UNIVERSIDAD POLITÉCNICA SALESIANA
SEDE QUITO

CARRERA:
INGENIERÍA ELECTRÓNICA

Trabajo de titulación previo a la obtención del título de:
INGENIEROS ELECTRÓNICOS

TEMA:
ANÁLISIS COMPARATIVO DE ALGORITMOS NEURO COMPUTACIONALES BIOLÓGICOS PARA PROCESO COGNITIVO DE LA MEMORIA Y SU APLICACIÓN EN EL ROBOT NAO

AUTORES:
ANDRÉS SANTIAGO GÓMEZ VEGA
OMAR FERNANDO GUERRERO RIVERA

TUTORA:
CARMEN JOHANNA CELI SÁNCHEZ

Quito, marzo del 2016

Cesión de derechos de autor

Nosotros, Gómez Vega Andrés Santiago, con documento de identificación N° 1723671234 y Guerrero Rivera Omar Fernando con documento de identificación N° 1719653600, manifestamos nuestra voluntad y cedemos a la Universidad Politécnica Salesiana la titularidad sobre los derechos patrimoniales en virtud de que somos autores del trabajo de grado titulado: ANÁLISIS COMPARATIVO DE ALGORITMOS NEURO COMPUTACIONALES BIOLÓGICOS PARA PROCESO COGNITIVO DE LA MEMORIA Y SU APLICACIÓN EN EL ROBOT NAO, mismo que ha sido desarrollado para optar por el título de: Ingenieros Electrónicos, en la Universidad Politécnica Salesiana, quedando la Universidad facultada para ejercer plenamente los derechos cedidos anteriormente.

En aplicación a lo determinado en la Ley de Propiedad Intelectual, en nuestra condición de autores nos reservamos los derechos morales de la obra antes citada. En concordancia, suscribimos este documento en el momento que hacemos entrega del trabajo final en formato impreso y digital a la Biblioteca de la Universidad Politécnica Salesiana.

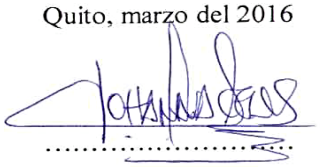
.....
Nombre: Andrés Gómez
Cédula: 1723671234
Marzo de 2016

.....
Nombre: Omar Guerrero
Cédula: 1719653600
Marzo de 2016

Declaratoria de coautoría del docente tutora

Yo, declaro que bajo mi dirección y asesoría fue desarrollado el trabajo de titulación ANÁLISIS COMPARATIVO DE ALGORITMOS NEURO COMPUTACIONALES BIOLÓGICOS PARA PROCESO COGNITIVO DE LA MEMORIA Y SU APLICACIÓN EN EL ROBOT NAO realizado por Andrés Santiago Gómez Vega y Omar Fernando Guerrero Rivera, obteniendo un producto que cumple con todos los requisitos estipulados por la Universidad Politécnica Salesiana para ser considerados como trabajo final de titulación.

Quito, marzo del 2016



.....

Carmen Johanna Celi Sánchez

Cédula de identidad: 1717437808

ANÁLISIS COMPARATIVO DE ALGORITMOS NEURO COMPUTACIONALES BIOLÓGICOS PARA PROCESOS COGNITIVOS DE LA MEMORIA Y SU APLICACIÓN EN EL ROBOT NAO.

Omar Guerrero
Departamento de Ingeniería Electrónica
Universidad Politécnica Salesiana
Quito-Ecuador
oguerrero@est.ups.edu.ec

Andrés Gómez
Departamento de Ingeniería Electrónica
Universidad Politécnica Salesiana Ecuador
Quito-Ecuador
agomezv@est.ups.edu.ec

Carmen Celi
Departamento de Ingeniería Electrónica
Universidad Politécnica Salesiana Ecuador
Quito-Ecuador
cceli@ups.edu.ec

***Abstract*— This article presents a detailed neuro-computational algorithms study, the first based on reinforcement learning called Q-learning and the second based on information or updated algorithm called A* Star, and then compare them in principle from its theoretical concepts through its mathematical form and then with the prior knowledge of its structure, implement them in a humanoid robot NAO. This implementation is carried out by programming software called choreographe, which is understandable to the robot named Naoqi and allows the programming language Python. First the algorithms used individually to verify their behavior in the humanoid robot NAO and identify efficiency in the interaction in the real world, for which an environment is created in the form of maze. With both algorithms, A* and Q-Learning, compares two parameters: runtime and number of iterations to complete learning.**

***Keywords*— Q-learning, Star, reinforcement learning, neuro-computational**

I. INTRODUCCIÓN

Los procesos cognitivos son procedimientos que se llevan a cabo para obtener un exitoso aprendizaje. Un aprendizaje en particular es el de memoria espacial que se refiere a la parte del sistema de memoria que reconoce el ambiente en que se encuentra el individuo o agente para así poder orientarse y encontrar un objetivo en particular.[1][2] Los algoritmos neuro computacionales tienen como finalidad demostrar el éxito del proceso cognitivo a través del planteamiento matemático de los mismos, para su posterior utilización en simuladores que arrojen resultados similares a lo que sucede en el mundo real y compararlos con la implementación en agentes que pueden ser robots para que actúen en situaciones en donde pongan a prueba las cualidades y debilidades de cada uno de ellos.

