Sistema de aprendizaje para evitar obstáculos verticales en robots de locomoción diferencial redundante

Arroyo Dominguez Victor Hugo[1] and Ocampo Jiménez Jorge[2]

Instituto Nacional de Astrofísica, Óptica y Electronica, Tonantzintla Puebla, México

Abstract. Este documento describe el desarrollo y las pruebas para el control de un comportamiento para un robot multi-articulado de locomoción diferencial. El control fue desarrollado utilizando dos algoritmos de aprendizaje por refuerzo, utilizando las técnicas de programación dinámica y diferencia temporal. El sistema de control fue probado en un simulador con el fin de implantarlo en el robot real. El desempeño entre los dos tipos diferentes de algoritmos, se realizo utilizando un obstáculo vertical el cual el robot tuvo que sortear en el mínimo número de movimientos alcanzados por los algoritmos de aprendizaje por refuerzo.

1 Introducción

En el diseño de un robot, siempre se tiene el compromiso entre funcionalidad y consumo de potencia. En particular, en este proyecto, se busco poder sortear obstáculos que generalmente son difíciles o imposibles para un robot móvil del tipo diferencial, como lo son los obstáculos verticales.

El robot que se busca controlar se muestra en la figura 1. Cada segmento posee dos grados de libertad; uno por articulación, y otro proporcionado por las llantas para el movimiento horizontal. Cada articulación fue planeada para moverse solamente en dos grados, 0 y 90 respectivamente, con el fin de limitar la complejidad del control del robot.

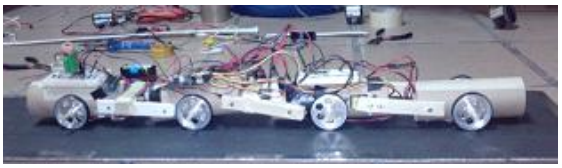

Fig. 1. Robot articulado terrestre.

El diseño del robot fue pensado para realizar un sistema de control de complejidad baja, sin embargo, en el momento del desarrollo se pudo observar incertidumbre en la funcionalidad del mismo. Al intentar subir obstáculos verticalmente, dirigido por un operador, se observó que la estrategia imaginada no

funcionaba. Aun así, se procedió a implementar el control con base en el diseño inicial del robot.

Se pensaron en diversos métodos para el control del robot. El objetivo siempre fue que el robot fuera capaz de descubrir la estrategia para poder subir los obstáculos de manera autónoma. Debido al ambiente incierto se optó por un algoritmo de aprendizaje por refuerzo (RL, por sus siglas en inglés), el cual exploraría el ambiente y nos daría información acerca de las frecuencias consecuentes a las acciones del robot, con lo que podría funcionar como las *observaciones* para un algoritmo de aprendizaje por refuerzo utilizando modelos gráficos probabilistas.

El objetivo principal del desarrollo del control, fue lograr implementar un comportamiento en el robot real el cual le permitiera sortear obstáculos verticales automáticamente, y a su vez, proporcionarnos información acerca de las ventajas y desventajas de usar un algoritmo de aprendizaje por refuerzo con el enfoque de programación dinámica para el control de movimientos de un robot. ¿Qué sensores necesitábamos para obtener retroalimentación del ambiente? ¿Realmente se puede generar una política que lleve a la resolución de la tarea con el diseño propuesto? ¿Existe una ventaja entre los diferentes enfoques de aprendizaje por refuerzo?. Estas preguntas, son las premisas de este trabajo.

2 Trabajo relacionado

En la literatura, al tipo de robots con varias articulaciones y locomoción del tipo diferencial, se les conoce comúnmente como *"snake like robot"*. Esta variedad interesante de locomoción trata de utilizar no solo el movimiento de serpenteo, si no también la habilidad de moverse rápidamente de manera horizontal, proporcionándole una amplia gama de grados de libertad. Como ejemplos podemos encontrar a ACM-R3 mostrado en la figura 2, Soryu I mostrado en la figura 3 o Genbu mostrado en la figura 4.

