Aprendizaje por refuerzo y control difuso para generar comportamiento de robots

Juan Carlos Gómez[1 2]; Claudio Abel Verrastro[1 3]
[1]UTN FRBA GIAR; [2]INTI; [3]CNEA
juanca@inti.gob.ar; cverra@cae.cnea.gov.ar

***Resumen*: La utilización de RL para generar comportamientos en sistemas de control y en robótica ha ganado un espacio significativo en las últimas décadas. La maldición de la dimensionalidad se hace presente y es necesario enfrentarla. En este trabajo se aborda el problema de utilizar aprendizaje por refuerzo cuando las variables de entrada y salida son continuas. Se propone el uso de sistemas de control difuso y un sistema de RL para calificar las reglas de control. Los estados continuos de entrada y las acciones de salida se representan mediante variables difusas. Se describe el algoritmo y se muestra como ejemplo un robot simulado "Lumbrí F3". Se establece una línea de base y se analizan los resultados obtenidos.**

***Palabras clave:* RL, FLC, SARSA(λ)**

I. INTRODUCCIÓN

La creciente sofisticación de las tareas que se le exige a un robot así como el cambio de las condiciones de carga, del ambiente y el incremento de los grados de libertad hacen que las herramientas de ingeniería clásica para la programación, la configuración y la coordinación de las acciones de cada una de las articulaciones, sean insuficientes para lograr un comportamiento adecuado del robot. La técnica de aprendizaje por refuerzo (RL) surge como una técnica prometedora para la generación de comportamientos, en forma automática, cuya complejidad excede las capacidades de los algoritmos de control clásicos.

El núcleo de esta técnica consiste en la incorporación del comportamiento, mediante prueba y error, interactuando con el ambiente, sin una programación explicita de la solución al problema.

La técnica de RL necesita recibir como realimentación la bondad de cada uno de los pasos realizados, llamada recompensa R. La "política" π es la función que genera las acciones a, teniendo en cuenta el estado actual s. Tanto los estados s como las acciones a pueden ser discretos o continuos. La recompensa R y la política π pueden ser funciones probabilísticas o deterministas. El aprendizaje consiste en la mejora continua de la política π que incrementa las recompensas R recibidas durante la realización de la tarea [1].

La técnica RL clásica se basa en un proceso de decisión Markoviano (MDP) consistente en un conjunto de estados S, un conjunto de acciones A, una función de recompensas R y una probabilidad de transición entre estados T que modela la dinámica del sistema describiendo los efectos de las acciones A. En este modelo, el estado siguiente s' y la recompensa r sólo dependen del estado s y la acción a.

En el caso de sistemas que tienen muchos estados y acciones posibles en cada estado, la exploración exhaustiva de todas las posibilidades es abrumadoramente larga, sino imposible. En el caso de RL la expansión combinatoria de estados-acciones tiene nombre propio: "Curse of Dimensionality" [2]. Se desarrollaron varias herramientas para limitar esta expansión [3] y tornar el espacio de búsqueda en un ambiente que sea suficientemente reducido para ser explorado en un tiempo razonable. Para esto se discretizan estados y acciones continuos [4] con una granularidad suficiente como para que el sistema pueda aprender de un número reducido de ensayos tipo "prueba y error" reduciendo la representación a un "grid-world" pero al aumentar la granularidad temporal se reduce la capacidad de representar las capacidades dinámicas [5], aumentando otros problemas aparejados al "sub-modelado". Los aspectos claves para afrontar, con éxito, el problema de la expansión combinatoria son: Una *representación eficiente, modelos aproximados* pero que capturen los aspectos esenciales de la realidad, y la incorporación de *conocimiento previo*. Por ejemplo para el caso de las acciones, en lugar de controlar cada uno de los grados de libertad del robot, se puede englobar el movimiento de varias articulaciones en una "meta acción" como "girar a la izquierda". El caso de la especificación y diseño de la función de recompensa es particularmente sensible, ya que a través de ella se modela el comportamiento deseado, con el agravante de que esta recompensa, en el mundo real puede estar varios pasos demorada. Por otra parte debe ser tal que permita el aprendizaje en forma incremental (los pasos intermedios de una mejor solución del problema deben ser mejor recompensados que otros que conducen al fracaso), es aquí donde se puede incorporar conocimiento previo en el dominio. Los métodos de diferencia temporal [6][3] abordan el problema de las recompensas demoradas