El presente artículo analiza el resultado de la comparación de dos algoritmos neuro-computacionales, en principio a partir de su estudio desde la teoría con el análisis de su formulación matemática, lo cual permite tener una base concreta para la programación de los mismos.

A partir de su estudio y análisis matemático es posible validar su implementación en el robot humanoide NAO[3]

Los algoritmos escogidos para la realización del presente trabajo son: algoritmo Q-Learning y algoritmo A* estrella, seleccionados por la amplia información y además por sus características similares en la parte de navegación para los robots.

Para este trabajo el robot tiene que evitar obstáculos dentro de un laberinto hasta llegar a su objetivo principal.

El ambiente creado para la implementación es un laberinto con tres columnas y cinco filas, lo que da un total de 15 bloques, llamados nodos o estados en el futuro, que para el caso del algoritmo Q-Learning representa quince estados; cada estado con un respectivo valor. Éste laberinto tiene también siete obstáculos representados físicamente por medio de paredes del tamaño de la altura del robot NAO (57,4cm) y 45cm de ancho. La base del laberinto es de lona para mejor adherencia del material de los pies del robot al suelo.

Para la programación de ambos algoritmos se utiliza el lenguaje Python, lenguaje que también es utilizado por CHOREGRAPHE, programa que permite la manipulación de las acciones para un robot NAO. [4][5]

Dentro de un laberinto el robot humanoide NAO utiliza sensores de proximidad para identificar si se encuentra cerca o no de un obstáculo.

II. FORMULACIÓN DEL PROBLEMA

Para la formulación del problema, se considera que los robots humanoides no tienen autonomía, por lo tanto, se necesita de factores externos para que el mismo pueda resolver algún problema en particular, por ejemplo, se necesita que el robot humanoide en este caso el robot NAO, tenga la capacidad de navegar por entornos sin asistencia humana, esto se lleva a cabo utilizando materiales y herramientas que permitan la manipulación del robot, para que por medio de la manipulación pueda explorar el ambiente y luego navegar a través del mismo, creando la capacidad de decidir el mejor camino. La navegación que realizan los robots humanoides en ambientes conocidos y desconocidos influye directamente en la memoria cognitiva del robot, y ya que el aprendizaje es una parte de los procesos cognitivos de la memoria, en el presente caso se plantea que la memoria cognitiva del robot debe ser autónoma para resolver cierto tipo de laberintos.

Se considera la resolución de un laberinto determinando el área a explorar realizando una búsqueda de las diferentes y posibles rutas determinando el mejor camino a recorrer. El presente trabajo se centra principalmente en el área de búsqueda informada y por refuerzo, éstos tienen por objetivo encontrar una solución al problema, conociendo el entorno con anterioridad o basado en el conocimiento del problema.

La solución se basa en la implementación de estos algoritmos de búsqueda en la memoria del robot humanoide NAO, algoritmos que son capaces de realizar el aprendizaje con diferentes procedimientos.

A. Algorítmo A estrella*

El algoritmo estrella (A*) es un algoritmo que busca el camino más corto desde un punto inicial, también llamado estado inicial, hasta un estado final o meta, a esto se le llama búsqueda inteligente.

Basados en el siguiente ejemplo se explicará el anterior concepto.

El primer paso del algoritmo estrella es obtener una área de búsqueda en la cual esta es dividida en una rejilla cuadrada que permite trabajar con una matriz bidimensional en el cual cada elemento de la matriz representa ya sea un obstáculo o una área libre transitable realizando los cálculos necesarios para encontrar la mejor ruta, moviéndose de una rejilla a otra (o en este caso se los llama nodos). El siguiente paso es determinar cuáles son los nodos transitables, una vez que se encuentren dichos nodos, generalmente se empieza a realizar los cálculos desde el punto A, buscando los nodos adyacentes, hasta encontrar el punto B[6]

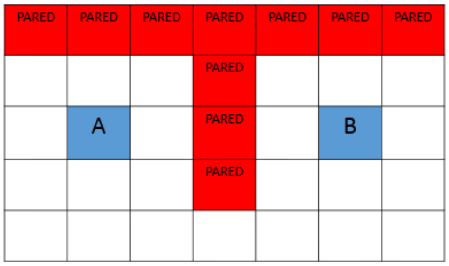


Figura 1. División del área de búsqueda en una cuadricula en 2D.

Se debe interpretar dos grupos en este algoritmo una lista abierta y una lista cerrada; una lista abierta albergará todos los posibles nodos a transitar y la lista cerrada los nodos que ya fueron transitados. En la lista abierta se encontrará incluido el nodo A (o el nodo padre) determinando así el punto de inicio, luego se lo transfiere a la lista cerrada ya que se conoce su posición y no necesita ser revisado de nuevo.

El siguiente paso es almacenar los nodos adyacentes al punto inicial, en este caso serán 8 nodos adyacentes. El algoritmo estrella hace uso de la información de la gráfica para calcular el costo del nodo de partida (G), que empieza en el nodo A hasta el siguiente nodo n y el costo estimado (H) que comienza en el nodo n hasta el nodo B. La suma de estas dos cantidades (F) se guarda en la función de costo final (F):

$$F = G + H$$

G = Es la ganancia que adquiere al realizar un movimiento, también se considera como el costo de movimiento realizado desde un nodo inicial hasta otro nodo n. H= Es la función heurística y representa el costo estimado del mejor camino para ir desde un cuadro de la rejilla hasta el destino final. La razón por la que es nombrado heurística, se debe a que es una estimación de la distancia que debe recorrer.

