Applying fuzzy logic and genetic algorithm in robot navigation

GONZALO LUZARDO M.

gluzardo@espol.edu.ec

gonchalox@gmail.com

INDICE

1. Introducción

El objetivo de este trabajo consiste en desarrollar el control de movimiento para que un robot alcance un objetivo en un entorno. Las variables de entrada y salida del robot utilizarán lógica borrosa (fuzzy logic) y serán obtenidas mediante un algoritmo evolutivo.

En la primera parte del trabajo se explicará el diseño del controlador borroso que utilizaremos, así como la configuración de los individuos a partir de los conjuntos borrosos obtenidos y el diseño de los algoritmos genéticos que serán aplicados sobre los individuos para realizar la evolución.

Por último se presentarán las tablas con los resultados obtenidos utilizando diferentes combinaciones: número de generaciones, tamaño de población y función de adaptabilidad (fitness), que iremos utilizando, se contrastarán los resultados y se presentaran las conclusiones del trabajo.

2. Control Borroso

Para realizar el comportamiento del robot utilizaremos un controlador borroso cuya entrada estará en función de la orientación relativa a nuestro objetivo (fuente de luz) con respecto de los sensores del robot.

Se usarán dos variables de entrada:

1. **El error en la orientación**, el cual nos ayudará establecer si la fuente de luz está frente al robot o a uno de sus lados y;
2. **La derivada de error**, la cual nos indicará si el robot está acercándose, alejándose o manteniendo su orientación con respecto a la luz.

Los sensores de luz que posee el robot miden la proximidad a una fuente de luz que en nuestro caso es el objetivo que debe alcanzar el robot, mientras más cercana sea la fuente de luz al sensor, el valor obtenido será mayor.

Conociendo los valores registrados por los sensores de luz derecho (`right_sensor_val`) e izquierdo (`left_sensor_val`), podemos calcular el error de orientación (`error`); y a su vez utilizar el error medido en la posición anterior (`dev_error`) junto con el tiempo transcurrido entre las dos mediciones (`t`) para calcular la derivada del error (`dev_error`):

$$error = lef\_sensor\_val - right\_sensor\_val$$

$$dev\_error = \frac{error - prev\_error}{t}$$

La figura 2.1 muestra un robot y el objetivo luminoso, ambos ubicados en un mundo. El valor que registra el sensor derecho, por estar el robot ubicado hacia la izquierda del objetivo, será mayor al registrado por el sensor izquierdo.

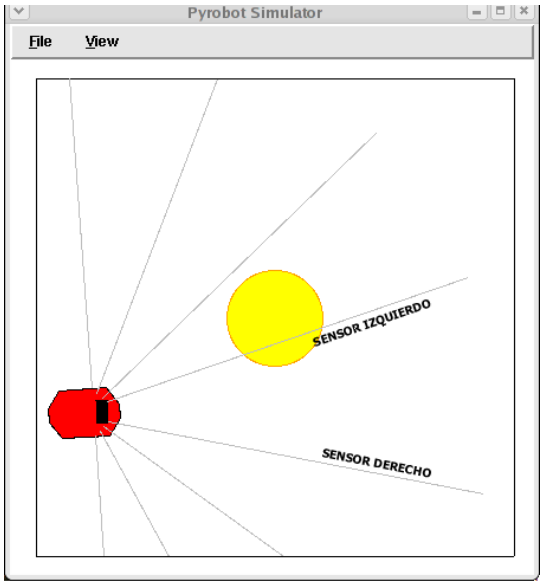


Figura 2.1. Robot utilizando sus sensores para localizar un objetivo luminoso

Considerando esto último, vamos a definir: los las funciones de pertenencia para el **error** que permitan expresar si la fuente está muy a la derecha, derecha, centrada, a la derecha y muy a la derecha; los conjuntos borrosos para la **derivada del error** que permiten expresar la velocidad con que estamos haciendo el giro pudiendo ser mucho hacia la derecha, hacia la derecha, constante, izquierda y mucho hacia la izquierda.

Podemos notar que si el objetivo está frente al robot el error calculado será de cero (**Z**); si el objetivo se encuentra hacia la derecha del robot el *error* calculado será negativo (**N**) el cual se hará aún más negativo (**BN**) si el robot se ubica más hacia la derecha del objetivo, y si el objetivo se encuentra hacia la izquierda el error será positivo (**P**) el cual aumentará (**BP**) a medida que nos ubiquemos más hacia la derecha del objetivo.