realizando una estimación de la recompensa futura esperada a partir de un estado dado o del par estado-acción, y realizan una actualización del valor del estado a partir de la diferencia entre la recompensa estimada y la verdaderamente obtenida. En este trabajo se presenta un método para abordar el problema del "Curse of Dimensionality" utilizando la segmentación difusa del espacio de búsqueda caracterizado por "features" o descriptores continuos y la utilización de meta-acciones, también continuas, para reducir los grados de libertad del robot desde el punto de vista del controlador. En los párrafos siguientes se describe el algoritmo de actualización de la función de valor de cada uno de los estados-acción que se representan como reglas de un controlador difuso. También se exponen los resultados de la implementación de un robot con 10 articulaciones coplanares que aprende a identificar un controlador difuso eficiente para desplazarse en el plano x-y.

II. MÉTODOS

En este trabajo se propone un algoritmo de aprendizaje por refuerzo (RL) para resolver un problema de control para el caso en que las variables, tanto de entrada como de salida, sean continuas. En la literatura existen ya mecanismos para dar solución a este problema, algunos segmentan el espacio de las variables para hacerlas discretas ("*tile coding*"); otras buscan características que representen un conjunto de situaciones (estados); también se utilizan métodos de aproximación de funciones para representar tanto políticas como funciones de valor, ver sección 2.4 en [1], y capítulo 8 en [6]. El uso de sistemas de control difuso [7] (FLC) resuelve el problema, desde el punto de vista del mapeo de un dominio continuo a un conjunto manejable de variables difusas. La cantidad de variables involucradas y la complejidad de los sistemas hace difícil contar con un experto para generar las reglas. Es entonces aquí donde uniendo ambos mundos se logra una solución. En este trabajo se propone utilizar un sistema de RL para que (aprenda) califique las reglas de un FLC en función de las recompensas obtenidas a medida que el sistema interactúa con el entorno.

En este caso se utiliza un conocido método de aprendizaje SARSA (λ) (Ver Capítulo 6 y 7 de [6]). A continuación se realiza una breve descripción del algoritmo de aprendizaje y de los sistemas de control difusos.

A. SARSA (λ)

Los métodos de RL pueden clasificarse, entre otras formas, según dos características, (Ver capítulo 10 [6]). La primera es la profundidad, que indica cuántos pasos adelante se utilizan para realizar una actualización (backup) de la función de valor buscada. La otra es la referida a cuántos estados se analizan en cada uno de los pasos anteriores, desde una muestra única hasta la consideración de todas las alternativas posibles (sample backup - full backup). Los métodos denominados de diferencia temporal (TD) (Ver capítulo 6 de [6]) son: métodos básicos para la estimación de las funciones de valor que consideran una instancia o prueba de uno o más pasos adelante. Con profundidad de un paso se encuentra TD(0), con varios pasos los métodos TD(λ) hasta la consideración de métodos Monte Carlo que consideran todos los pasos a futuro (full-return backup).

La función de valor buscada por estos métodos puede ser de dos tipos: La función de valor de **estado** se define como el valor de recompensa esperado a futuro a partir del estado considerado; La función de valor **estado-acción** es similar, pero considera el valor de tomar una determinada acción a partir del estado considerado.

SARSA(λ) [6] es un método TD que estima la función de valor estado-acción *Q(s-a)* utilizando el error temporal (diferencia) entre la estimación anterior de la función de valor y el valor actual de la misma incluyendo la recompensa recibida en la experiencia (muestra) recién realizada. Ver ecuaciones (1)(2). Esta actualización se realiza iterativamente tomando muestras hacia adelante. La actualización se retro propaga hacia todos los pares *s-a* recientemente visitados con el criterio de que la diferencia temporal, y la recompensa recibida no sólo es fruto de la última transición sino de las anteriores también. Cuanto más recientemente y más veces es visitado un par *s-a* mayor es la proporción que se propaga. El mecanismo utilizado con ese fin se denomina "*Elegibility trace*" (ET). El parámetro λ gobierna esta proporción, con valores entre (0,1) Ver ecuaciones (3)(4). A continuación se describe la regla de actualización:

$$Q'(s,a) = Q(s,a) + \propto \delta(s,a) et(s,a) \quad (1)$$

Donde:

$$\delta(s,a) = r + \gamma\, Q(s',a') - Q(s,a) \quad (2)$$

y el ET:

$$et'(s,a) = \lambda\, et(s,a) + 1 \quad (3)$$

si s-a es el par estado-acción visitado y si no:

$$et'(s,a) = \lambda\, et(s,a) \quad (4)$$

Se adoptó la nomenclatura del trabajo [3] donde *x'* representa el valor de *x* en el instante *t+1* y *x* en el instante t. Ver detalle en la Figura 1.