En este ejemplo se asigna un costo G=10 en movimientos horizontal y vertical, y un costo de G=14 para movimiento diagonales, utilizando el teorema de Pitágoras.[7]

La función heurística puede ser calculada de dos formas, si sólo se tienen movimientos en forma recta se utiliza el cálculo de distancias de Manhattan, de lo contrario, si el agente se está moviendo diagonalmente en el ambiente, se utilizará el teorema de Pitágoras.

Para éste trabajo, el robot Nao se moverá vertical u horizontalmente a través del ambiente, por lo tanto, se utilizará el cálculo de la distancia de Manhattan. [8]

$$d = \sum_{i=1}^{n} |x_i - y_i|$$

El nodo A necesita recorrer 3 espacios de forma horizontal al lado derecho hasta llegar al nodo B por lo cual se multiplica por 10 y se obtiene un valor de H=30 y los nodos de lado izquierdo, de arriba y abajo utilizan 5 espacios por lo que H=50. Siguiendo el mismo procedimiento se calcula los valores de (G) y (H) para los nodos restantes.

El siguiente paso es elegir el nodo con menor costo F en la lista abierta, y añadirlo a la lista cerrada, el resto de nodos son ignorados y los nodos de la lista abierta son de nuevo evaluados.

Este proceso termina cuando el nodo B es agregado a la lista cerrada. Para determinar la mejor ruta se trabaja desde el nodo final B hacia atrás hacia el nodo inicial A.

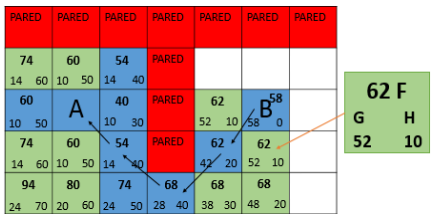


Figura 2. Mejor ruta encontrada desde el nodo A hasta el nodo B.

Con base en el análisis anterior se ha desarrollado el siguiente algoritmo en Pseudocódigo:

```
----------------------------------------------------
  Proceso Algoritmo_Estrella
  Definir nodo_inicial,nodo_final,nodo_n,nodo_padre
  Definir lista_abierta,lista_cerrada
          nodo_padre<-nodo_inicial
          lista_abierta<-nodo_inicial
          Repetir
          Buscar un conjunto de nodos_n
          Calcular F=G+H de cada nodo_n
          Nodo_inicial<-nodo con menor costo F
          Comparar nodo_inicial con los nodos_n
          Agregar nodo_inicial lista_cerrada
          Si nodo_n<costo Entonces
            Agregar nodo_n en lista_cerrada
                    nodo_inicial<-nodo_n
                     Sino
          Agregar nodo_n en lista_abierta'
                     FinSi
                     Si lista_abierta=0 Entonces
                                Escribir 'Error'
                     FinSi
          Hasta Que nodo_final=nodo_inicial
Escribir Ruta desde el nodo_inicial hasta el nodo_final
  FinProceso
----------------------------------------------------
```

FIGURA 3. Pseudocódigo algoritmo estrella

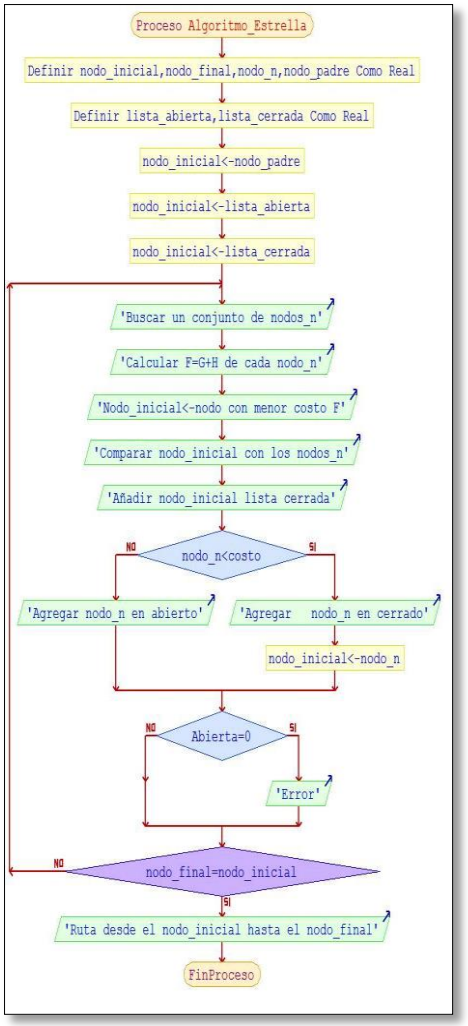


Figura 4. Diagrama de flujo del algoritmo estrella A*.

B. Algoritmo Q-Learning.

El algoritmo Q-learning es un algoritmo de aprendizaje por refuerzo, por lo tanto, su aprendizaje está basado en prueba y error [9]. El aprendizaje por refuerzo determina que un agente observa el "mundo" en el que se encuentra y a partir del estado realiza una acción y repite el mismo proceso hasta llegar a su objetivo (si existe). El aprendizaje por refuerzo recibe información del mundo exterior para luego elegir acciones óptimas, por lo tanto se puede decir que el aprendizaje por refuerzo es un aprendizaje supervisado, con la diferencia que en el aprendizaje por refuerzo, el agente en principio no sabe qué acción tendrá que tomar, este debe ir descubriendo en el proceso cuáles son las acciones que le generen mayores recompensas y menores castigos.