En cuanto a la *derivada del error*, si el robot no ha realizado ningún giro la derivada del error calculada será igual a cero (**Z**), y por otro lado si está haciendo un giro hacia la derecha la derivada del error calculada tendrá un valor negativo (**N**) la cual será aún más negativa (**BN**) si el robot hace dicho giro a una mayor velocidad; por el contrario, si el robot está realizando un giro hacia la izquierda la derivada del error calculada tendrá un valor de positivo (**P**), el cual será aun mayor (**BP**) si el giro lo realiza con una mayor velocidad.

Se tendrá una sola salida que se corresponderá con la *velocidad de giro del robot*, manteniendo constante la velocidad de avance. Para la salida utilizaremos funciones de pertenencia que permitan expresar acciones del robot del tipo girar mucho a la derecha (**BR**), girar a la derecha (**R**), ir de frente (**NC**), girar a la izquierda (**L**) y girar mucho a la izquierda (**BL**).

Utilizaremos funciones de pertenencia simples (trapezoidales) en los conjuntos borrosos que utilizaremos para el error, la derivada del error y la salida. En la figura 2.2 podemos observar como lucirán estas funciones; Inicialmente colocaremos valores arbitrarios a cada una de ellas, que posteriormente iremos modificando con el objetivo de mejorar el comportamiento del robot de ir hacia el objetivo.

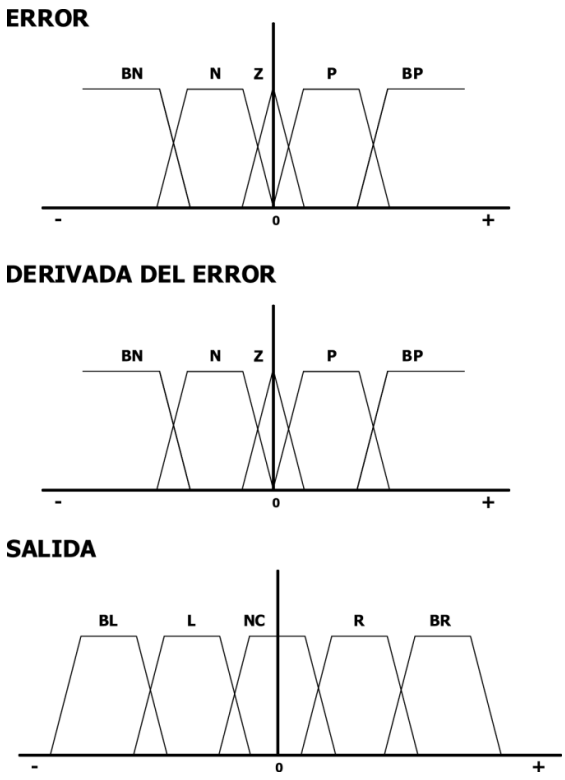


Figura 2.2. Funciones utilizadas en cada uno de los conjuntos borrosos

El control borroso va a utilizar un conjunto de reglas definidas en una tabla FAM (Fuzzy Asociative Memory) la cual será utilizada para calcular la contribución de cada conjunto borroso sobre la salida. Las reglas se construirán sobre todas las combinaciones posibles entre el error y la derivada del error. La tabla 2.1 muestra los contenidos de esta tabla, en dicha tabla se utilizan las iníciales correspondientes para cada una de las funciones de pertenencia de los conjuntos borrosos.

		ERROR				
		BN	**N**	**Z**	**P**	**BP**
DERIVADA DEL ERROR	**BN**	NC	R	BL	L	BL
	N	R	R	L	NC	L
	Z	BR	R	NC	L	BL
	P	R	R	R	L	BL
	BP	BR	BR	BR	NC	NC

Tabla 2.1. Fuzzy asociative memory

Para “desborrosificar” la salida utilizaremos el centro de masa sobre la contribución que realiza las entradas en el conjunto borroso de salida. Dicho centro de masa lo obtendremos promediando el área bajo la curva en cada una de las funciones de pertenencia considerando la contribución de las entradas sobre cada una de estas funciones. La figura 2.3 nos muestra el centro de masa obtenido a partir de un conjunto borroso de salida ejemplo.

