UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

TESIS DE GRADO

“TECNICA EVOLUTIVA DE AGENTE INTELIGENTE CON REDES NEURONALES Y ALGORITMO GENETICO EN UNA PLATAFORMA 2D”

Tesis de Grado para obtener el Título de Licenciatura en Informática Mención Ingeniería de Sistemas Informáticos

POR: JOSE GONZALO OLIVAREZ BOLIVAR

TUTOR: M.sc. GROVER ALEX RODRIGUEZ

LA PAZ – BOLIVIA

2021

HOJA DE CALIFICACIONES

UNIVERSIDAD MAYOR DE SAN ANDRÉS

FACULTAD DE CIENCIAS PURAS Y NATURALES

CARRERA DE INFORMÁTICA

Tesis de grado

TECNICA EVOLUTIVA DE AGENTE INTELIGENTE CON REDES NEURONALES Y ALGORITMO GENETICO EN UNA PLATAFORMA 2D

Presentado por: Jose Gonzalo Olivarez Bolivar

Para optar el grado Académico de Licenciado en Informática

Mención Ingeniería de Sistemas Informáticos

Nota Numeral: ……80…….

Nota Literal: ……OCHENTA……..

Ha sido: ……SOBRESALIENTE…….

Director de la carrera de Informática: Ph.D. Jose Maria Tapia Baltazar

Tutor: M.Sc. Grover Alex Rodriguez Ramirez

Tribunal: Lic. Brigida Alexandra Carvajal Blanco ……

Tribunal: Lic. Eufren Llanque Quispe ……

Tribunal: Lic. Victor Pablo Pozo Diaz ……

UNIVERSIDAD MAYOR DE SAN ANDRÉS
FACULTAD DE CIENCIAS PURAS Y NATURALES
CARRERA DE INFORMÁTICA

LA CARRERA DE INFORMÁTICA DE LA FACULTAD DE CIENCIAS PURAS Y NATURALES PERTENECIENTE A LA UNIVERSIDAD MAYOR DE SAN ANDRÉS AUTORIZA EL USO DE LA INFORMACIÓN CONTENIDA EN ESTE DOCUMENTO SI LOS PROPÓSITOS SON ESTRICTAMENTE ACADÉMICOS.

LICENCIA DE USO

El usuario está autorizado a:

a) visualizar el documento mediante el uso de un ordenador o dispositivo móvil.
b) copiar, almacenar o imprimir si ha de ser de uso exclusivamente personal y privado.
c) copiar textualmente parte(s) de su contenido mencionando la fuente y/o haciendo la referencia correspondiente respetando normas de redacción e investigación.

El usuario no puede publicar, distribuir o realizar emisión o exhibición alguna de este material, sin la autorización correspondiente.

TODOS LOS DERECHOS RESERVADOS. EL USO NO AUTORIZADO DE LOS CONTENIDOS PUBLICADOS EN ESTE SITIO DERIVARA EN EL INICIO DE ACCIONES LEGALES CONTEMPLADOS EN LA LEY DE DERECHOS DE AUTOR.

DEDICATORIA

Dedico la presenta investigación

A Dios por ayudarme a perseverar en toda mi vida personal

A mis padres Agustina Bolivar y Gonzalo Olivarez

por todo su sacrificio para ayudarme a culminar con mis estudios

A mis docentes de la carrera de informática

A todos mis amigos que conocí en el ambiente de la informática

AGRADECIMIENTO

A Dios por las herramientas mental y física que me concedió para lograr realizarme en la carrera de informática

A mi familia que me inculco los valores y ética para crecer como persona de provecho en la sociedad

Al M.sc Grover Alex Rodriguez por ser mi tutor metodológico de la presente tesis de grado y por saber dar el conocimiento y concejos para la culminación de esta tesis de grado

olivarez.bolivar@gmail.com

Resumen

Utilizaremos redes neuronales para entrenarlas utilizaremos algoritmos genéticos lo utilizaremos para imitar un comportamiento evolutivo ¿por qué entrenamos de esta forma? porque para un ambiente de simulación los estados prácticamente tienden al infinito son demasiado grandes para crear un agente que se adapte a lo que queremos.

La presente tesis de grado muestra cómo se puede emplear una técnica de entrenamiento de algoritmo genético con red neuronal de una manera fácil para no tener que emplear estados que suelen tener a ser infinitos así se demostrara que es una opción de entrenamiento de red neuronal y aplicado a un agente inteligente se verá como este entrena y evoluciona en un escenario y ambiente 2D donde se mostrara su evolución y posterior desempeño del agente inteligente mejor entrenado.

Palabras clave: Algoritmo Genético, red neuronal, computación evolutiva, agente inteligente

Metodología: Prototipado ,Prometheus

Abstract

We will use neural networks to train them, we will use genetic algorithms, we will use them to imitate evolutionary behavior. Why do we train in this way?

The present degree thesis shows how a genetic algorithm training technique with a neural network can be used in an easy way so as not to have to use states that tend to be infinite, so it will be shown that it is an option of neural network training and applied an intelligent agent will be seen as he trains and evolves in a 2D environment and scenario where his evolution and subsequent performance of the best-trained intelligent agent is shown.