El aprendizaje por refuerzo permite conocer si en cierto estado el valor de la función Q es aceptable, además existe la probabilidad que en estados cercanos el valor de Q sea óptimo.

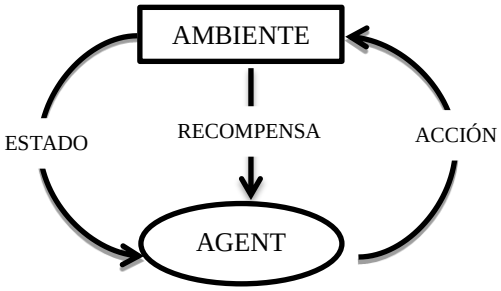


Figura 5. Modelo del aprendizaje por refuerzo.

El algoritmo Q-Learning permite realizar una acción en un estado determinado, el cual se llama Q-valor, éste valor inicialmente puede ser cero o algún otro valor aleatorio. La función Q relaciona los estados y las acciones y se va actualizando según la recompensa que obtenga al realizar una acción, esto debido a la política utilizada. La política representa la forma en la que actúa el agente, es decir: el agente puede tomar una acción al azar cuando se encuentre en un estado determinado, pero la probabilidad de que el agente haya tomado la decisión correcta dependerá del tipo de política que se esté usando, es decir en cada estado se tendrá un valor que exprese qué tan cerca está el agente de su objetivo.

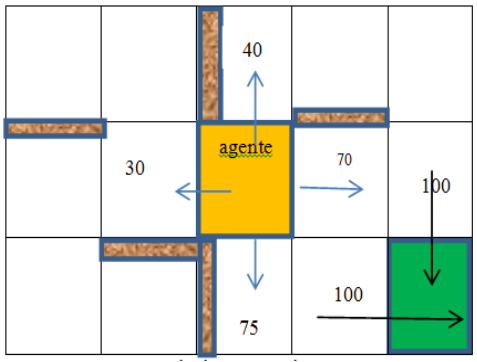


Figura 6. Toma de decisiones de un agente.

El agente dispone de cuatro opciones para realizar una acción y elegirá una al azar o elegirá la acción que tenga mayor valor, que es la que informa al agente que está más cerca de llegar a su objetivo (cuadro de color verde), que para objeto de este ejemplo obtiene una recompensa de 100 si alcanza el objetivo, para los demás casos la recompensa será -1.

Para el caso de resolución de un laberinto se necesita que la política que se use sea de exploración y uno de los métodos más utilizados es el de exploración ϵ-greedy, el cual se divide en: greedy con probabilidad ϵ que significa que el agente elige una acción que provee mayor cantidad de refuerzos y greedy con problabilidad 1-ϵ que significa que el agente escoge la mejor acción (el valor de ϵ debe ser entre 0 y 1). Es decir al utilizar política para exploración se tiene que:

- El valor de la constante ϵ es el valor que condiciona la acción que va a tomar el agente.
- Durante todo el proceso del algoritmo el agente realizará la misma toma de decisiones.

Al ejecutar la mejor acción el agente procede a observar el estado resultante (s') y su recompensa r, para proceder a la siguiente toma de decisión. La recompensa está dada por el estado en el que se encuentra el agente. Por ejemplo:

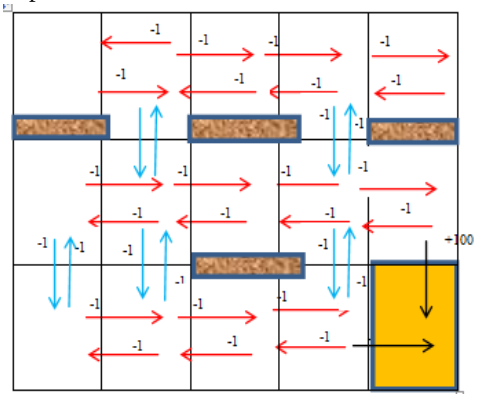


Figura 7. Ejemplo de posibles recompensas en un determinado laberinto al pasar de un estado a otro. (cuadro color amarillo: objetivo)

Al obtener la mejor acción en cada episodio, el agente vuele a realizar la misma acción hasta llegar al objetivo para lo cual es necesario actualizar la función Q.

La función de actualización del valor Q es entonces:

$$Q_{(s,a)} = Q_{(s,a)} + \alpha(r + \gamma máx Q_{(s',a')} - Q_{(s,a)})$$

Donde $Q_{(s,a)}$ representa el valor anterior Q en el estado anterior **s** y para la acción anterior **a**, $Q_{(s',a')}$ representa los valores actuales de Q para la acción actual **a'** en el estado actual **s'**, **α** es la tasa de aprendizaje ($\alpha>0$), Υ representa el factor de descuento ($0\leq\Upsilon\leq1$), **r** representa la recompensa para cada par (estado, acción), que puede tomar cualquier valor al inicializar la función Q.[10]

Con base en el análisis anterior se ha desarrollado el siguiente algoritmo en Pseudocódigo:[11]

INICIO
Crear Q,α, γ, ε, estado_final, s, a
Inicializar
$Q_{(s,a)} = 0$
α=0.1
γ=0.9
ε=0.1