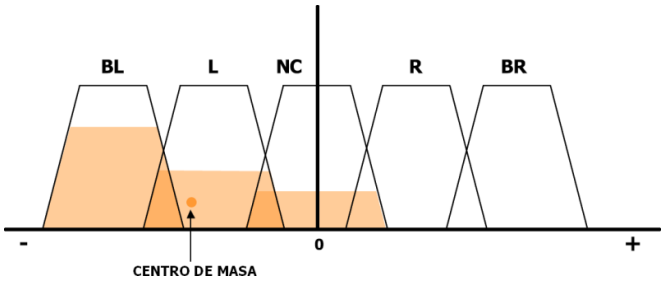


Figura 2.3. Desborrosificación del conjunto borroso de salida

3. Evolución del comportamiento con computación evolutiva

Se utilizará programación genética como herramienta para la búsqueda de los valores de las funciones en los conjuntos borrosos que optimicen el control de movimiento del robot, esto es, buscar la combinación en las funciones de pertenencia en cada uno de los conjuntos borrosos que haga que un robot llegue lo más pronto a su objetivo.

La idea es definir una población de individuos, cada uno de los cuales contendrá la codificación del controlador borroso encargado de controlar al robot por el entorno. Los elementos que vamos a evolucionar serán los parámetros de las funciones de pertenencia de los conjuntos borrosos, en nuestro caso particular los puntos de corte en los que la función cambia de valor.

Con el objetivo de minimizar el tamaño de los individuos, vamos a sacar provecho de la simetría sobre el *eje y* de las funciones de pertenencia de los conjuntos borrosos. De esta forma un individuo tendrá en su genotipo los puntos de corte que corresponden al *eje x* positivo de las funciones de pertenencia de cada uno de los conjuntos borrosos del error, derivada del error y la salida, en ese orden respectivo; los valores restantes podrán ser obtenidos a partir de estos de manera sencilla. La figura 3.1 ilustra la codificación a ser utilizada.

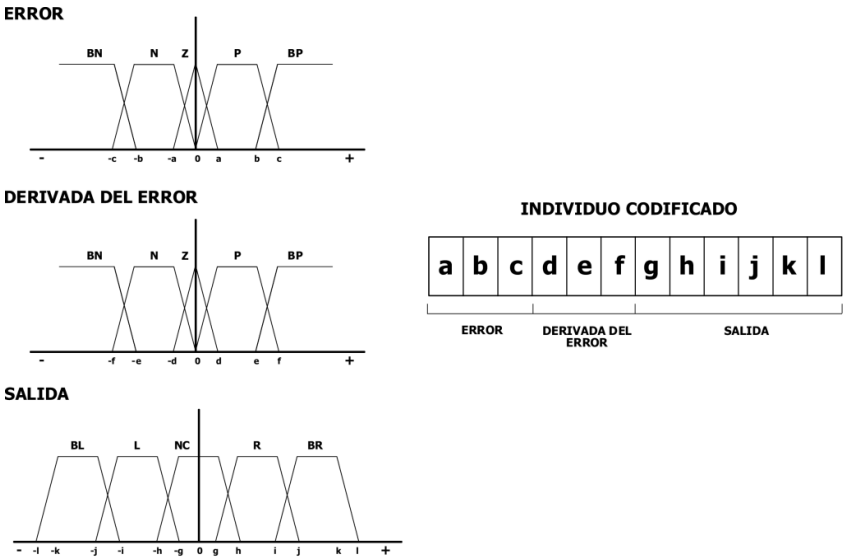


Figura 3.1. Codificación del controlador borroso en un individuo de la población

Consideraremos un ***individuo válido*** a aquel individuo que genere un controlador borroso válido, en nuestro caso para que un individuo sea válido se debe cumplir la siguiente condición:

$$c > b > a \quad \text{AND} \quad \text{f} > e > \text{d} \quad \text{AND} \quad \text{l} > k > j > i > h > g$$

El esquema del algoritmo evolutivo que utilizaremos será el siguiente:

1. Se creará una **población inicial** de individuos válidos de forma aleatoria. Cada uno de los individuos en esta población inicial se inicializará con valores aleatorios obtenidos con una distribución uniforme.
2. Cada uno de los individuos de la población será **evaluado** mediante una **función de adaptabilidad**. La evaluación se hará mediante la simulación del robot usando los parámetros del controlador difuso codificados en el individuo.
3. Se elegirán aleatoriamente individuos de la población para generar una nueva. Aquellos individuos **mejor adaptados** tendrán una probabilidad más alta de ser seleccionados.
4. Se **mutará** cada uno de los individuos seleccionados para generar un nuevo individuo, sumando una cantidad aleatoria obtenida a través de una distribución normal con media cero y una varianza muy baja a cada uno de sus genes; controlando siempre que el nuevo individuo generado sea válido.
5. Repetir (ir a 2) hasta un número de **pasos máximo,** hasta que todos los **individuos de la población sean similares** o hasta que la calidad del mejor individuo o de la población en su conjunto supere un determinado **umbral** prefijado con anterioridad.