Keywords: Genetic Algorithm, neural network, evolutionary computing, intelligent agent Methodology:Prototyping,Prometheus

ÍNDICE

CAPITULO I ANTECEDENTES GENERALES 1
1.1 INTRODUCCIÓN 1
1.2 PROBLEMA 3
1.2.1 Antecedentes 3
1.2.2. Planteamiento del problema 4
1.2.2.1 Problema principal 4
1.2.2.2 Lista de Problemas 5
1.2.3. Formulación del problema 5
1.3 OBJETIVOS 5
1.3.1 Objetivo principal 5
1.3.2 Objetivos específicos 5
1.4 HIPÓTESIS 6
1.5 JUSTIFICACIÓN 6
1.5.1 Justificación social 6
1.5.2 Justificación Tecnológica 6
1.5.3 Justificación Científica 7
1.6. ALCANCES 7
1.7. METODOLOGÍA 7
CAPITULO II MARCO TEÓRICO 11
2.1. Base teórica que sustenta la Tesis 11
2.1.1Redes Neuronales 11
2.1.2 Redes neuronales Biológicas 11
2.1.3 Red neuronal artificial 13
2.1.3.1 EL PERCEPTRÓN 16
2.1.3.2 Regla de aprendizaje perceptrón 16
2.1.4 Clasificación 17
2.1.5 Aprendizaje o entrenamiento 18
2.2 Temática: Redes Neuronales y Algoritmos Genéticos 19

2.2.1 Computación Evolutiva 19
2.2.2 Programación Evolutiva 20
2.2.3 Algoritmo Genético 22
2.2.3.1 Cromosoma 22
2.2.3.2 Evaluación y Aptitud 22
2.2.3.3 Población 23
2.2.3.4 Selección 23
2.2.3.4.1 Selección por ruleta 23
2.2.3.4.1 Selección proporcional 23
2.2.3.4.2 Selección por Ranking 24
2.2.3.4.3 Cruce 24
2.2.3.4.4 Mutación 24
CAPÍTULO III DISEÑO METODOLÓGICO 26
3.1 INTRODUCCIÓN 26
3.1.1 Metodología de Prototipado 26
3.1.2 Metodología Prometheus. 27
3.2 Metodología de Prototipado combinado con la Metodología Prometheus 28
3.2.1 Desarrollo Metodología Prototipado 29
3.2.1.1 Fase de Desarrollo de Modelo 29
3.2.1.1.1 Análisis y Especificación 29
3.2.1.1.2 Diseño y construcción. 30
3.2.2 Desarrollo de la metodología prometheus.- 30
3.2.2.1 Especialización del sistema 30
3.2.2.2 Diseño y Arquitectura 33
3.2.2.3 Diseño detallado 34
3.2.2.3.1Algoritmo Genético para el entrenamiento de la red neuronal. - 34
3.2.2.3.2 Codificación del Cromosoma. 35
3.2.2.3.3 Genotipo y Fenotipo. 36
3.2.2.3.4 Evolución y Aptitud. 36
3.2.2.3.5 Selección por Ranking. 36
3.2.2.3.6 Operación de cruce. 37
3.2.2.3.7 Cruce Multipunto 37
3.2.2.3.8 Mutación. 38

3.3 Diseño y desarrollo final 40
3.3.1 Interface de inicio.- 40
3.3.2 Escenarios 2D.- 40
3.3.3 Diseño de la apariencia del agente.- 44
3.4 Entrega de Datos 46
3.4.1 Tabla para mostrar estadísticas. se muestra la mutación en porcentaje.- 46
3.4.2 Ejecución del agente inteligente en escenario 2D.- 47
Se ejecutara en un ordenador con 47
CAPÍTULO IV EVALUACIÓN DE RESULTADOS 54
4.1 RECOLECCIÓN DE DATOS. 54
4.2ANALISIS DE DATOS. 67
4.3 Demostración de la Hipótesis. 68
4.4 Resuelve el problema 74
CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES 75
5.1 CONCLUSIONES 75
5.2 RECOMENDACIONES 76
Bibliografía 77
ANEXOS 80

ÍNDICE DE TABLAS

Tabla 1Tipos de aprendizaje 19

Tabla 2 Estadísticas a mostrar en la interface 46

Tabla 3 .muestra las generaciones de 1 -35 54

Tabla 4 Muestra las generaciones del 101-135 Fuente:Propia 56

Tabla 5. muestra las generaciones del 200-234 58

Tabla 6.muestra las generaciones 304-338 60

Tabla 7. muestra las generaciones 381-415 62

Tabla 8 generaciones del 490-524 64

Tabla 9Muestra las generaciones del 530-564 66

Tabla 10 Interpretación de los datos evolutivas 68

Tabla 11. Generaciones posteriores de agentes inteligentes 71

Tabla 12. Generaciones posteriores de agentes inteligente ,evaluando resultados para varianza 72