//(con 0's o valores aleatorios), //Q(s,a) para todo s ∈ S y para //todo a ∈ A(s)

Para (s,a) hacer
$Q_{(s,a)}$ ⟵ Inicializar //$Q_{(s,a)}$ es la estimación inicial de Q
Fin Para

s observar estado
a seleccionar accion // Elegir a desde s usando una política derivada
//desde Q (p. ej. ∈-greedy)

Mientras s ≠ estado_final hacer
Ejecutar a
r ⟵ recompensa
s' ⟵ observar nuevo estado
a' ⟵ seleccionar siguiente acción
$Q_{(s,a)} = Q_{(s,a)} + \alpha(r + \gamma máx Q_{(s',a')} - Q_{(s,a)})$
s ⟵ s'
a ⟵ a'
Fin Mientras

Figura 8. Pseudocódigo algoritmo Q-Learning

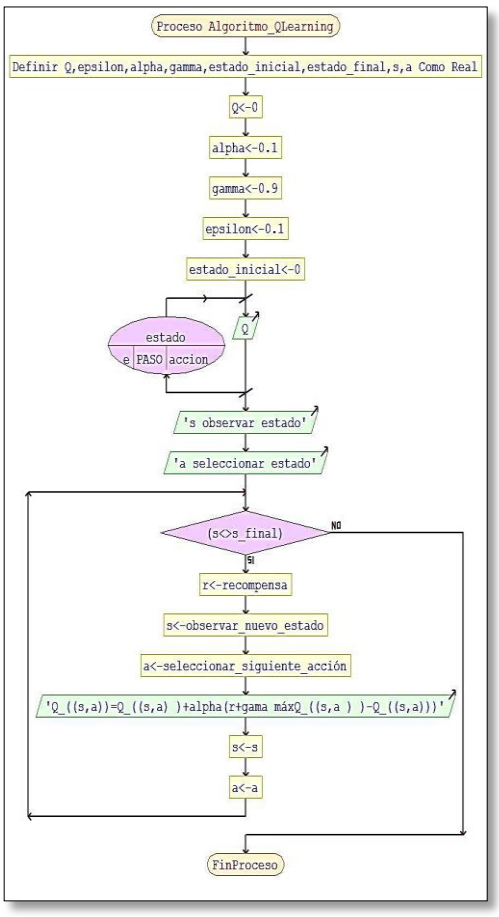


Figura 9. Diagrama de flujo del algoritmo Q-Learning.

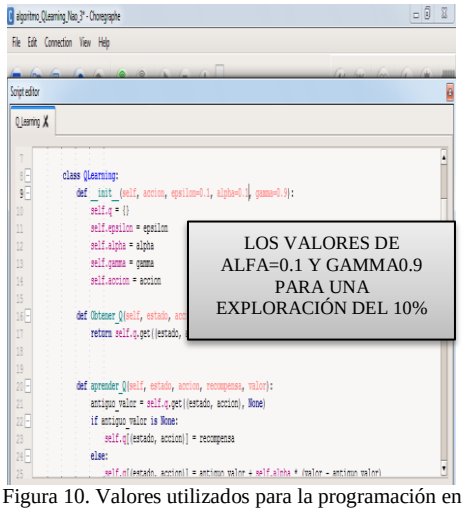


Figura 10. Valores utilizados para la programación en Choregraphe con lenguaje Python.

III. ANÁLISIS DE RESULTADOS

A. Análisis computacional del algoritmo A estrella*

Para el análisis del algoritmo A* estrella se utilizó la heurística con la fórmula de la distancia entre dos puntos o cálculo de distancia de Manhattan debido a que para el presente caso el robot no se moverá diagonalmente.

Con varios valores de ganancia (0.1, 0.5, 1.5, 10) se realizó las pruebas para el análisis en tiempo de ejecución y para el número de iteraciones utilizadas por el algoritmo A*, debido a que con dichos valores de demuestra la estabilidad del algoritmo en ambos casos.

TABLA I. PRUEBAS CON VALORES DE GANANCIA DIFERENTES PARA EL TIEMPO DE EJECUCIÓN

GANACIA	TIEMPO DE EJECUCION
0.1	0.1349
0.5	0.0920
1	0.0399
5	0.1259
10	0.0950

El tiempo de cálculo computacional en encontrar la ruta optima, más bajo se obtiene con una G=1.

TABLA II. PRUEBAS CON VALORES DE GANANCIA DIFERENTES PARA EL NÚMERO DE ITERACIONES

GANACIA	ITERACIONES
0.1	24
0.5	24
1	24
5	37
10	37

En la tabla II, los valores de ganancia G=0.1, G=0.5 Y G=1, son valores que reflejan un menor

número de iteraciones.

Por lo tanto en las pruebas computacionales el valor óptimo de ganancia es de G=1.

