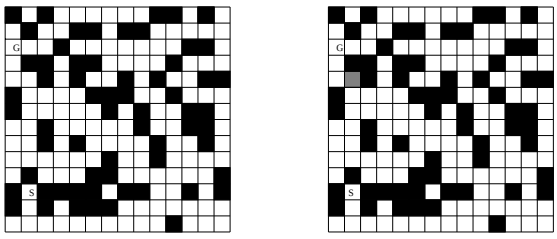
Figura 3: Políticas óptimas

La **Figura 3** por su parte muestra el conjunto de políticas óptimas para llegar desde el estado S al G. Cuando se trabaja sobre laberintos no estacionarios, los cambios que en general se suelen considerar son:

- *Bloqueo del laberinto (Blocking Maze):* Este cambio consiste en dejar que el agente aprenda una política óptima y posteriormente bloquear con una o más barreras esta trayectoria.
- *Generar un "atajo" en el laberinto (Shortcut maze):* Este cambio se produce cuando la eliminación de barreras en el laberinto genera un camino más corto que el actual. Esto causa que la política aprendida por el agente hasta el momento deje de ser óptima.

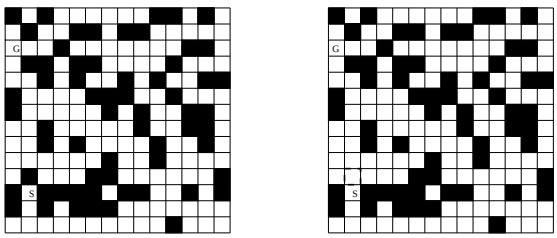
Nuestros experimentos se han basado en el laberinto de la **Figura 2** considerando 4 escenarios: el caso estacionario, al que denominaremos **Sin Cambio**, y 3 tipos de ambientes no estacionarios que representan distintas combinaciones de las 2 clases de cambios presentadas previamente. Estos escenarios son:

- **Cambio 1**
  Este cambio es un ejemplo de un *bloqueo del laberinto*. La política óptima que se supone que el agente aprendió (ver Figura 3) es bloqueada en el episodio 100.



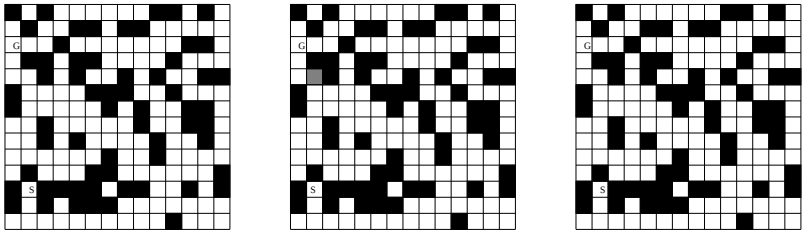
Cambio 1: Laberinto inicial (Izquierda), después del episodio 100 (Derecha)

- **Cambio 2**
  Este cambio es un ejemplo de la generación de un "atajo" en el laberinto. Un camino más corto es generado a partir del episodio 100 al eliminar la barrera que está por encima del estado de comienzo S.



Cambio 2: Laberinto inicial (Izquierda), después del episodio 100 (Derecha)

- **Cambio 3**
  Este experimento combina los dos tipos de cambios anteriores. Las siguientes figuras muestran que un bloqueo se produce en el episodio 100 y un atajo en el episodio 150 al remover la barrera introducida recientemente. Nótese la complejidad de este cambio para el algoritmo, dado que en 50 episodios tiene que aprender la nueva politica y luego descubrir que el camino original se volvió a abrir.



Cambio 3: Laberinto inicial (Izquierda), después del episodio 100 (Centro), después del episodio 150 (Derecha).

6.1 Resultados

A continuación presentaremos los resultados obtenidos con nuestra propuesta (Dyna-EDO) comparándola con la arquitectura Dyna con bono de exploración (Dyna-Bono).