Índice figuras

Figura2.1.2Estructura de una neurona biológica típica ---------- 12

Figura 2.1.3. Red Neuronal Artificial---------- 13

Figura 2.1.3.1. Modelo de neurona de McCulloch y Pitts ---------- 14

Figura 2.1.3.2. Función umbra ---------- 15

Figura 2.1.3.3. Funciones de activación---------- 15

Figura 2.1.3.3 ---------- 16

Figura 2.1.4Red Feed Fordward ---------- 17

Figura 2.1.4.1Red Recurrente ---------- 18

Figura 2.2.3.4.3 Cruce y Mutacion ---------- 24

Figura 2.2.3.4.4Ejemplo Mutación ---------- 25

Figura 3.1.1 Flujo Metodología Prototipado---------- 27

Figura 3.2 Metodología de Prototipado combinado con la Metodología Prometheus ---------- 28

Figura 3.2.2.1 Diagrama de Escenarios ---------- 32

Figura 3.2.2.2 Diagrama Agente -rol---------- 34

Figura 3.2.2.3.1 Diagrama de Flujo Algoritmo Genético ---------- 35

Figura 3.2.2.3.7 Cruce Multipunto ---------- 38

Figura 3.2.2.3.8 Mutacion por complemento ---------- 39

Figura 3.3.1 Interface de Inicio---------- 40

Figura 3.3.2 Plataforma 2D para desplazamiento del agente inteligente pista de conducir ---------- 41

Figura 3.3.2.1 TileMaps Plataforma 2D para desplazamiento del agente inteligente pista de conducir -- 42

Figura 3.3.2.2 Plataforma 2D para desplazamiento del agente inteligente pista de conducir ---------- 42

Figura 3.3.2.3 Plataforma 2D para desplazamiento del agente inteligente pista de conducir ---------- 43

Figura 3.3.3 agente inteligente que se desplazara en el escenario ---------- 44

Figura 3.3.3.1 agente inteligente auto que se desplazara en el escenario ---------- 45

Figura 3.4.2 agente inteligente desplazándose en la plataforma 2D ---------- 47

Figura 3.4.2.1 agente inteligente desplazándose en la plataforma 2D ---------- 48

Figura 3.4.2.2 agente inteligente desplazándose en la plataforma 2D ---------- 49

Figura 3.4.2.3 agente inteligente desplazándose en la plataforma 2D ---------- 50

Figura 3.4.2.4 agente inteligente auto desplazándose en la plataforma 2D ---------- 51

Figura 3.4.2.4 agente inteligente auto desplazándose en la plataforma 2D ---------- 51

Figura 3.4.2.4 agente inteligente auto desplazándose en la plataforma 2D ---------- 52

Figura 3.4.2.5 agente inteligente auto desplazándose en la plataforma 2D ---------- 52

Figura 3.4.2.6 agente inteligente auto desplazándose en la plataforma 2D ---------- 53

Figura4.3 Tendencia de Evolución ---------- 70

CAPITULO I ANTECEDENTES GENERALES

1.1 INTRODUCCIÓN

En la actualidad existen muchos campos aplicativos para el desarrollo de la IA (Inteligencia Artificial) como pueden ser agentes inteligentes, sistemas expertos, sistemas de control automáticos entre los conocidos. Muchos de estos elementos incluyen arboles de búsqueda, redes neuronales entre otros para su toma de decisiones inteligentes y también para simular un aprendizaje (Garcia, 2007).

Un agente software inteligente es un programa que puede realizar tareas específicas para un usuario y posee un grado de inteligencia suficiente para ejecutar parte de sus tareas de forma autónoma y para interactuar con su entorno de forma útil (Adamaris, 2014).

Según Garcia (2008) un agente es cualquier entidad que percibe su entorno a través de sensores y actúa sobre ese entorno mediante efectores. Un agente es racional cuando realiza la mejor acción posible a partir de los datos percibidos.

En las Redes Neuronales Artificiales, la unidad análoga a la neurona biológica es el elemento procesador. Un elemento procesador tiene varias entradas y las combina, normalmente con una suma básica. La suma de las entradas es modificada por una función de transferencia y el valor de la salida de esta función de transferencia se pasa directamente a la salida del elemento procesador (León Q. , 2012).

Los algoritmos evolutivos son métodos de optimización y búsqueda de soluciones basados en los postulados de la evolución biológica. En ellos se mantiene un conjunto de entidades que representan posibles soluciones, las cuales se mezclan, y compiten entre sí, de tal manera que las más aptas son capaces de prevalecer a lo largo del tiempo, evolucionando hacia mejores soluciones cada vez (Whitley, 2008).

Los algoritmos evolutivos, y la computación evolutiva, son una rama de la inteligencia artificial. Son utilizados principalmente en problemas con espacios de búsqueda extensos y no lineales, en donde otros métodos no son capaces de encontrar soluciones en un tiempo razonable.