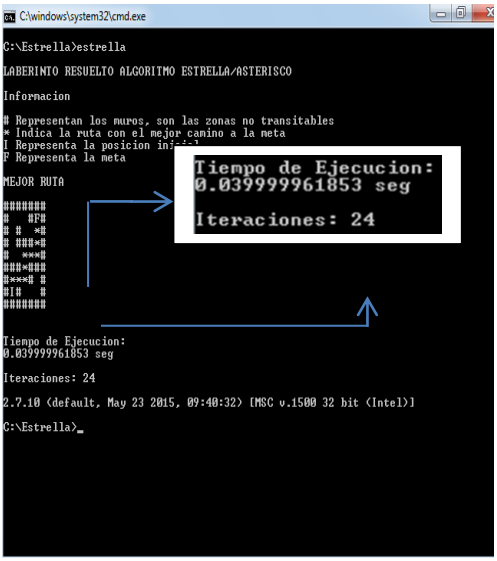


Figura 11. Ejecución del algoritmo A* estrella en el intérprete de Python. Resultados: tiempo de ejecución, número de iteraciones y mejor ruta obtenida.

B. Análisis computacional del algoritmo Q-Learning

Para el presente trabajo el algoritmo Q-Learning utiliza la técnica ϵ-greedy que permite la exploración con un valor de ϵ entre 0 y 1.

Se realiza una simulación con el intérprete de Python, y se realizan diez pruebas para el tiempo de ejecución, como primer factor de análisis.

TABLA III. TIEMPO DE EJECUCIÓN Q-LEARNING

# PRUEBA	Q learning
1	0.7850 seg
2	0.9619 seg
3	0.8499 seg
4	0.8840 seg
5	0.6949 seg
6	0.7210 seg
7	0.7369 seg
8	0.8010 seg
9	0.7120 seg
10	0.8229 seg

La tabla anterior muestra que el tiempo de ejecución es aleatorio para cada situación, es decir, el algoritmo Q-Learning no es predictivo, puede entregar varios resultados en cada implementación.

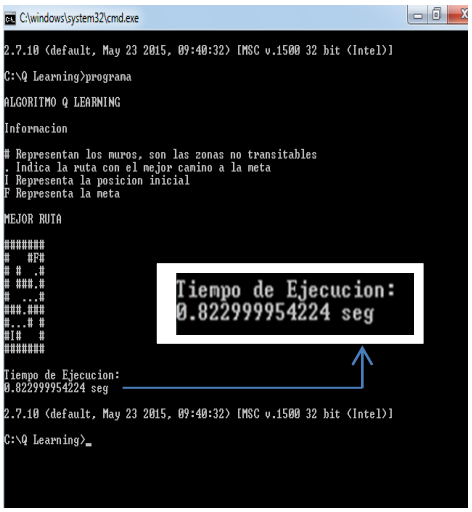


Figura 12. Ejecución del algoritmo Q-Learning en el intérprete de Python. Resultados: tiempo de ejecución y mejor ruta obtenida.

En el algoritmo Q-Learning se puede observar la efectividad del aprendizaje estableciendo los valores de alfa (α) y gamma (γ) que varían entre 0 y 1, primero con valores aproximados a cero y luego con valores cercanos a 1, para demostrar la variación en cada uno de los casos. Los experimentos dan los siguientes resultados:

ALFA:

TABLA IV. PRUEBAS CON VALORES DE TASA DE APRENDIZAJE PARA EL NÚMERO DE ITERACIONES

A) TASA DE APRENDIZAJE α=0.1, α=0.9

# PRUEBA	# ITERACIONES CON α=0.1	# ITERACIONES CON α=0.9
1	265	96
2	205	85
3	206	114
4	176	130
5	205	96

GAMA:

TABLA V. PRUEBAS CON VALORES DE FACTOR DE DESCUENTO PARA EL NÚMERO DE ITERACIONES CON ϒ=0.1, ϒ=0.9

# PRUEBA	# ITERACIONES CON ϒ=0.1	# ITERACIONES CON ϒ=0.9
1	162	119
2	205	131
3	206	171
4	176	158
5	114	90

Cuando el valor de alfa (α) y gamma (ϒ) se acercan a cero el aprendizaje se realiza en más iteraciones, que cuando el vaolor de alpha y gamma se acerca a uno.

Los valores propuestos para el análisis en la implementación con el robot fueron; tasa de aprendizaje (α=0.1) factor de descuento (γ=0.9) y el valor que permite la exploración (épsilon=0.1)

El valor de épsilon fue 0.1 porque representa el 10% de la exploración que realiza el agente en un determinado estado.

En el intérprete de Python también se visualiza el ambiente que se ha utilizado para el presente trabajo, a través de una característica de la librería pygame de Python, la cual permite dibujar el proceso realizado.

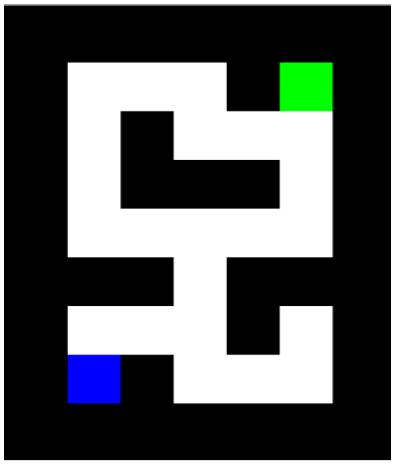
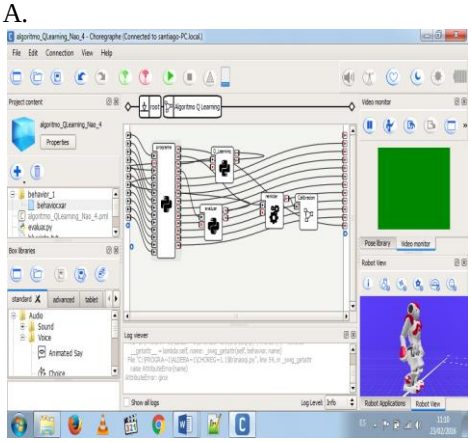


Figura 13. Ambiente creado para la simulación en Python y posterior implementación con el robot humanoide NAO.