Este algoritmo es útil cuando la cantidad de estados y acciones lo hacen posible. La maldición de la dimensionalidad acecha.

En particular, para estados continuos, existen otros enfoques. Sutton y Barto proponen el "tile coding" [6],

donde dividen el dominio en estados discretos y aplican los mismos algoritmos. En la sección 2.4 de [1] se presenta la alternativa de "Aproximación de funciones". La función puede mapear estado a política o acción, funciones de valor *V(s)* y *Q(s,a)*, o directamente el modelo *T(s,a,s')* y *R(s,a)*. El algoritmo de aprendizaje ajusta esta función con los valores observados en la experiencia.

SARSA(λ)
Inicializar $Q(s,a) = xx$ y, $e(s,a) = 0$
Repetir (para cada episodio):
  Inicializar *s*, y luego *a* usando la política de exploración actual (ej: eGreedy)
  Repetir (para cada paso del episodio):
    Ejecutar acción *a* y observar *r* y *s'*
    Elegir acción *a´* (desde s´) con la estrategia de exploración actual (ej.: eGreedy)
    Calcular: $\delta(s,a)= r+\gamma Q(s',a')-Q(s,a)$
    Actualizar *elegibility trace et(s,a)*
    Para todo par estado, acción actualizar
    $Q(s,a):=Q(s,a)+\alpha\ \delta(s,a)\ et'(s,a)$
    s:=s' y *a:=a'*
  fin
fin

Figura 1 Algoritmo SARSA(λ)

B. CONTROLADORES DIFUSOS (FLC)

El empleo de FLC como método de representación del comportamiento, del sentido común o como una forma de razonamiento aproximado al de los seres humanos es ampliamente utilizado en control y robótica[8].

Un controlador difuso basa su funcionamiento en dos partes bien definidas, la lógica difusa y los sistemas basados en reglas [7] [9].

La lógica difusa permite calificar el valor tomado por una variable con distintas etiquetas. Por ejemplo un valor de temperatura del agua para el mate puede clasificarse como "*fría*", "*tibia*", "*a punto*" o "*muy caliente*". A diferencia de la lógica convencional, un mismo valor de temperatura (e.g. 84ºC) puede pertenecer a dos o más conjuntos a la vez y cada uno con cierto grado de pertenencia. (e.g. 0.8 a "*a punto*" y 0.4 a "*muy caliente*"). El grado de pertenencia se define en [0 1]. Esto permite describir el valor de la temperatura en función del grado de pertenencia a cada clase. De esta forma la veracidad de una afirmación como "*el agua para el mate está muy caliente*" puede cuantificarse por el grado de pertenencia del valor de temperatura referido al conjunto de temperaturas "*muy caliente*" dado por una función de membrecía (*mf*).

Una regla es una cláusula del tipo: "***si*** *antecedente* ***entonces*** *consecuente*". En el contexto de los sistemas de control el antecedente refleja una situación o estado de las variables de entrada y el consecuente se refiere a ejecutar determinada acción de control. Un sistema basado en reglas evalúa el grado de verdad del antecedente y si resulta verdadero dice que es una regla en condiciones de "disparar". Si se utiliza lógica difusa el grado de verdad del antecedente ya no es solamente 0 ó 1 sino que puede adoptar valores intermedios. El grado de verdad del antecedente se utiliza para indicar ahora la fuerza de disparo de la regla. Finalmente, un sistema basado en reglas resuelve salomónicamente el conflicto entre todas las reglas considerando su fuerza de disparo y la acción de salida considerada.