Los mecanismos de control para este tipo particular de robots son complejos, debido a los grados de libertad involucrados, así también para mecanismos de planeación de movimientos. Un caso interesante de estudio es el robot OT-4 [1], el cual utiliza el sistema de control 7G, el cual incluye una red neuronal para aprendizaje por refuerzo, así como un algoritmo genético para optimizar los parámetros de entrenamiento del sistema, en la figura 5 se muestra el robot OT-4. En la figura 6 se muestra el diagrama esquemático del sistema de control 7G.

Recordemos que aprendizaje es la habilidad deseada para todo robot autónomo, con el fin de poder enfrentar situaciones que no están explícitamente definidas por su control. La idea de lo que es aprendizaje en lo seres humanos surge principalmente de la interacción del sujeto con el ambiente para lograr una tarea, a lo largo de la vida del ser humano, la interacción repetida de las mismas tareas nos da más información del ambiente y de nuestro comportamiento. En aprendizaje por refuerzo un agente trata de aprender un comportamiento mediante interacciones de prueba y error en un ambiente dinámico e incierto.

Fig. 2. Robot ACM-R3

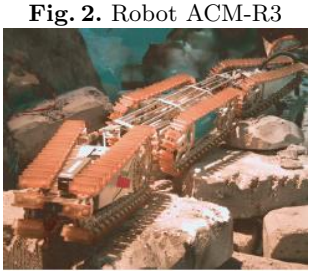

Fig. 3. Robot Soryu I

Fig. 4. Robot Genbu

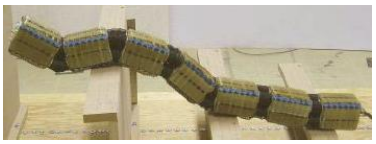

Fig. 5. Robot OT-4.

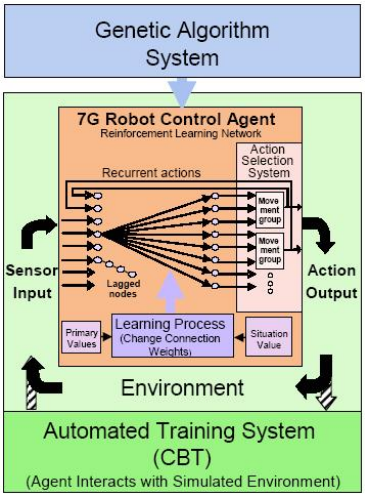

Fig. 6. Diagrama esquemático del sistema de control 7G tomado de [1].

3 Metodología y desarrollo

La metodología para el desarrollo de robots con control basado en MDP's se puede observar en la figura 7

El primer paso, fue definir el sistema: elegir el tipo de robot que llevaría a cabo la tarea deseada. Se partió de la premisa de que los robots articulados nos darían la flexibilidad suficiente en la construcción del prototipo así como las ventajas de los robots de locomoción con ruedas.

Fig. 7. Metodología de desarrollo.

Partiendo del sistema, se procedió paralelamente a definir el modelo, así como a elegir el controlador para el robot. El modelo, primeramente seria resuelto en un simulador, debido a que la tarea de resolver un modelo de este estilo conlleva a hacer pruebas exhaustivas, el simulador permitiria acortar el tiempo de desarrollo, ademas de superar las limitaciones del controlador.

El diseño preliminar del prototipo, nos ofreció información acerca del tipo de actuadores que necesitábamos: 4 servomotores, y 4 motores con movimiento paralelo; traducido al modelo, los actuadores nos ofrecieron las acciones que teníamos que representar en el modelo, las cuales se pueden observar en la tabla 1.