Las gráficas que se muestran corresponden a los resultados promediados de 10 repeticiones de cada experimento. Las rotuladas como *recompensas acumuladas* muestran el retorno descontado que el agente ha *acumulado* hasta un determinado episodio. Las rotuladas como *recompensas descontadas* muestran el retorno que el agente obtiene *en cada* episodio. El problema fue definido con un valor de $\gamma = 0.95$. El valor del parámetro α es 1. El parámetro ϵ utilizado en Dyna-Bono fue inicializado en 0.0005 el cual demostró empíricamente ser el más adecuado para los experimentos. Dyna-EDO por su parte demostró su mejor performance seleccionando el parámetro $\lambda = 0.4$.

Sin Cambio

Como podemos apreciar en las siguientes figuras Dyna-EDO tiene un mejor desempeño en el ambiente estacionario que Dyna-Bono. Si bien de acuerdo a las recompensas descontadas ambos métodos aprenden la política óptima antes del episodio 25, podemos observar un comportamiento oscilatorio de Dyna-Bono (generado por la exploración de estados poco relevantes) que produce una performance más pobre que Dyna-EDO.

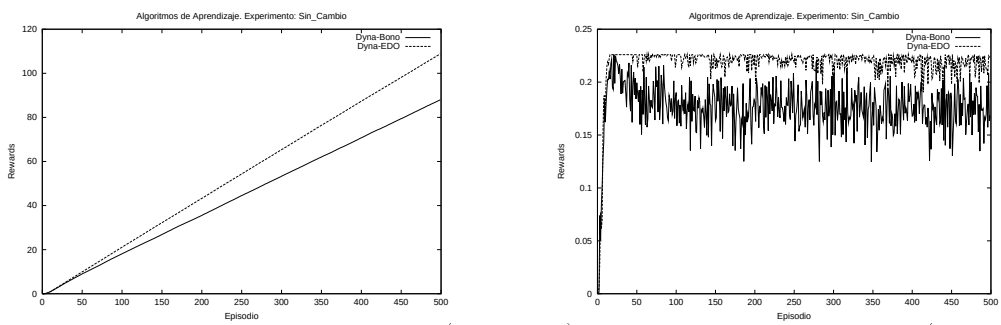


Figura 4: Recompensas acumuladas (izquierda) Recompensas descontadas (derecha)

Cambio 1

En el Cambio 1 (bloqueo de la trayectoria del agente) ambos métodos detectan rápidamente el bloqueo y aprenden la nueva política óptima. Las mismas consideraciones realizadas para el caso anterior sobre el comportamiento oscilatorio de Dyna-Bono son aplicables en este caso, resultando en una mejor performance de Dyna-EDO.

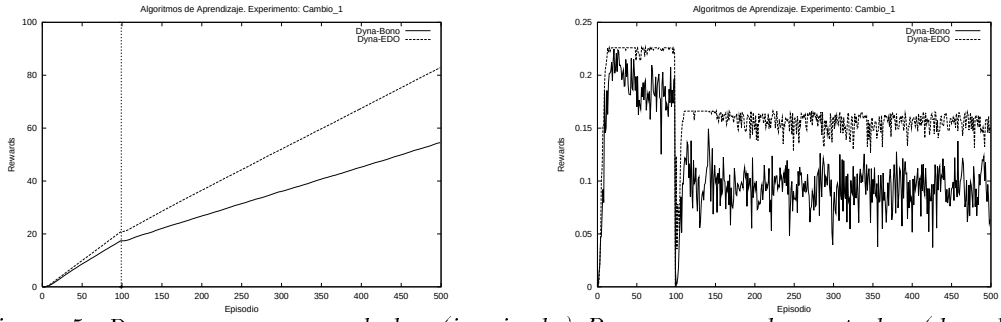


Figura 5: Recompensas acumuladas (izquierda) Recompensas descontadas (derecha)

Cambio 2

En este escenario la performance de ambos algoritmos es comparable, si bien aproximadamente a partir del episodio 175, comienza a notarse una pequeña diferencia a favor de Dyna-EDO.