Un FLC realiza un ciclo de funcionamiento que comienza con difuminar el valor de entrada asignado un valor de pertenencia a cada etiqueta que califica a esa variable de entrada. A continuación evalúa el antecedente de todas las reglas del sistema considerando la composición lógica de cada uno. Típicamente el antecedente está compuesto por una conjunción de valores difusos, se toma entonces el mínimo de entre los que componen ese antecedente como valor de verdad de esa proposición. Este valor se define como fuerza de disparo. Luego para cada acción de salida posible se considera de entre aquellas reglas que la indican, la de mayor fuerza de disparo como la regla que mejor puede "opinar" en esa situación. Una vez que se realizó la misma operación para cada acción de salida, se resuelve entre las distintas "opiniones" pesando la acción a tomar en función de la fuerza de disparo. De esta manera se obtiene un valor concreto para la acción de salida. Esta última operación se resuelve generalmente con el método de centro de gravedad (COG)[7].

Para el diseño de un FLC se necesita un experto que pueda expresar su conocimiento acerca del problema en forma reglas.

Entre otras, las ventajas de este tipo de sistemas son: acciones de control suaves, facilidad de implementación y que no se necesitan un modelo preciso de la planta. Se necesita de un experto que exprese su conocimiento en forma de reglas y *mf*.

Existen en la literatura, otros acercamientos para sintonizar FLC. Entre ellos los trabajos de Andrea Bonarini que emplea algoritmos evolutivos para sintonizar y modificar reglas [8].

C. ALGORITMO PROPUESTO

Como se mencionó al comienzo de esta sección, la propuesta es utilizar un sistema de RL para que califique las reglas de un FLC.

El FLC se plantea con todas las reglas *r* posibles.

Para cada variable de entrada se define un grupo de entradas difusas y sus *mf* que permitan clasificar al valor de esa variable de entrada como perteneciente a uno o varios de esos conjuntos difusos. Esta operación pasa de un continuo de valores de entrada a un grupo pequeño de entradas difusas con sus respectivos valores de pertenencia. Esto constituye el estado del sistema.

Los antecedentes de las reglas se construyen con la conjunción de una variable difusa por cada variable de entrada. Se generan todas las combinaciones posibles.

Para la salida se procede de manera similar. Para cada variable de salida se define un conjunto de variables difusas de salida. Y se genera un conjunto de reglas con todas las combinaciones de entradas, para cada variable difusa de salida.

Cada regla será calificada por el sistema de aprendizaje y para eso se construye una tabla multidimensional *Q(sd,ad)* donde *sd* y *ad* se refieren a estado difuso y acción difusa respectivamente. Para cada regla se evaluará su valor *Q(r)*. El sistema de aprendizaje elegido fue SARSA (λ) dado que es un tipo de sistema de RL que aprende una función de valor.

El algoritmo así construido tiene una regla de actualización del tipo de las ecuaciones (1) y (2).

El algoritmo se muestra en la Figura 2.

La acción a realizar se obtiene del FLC a partir de una situación determinada siguiendo la estrategia de exploración-explotación definida, por ejemplo εGreedy [3]. Esta estrategia establece: elegir una acción al azar con una probabilidad ε, o la mejor disponible para la política actual. En el caso en que resulte que deba elegirse al azar, simplemente se conforma un FLC con una regla elegida al azar para cada antecedente posible. Para el caso de elegir la mejor acción, se conforma un FLC con la regla mejor calificada para cada estado posible.

Se resuelve el FLC de manera convencional y se obtiene la acción de control a aplicar. Devuelve además cuales fueron las reglas *r* utilizadas y sus respectivas fuerzas de disparo (*fd*). Notar que FLC devuelve una o más reglas *r*).

Se realiza la acción de control *a* sobre el sistema y se observa el cambio de estado *sd'* y la recompensa *rew* obtenida.

Con el objeto de considerar varios pasos para retro propagar la recompensa recibida y la diferencia de valor, se construye una especie de ET pero donde sólo se almacena, en una lista de estructuras ordenada, las reglas *r* recientemente utilizadas junto a las *fd* correspondientes.

Se calcula el *δ(r)* para cada acción difusa aplicada.

