ISSN 1870-4069

Generación de control de dirección para vehículo autónomo por medio de programación genética

Edgar Miguel Aguilar Diaz, Katya Rodríguez Vázquez

Universidad Nacional Autónoma de México, Instituto de Investigaciones en Matemáticas Aplicadas y en Sistemas, México
xeicker@gmail.com, katya.rodriguez@iimas.unam.mx

Resumen. En este artículo se presenta el desarrollo de un controlador para vehículo autónomo con dirección Ackermann [11] evolucionado por medio de Programación Genética. La dirección Ackermann es la utilizada por un coche común de cuatro ruedas. En este caso, la Programación Genética hace uso de una función de evaluación que describe la geometría característica con la que se mueve el vehículo, sin considerar los efectos dinámicos, sino únicamente los cinemáticos. Se resuelve el problema mediante su división en dos partes. Primero, llevar al auto a una recta cualquiera, esta parte se soluciona independientemente y crea el controlador para implementarlo en la simulación de la siguiente parte. Y segundo, se generaliza para una trayectoria representada por un conjunto de rectas, en forma de una recta poligonal. Para hacer esta generalización se utilizan coordenadas relativas, lo que implica transformación de coordenadas. Esto con la finalidad de utilizar la misma referencia que en la primera parte, sin importar la recta en que se encuentre el auto. Los resultados obtenidos presentan comportamientos adecuados que cumplen con la tarea definida como objetivo.

Palabras clave: programación genética, dirección Ackermann, control.

Generation of the Steering Control for an Autonomous Vehicle using Genetic Programming

Abstract. This article presents the development of the steering control for an autonomous vehicle with Ackermann steering [11] evolved using Genetic Programming. Ackermann steering is the one used in most four-wheeled cars. In this case, Genetic Programming uses an evaluation function which describes the specific geometry the vehicle moves with, which means that it doesn't take into consideration dynamic effects, only kinematics. The problem is solved by first dividing it into two different parts. First, to take the car toward any straight line, this part is treated entirely independent from the rest and generates a controller to be implemented in the simulation of the next part. And second, it is generalized to any trajectory made of straight lines, in the form of a polyline. This generalization is achieved using relative coordinates, which implies that a transformation is needed, with the aim of always using the same reference as in the previous part, without the need of knowing the line where the car lies. The

results obtained show an adequate behavior which meets the assigned task defined as its objective.

Keywords: Genetic Programming, Ackermann Steering, Control

1. Introducción

La Programación Genética (PG) propuesta por John Koza [3] es una técnica evolutiva basada en poblaciones, la cual, a diferencia de un Algoritmo Genético (AG) [1,2] no trabaja sobre una estructura de datos de longitud fija, trabaja sobre una estructura de longitud variable y tiene como objetivo evolucionar funciones, modelos o programas, más que encontrar un conjunto de parámetros.

Hay diferentes antecedentes de desarrollo de controladores [3,7,8,10], se presentan diferentes propuestas del uso de la PG para evolucionar estructuras de controladores. En el caso de un vehículo autónomo, el desarrollo de un controlador de dirección es complejo en el sentido de que es un sistema no lineal, y las acciones de control están muy limitadas por el ángulo máximo que permiten las ruedas de dirección. Por esta razón se optó por utilizar programación genética, dado que, aún sin tener una propuesta de controlador, este enfoque de cómputo bioinspirado sería capaz de generar un controlador satisfactorio.

En [8] se describe una forma de diseñar controladores no lineales, en la que por medio de programación genética se crean diferentes modos de control. A su vez, estos se activan y desactivan obedeciendo a otra función generada de la misma forma. En estas características esta investigación es similar, pues de igual manera se consiguieron dos controladores, en los que uno de ellos toma decisiones sobre el otro. Sin embargo, la forma en que se consigue tiene mayor similitud con [3,7], donde la evaluación se hace directamente probando el control sobre el sistema, en lugar de sobre fórmulas que los validen.

La capacidad de generalización de la programación genética se observó al momento de someter al mejor individuo final a una serie de condiciones aleatorias, y observar su capacidad de resolverlas todas satisfactoriamente.