Según Wilson(2005) la terminología de la teoría de la evolución, las entidades que representan las soluciones al problema se denominan individuos o cromosomas, y el conjunto de éstos, población. Los individuos son modificados por operadores genéticos, principalmente el sobre cruzamiento, que consiste en la mezcla de la información de dos o más individuos; la mutación, que es un cambio aleatorio en los individuos; y la selección, consistente en la elección de los individuos que sobrevivirán y conformarán la siguiente generación. Dado que los individuos que representan las soluciones más adecuadas al problema tienen más posibilidades de sobrevivir, la población va mejorando gradualmente.

Según Zilouchian(2010) el algoritmo genético es un algoritmo matemático altamente paralelo que transforma un conjunto de objetos matemáticos individuales con respecto al tiempo usando operaciones modeladas de acuerdo al principio Darwiniano de reproducción y supervivencia del más apto, y tras haberse presentado de forma natural una serie de operaciones genéticas de entre las que destaca la recombinación sexual. Cada uno de estos

objetos matemáticos suele ser una cadena de caracteres (letras o números) de longitud fija que se ajusta al modelo de las cadenas de cromosomas, y se les asocia con una cierta función matemática que refleja su aptitud.

1.2 PROBLEMA

1.2.1 Antecedentes

Existen muchos campos aplicativos de los agentes inteligentes que están siendo relacionados con los árboles de búsqueda, arboles de decisión, redes neuronales con algoritmos evolutivos. Con la integración de redes neuronales y algoritmos genéticos se pueden realizar diversas simulaciones de aprendizaje y aplicarlas como un agente evolutivo.

La historia de la IA evolutiva y de aprendizaje se remonta a finales de los años 50, cuando Arthut L. Samuel , trabajador de la IBM , desarrollo un programa para la IBM 701 , la primera máquina comercial de IBM, capaz de jugar a las damas mediante aprendizaje evolutivo (Arthut, 1990).

En Bolivia se realizó una tesis de grado, que como tema de investigación implicaba un videojuego con IA como agente inteligente.

David Joel Mamani Quisbert ”PROTOTIPO DE UN VIDEOJUEGO DE FUTBOL UTILIZANDO METODOS DE TOMA DE DECISIONES DE LA INTELIGENCIA ARTIFICIAL ”, UMSA-informática/2015 ,Es el desarrollo de un videojuego con IA basada en Arboles de Comportamiento y Arboles de Decisión para que los jugadores puedan asumir comportamientos inteligentes.

En la ciudad de Jaen España se realizo el trabajo de fin de grado que describe algoritmos evolutivos en videojuegos rpg.

Jose Javier Perez Cruz" VIDEOJUEGO RPG(ROLE PLAYING GAME) BASADO EN ALGORITMOS EVOLUTIVOS " Universidad Jaen-Ingenieria Informatica/2018,Es un juego de combate por turnos donde el jugador tomara el rol de entrenador .Posteriormente, será el personaje quien decidirá mediante un algoritmo genético la mejor solución para ganar.

1.2.2. Planteamiento del problema

En las simulaciones de agentes inteligentes que estén ligados con algoritmos evolutivos muchos de estos estados están compuestos por una maquina finita de estados que hace que se simulen estos comportamientos inteligentes por que los dividen los problemas de programación en pequeñas partes más manejables.

Estas máquinas de estados pueden tener ventajas por su simplicidad gracias a su diagrama de estados suelen ser muy intuitivos. Pero esta simplicidad es su mayor inconveniente ya que al crecer de manera significativa según la complejidad del campo aplicativo de la simulación esta máquina finita de estados aumenta sus estados y transiciones llegando a ser gradualmente inmanejables.

1.2.2.1 Problema principal

A causa de las limitantes de las máquinas de estados finitos surge en el ambiente de las simulaciones un término llamado estupidez artificial (Linden, 2003) que tiene como problemáticas la sensación de irrealidad, la aparición de comportamientos anormales en situaciones no consideradas o el conocimiento de comportamientos predecibles.

1.2.2.2 Lista de Problemas

-Bajo desempeño de agentes inteligentes con comportamientos predictivos y repetitivos

-Carece de sensación atractiva de evoluciones inteligentes

-La deficiencia de las simulaciones de simulaciones con máquinas de estados finitos

1.2.3. Formulación del problema

¿Cómo mejorar el desempeño de un agente inteligente respecto a la carencia de sensación atractiva y evolutiva?

1.3 OBJETIVOS

1.3.1 Objetivo principal

Desarrollar un agente inteligente que pueda tener un desempeño atractivo con sensación evolutiva, basada en redes neuronales y algoritmo genético en una plataforma 2D.

1.3.2 Objetivos específicos

-Construir una simulación de calidad en IA aplicada en GotdotEngine y su lenguaje nativo

-Diseñar un escenario en 2D de la categoría de plataformas.

-Entrenar la red neuronal según estímulos del ambiente del escenario.

-Evaluar el prototipo construido.

1.4 HIPÓTESIS

El agente inteligente controlado por redes neuronales y algoritmo genético mejora el comportamiento y desempeño atractivo evolutivo con una función de aceptación del 91%.