Finalmente, de manera similar a las ecuaciones (1) y (2), se actualiza el valor de todas las reglas recientemente visitadas y se les aplica una proporción λ^n del *δ(r)* actual correspondiente. Donde *n* es la cantidad de pasos desde que se disparó la regla considerada hasta el paso actual. Notar que si una misma regla es disparada recientemente más de una vez, se realiza más de una acumulación.

```
SARSA(λ)+FLC
Inicializar Q(r) = xx  y, Lista et(sd,ad) =[]
Repetir (para cada episodio):
  Inicializar sd
  Inicializar [a, r, fd] = FLC(s, métodoExploración)
  Repetir (para cada paso del episodio):
    Ejecutar acción a y observar rew y sd'
    [a', r', fd']  = FLC(s', métodoExploración)
    Calcular:  δ(r')= rew + γ Q(r')-Q(r)
    Actualizar lista et ←(r',fd')
    Para todo elemento de la lista
    Q(r):=Q(r)+α λ^n δ(r') fd'
    sd:=sd' ; ad:=ad' ; fd = fd'
  fin
fin
```

Figura 2 Algoritmo Propuesto SARSA(λ)+FLC

Si el sistema tiene más de una variable de salida esto puede replicarse para cada una de ellas.

D. EJEMPLO: LUMBRÍ F3

Como ejemplo de aplicación se implementó el algoritmo propuesto sobre un robot simulado denominado "Lumbrí F3". Este robot está basado en uno anterior desarrollado con fines didácticos presentado en dos tutoriales en los SASE 2013 y 2014 [10]. Se trabajó en MATLAB y sólo se representa su cinemática.

El modelo se asemeja en algo a una lombriz pero con ciertas particularidades que facilitan su descripción.

Lumbrí F3 está conformada por 10 vínculos y 9 segmentos, donde el primer vínculo es la cabeza y el último la cola. Ambos extremos con posibilidad de fijación. Se restringió a que todos los vínculos formen el mismo ángulo entre segmentos. Este modelo se mueve en un plano horizontal (x, y), con ángulos continuos entre -45º y +45º. (con todos en 0º queda estirada).

Las acciones posibles le permiten girar sobre el extremo fijo (cabeza o cola), establecer el ángulo entre vínculos y establecer el extremo fijo.

La longitud de un segmento es de 3unidades, por lo que la longitud de la Lumbrí F3 estirada es de 27 unidades.

Se pretende que el robot se mueva a izquierda en el eje de las x, por lo tanto se plantea una recompensa calculada como:

$$rew' = -(x' - x)/L + r' - 0.1 \qquad (5)$$

Donde *x* es la posición más a la izquierda de la Lumbrí F3, *L* es su longitud y *r* es un valor devuelto por el modelo que refleja dolor, costo o satisfacción interna al ejecutar las últimas acciones. Este último, r, no se utiliza en esta oportunidad. El 0.1 se resta para incentivarla a que resuelva más rápidamente.

El sistema se define de la siguiente forma:

Variables de entrada:

1. La primera variable de entrada indica el grado de estiramiento, y tendrá 3 valores difusos posibles: *muyEstirada*, *estirada* y *pocoEstirada*. Se calcula midiendo la distancia desde la cola a la cabeza relativa a la distancia máxima. Valores entre 0 a 1.
2. Contempla la posición del extremo libre de la Lumbrí F3, respecto de su extremo fijo a izquierda o derecha, Son 3 valores difusos: *izquierda*, al *centro* o a la *derecha*.
3. Esta es similar a la anterior pero con valores *arriba*, al *medio* y *abajo*.
4. Cabeza fija o cola fija. 2 Valores discretos.

En todos los casos se definen *mf* simples del tipo trapezoidal o triangular.

Son entonces 4 dimensiones con 3x3x3x2=54 valores difusos posibles. (estos conformarán los antecedentes de cada regla). Cada elemento es un estado posible.

Las variables de salida (acciones) se definen:

1. Incremento del ángulo del extremo actualmente fijo. Se definen 3 acciones difusas, *incrementar* en 30º, 0º y -30 º.
2. Curvar el cuerpo (moviendo los ángulos entre segmentos adyacentes). 3 acciones difusas. Curvar a *Izquierda* (45º), *enderezar* (0º), curvar a *derecha*(-45º).
3. Fijar cabeza/cola. *Fijar cabeza* (1), *dejar como está* (0) *fijar cola* (-1). En este último caso, como la salida no tiene valores intermedios, se establecieron umbrales para hacer la fijación de la cabeza o cola. Son también 3 acciones difusas.

En todos los casos son 3 valores "crisp" [7].