Partiendo de este planteamiento, este artículo está estructurado de la siguiente manera. En la sección 2 se presenta la descripción del simulador empleado. En la sección 3 se presenta una breve introducción de Programación Genética y sus operadores genéticos. En la sección 4 se expresa el planteamiento del problema. La sección 5 describe a detalle las primitivas y los parámetros de la PG empleados para este caso de estudio. En la sección 6 se presentan los resultados obtenidos y en la sección 7 se detallan las conclusiones y trabajos futuros.

2. Descripción del simulador

Para poder generar el controlador es necesario tener un simulador donde probarlo. Por esta razón se programó una función que describiera el movimiento del auto (ecuaciones (3 y 4)). En la Figura 1 se muestra cómo puede reducirse el cálculo del

movimiento de un vehículo de cuatro ruedas a uno de 2 [4,11]. Si el auto conserva una dirección θ por un tiempo t a una velocidad V, recorrerá una distancia D que representa un arco de ángulo φ contenido en el círculo con radio R (ecuación (1)).

$$\varphi = \frac{V\,t}{R}\ ; para\ R = \frac{\tan(\theta)}{L}. \tag{1}$$

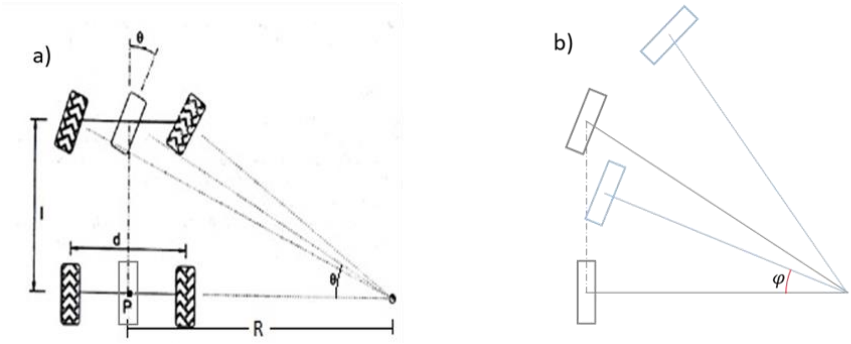

Fig. 1. a) Descripción de como la dirección Ackermann permite modelar un vehículo de cuatro ruedas como uno de dos. b) Ilustración del movimiento de un vehículo de dos ruedas.

Cabe mencionar el siguiente límite:

$$\lim_{R\to 0} \varphi \rightarrow \infty. \tag{2}$$

En el límite representado en la ecuación (2) se podría pensar que habría una indeterminación; sin embargo, dado que R al ser función del ángulo de las ruedas, no puede disminuir más allá de lo que permita la torsión máxima de la dirección (considerada en esta investigación como 40°). Con esta consideración pueden ahora plantearse los desplazamientos lineales en coordenadas cartesianas como:

$$\Delta X\ =\ R\ sin(\varphi\ ), \tag{3}$$

$$\Delta Y\ =\ R(1\ -\ cos(\varphi\ )). \tag{4}$$

3. Programación genética

La programación genética, como ya se dijo, es una técnica evolutiva basada en poblaciones, la cual utiliza estructuras de forma y tamaño variable. Dichas estructuras pueden ser árboles, grafos en general [8] o incluso líneas de código [4]; para cada una de ellas tiene diferentes operadores de cruza y mutación. Con todo esto, la PG puede generalizar una solución a un problema con diferentes condiciones variables; es aplicable a un sin número de situaciones, dependiendo de la interpretación que se le dé a la evaluación y la forma en que ésta se lleve a cabo. En la Figura 2 se presentan los pasos del algoritmo clásico de Programación Genética.

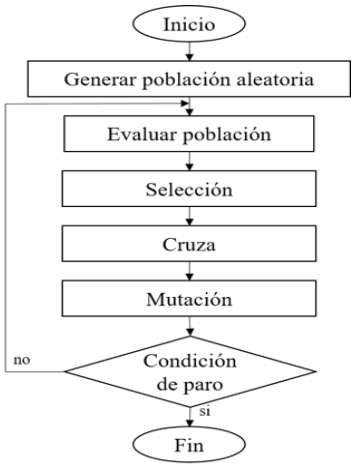

Fig. 2. Algoritmo de programación genética.