Uno de los primeros problemas que surgieron fue la definición de los estados del modelo, si bien se tenia el diseño, aun permanecía la pregunta de cuales serian los sensores que darían la información suficiente para representar un estado. La primera propuesta fue utilizar sensores de rango, lo cuales darían información acerca de que tanto el robot avanzaba, sin embargo, la simulación mostró que era impractico tratar de discretizar, y ademas las limitaciones en el controlador no nos dejarían llevar a acabo aquella implementación. Decidimos utilizar fotorresistencias, y cambiar el ambiente, las fotorresistencias nos regresarían un valor para colores claros, y otro para colores fuertes, con lo que podemos representar, subir el escalón con un color claro, y no subirlo con un color fuerte, lo que nos

Acciones
Servo 1 arriba
Servo 2 arriba
Servo 3 arriba
Servo 4 arriba
Servo 1 alineado
Servo 2 alineado
Servo 3 alineado
Servo 4 alineado
Avanzar

Table 1. Acciones del modelo del robot.

deja una representación binaria de 0 y 1. Se coloco una fotorresistencia por segmento, lo que nos dio 4 bits (16 posibles estados) inicialmente. En la figura 8 se ilustra esta representación.

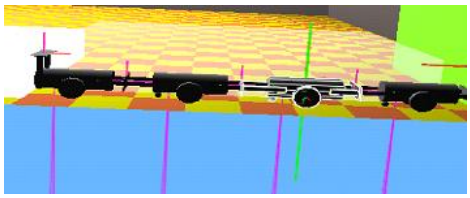

Fig. 8. Las líneas rosadas indican la posicion de las fotorresistencias.

Sin embargo esta representación también resultó impractica por la falta de expresividad, se tenia una "representación horizontal" del estado del robot, pero no se contaba con una vertical, asi que se asignaron 4 bits mas para la representación vertical, que es equivalente a decir si el servomotor n esta en posición vertical u horizontal. Finalmente, la representación final fue de 8 bits (256 estados), donde los 4 bits menos significativos representan las posiciones verticales, y los 4 bits mas significativos las posiciones horizontales en el ambiente. El bit en la posición n, representa al segmento n (visto del frente hacia el final del robot), así que los estados finales son representaciones decimales de números mayores a 127.

Segmento 4	Segmento 3	Segmento 2	Segmento 1	Servo 4	Servo 3	Servo 2	Servo 1
0 Abajo	0 Abajo	1 Arriba	1 Arriba	0 Abajo	0 Abajo	1 Arriba	0 Abajo

Table 2. Representacion de 8 bits de los estados del robot

El siguiente paso fue resolver parte del problema, primero utilizamos un algoritmo de diferencia temporal, esto con el fin de encontrar una solución al control de movimientos del robot para subir el obstáculo vertical, y la segunda, explorar el ambiente para obtener observaciones de las transiciones del robot para aplicarlo en el algoritmo de programación dinámica. Para esta tarea se eligió un algoritmo on-line con propagación, Sarsa λ. Se eligió un algoritmo on-line ya que obtendríamos más frecuencias de los estados *"más importantes"* para la solución que se obtuvieran en la fase de exploración, además de que la convergencia sería mas rápida por utilizar trazas de legibilidad. Se utilizo una exploración tipo ϵ-greedy con parámetro 0.6 con el fin de ayudar a obtener el mayor número de frecuencias para el modelo del MDP. Debido a que Sarsa λ, disminuyó el valor del trayecto de la recompensa conforme se aleja de ella, se eligieron las funciones de recompensa como 0, para todos los estados menores a 128, con lo que se obtenían valores cercanos a cero para estados iniciales, y 10 para estados mayores o iguales a 128, que representan los estados finales, con el bit mas significativo (ultimo segmento) arriba del obstáculo.

Una vez obtenidas las frecuencias, se utilizo el algoritmo de programación dinámica, *value iteration*, para resolver el MDP. El teorema de *value iteration* asegura que se obtendrá la política optima en el limite.