1.5 JUSTIFICACIÓN

1.5.1 Justificación social

La presente investigación de agente inteligente estará dirigida para beneficiar directamente a estudiantes de la carrera de informática-UMSA ya que puede ser precedente de varios tipos de aplicación con esta técnica de entrenamiento evolutivo. También serán beneficiados diseñadores de video juegos por que el escenario de la simulación estará hecho en un motor de video juegos amigable para aprendizaje de personas interesadas en video juegos.

1.5.2 Justificación Tecnológica

El aporte tecnológico será palpable en los diferentes tipos de usos que ya tienen las redes neuronales actualmente con el veneficio que puede no necesariamente necesita computadoras de alto rendimiento si no que puede ser compilada sin problemas en computadoras mas comunes en uso. Todos los juegos necesitan un nivel de programación en IA para su mejor popularidad sobre todo en los llamados BosFinal(enemigo final)con el presente proyecto se aplicara las redes neuronales y algoritmo genético en un agente inteligente que podrá ser adaptado a BosFinal de videojuego también se implementara un ambiente de juego para el desenvolvimiento del personaje.

1.5.3 Justificación Científica

El aporte al conocimiento científico de esta investigación sentara un precedente acerca de los resultados obtenidos al implementar redes neuronales y algoritmos genéticos como técnicas para el aprendizaje evolutivo de un agente inteligente.

1.6. ALCANCES

La pretensión de este trabajo es adaptar un algoritmo genético para el aprendizaje de una red neuronal y que después del entrenamiento pueda evolucionar a un óptimo tipo comportamiento del agente inteligente.

Para demostrar el entrenamiento evolutivo se construirá una Red Neuronal Artificial y un algoritmo genético que entrene a la Red Neuronal Artificial, de manera que pueda aproximarse a un agente inteligente entrenado de manera evolutiva.

Para este agente inteligente se implementará un escenario 2D a partir el cual podrá demostrarse las pruebas suficientes para poder sacar conclusiones.

1.7. METODOLOGÍA

La comprobación de la hipótesis planteada en la presente investigación será realizada a través de los pasos establecidos por el método científico de Mario Bunge, que partirá del diseño del método de entrenamiento de una red neuronal basado en un algoritmo genético para luego aplicarla en un prototipo computarizado, cuyo proceso será basado en el proceso Unificado de desarrollo de software (Hubel, 2005).

Por lo tanto, para comprobar la hipótesis o refutarla se basará principalmente en los estudios que se obtendrán a partir del prototipo que se generara en un escenario 2D.

De esta manera los puntos relacionados con la comprobación de la hipótesis presentada involucran:

- El diseño de la prueba, a la vez relacionado con el planteamiento de los medios para poner a prueba las predicciones.
- La ejecución de la prueba, que incluye la realización de operaciones y recolecciones de datos.
- La Elaboración de Datos que esta relación con la clasificación, análisis, evaluación de los datos

Las metodologías para el desarrollo del prototipo y el agente inteligente serán: Metodología de Prototipado y Metodología Prometheus.

Metodología de Prototipado está relacionada con la mejora continua y el **Ciclo de Deming** que consiste en un proceso iterativo enfocado en diseñar, implementar, medir y ajustar un plan (James, 2010).

En el caso de un prototipo, podríamos aplicar **5 etapas en el diseño e implementación de una idea** (previamente priorizada para pasar al proceso de prototipado). Esta metodología se puede aplicar en el desarrollo de nuevos negocios, prototipado web, prototipado de apps, entre otros entornos.

Etapas para diseñar e implementar un prototipo:

1. **Define los requerimientos y variables.** Lo primero es saber qué vamos a explorar, cuál es el propósito del prototipo, qué elementos se requieren.
2. **Define las herramientas para el diseño y testeo.** Tomando como ejemplo un prototipado web se pueden utilizar herramientas como inVision, UXPin, Figma o incluso herramientas que no requieren conocimientos de diseño o código, como proto.io. En el caso de un prototipado de apps existen herramientas como Marvelapp.
3. **Diseña el prototipo de tu idea.** Analiza qué tipo de prototipo es el más adecuado y evalúa factores como el diseño, montaje, ergonomía, materiales, formas, dimensiones, entre otros.
4. **Testea el prototipo.** Da a conocer el prototipo a los usuarios, observa y registra el desempeño del prototipo, y permite al usuario aportar ideas.
5. **Analiza los resultados y aprendizajes.** A partir de la definición inicial y el desempeño del prototipo, perfecciona el concepto de solución.

Metodología Prometheus es conocido para el desarrollo de agentes inteligentes tiene 3 etapas básicas:

- Especificaciones del sistema
- Arquitectura del sistema
- Diseñado detallado del sistema

Dentro de las aplicaciones del sistema se deben desarrollar casos de uso, identificar totalmente los objetos primordiales, diagrama de escenarios, objetos, roles, identificar la interface del agente con su ambiente (Padgam, 2002).

Por último, se tendrá la inferencia de conclusiones donde la interpretación de los datos elaborados a luz del modelo teórico.

CAPITULO II MARCO TEÓRICO

2.1. Base teórica que sustenta la Tesis

2.1.1Redes Neuronales

Las redes neuronales artificiales son módulos matemáticos que interpretan el comportamiento del sistema nervioso. Como todo modelo, realizan una simplificación del sistema real que simulan y toman las características principales del mismo para la resolución de tareas determinadas (Schmidhuber, 2015).