Los parámetros que vamos a estudiar serán los siguientes:

1. Número de generaciones utilizadas.
2. Número de individuos en la población.
3. Diferentes valores para la varianza en la función de distribución normal que utilizaremos en el proceso de mutación de los genes de un individuo.
4. Diferentes formas de evaluar la adaptabilidad del individuo:
    a. Colocar al robot en una posición específica y contar los pasos que tomó para llegar al objetivo, en este caso el valor de la adaptabilidad del individuo es inversamente proporcional al número de pasos que necesitó el robot para llegar al objetivo.
    b. Colocar al robot en una posición aleatoria y contar los pasos que tomó para llegar al objetivo, en este caso el valor de la adaptabilidad del individuo es inversamente proporcional al número de pasos que necesitó el robot para llegar al objetivo y directamente proporcional a la distancia del punto de partida con respecto al objetivo.
    c. Utilizar una escala logarítmica para tratar de aumentar la probabilidad de que un individuo mejor adaptado sea seleccionado para formar parte de la nueva población.
5. Elitismo.

4. Experimentos y resultados obtenidos

La implementación del control borroso y los algoritmos genéticos fue realizada en **Python**. La simulación del robot fue realizada en **Pyrobot**. Adjunto a este trabajo están el código fuente de la aplicación utilizada para realizar los experimentos.

En primer lugar cada experimento vamos a fijar los parámetros objeto de nuestro estudio y analizar los resultados obtenidos. La aplicación nos retorna un conjunto de ficheros que contienen resultados de la evolución realizada sobre in conjunto de individuos de una población inicial.

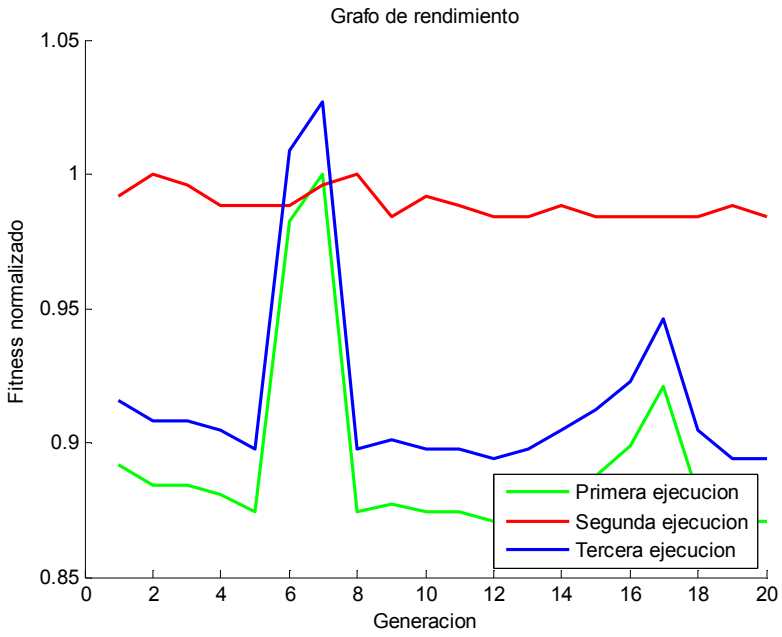
La aplicación recibe como parámetros de entrada lo siguiente:

- **`max_gens`**, número máximo de generaciones.
- **`max_size_gen`**, tamaño de cada generación, esto es el número de individuos que contendrá cada generación.
- **`max_steps`**, número máximo de pasos que podrá tomar un robot para llegar a la meta, si un robot toma más de dicha cantidad de pasos se lo considerara como el peor adaptado, esto es, se le asignará el peor factor de adaptación posible.
- **`invalid_steps`**, número mínimo de pasos que debe tomar un robot para llegar a la meta. Aunque resulte un poco extraño la utilización de este factor, nos será de mucha utilidad cuando realicemos experimentemos en donde vayamos a posicionar el robot de manera aleatoria, de esta forma si colocamos el robot muy cerca o incluso sobre la meta no se lo considere el mejor adaptado aunque haya tomado una cantidad muy pequeña de pasos para llegar a dicha meta.
- **`mut_fact`**, factor de mutación, este parámetro establece que tanto va a mutar un gen de un individuo. Cada gen muta de acuerdo a una función gausiana con media igual a cero, y una varianza igual al factor de mutación.
- **`use_elite`**, una bandera que especifica si se desea realizar la evolución utilizando elitismo, esto es tomar el individuo mejor adaptado de la población y enviarlo íntegramente (sin realizar ningún tipo de mutación) a la nueva población que va a ser generada.
- **`fname`**, prefijo que tendrán los ficheros con los resultados de la ejecución de la aplicación.