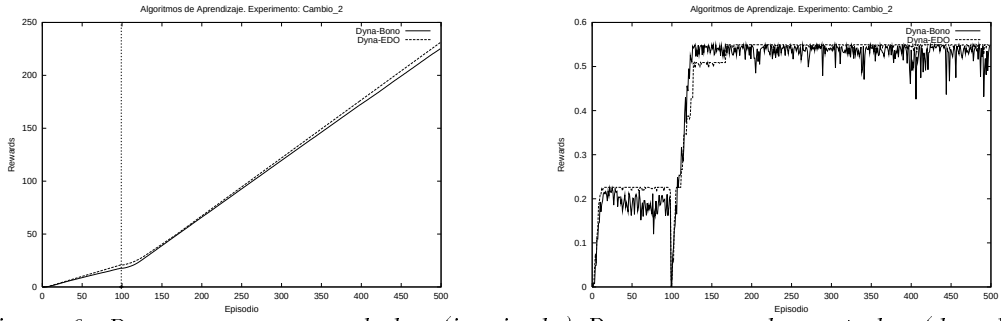


Figura 6: Recompensas acumuladas (izquierda) Recompensas descontadas (derecha)

Cambio 3

En el Cambio 3 se puede observar que después del cambio ocurrido en el episodio 150, a Dyna-EDO le cuesta abandonar el mínimo local formado. Sin embargo la heurística utilizada mantiene al agente rodeando el objetivo, sin moverse demasiado hacia zonas no productivas con lo cual finalmente logra superar a Dyna-Bono manteniendo la diferencia positiva de la pendiente.

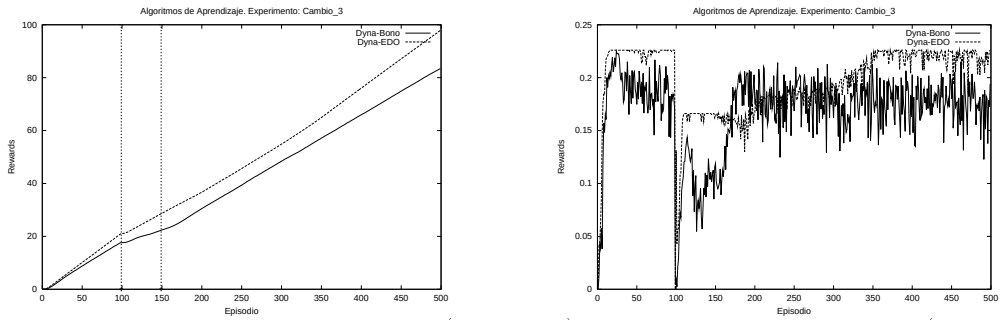


Figura 7: Recompensas acumuladas (izquierda) Recompensas descontadas (derecha)

Los resultados obtenidos en los 4 escenarios considerados, demuestran una clara superioridad de Dyna-EDO con respecto a Dyna-Bono. Si bien una demostración formal de porqué sucede este fenómeno está fuera de los alcances de este trabajo, intuitivamente uno puede decir que Dyna-EDO se concentra más sobre las trayectorias que conducen al objetivo y no explora tanto en las regiones más alejadas del mismo. Este hecho puede ser apreciado en las siguientes gráficas que muestran las veces que cada método visitó los distintos estados del laberinto en los experimentos realizados para el cambio 3. Estos números fueron discretizados en 8 intervalos y se les asignó una mayor intensidad de acuerdo a la cantidad de veces que fueron visitados.

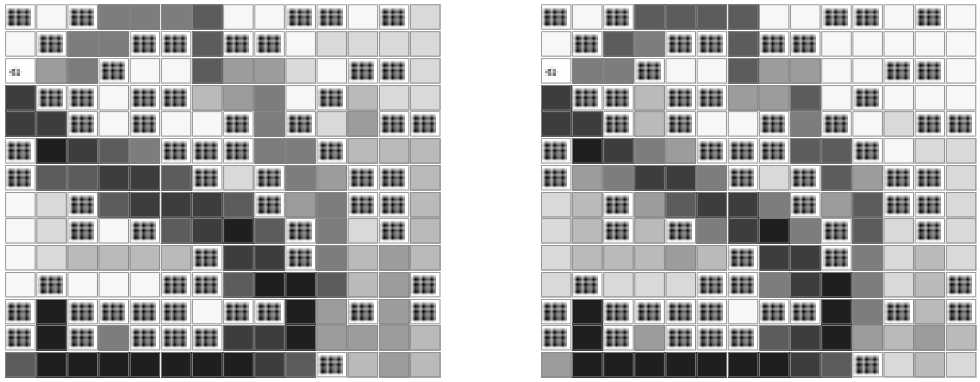


Figura 8: Dyna-Bono (Izquierda), Dyna-EDO (Derecha)