2.1.2 Redes neuronales Biológicas

Se estima que el sistema nervioso contiene alrededor de cien mil millones de *neuronas*, organizadas mediante una red compleja en la que las neuronas individuales pueden estar conectadas a varios miles de neuronas distintas. Se calcula que una neurona del córtex cerebral recibe información, por término medio, de unas 10.000 neuronas, y envía impulsos a varios cientos de ellas (LAGE, 2009).

Desde un punto de vista funcional, las neuronas constituyen procesadores de información sencillos. Como todo sistema de este tipo, posee un canal de entrada de información, las *dendritas*; un órgano de cómputo, el *soma*, y un canal de salida, el *axón(Fig.2.1.2)*

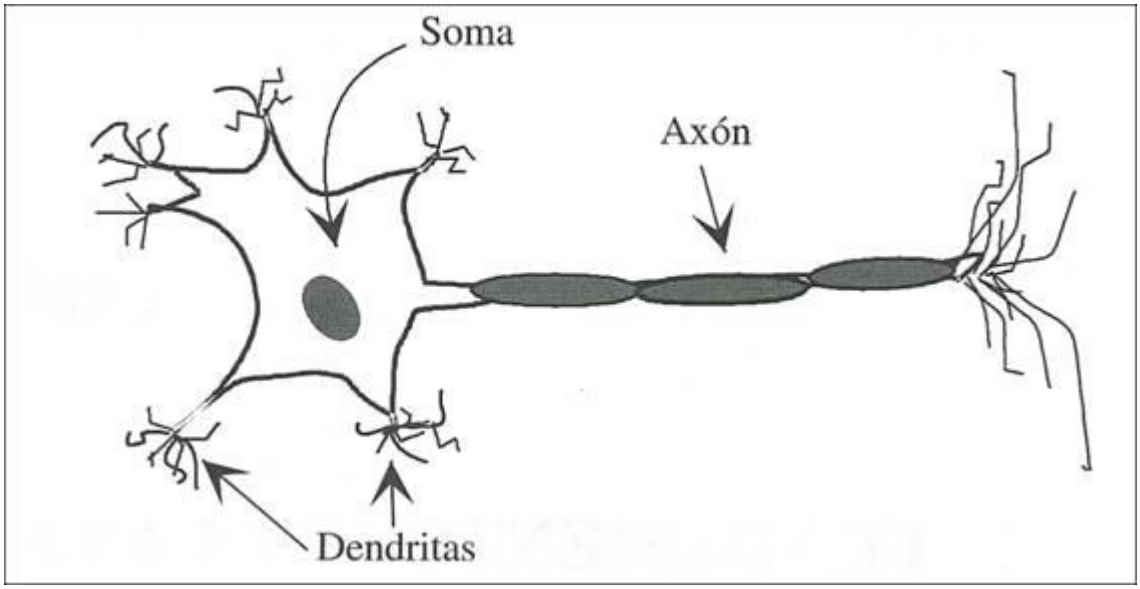


Figura2.1.2Estructura de una neurona biológica típica

Fuente:Rosentblant 2010 biologia artificial

Dendritas. Recogen señales procedentes de otras neuronas, o del mundo externo y las transmiten al cuerpo de la neurona.

Cuerpo. Recibe las señales de todas las dendritas y según la tipología de la neurona determina una respuesta global.

Axón. Transmite la respuesta global generada por el cuerpo de la neurona. La respuesta se transmite a otras neuronas o a algún miembro efector.

Sinapsis. Es el punto de unión entre dos neuronas(dendrita - axón), y por el cual el impulso transmitido por el axón se pasa a otras neuronas.

La dinámica de la red esta entonces determinada por este traspaso de impulsos entre las neuronas que la conforman.

Neurona artificial

En la actualidad existen dos conceptos bastante aceptados de neurona artificial (Bueno, 2014).

- Unidad de procesamiento encargada de simular el comportamiento de una neurona biológica.
- Modelo matemático que trata de emular de manera simplificada el comportamiento de una neurona biológica.

2.1.3 Red neuronal artificial

Es la interconexión de varias unidades de procesamiento (neuronas). Esta toma entradas, las procesa y emite salidas (Bueno, 2014).