El **primer experimento** fue hecho con poblaciones relativamente pequeñas de 20 individuos, y con un número de generaciones de 50, sin utilizar elitismo y un factor de mutación relativamente bajo (0.1). En la evaluación de cada individuo se ubico el robot siempre con la misma orientación y en la misma posición lo más alejado posible de la meta, de tal forma que cada individuo sea evaluado siempre en las mismas condiciones que los otros. La adaptabilidad (fitness) de cada individuo se midió contando el número de pasos (steps) que el individuo tomaba en llegar a la meta, los mejores adaptados tenían un número menor de pasos, de tal forma que la adaptabilidad (f) fue calculada de la siguiente forma:

$$f = \frac{1}{steps}$$

Se realizaron tres ejecuciones[1], todas ellas con los mismos parámetros, en la grafica 4.1 podemos observar un resumen de los resultados obtenidos. El **eje x** corresponde a la generación, y el **eje y** corresponde al valor de fitness normalizado (se divide para el fitness alcanzado por el mejor individuo en todas las generaciones) del mejor individuo de dicha población. Se utiliza el fitness normalizado con el objetivo de visualizar mejor los resultados, de tal forma que podemos comparar el mejor individuo de una población con respecto al mejor individuo de entre todas las poblaciones.



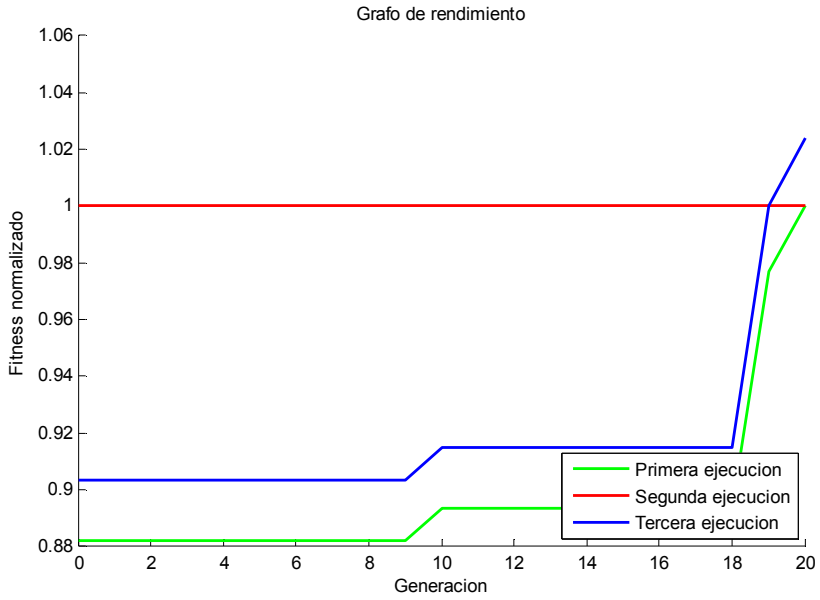
Grafica 4.1. Resultados del primer experimento

Como podemos observar en la grafica 4.1 los resultados no son nada prometedores, es una búsqueda totalmente aleatoria. En ciertas poblaciones hemos tenido individuos muy buenos que a pesar de que hayan sido seleccionados para mutar y formar parte de una nueva generación, esta nueva no haya mejorado con respecto a la anterior. Para mejorar estos resultados utilizaremos elitismo, esto es, tomaremos al mejor individuo de una población para colocarlo íntegramente en la nueva población generada a partir de la anterior.

En este nuevo experimento hemos asignaremos el mismo valor colocado a los parámetros del experimento anterior con la diferencia de la utilización del elitismo. En la grafica 4.2 podemos observar un resumen de los resultados obtenidos.

Analizando los resultados de la gráfica 4.2, en primer lugar notamos como el elitismo mantiene la calidad de la población. En el caso de la segunda ejecución notamos como el mejor individuo de la primera generación fue el mejor de todas las generaciones, es muy seguro que en este caso hayamos caído en un óptimo local. Para mejorar esto aumentaremos el factor de mutación a 0.3 para ampliar el espacio de búsqueda.

[1] Por motivos de tiempo se realizaron solo tres ejecuciones. Aunque no resulte ser estadísticamente válido, podemos sacar algunas conclusiones a partir de esto.