C. ANÁLISIS COMPARATIVO DE LOS ALGORITMOS A* ESTRELLA Y Q-LEARNING.

En el análisis con la implementación de los algoritmos se tiene como principal objeto de estudio la programación en coregraphe, programa en el cual se realiza una programación secuencial para los algoritmos Q-Learning y A* estrella.

A.



B.

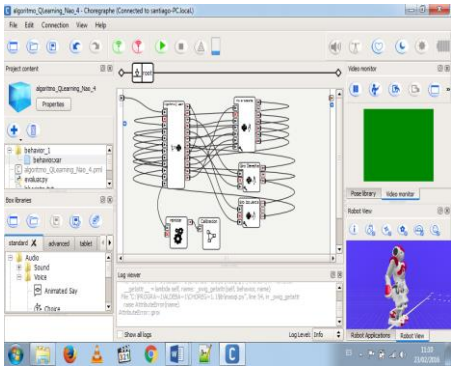


Figura 14. Programación del algoritmo Q-Learning en choregraphe. A) PRINCIPAL B) Q-LEARNING

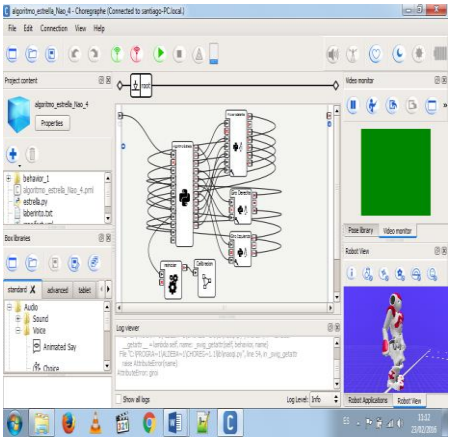


Figura 15. Programación del algoritmo A* Estrella

Para la comparación de los dos algoritmos se utilizaron dos parámetros, el primero se refiere al tiempo de ejecución de cada uno de los algoritmos y el segundo parámetro es el número de iteraciones en el que realiza el aprendizaje.

Se realizaron diez pruebas en cada uno de los casos para tener mayor posibilidad de efectividad en la comparación de los algoritmos.

TABLA VI. TABLA DE VALORES DE TIEMPO DE EJECUCIÓN CON LA IMPLEMENTACIÓN EN COREGRAPHE

N°	Tiempo de ejecución algoritmo Estrella	Tiempo de ejecución algoritmo Q learning
1	0.0520 seg	0.7850 seg
2	0.0610 seg	0.9619 seg
3	0.0709 seg	0.8499 seg
4	0.0500 seg	0.8840 seg
5	0.0529 seg	0.6949 seg
6	0.0590 seg	0.7210 seg
7	0.0840 seg	0.7369 seg
8	0.0620 seg	0.8010 seg
9	0.0529 seg	0.7120 seg
10	0.0829 seg	0.8259 seg
Promedio	**0.06866seg**	**0.79725seg**

La tabla anterior muestra que el algoritmo estrella es más rápido en su cálculo para determinar la mejor ruta en comparación con el algoritmo Q-Learning, esto es debido a que el algoritmo estrella utiliza menor cantidad de recursos computacionales, porque el algoritmo Q-Learning realiza un cálculo más extenso desde la política hasta la actualización del valor Q en cada estado.

TABLA VII. PRUEBAS PARA EL NÚMERO DE ITERACIONES EN LA IMPLEMENTACION CON EL ROBOT (COREGRAPHE)

N°	Iteraciones algoritmo Estrella	Iteraciones algoritmo Q learning
1	24	151
2	24	183
3	24	173
4	24	176
5	24	156
6	24	136
7	24	181
8	24	116
9	24	96
10	24	118

La tabla anterior muestra la comparación de los algoritmos basándose en el número de iteraciones que realizan para completar el proceso de aprendizaje. El algoritmo Q-Learning para alcanzar su política óptima debe realizar un proceso más largo, por lo que se demora más en determinar la ruta más apropiada, en comparación con el algoritmo estrella A*.

D. ANALISIS COMPARATIVO DE LOS ALGORITMOS A* ESTRELLA Y Q-LEARNING APLICADOS EN EL LABERINTO FÍSICO.

Para la comparación de los dos algoritmos se utilizaron dos parámetros, el primero el número de rutas completadas y el segundo parámetro es el tiempo que tarda en completar la ruta.
El laberinto utilizado tiene las siguientes dimensiones: 225cm de largo x 125cm de ancho y se dividió en rejillas de 45cm por 45cm.
Para los obstáculos se consideró la utilización de paredes de cartón paja por su fácil manipulación al momento de colocarlo en el ambiente. La altura de estos obstáculos es de 45cm de ancho y 60cm de alto, debido al tamaño del robot.
El robot NAO para realizar su movimiento utiliza los motores ubicados en sus piernas, con los que a través de la orden que se envía por el programa Choregraphe, calcula cuántos pasos debe dar para completar la distancia requerida por el programador.