El caso concreto de la PG utilizada para este artículo es la estándar, que usa árboles como estructura a evolucionar [6]. Se incluyen las funciones definidas automáticamente [3], o ADFs por sus siglas en inglés, las cuales son árboles completos que representan una hoja en el árbol principal. Las ADFs tienen la particularidad de que dentro del árbol principal son tratadas exactamente igual a terminales, como si fuera un solo nodo y ese nodo tuviera simplemente un dato; por otro lado, fuera de ese árbol pueden cruzarse o mutar en cualquier punto de su estructura.

El operador de cruza seleccionado es uno muy simple, para efectuarlo se eligen aleatoriamente dos individuos, y con probabilidad *Pc* (Probabilidad de Cruza) se ejecuta lo siguiente: en cada uno de ellos se toma un nodo (con probabilidad uniforme) y se intercambian entre sí las ramas (o sub-árboles) asociados a esos nodos.

Además, se introdujo una pequeña adaptación al operador para evitar el crecimiento descontrolado en la profundidad de los árboles. La selección del nodo en el primer individuo sigue haciéndose de forma totalmente aleatoria, pero en el segundo solo puede seleccionarse un nodo cuya profundidad no provoque, después de la cruza, un crecimiento más allá del máximo establecido por individuo.

Para el caso de la mutación se utilizaron 3 operadores diferentes. En el primero se selecciona un nodo aleatorio y, junto con la rama que representa, es remplazado por otro generado aleatoriamente, tal que la profundidad total del individuo no sobrepase el límite. El segundo es muy similar, pero difiere en que en éste el nodo elegido se remplaza con una hoja. En el tercero, las terminales constantes son cambiadas por otras aleatorias.

El primero y segundo ejecutan su operación con una probabilidad *Pm* (Probabilidad de mutación); en el tercero la probabilidad es para cada nodo, no para la operación total.

4. Planteamiento del problema

Se desea generar el controlador de dirección de un vehículo autónomo. Se plantea que funcione sobre una trayectoria de rectas generada por un procesamiento previo de la información. Esto para reducir la dificultad de manejar coordenadas relativas, mismas utilizadas en [7], por otro lado, se contrasta con ese estudio en que ahí se utilizan curvas y el conjunto de trayectorias con que se entrena es reducido. Visto así, el controlador planteado es un poco complejo, pues el sistema tiene como entrada una trayectoria deseada y a la salida del controlador sólo hay una interacción con el ángulo de las ruedas delanteras. Lo más probable es que si se le da a resolver el problema completo a la PG, no podría resolverlo o requeriría demasiado poder de cómputo. Por lo que se dividió en dos partes.

Lo primero sería lograr un controlador capaz de hacer al vehículo seguir una recta dada, a partir de cualquier posición y orientación relativas a ésta. No obstante, este subproblema puede simplificarse aún más. Si se considera que el auto debe seguir al eje X, puede generalizarse a cualquier recta haciendo la correspondiente transformación de coordenadas. Además, no es necesario que el controlador funcione para cualquier posición y orientación, puesto que la trayectoria se mantendrá cercana al auto todo el tiempo. De esta forma, solo se utilizaron posiciones a menos de 2m de distancia a la recta y orientación de máximo 90° y mínimo -90°. El segundo subproblema sería entonces un controlador que decida en que momento pasar de seguir una recta a seguir la posterior. Las trayectorias se generan aleatoriamente para cada generación, lo cual promueve la generalización del resultado.

Para la simulación en ambos casos se debía discretizar el tiempo, y se hizo con intervalos de una décima de segundo de resolución. Se calcularon las derivadas, ecuación (6), e integrales, ecuación (5), tomando sólo dos puntos. Esto dado que la función que describe el movimiento del auto no es tan compleja, por lo cual con dos puntos se obtiene una aproximación aceptable. De la misma forma, la resolución es la suficiente para hacer una simulación fiel. Dado que la velocidad del auto se consideró en todo momento 1m/s, se tiene un desplazamiento de apenas 10cm para cada evaluación.

$$\int_a^b f(x)dx \approx \sum_{i=1}^{N} f(x_i)\Delta x \; ; \; N = \frac{b-a}{\Delta x}, \tag{5}$$

$$\frac{d}{dx} f(x) \approx \frac{f(x)-f(x-\Delta x)}{\Delta x}. \tag{6}$$

5. Parámetros de PG

Para el primer subproblema, en cada intervalo se tomó una evaluación del controlador y el valor arrojado se consideró la velocidad con que se cambiaba la dirección de las ruedas en radianes/segundo. La simulación se corrió durante 20 minutos, después de los cuales se ejecutaron 100 generaciones. La función de evaluación fue multiobjetivo escalar [6], en la cual se tomaron en cuenta 3 parámetros: la integral de la curva descrita por el auto (dividida entre 100); la diferencia entre las oscilaciones hechas y las mínimas necesarias (divididas entre 15); y el error final del auto.