Grafica 4.2. Resultados del segundo experimento

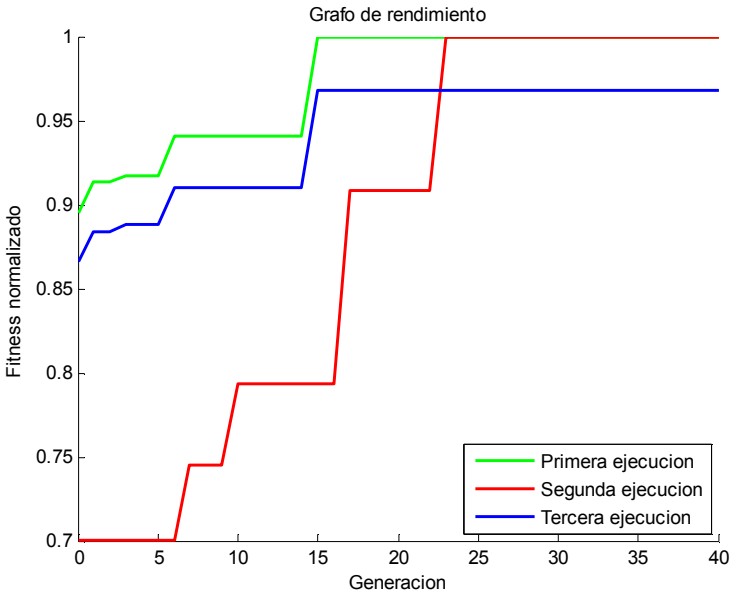
Por otro lado, en primera instancia pudiéramos tratar mejorar la velocidad de convergencia. Si analizamos a profundidad lo resultados, considerando el numero de pasos que le toma a un individuo en llegar a la meta; un individuo mejor adaptado puede tomar 240 pasos en llegar a la meta, mientras que el peor adaptado puede tomar 250 pasos, tan solo 10 pasos de diferencia; esto al final se traducirá en que ambos individuos tengan casi la misma probabilidad de ser seleccionados para formar parte de la nueva generación.

Para mejorar los resultados podemos utilizar una función de evaluación que reaccione mejor a estos pequeños cambios, de tal forma que un individuo con un número un poco menor de pasos tenga una mayor probabilidad (aunque no mucha ya que podríamos caer en un optimo local) a ser seleccionado que uno con un número un poco mayor. Para lograr esto utilizaremos una función de evaluación de adaptabilidad logarítmica.

Adicionalmente a esto aumentaremos el número de generaciones a 40 de tal forma que observemos mejor la evolución de las poblaciones utilizando estas mejoras. En la grafica 4.3 podemos observar en los resultados obtenidos al aplicar las mejoras descritas anteriormente.

Analizando los resultados de la gráfica 4.3 podemos observar que la convergencia resulta ser un poco más rápida.

En este punto vamos a probar los resultados obtenidos seleccionando al mejor individuo de la población final, e introduciendo sus parámetros en un robot simulado para analizar su comportamiento. La figura 4.4 muestra el camino recorrido por el robot para llegar a la meta.



Gráfica 4.3. Resultados del tercer experimento

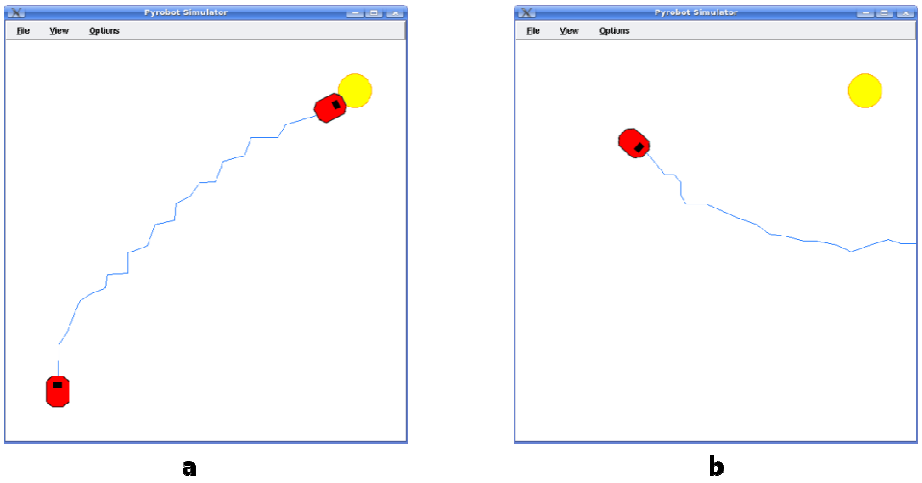


Figura 4.4. Camino recorrido por un robot para llegar a la meta, introduciendo los resultados obtenidos en el tercer experimento, (a) ubicado con la orientación y posición con la cual hemos realizado todos los experimentos y (b) en una posición y con una orientación diferente a esta