Respecto de las reglas: Como ya se mencionó, se consideran todas las reglas posibles. Hay un conjunto de reglas para cada variable difusa de salida. En todos los casos con todos los antecedentes difusos posibles, uno por cada variable de entrada. En este caso 4.

Para llevar la calificación de las reglas se construye una matriz para cada variable de salida, cada una con un elemento para cada acción difusa posible. La cantidad de elementos de cada una es de 3x3x3x2x3=162. Donde el último 3 corresponde a la cantidad de variables difusas de salida. Por ejemplo: Incrementar el ángulo del extremo fijo, tiene 3 variables difusas: *incrementar*, *dejar como está* y *decrementar*. Sus respectivos valores son +30º, 0º y -30º.

Como son 3 variables de salida, cada una con 3 variables difusas, la cantidad total de elementos es: 3x3x3x2x3x3= 486 reglas posibles. Cada elemento corresponde al valor de la calificación de cada regla.

El FLC se resuelve de manera ordinaria, usando el mínimo de sus antecedentes para calcular la *fd* de cada regla y para los consecuentes se usan las reglas con la máxima *fd*. Finalmente el valor para cada acción se resuelve con el método de centro de gravedad [7] .

Dado que en este ejemplo hay 3 acciones de control, las mismas se aplican siempre en un mismo orden sobre la Lumbrí F3 y se obtiene un nuevo estado *sd'* y una recompensa *rew*.

III. EXPERIMENTACIÓN

Se construyó un modelo de la Lumbrí F3 y se probó su funcionamiento.

Todas las experiencias se realizan a partir de un estado inicial aleatorio de Lumbrí F3 con el extremo fijo en la posición (0,0). Siempre se realizan episodios de 1000 pasos cada uno en la siguiente secuencia:

a) 10 iteraciones con método de exploración "naive".
b) 200 iteraciones con método de exploración "εGreedy". Las 10 últimas con una animación que permite visualizar los movimientos del robot.

Se almacenan los valores alcanzados en las últimas 200 experiencias para graficar la curva de aprendizaje.

El ET se construyó con una profundidad de 7 pasos. El factor de aprendizaje α se hizo dinámico, comenzando por 0.01 hasta 0.0005 al final de la experiencia. El factor de descuento γ en 0.9. El ε también se hizo dinámico desde 0.1 a 0.005. El factor λ para el desvanecimiento de la memoria del ET se fijó en 0.95.

Con el objeto de establecer una referencia se calcula la distancia que recorre la Lumbrí F3 si establece una secuencia de acciones fija tal y como lo podría hacer un programador.

Si se parte de la posición estirada, con ángulo 180º (acostada sobre el eje x) y se programa una secuencia rígida de 6 acciones, cada una con un giro sobre el extremo fijo de 30º y en la 6ª acción, se agrega el cambio del extremo de fijación, se avanzarán 27 unidades, que es el equivalente al largo de la Lumbrí F3. Si se ejecutan 1000 acciones, se puede avanzar 4500 unidades.

IV. RESULTADOS

Luego del diseño y la implementación se obtuvo un tamaño de 486 elementos para la matriz *Q(r)*.

En la Figura 3 se muestran las distancias alcanzadas para cada iteración. Es una curva de aprendizaje típica

Se puede mostrar un gráfico histórico con la distancia alcanzada en los 1000 pasos de cada experiencia. En cada oportunidad se parte de posturas aleatorias. El promedio de las últimas 20 experiencias del aprendizaje es 10884 unidades.

En otra experiencia de aprendizaje, una vez extractado el FLC, y partiendo de posturas aleatorias, pero siempre desde (0.0) se obtuvo en un promedio de 20 corridas una distancia de 10054 unidades.