Para el segundo se hicieron dos versiones distintas. En ambos si el resultado de la evaluación del individuo es positivo, se decide cambiar a seguir la siguiente recta, de otra forma el auto se mantiene en la actual. Los cálculos se realizaron con las coordenadas relativas a la recta en que el auto determinaba estar.

Para la primera versión la simulación tardó poco más de una hora en llevar a cabo 100 generaciones. La evaluación de cada individuo tardó más dado que se evaluaron 10 trayectorias distintas, cada una conformada por 4 rectas. Este número es debido a que se quería considerar la mayor cantidad de casos distintos como: curvas en sentidos opuestos, rectas cortas entre vueltas, y demás.

La función de evaluación de nuevo fue multiobjetivo escalar, tomando en cuenta: la diferencia entre las oscilaciones hechas y las mínimas necesarias (divididas entre 5); la integral de la curva descrita por el auto (dividida entre 10); y, considerando que la trayectoria se dio en un arreglo de rectas, la diferencia entre el índice de la recta a la que llegó y aquella a la cual debía llegar, multiplicado por 100. Este peso tan alto se utilizó para forzar a la simulación a llevar al individuo hasta el final de la trayectoria. Para obtener un mejor controlador, se evaluó sobre 10 posiciones iniciales diferentes y se le asignó como puntaje de adaptación al individuo el peor de los 10.

Para la segunda versión se buscaba ya un controlador capaz de evitar colisiones. Se consideró agregar sensores como variables o formas de sensado como funciones. Sin embargo, la necesidad de procesamiento se incrementó demasiado. Por lo que se consideró al final, utilizar rectas delimitadoras de la trayectoria colocadas a 3 metros de ésta. El objetivo ahora sería no salirse de esas líneas, salirse se consideraría una colisión. De esta forma el generador de trayectorias previo a este controlador debería solo asegurarse de que los obstáculos estuvieran por fuera de dichas líneas.

En la simulación se llevaron a cabo sólo 50 generaciones en alrededor de 3 horas. Esto debido a que el cálculo de colisiones se llevaba muchos recursos de cómputo. La función objetivo se definió únicamente como la minimización de los choques. En este caso se evaluó a cada individuo en 10 trayectorias distintas, y como puntaje del individuo se tomó el promedio. A diferencia de las dos simulaciones anteriores se tuvo cuidado en que, aunque las trayectorias eran distintas en cada generación, eran las mismas para cada individuo, de esta forma se haría una clasificación más justa que llevara a una convergencia más rápida.

Los parámetros compartidos por todas las simulaciones se muestran a continuación, junto con las funciones base (Tabla 1), y las variables tomadas en cuenta en la simulación (Tabla 2):

- Tamaño de población: 500
- Probabilidad de cruza: 0.8
- Probabilidad de mutación: 0.1
- Número de generaciones: 100
- Máxima profundidad:
  - Para primer subproblema: 7
  - Para segundo versión 1: Adf0: 2; Adf1: 2; Árbol principal: 4
  - Para segundo versión 2: Afd0: 3; Adf1: 3: Árbol principal: 7
- Elitismo: no
- Forma de selección: Torneo tamaño 2

Tabla 1. Funciones utilizadas.

Función	Resultado
Neg(x)	-x
Suma (x, y)	x+y
Resta (x, y)	x-y
Mul (x, y)	x*y
División (x, y)	x/y si $y \neq 0$ 1 si y=0
IfR (a, b, c, d, e)	d si $b \leq a \leq c$ e de otra forma
IfGT (a, b, c, d)	c si $a \geq b$ d de otra forma
Hyp (x, y)	$\sqrt{x^2 + y^2}$
Sin(x)	sin(x)
E(x)	$e^{-abs(x)}$
Raiz2(x)	$\sqrt{x}$

Tabla 2. Variables utilizadas en la simulación.