Como podemos observar en la figura 4.4, el robot llega de manera correcta a la meta partiendo de la posición fija desde donde hemos realizado todas nuestras pruebas. En cambio cuando el robot parte de una posición y con una orientación diferente le resulta complicado llegar a la meta, en este caso nunca lo logra. Esto quiere decir que el robot esta optimizado de tal forma que pueda llegar a la

meta de manera más rápida siempre y cuando parta desde la misma posición y con la misma orientación a la cual hemos hecho todos nuestros experimentos.

Para mejorar esto, en cada evaluación de la adaptabilidad del individuo, el robot iniciara en una posición y orientación completamente aleatorias, de tal forma que podamos tener una solución en la cual el robot este mejor adaptado para llegar a la meta desde cualquier posición en el entorno y con cualquier orientación inicial.

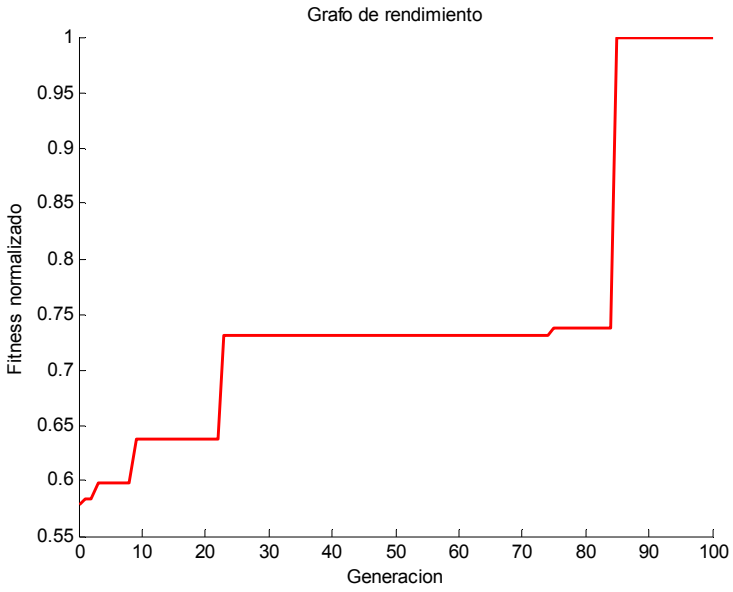
Al hacer esta consideración debemos realizar un pequeño cambio en nuestra función de evaluación ya que debe considerar la posición inicial en la cual se ha colocado el robot antes de iniciar la simulación.

Definimos **d**, como la distancia a la meta del robot al inicio de la simulación. La función de evaluación de adaptabilidad (fitness) lucirá de la siguiente forma:

$$f = \log\left(1 + \frac{d}{steps}\right)$$

Sumamos 1 al resultado de la operación `d/steps`, para no tener valores negativos de fitness. Como vemos el fitness es directamente proporcional a la distancia e inversamente proporcional al número de pasos que toma para llegar a la meta. Adicionalmente a esto penalizaremos a los individuos que no lleguen a la meta en un número máximo de pasos (500).

El último experimento realizado considera todas estas mejoras. La figura 4.5 nos muestra los resultados obtenidos.



Gráfica 4.5. Resultados del cuarto experimento

Como podemos observar en la grafica 4.5, la convergencia resulta ser lenta, esto es normal debido a que estamos buscando una solución en un espacio de búsqueda mucho más amplio el cual esta considerando la posición y orientación inicial del robot.

Por último evaluaremos el individuo en el simulador, colocando al robot en tres posiciones y orientaciones diferentes y observaremos el camino que sigue hasta llegar a la meta. La figura 4.5 nos muestra los resultados obtenidos.