En esta sección se debe tomar en cuenta las siguientes características:

1) El tipo de material del piso en este caso se utilizó vinilo por la razón que el robot Nao tiene el mismo material en las plantas de sus pies.
2) Si el suelo no está completamente nivelado el robot Nao se desviará de su ruta.
3) El tiempo máximo de uso continuo del robot NAO es de una hora debido al calentamiento de los motores. Para utilizarlo nuevamente se debe esperar treinta minutos. Éste sobrecalentamiento genera error de posición, errores en la exactitud del movimiento.

TABLA VIII. TABLA DE VALORES DEL NÚMERO DE RUTAS COMPLETADAS Y NO COMPLETADAS REALIZADAS EN EL AMBIENTE FÍSICO Y EL TIEMPO UTILIZADO PARA CADA RUTA.

Algoritmos	Numero de prueba	Ruta completada exitosamente	Ruta no completada	Tiempo
Estrella	1	X		58 seg
Estrella	2	X		58 seg
Estrella	3	X		57 seg
Estrella	4		X	16 seg
Estrella	5		X	36 seg
Q Learning	1	X		60 seg
Q Learning	2	X		56 seg
Q Learning	3	X		60 seg
Q Learning	4		X	57 seg
Q Learning	5		X	36 seg

La tabla anterior muestra que los algoritmos estrella y Q-Learning completan un número igual de rutas, y a partir de la cuarta prueba, el robot tiene problemas en completar las rutas, problemas causados por las características antes mencionadas.
Además porque luego de realizar cinco pruebas con el algoritmo estrella el robot se mantiene inactivo durante 30 minutos para enfriar sus motores y posteriormente se realizan las cinco pruebas con el algoritmo Q-Learning.

IV. CONCLUSIONES

El robot humanoide Nao obtuvo autonomía para moverse en un entorno conocido a través de los algoritmos de memoria cognitiva, estrella y Q-Learning implementados en su sistema hardware-software. Logrando así calcular el

camino más corto a diferencia de una programación convencional.

El algoritmo estrella alcanzó la respuesta deseada en 24 iteraciones a diferencia que el algoritmo Q-learning con 186 iteraciones. Estos datos probados en un laberinto de 15 rejillas, donde resulta el algoritmo de búsqueda informada como el más eficaz a ser implementado por su consumo de recursos computacionales.

El algoritmo estrella alcanzó la respuesta deseada en un promedio de 68.66 milisegundos a diferencia que el algoritmo Q-learning con un promedio de 797.25 milisegundos. Estos datos probados en un laberinto de 15 rejillas, donde resulta el algoritmo de búsqueda informada como el más eficiente a ser implementado por su rapidez computacional.

El hardware del robot no permitió alcanzar los resultados ideales en cuanto a la precisión en cada movimiento. A esto se suman los factores externos del ambiente que influyen directamente en el resultado ideal esperado. En un entorno de amplia dimensión, estos errores no afectarán en la navegación del robot, sin embargo, en ambientes estrechos podrían convertirse en un gran problema de colisión.

El presente trabajo permite conocer de una forma más concreta procesos cognitivos de la memoria en el aprendizaje de exploración con base en los algoritmos A* estrella y Q-Learning implementados en un robot humanoide NAO, en donde se destaca el algoritmo A* estrella como el más rápido y con menor número de iteraciones, cabe recalcar que se ha realizado pruebas en un laberinto pequeño, en trabajos futuros se podrían implementar entornos complejos con muchas rejillas y los resultados podrían cambiar.

V. REFERENCIAS

[1] T. Madl, K. Chen, D. Montaldi, and R. Trappl, "Computational cognitive models of spatial memory in navigation space : A review," *Neural Networks*, vol. 65, pp. 18–43, 2015.

[2] Á. Iglesias Sánchez, "Modelo computacional cognitivo de toma de decisiones basado en el conocimiento: aplicación en la inferencia de explicaciones," 2013.

[3] J. M. Cañas Plaza, "Programación del robot humanoide Nao para la RoboCup Contenidos," 2009.

[4] Guido Van Rossum, "El tutorial de Python," *Fred L. Drake, Jr.*, 2009.

[5] S. Krivanec, A. Petráckova, T. T. Phuong Linh, and D. Prusa, "Nao Robot Applications Developed In Choregraphe Environment," *m p Cent. Mach. Percept.*, vol. 21, p. 21, 2010.

[6] Instituto Tecnológico de nuevo Laredo, "Algortimo A*," *Intel. Artif.*, pp. 1–12.

[7] "Planeación de rutas."

[8] A. Grané, "Distancias Estadísticas y Escalado Multidimensional (Análisis de coordenadas Principales)." p. 38.

[9] F. Fernández Rebollo and D. B. Millán, "Aprendizaje por Refuerzo," *PLG, Departamento de Informática*, 2009.

[10] A. Plasencia Salgueiro, "SIMULACIÓN DE LA NAVEGACIÓN DE LOS ROBOTS MÓVILES MEDIANTE ALGORITMOS DE APRENDIZAJE POR REFUERZO PARA FINES DOCENTES," *Instituto de Cibernética, Matemática y Física (ICIMAF), Agencia de Energía Nuclear (AENTA)*, p. 7, 2013.

[11] A. M. Printista, M. L. Errecalde, and C. I. Montoya, "Una implementación paralela del algoritmo de Q-Learning basada en un esquema de comunicación con caché," *UNSL*, vol. 338403, p. 12.