Variables	Representación
Largo del auto	L
Ancho del auto	A
Velocidad	V
Ángulo de las ruedas	THr
Distancia en dirección de la recta	X
Distancia a la recta	Y
Orientación del auto	THa
Máxima torsión de las ruedas	Ms
Velocidad en dirección de la recta	Vx
Velocidad perpendicular a la recta	Vy
Velocidad angular del auto	W
Distancia a la siguiente esquina	Desq
Largo de la siguiente recta	D_Sesq
Angulo de la siguiente esquina	Thesq
Angulo siguiente a Thesq	Th_Sesq

6. Análisis de resultados

El primer controlador fue crucial para continuar con la investigación, pues en él se basaría el resultado final. Se corrieron diferentes simulaciones con diferentes configuraciones; la final es la que se describe en la sección anterior. Además de ajustar esos parámetros, se corrieron 10 simulaciones distintas con ellos; y de éstas se eligió la que lanzó los mejores resultados, basándose directamente en el error y en la apariencia de la trayectoria, mostrada en la Figura 3. El controlador resultante fue el siguiente:

Neg(Resta(Mul(E(THr),Division(Y,L)),Resta(Neg(THr),IfR(X,Ms,THa,W,THa))))

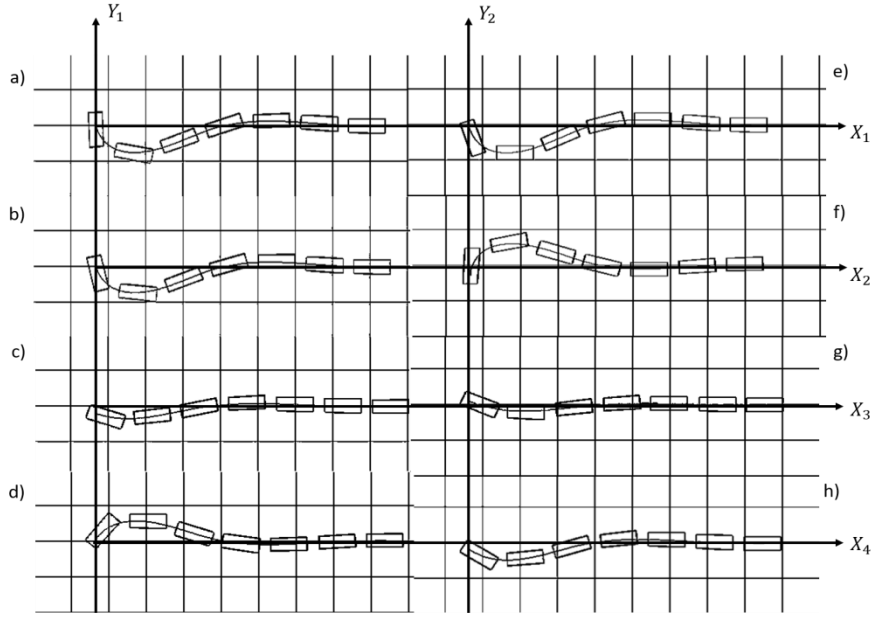

Fig. 3. Se muestran tres diferentes simulaciones con condiciones iniciales distintas (en cuadrícula 5x5m). Puede apreciarse la respuesta del sistema con este controlador y se observan muy pocas oscilaciones al tratar de seguir el eje Xn.

En este controlador se observan algunas variables cuya presencia podría predecirse fácilmente. Por ejemplo, la coordenada en Y comunica al control lo alejado que se encuentra de la recta; la velocidad angular, ayuda a evitar sobrepasos. Por otro lado, resulta interesante ver la presencia de la variable X, puesto que, indirectamente expresa como el error es diferente al inicio del recorrido. Esto dado que, dentro de una función condicional, esta variable modifica el resultado solo en un rango de valores.

También se ven involucradas algunas constantes que al parecer a las leyes de selección les parecieron útiles (el largo del auto y la máxima velocidad de torsión). Estos valores en un ambiente real varían, sin embargo, para una implementación

deberían sustituirse esas variables por constantes. Ya que la simulación aquí descrita, no toma en cuenta el cambio de las mismas ni el efecto que tendrían en el control.