7 Conclusiones

Trabajos previos realizados con ARBM han demostrado su superioridad con respecto al AR sin modelo en ambientes estacionarios. Nuestras propias experiencias, reportadas en [2] han confirmado estos resultados utilizando procesos de planning que, con escasos requerimientos de tiempo y espacio mantienen la reactividad del agente como para su utilización en tiempo real. No obstante esto, en ambientes no estacionarios las bondades del ARBM son a menudo cuestionadas debido a las potenciales inconsistencias del modelo aprendido con el mundo real. En este sentido, consideramos importante el aporte de Sutton que con su arquitectura Dyna-Bono utilizó este mismo modelo para una exploración planificada que permitiera detectar los posibles cambios producido en el ambiente. Nuestro aporte con este trabajo es mostrar que la incorporación de un mayor conocimiento en la representación interna del agente posibilita mantener la adapabilidad del agente, y realizar una exploración más inteligente concentrándose en las partes relevantes del problema. En este sentido, nuestra propuesta de exploración dirigida por el objetivo demostró empíricamente la factibilidad de este enfoque con un mejor desempeño que la arquitectura Dyna-Bono. Los resultados obtenidos proveen evidencia de que en los ambientes no estacionarios, la exploración basada en una visión más global del sistema como la realizada cuando se conoce la ubicación del objetivo, es menos susceptible a las inconsistencias que pueden exhibir las funciones de valor por los cambios en el ambiente, y más inteligente que cuando sólo se toma en cuenta el tiempo transcurrido desde la última ejecución de la acción. Si bien nuestro trabajo, está actualmente circunscripto a problemas del tipo de los laberintos y es potencialmente aplicable a problemas de navegación de robots estamos actualmente analizando su aplicación en otros dominios más generales donde se puedan establecer otras medidas de distancia entre dos estados del problema.

Referencias

[1] Peter Dayan and Terrence J. Sejnowsky. Exploration bonuses and control dual. *Machine Learning*, 25(1):5–22, 1996.

[2] Marcelo Errecalde, Maria Liz Crespo, and Cecilia Montoya. Aprendizaje por refuerzo: Un estudio comparativo de sus principales métodos. In *Proceedings del Segundo Encuentro Nacional de Computación*, Mexico, 1999.

[3] Leslie P. Kaelbling. *Learning in Embedded Systems.* The MIT Press, Cambridge, Massachusetts, 1993.

[4] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[5] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: Reinforcement learning with less data and less time. *Machine Learning*, 13(1):103–130, 1993.

[6] Jing Peng and Ronald J. Williams. Efficient learning and planning within the dyna framework. In *Adaptative Behavior*, volume 1, pages 437–454. 1993.

[7] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice-Hall, 1995.

[8] Richard S. Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *Proceedings of The Seventh International Conference on Machine Learning*, pages 216–224, San Mateo, CA, 1990. Morgan Kaufmann.

[9] Richard S. Sutton. Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin*, 2:160–163, 1991. ACM Press.

[10] Richard S. Sutton and Andrew G. Barto. *Reinforcement learning: an introduction.* The MIT Press, 1998.

[11] Sebastian B. Thrun. The role of exploration in learning control. In David A. White and Donald A. Sofge, editors, *Handbook of Intelligent Control; Neural, Fuzzy and Adaptative Approachs.* Van Nostrand Reinhold, New York, 1992.

[12] Marco Wiering and Jurgen Schmidhuber. Efficient model-based exploration. In R. Pfeiffer, B. Blumberg, J. A. Meyer, and S. W. Wilson, editors, *Proceedings of The Fifth International Conference on Simulation of Adaptative Behavior: From Animals to Animats 5*, pages 223–228. The MIT Press, 1998.