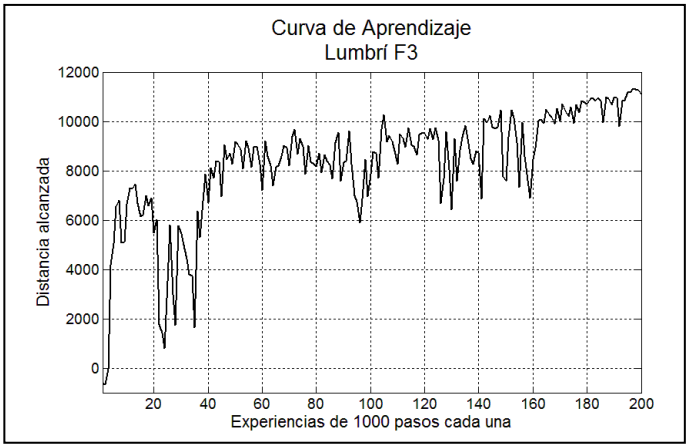

Figura 3 Curva de aprendizaje típica. 200 experiencias de 1000 pasos

V. CONCLUSIONES

La utilización de un conjunto de variables difusas para identificar el estado de las variables continuas de entrada redujo considerablemente la cantidad de estados posibles sin perder la condición de variables continuas. La ventaja inmediata es la disminución en cantidad de recursos necesarios para su almacenamiento y en el tiempo de exploración.

Durante la experiencia se estableció un valor de referencia de 4500 unidades para una secuencia de 1000 pasos realizada con una programación rígida y el algoritmo propuesto superó el doble de esa referencia.

En el trabajo previo se utilizó un modelo distinto que utiliza una representación tabular sobre el espacio *s-a*.

Sus características fueron: 5 vínculos (incluyendo cabeza y cola. 4 segmentos. 7 ángulos discretos entre vínculos no extremos. 5 ángulos para el extremo fijo y 27 acciones posibles. Todo esto hace un total de 3430 estados y la matriz de valores *Q(s,a)* de 92610.

Si el modelo anterior tuviese la misma cantidad de vínculos que Lumbrí F3, y se intentara resolver con ese mismo enfoque, se puede estimar la cantidad de estados en 57648010 y el tamaño de la matriz *Q(s,a)* en 3.574.176.620 elementos.

Con la metodología propuesta se empleó una matriz de sólo 486 elementos. Si con este mismo enfoque se generara una salida para cada uno de los ángulos entre vínculos la cantidad de reglas sería 3x3x3x2x3x10= 1620 elementos. Donde el 10 es la cantidad de acciones.

Otra ventaja del enfoque propuesto es que una vez realizado el aprendizaje (calificación de las reglas) se pueden extraer las reglas mejor calificadas para cada una de los estados posibles e implementar un controlador difuso ordinario. Si se supone que las características del ambiente y del robot no cambian se puede utilizar al robot en modo producción con un FLC fijo.

Como trabajo futuro se planea combinar este método con un mecanismo que modifique automáticamente las *mf*[8], y optimizar así el FLC obtenido.

REFERENCIAS

[1] J. Kober, J. A. Bagnell y J. Peters, «Reinforcement Learning in Robotics: A Survey,» *International Journal of Robotics Research,* vol. 32, nº 11, pp. 1238-1274, 2013 July.

[2] R. Bellman, «Dynemic Programming and the Numerical Solution of Variational Problems,» Operational Research, vol. 5, nº 2, pp. 277 - 288, Apr 1957.

[3] L. Kaelbling, M. Littman y A. Moore, «Reinforcement Learning: A Survey,» de Journal of Artificial Intelligence Research 4, 1996.

[4] L. Busoniu, R. Babuska, B. De Schutter y D. Ernst, Reinforcement Learning and Dynamic Programming Using Function Approximators, Boca Raton, FL: CRC Press, 2010.

[5] S. Schaal, C. Atkeson y S. Vijayakumar, «Scalable Techniques from Nonparametric Statistics for Real Time Robot Learning,» Applied Intelligence, vol. 17, nº 1, pp. 49 - 60, 2002.

[6] R. Sutton y A. Barto, Reinforcement Learning: An Introduction, Cambridge, MA: MIT Press, 1998.

[7] J. C. Gómez, Fuzzy Control, Buenos Aires: edUTecNe - Editorial Universitaria de la Universidad Tecnológica Nacional, 2008.

[8] A. Bonarini, «Learning Behaviors Implemented as Fuzzy Logic Controllers for Autonomous Agents,» de Second Online Workshop on Evolutionary Computation, 1996.

[9] L. Reznik, Fuzzy Controllers, Oxford: Butterwoth-Heinemann, 1997.

[10] J. C. Gómez y C. A. Verrastro, «Simposio Argentino de Sistemas Embebidos SASE 2013,» 14 08 2013. [En línea]. Available: http://www.sase.com.ar/2013/files/2013/09/SASE2013-AprendizajePorRefuerzo.pdf. [Último acceso: 17 08 2014].