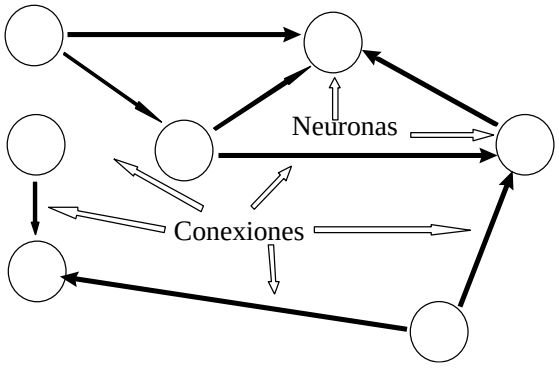


Figura 2.1.3. Red Neuronal Artificial

Fuente*:Pitts 2010,Red neuronal artificial*

Modelo de McCulloch-Pitts

Modelo de neurona binaria propuesto por McCulloch(1943), el cual calcula la suma ponderada de sus entradas producidas por otras unidades, y da como salida un uno (1) si ésta se encuentra por encima de un valor denominado umbral, o un cero (0) si está por debajo. La ecuación que gobierna el funcionamiento de dicho modelo de neurona es la siguiente:

$n_i(t+1)=f(\square_j(w_{ij}*n_j(t)) - u_i)$

donde,

w_{ij} es el peso de la conexión entre la neurona *j* y la neurona *i*

$n_j(t)$ es la salida producida por la neurona *j*

u_i es el umbral de la neurona *i*

$f(x) = 0$ si $x<u$

1 en otro caso

es la función umbral.

En la figura 4 se presenta esquemáticamente una neurona.

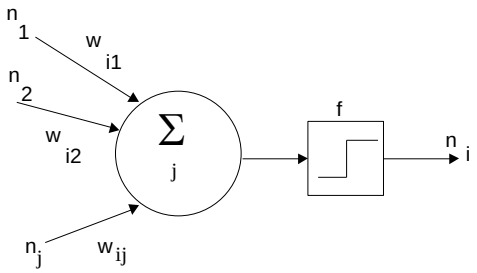


Figura 2.1.3.1. Modelo de neurona de McCulloch y Pitts Fuente*:Pitts 2010,Red neuronal artificial*

La función de umbral tiene la forma de la figura 5.

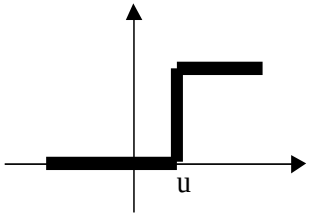


***Figura 2.1.3.2. Función umbra**l **Fuente**:Pitts 2010,Red neuronal artificial*

En la actualidad al modelo de neurona de McCulloch(1943)se le permite usar otro tipo de funciones de activación. Algunas de las mas usadas son presentadas en la figura 2.1.3.3.

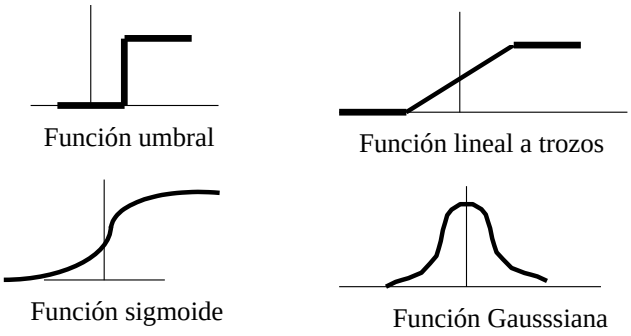


***Figura 2.1.3.3. Funciones de activación Fuente**:Pitts 2010,Red neuronal artificial*

2.1.3.1 EL PERCEPTRÓN

Es una red neuronal feedforward que consta de dos capas, una de entrada y una de salida. En sus inicios se consideró que podía resolver todo tipo de problemas, sin embargo, no es posible resolver el problema XOR con este modelo (Bueno, 2014).

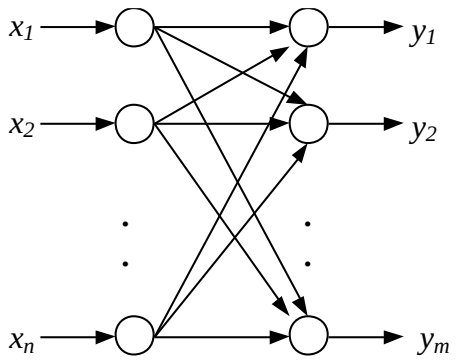


Figura 2.1.3.3. Perceptrón ***Fuente****:Pitts 2010,Red neuronal artificial*

2.1.3.2 Regla de aprendizaje perceptrón

1. Inicialización de los pesos y del umbral: Inicialmente se asignan valores aleatorios a cada uno de los pesos (w_i) de las conexiones y al umbral ($-w_0$=q).
2. Presentación de un nuevo patrón de entrenamiento.
3. Cálculo de la salida actual $y(t)=f(\ \square_i\ (\ w_i(t) * x_i(t)\ ) - q\ )$, siendo $f(x)$ la función de transferencia escalón.

4. Adaptación de los pesos:

$$w_i(t+1)= w_i(t)+ \square\ w_i(t)$$

donde□

$\square\ w_i(t)=2\square * x_i(t) *y(t)$ si $y(t)\square d(t)$

0 en caso contrario.

□ es la rata de aprendizaje que varía entre 0 y 1.

5. Repetir desde el paso 2.

2.1.4 Clasificación

Según la topología de la red se pueden clasificar en:

-Redes hacia delante (feedforward). Perceptrón simple, Perceptrón multicapa, redes con funciones de base radial.