4 Experimentos y Resultados

Se realizó la ejecución del algoritmo sarsa λ con 50 episodios desde el estado inicial 0 y estado final mayor o igual a 10000000 (base 2). Se utilizó una exploración ϵ-greedy con ϵ igual a 0.6 este parámetro se eligió así para favorecer la exploración de mas estados dado la política. Se eligió un λ de 1, un α de 0.1 y un γ de 0.9, después de 13 horas simuladas equivalente a 4 horas reales no se logro completar ningún episodio. Debido a el tiempo en el que procesa el simulador las variables físicas que intervienen en el modelo, combinado con el espacio de búsqueda hace intratable el numero de estados, por lo que se procedió a dividir el problema en subproblemas (expresados en el prototipo como la acción de subir cada uno de los segmentos), que si bien podría no ser la solución global optima, obtenemos esas soluciones más rápidamente en esos subespacios de estados, las cuales las podemos combinar para obtener una solución global.

Una vez dividido el problema se realizaron ejecuciones del algoritmo sarsa λ con 20, 50, y 100 episodios, los resultados obtenidos se muestran en la tabla 3.

Número de episodios	Acciones realizadas
20	19
50	17
100	17

Table 3. Resultados de la solución global alcanzados mediante la fusión de las subsoluciones.

El siguiente paso, fue utilizar las frecuencias obtenidas por el algoritmo Sarsa λ, con las cuales se calcularon las probabilidades, que fueron la frecuencia a la transición del estado *s'* al estado *s* con la acción *a*, entre la suma total de frecuencias de estados/acción para los estados *S*. Si bien se tenía una estimación de algunos estados, existían aun estados sin estimaciones de probabilidad, debido a que la exploración no recorrió todos los estados, con lo que se procedió a *"rellenar"* estas probabilidades. Se asignaron entonces a los estados/acciones no recorridos por el algoritmo, asignaciones equiprobables. La función de recompensa elegida fue de -1 para todos los estados menorea a 128, y 10 para el caso contrario. El parámetro θ=1, y ϕ=0.9. El resultado observable para la simulación fue que el robot caía en ciclos debido a la política, lo que implica que no llegaba nunca a la posición final, esto es atribuido, a que se le asignaron probabilidades arbitrariamente a transiciones que son imposible para el robot, por lo que a largo plazo, daban esas transiciones imposibles mayor recompensa que las registradas por las frecuencias.

El siguiente paso, fue replantear las probabilidades faltantes, si bien, no se tenia certeza del siguiente estado, se podía suponer, que todas las probabilidades faltantes eran cero, y que todas las acciones llevadas en un estado desconocido, llevaban al mismo estado. Con lo que se aplico el algoritmo con estas suposiciones arrojando los siguientes resultados.

Número de episodios	Acciones realizadas
20	14
50	11
100	14

Table 4. Resultados de la solución global alcanzados mediante la fusión de las subsoluciones.

Con lo que se muestra, una clara diferencia en el número de acciones alcanzadas con el algoritmo de programación dinámica y Sarsa λ. El mínimo de acciones alcanzadas por las observaciones registradas se alcanzo con tan solo 20 episodios. Se atribuye que se haya alcanzado con 50 episodios un número de acciones menores al de 100, debido a que en 100 episodios, hubo mas oportunidades de fallo con respecto a la política.

Debido a que se obtuvieron buenos resultados, se fue mas allá en las suposiciones, y se planteó que la posición de los servomotores es totalmente determinista, si a un servomotor se le da la orden de subir o bajar, estos pasan indudablemente a ese estado. Así que se asignaron como uno dichas probabilidades, con lo que se obtuvo lo siguiente:

Con lo que 11, se estima que es el numero óptimo de acciones para llegar a la meta (la estimación es basada en observaciones del experto). Lo cual esta cerca de alcanzar el algoritmo con una frecuencia obtenida de 20 episodios agregando conocimiento experto. En la figura 9 se muestra la comparación de los resultados mostrados en los cuadros anteriores.

Número de episodios	Acciones realizadas
20	12
50	11
100	12

Table 5. Resultados de la solución global alcanzados mediante la fusión de las sub-soluciones.