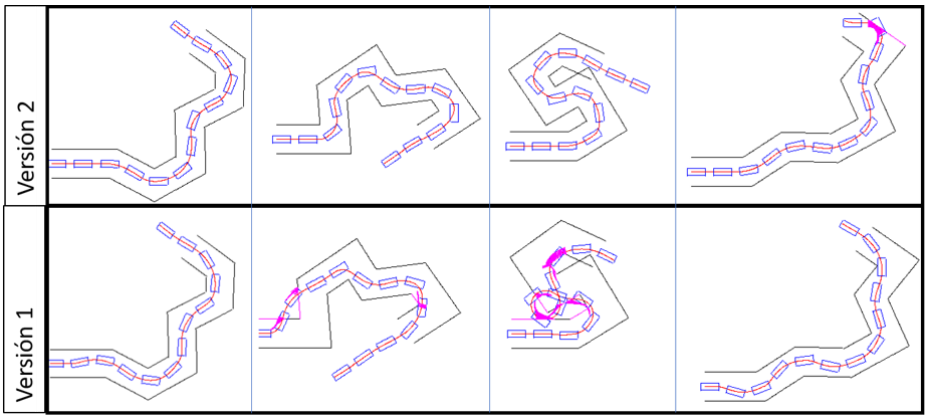

Fig. 4. Comparación de las dos versiones de los controladores. Puede observarse como la versión uno, al guiarse principalmente por la integral de la curva, evoluciona una estrategia que busca encontrar atajos. La versión dos, en contraste, al guiarse por las colisiones, se mantiene más apegada a la trayectoria que debe seguir.

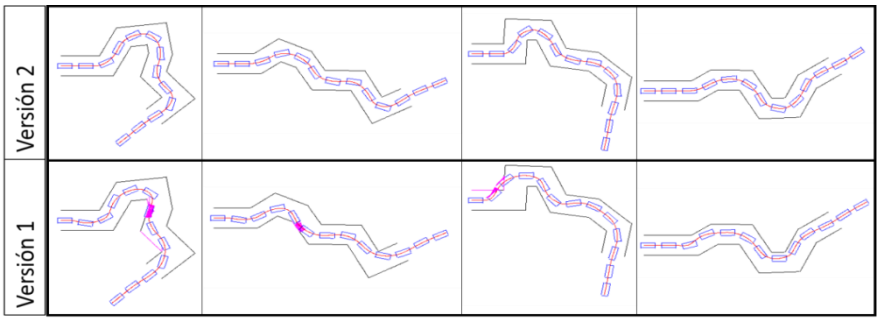

Fig. 5. Comparación de las dos versiones de los controladores. Se observa un comportamiento similar al presentado en la Figura 4.

Para el segundo subproblema, dado que se crearon 2 versiones distintas, se consiguió hacer una comparación de lo que se genera con PG cambiando la función objetivo. Aun cuando ambas versiones resuelven lo mismo, la forma en que se califica su desempeño es distinta, por lo tanto, la definición de "mejor" cambia para cada simulación, lo cual puede apreciarse en las Figuras 4 y 5. De esta forma los distintos controladores obtenidos fueron los siguientes:

- Versión 1:
  - ADF0:
    - Suma(A,A)

  - ADF1:
    - Hyp(Desq,THa)
  - Árbol principal:
    - IfR(Resta(Th_Sesq,ADF1),Neg(D_Sesq),Seno(Resta(ADF0,Desq)), Raiz2(V),IfR(D_Sesq,ADF0,ADF0,THa,-0.659634))
- Versión 2:
  - ADF0:
    - IfR(Division(X,V),IfR(V,-1.422355,V,1.950899,V),IfR(Thesq,X,X,-1.422355,X),-1.422355,-1.422355)
  - ADF1:
    - Hyp(IfR(Ms,Vx,D_Sesq,ADF0,D_Sesq),Ms)
  - Árbol Principal:
    - Resta(Resta(Suma(Desq,Mv),Suma(Desq,ADF0)),Raiz2(Neg(Hyp (Desq , Hyp(Suma(Desq,ADF0),Suma(THr,ADF0))))))

Las distintas estrategias que siguen las versiones 1 y 2 tienen ventajas y desventajas. En la parte derecha de la Figura 4 puede observarse como la preferencia de atajos de la versión 1, le da una ventaja sobre la versión 2. Sin embargo, en el resto de las trayectorias la misma preferencia genera una desventaja, sobre todo en las imágenes centrales; mientras en la imagen de la izquierda igual se genera una trayectoria con más oscilaciones.