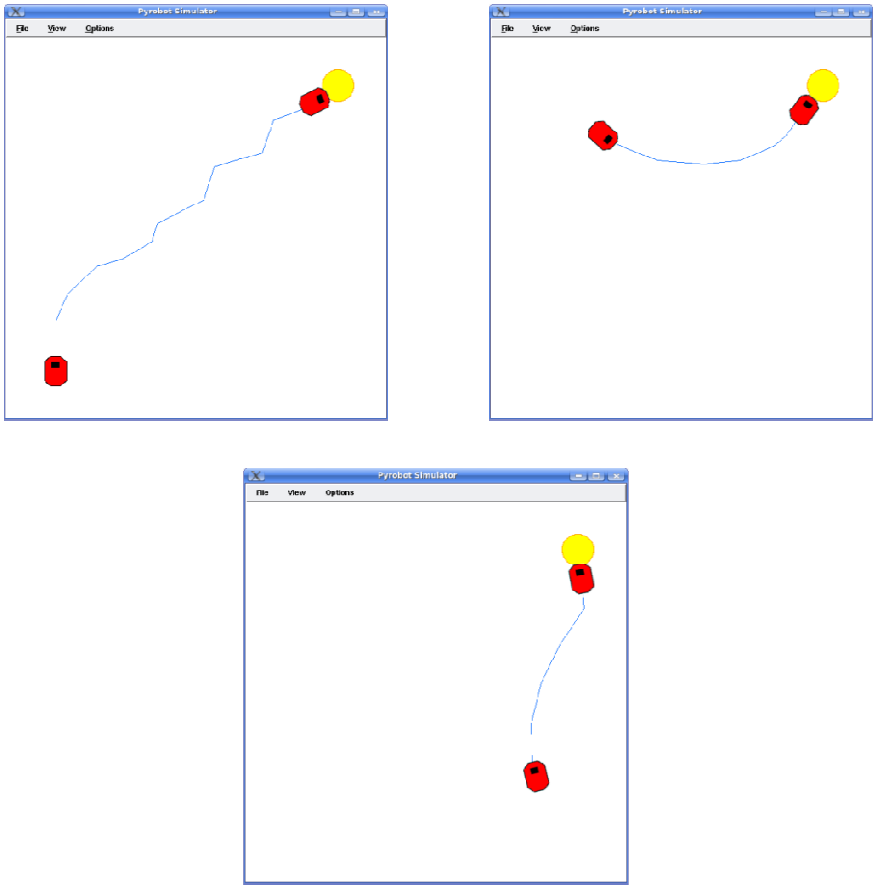


Figura 4.5. Camino recorrido por un robot para llegar a la meta, usando los parámetros obtenidos como resultado del cuarto experimento, ubicado en tres posiciones y orientaciones iníciales diferentes.

Como podemos apreciar en la figura 4.5, el robot está mejor adaptado para llegar a la meta desde cualquier punto del entorno y con cualquier orientación inicial.

5. Trabajos futuros

Un posible trabajo futuro podría ser el evolucionar la tabla FAM tomando como base los conjuntos borrosos de entrada y salida que se obtuvieron como resultado en el presente. En este caso se deberá considerar una codificación que permita almacenar la tabla completa en la estructura de un individuo; y a diferencia de lo que hemos hecho hasta momento se debe permitir realizar cruce y mutación, el primero con una probabilidad muy alta con respecto al segundo.

Otro posible trabajo podría ser el introducir los resultados obtenidos en un robot físico, lo cual permita analizar su comportamiento en un ambiente real de funcionamiento.

6. Conclusiones

Las conclusiones que podemos anotar al haber realizado el presente trabajo son las siguientes:

- El control borroso nos permite suavizar de manera considerable los movimientos del robot.
- Pudimos llegar a una solución, aunque no la mejor, la cual nos permite tener un robot que llega a un objetivo en el menor tiempo posible.
- Los algoritmos genéticos, aunque suelen ser lentos, son un mecanismo muy potente para encontrar una solución a un problema de optimización.
- Lo más importante en los algoritmos genéticos, y lo que me permite llegar a una buena solución en el menor tiempo posible es un buena función de adaptabilidad que permita reconocer de manera adecuada cuales son los mejores individuos dentro de una población.
- Los algoritmos genéticos y la lógica borrosa son mecanismos que al ser utilizados de manera combinada son un mecanismo potente para poder encontrar comportamientos que optimicen los movimientos de un robot no solo para llegar a un objetivo sino para realizar tareas más complejas.

7. Referencias

- **Hammond Vashisth and Peng-Yung Woo**, APPLICATION OF FUZZY LOGIC TO ROBOTIC CONTROL, Department of Electrical Engineering, Northern Illinois University, DeKalb, IL 60115.
- **Blank, Kumar, Meeden, Yanko**, The Pyro toolkit for AI and robotics, Bryn Mawr College, Swarthmore College, Univ. of Mass. Lowell.
- **Lee Edward**, Applying fuzzy logic to robot, L & P Culture and Technology Inc., San Diego, California, USA