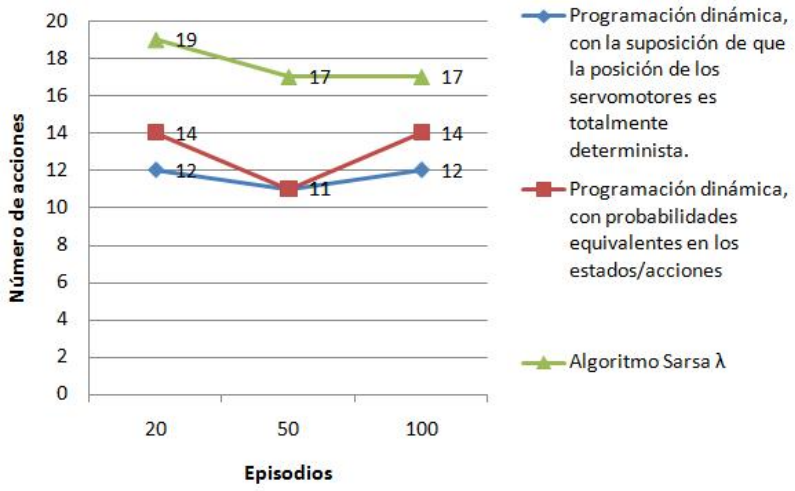

Fig. 9. Comparacion de resultados.

5 Conclusiones y trabajo a futuro.

- Se pudo obtener un modelo de las acciones y los estados a partir de un prototipo conceptual, estas acciones y estados se definieron también el desarrollo del prototipo real.
- Se obtuvo un comportamiento (política) para el prototipo dadas las interacciones con el ambiente (simulado).
- En esta clase de problemas es importante el modelado, ya que el numero de estados afecta exponencialmente el tiempo que tarda en alcanzar la política óptima.
- Para problemas con espacio de búsqueda muy grande es recomendable implementar la solución en un simulador.
- Proceder a realizar más experimentos comparando el número de episodios para la convergencia de sarsa λ.
- Se debe definir un algoritmo de exploración lo suficientemente eficiente, para obtener el mayor número de frecuencias de los estados importantes y menor número de frecuencias de los estados que son menos frecuentes.
- Se necesita un gran número de probabilidades, lo cual repercute enormemente en el tiempo en que se obtendrá una solución del modelo.
- Las observaciones de los expertos, no pueden ser dadas a la ligera, fácilmente pueden llevar a modelos inconsistentes.
- Las observaciones correctas de los expertos pueden favorecer enormemente la eficiencia del modelo.
- La combinación de model-learning y reinforcement-learning arroja resultados bastante aceptables al utilizar estos dos tipos de enfoques por separado, sin embargo, el tiempo de procesamiento se ve incrementado, debido a esta mezcla de algoritmos. Se puede observar claramente el como se llego a la idea de algoritmos como Dyna-Q y Prioritized Sweeping.
- La simplificación del modelo es importante para la reducción de componentes en el hardware de control.
- Puede pensarse en extender el funcionamiento del robot creando nuevos comportamientos.

References

1. Hutchison, William R. and Constantine, Betsy J. and Borenstein, Johann and Pratt, Jerry. Development of Control for a Serpentine Robot 7th IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA2007) June 20-23, 2007 Jacksonville, USA
2. Sutton, Richard and Barto, Andrew G. Reinforcement Learning: An Introduction The MIT Press Cambridge, Massachusetts London, England
3. Hirose, Shigeo and Fukushima Edwardo F. Snakes and Strings: New Robotic Components for Rescue Operations Tokio Institute of Technology 2-12-1 Ookayama Meguro-ku, JAPAN
4. Granosik, Grzegorz and Borenstein Johann Integrated Joint Actuator for Serpentine Robots IEEE/ASME Transactions on Mechatronics, Vol 10, No 5 October 2005, pp 473 - 481