7. Conclusiones y trabajos futuros

La programación genética se utilizó con éxito para el desarrollo de un controlador, mismo que fue probado ante diferentes circunstancias y demostró resolver el problema. Esto sin importar la configuración de la trayectoria, y hasta cierto punto la complejidad de la misma, puesto que el controlador fue capaz de mantenerse dentro de las líneas delimitadoras en la gran mayoría de los casos.

Aquí puede verse la aplicabilidad de la programación genética en el diseño de controladores. En este caso se inició con un espacio de búsqueda bastante grande, utilizando todas las variables que podrían o no verse involucradas; e igual se hizo con las funciones. Y aún con un espacio de búsqueda tan basto, el algoritmo pudo entregar un resultado aceptable.

También cabe resaltar que es importante la separación de un problema en subproblemas. De esta forma, aunque se tenga un espacio muy grande de soluciones, los problemas que se intentan resolver son relativamente sencillos. Por ejemplo, al inicio se consideró realizar un controlador que llevara al auto de un punto A a uno B. Sin embargo, eso involucraba demasiadas variables, y se trataba de un problema muy complejo por lo cual no se obtuvieron resultados. De esta forma se notó la necesidad de dividir el problema en dos partes.

Otro punto para tomar en cuenta es la limitación de recursos de cómputo. Para la versión 2 del segundo subproblema se consideró utilizar información de sensores, pero eso hacía que la necesidad de recursos de cómputo creciera demasiado, al punto en que cada generación tomaba más de 20 minutos en evaluarse. Así, si bien no se

consiguió que el controlador utilizara información del ambiente directamente, si es capaz de evitar obstáculos indirectamente.

Esta investigación puede ampliarse en un futuro de dos formas: una es correr el algoritmo de la segunda versión por una cantidad mayor de generaciones. Esto podría perfeccionar la estrategia que se fomenta con esa función objetivo. La segunda forma es agregar los efectos dinámicos, lo cual aseguraría la posibilidad de aplicar el controlador a altas velocidades. Asimismo, resulta interesante combinar este estudio con otras aproximaciones, como la desarrollada en [7], donde se controla, además, el acelerador y se manejan curvas. El resultado sería capaz de enfrentarse a un mayor número de ambientes de una manera más eficiente.

Referencias

1. Goldberg, D.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Boston (1989)
2. Holland, J.: Adaptation in natural and artificial systems. MIT Press, Cambridge (1992)
3. Koza, J.: Genetic programming: On the programming of computers by means of natural selection. MIT Press, Cambridge, Mass. (1992)
4. King-Hele, D.: Erasmus Darwin's improved design for steering carriages--and cars. Notes and Records of the Royal Society. 56, 1, pp. 41–62 (2002)
5. Brameier, M., Banzhaf, W.: Linear genetic programming. Springer, New York (2007)
6. Sekaj, I., Perkacz., J.: Genetic programming - based controller design. In: 2007 IEEE Congress on Evolutionary Computation (2007)
7. Ebner, M., Tiede, T.: Evolving driving controllers using Genetic Programming. In: 2009 IEEE Symposium on Computational Intelligence and Games (2009)
8. Verdier, C., Mazo, Jr., M.: Formal Controller Synthesis via Genetic Programming. IFAC-PapersOnLine. 50, 1, pp. 7205–7210 (2017)
9. Mabu, S., Hirasawa, K., Hu, J.: A Graph-Based Evolutionary Algorithm: Genetic Network Programming (GNP) and Its Extension Using Reinforcement Learning. Evolutionary Computation. 15, pp. 369–398 (2007)
10. Poli, R., Langdon, W.B., McPhee, N.F., Koza, J.R.: A field guide to genetic programming. Lulu Press, S.l. (2008)
11. Geometría Direccional: Estudio de las cotas, https://www.edu.xunta.gal/centros/cafi/aulavirtual2/pluginfile.php/14627/mod_folder/content/0/Geometria_direccional.pdf