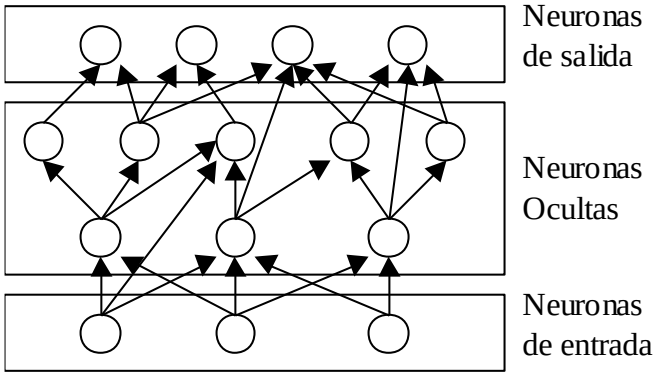


Figura 2.1.4Red Feed Fordward Fuente*:Pitts 2010,Red neuronal artificial*

-Redes recurrentes (feedback). Redes competitivas, red Kohonen, red Hopfield, modelos ART.

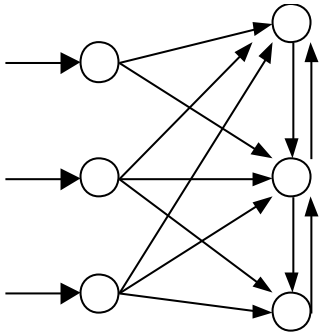


Figura 2.1.4.1Red Recurrente Fuente*:Pitts 2010,Red neuronal artificial*

2.1.5 Aprendizaje o entrenamiento

Tipos de aprendizaje:

Existen tres formas por las cuales una red neuronal puede aprender, supervisado, no supervisado y por refuerzo (Carbonell, 2000).

Aprendizaje supervisado. Este tipo de algoritmo utiliza un conjunto de ejemplos (entrada-salida) que la red debe aprender. El entrenamiento se lleva a cabo mediante un proceso iterativo de adaptación de la red neuronal de manera que sus salidas se acerquen cada vez más a las salidas deseadas especificadas en los ejemplos (Carbonell, 2000).

Aprendizaje no supervisado. En este tipo de algoritmos no se especifican salidas deseadas, el algoritmo trata de encontrar de manera autónoma regularidades en los datos de entrada creando categorías que agrupen a los datos más similares (Carbonell, 2000).

Aprendizaje con refuerzo. La red aprende a través de un proceso de ensayo y error que maximiza el índice de desempeño denominado señal de error.

Clasificación de algoritmos de aprendizaje

Tabla 1Tipos de aprendizaje
***Fuente**:Pitts 2010,Red neuronal artificiaL*

Tipo de aprendizaje	*Regla de Aprendizaje*	Algoritmo mas usado
Supervisado	Corrección de error, Boltzmann, Hebbian, Competitivo	Back Propagación Adaline y Madaline
No supervisado	Corrección de error, Hebbian, Competitivo	Aprendizaje con memoria asociativa
Híbrido	Corrección de error y competitivo	Algoritmo de aprendizaje RBF

2.2 Temática: Redes Neuronales y Algoritmos Genéticos

2.2.1 Computación Evolutiva

Según Fogel(1992), el término “computación evolutiva” o “algoritmos evolutivos” engloba una serie de técnicas inspiradas en los principios de la teoría Neo-Darwiniana de la evolución

natura. En términos generales, para simular el proceso evolutivo en una computadora se requiere:

• Codificar las estructuras que se replicarán (o sea, una estructura de datos que se utilice para almacenar a un “individuo”).

• Operaciones que afecten a los “individuos” (típicamente, se usa cruce y mutación).

• Una función de aptitud que nos indique qué tan buena es una solución con respecto a las demás.

• Un mecanismo de selección que implemente el principio de “supervivencia del más apto” de la teoría de Darwin.

2.2.2 Programación Evolutiva

Es una técnica denominada “Programación Evolutiva”, en la cual la inteligencia se ve como un comportamiento adaptativo. La Programación Evolutiva enfatiza los nexos de comportamiento entre padres e hijos, en vez de buscar emular operadores genéticos específicos como en el caso de los Algoritmos Genéticos (Fogel, 1992).

El algoritmo básico de la Programación Evolutiva es el siguiente:

• Generar aleatoriamente una población inicial.

• Se aplica mutación.

• Se calcula la aptitud de cada hijo y se usa un proceso de selección mediante torneo (normalmente estocástico) para determinar cuáles serán las soluciones que se retendrán.

La Programación Evolutiva es una abstracción de la evolución al nivel de las especies, por lo que no se requiere el uso de un operador de recombinación (diferentes especies no se pueden cruzar entre sí). Asimismo, usa selección probabilística (Fogel, 1992). Algunas aplicaciones de la Programación Evolutiva son:

• Predicción

• Generalización

• Juegos

• Control automático

• Problema del viajero

• Planeación de rutas

• Diseño y entrenamiento de redes neuronales

• Reconocimiento de patrones

2.2.3 Algoritmo Genético

L (Pajares, 2006)os algoritmos genéticos son una técnica de resolución de problemas inspirada en la naturaleza. Están basados en el principio darwiniano de reproducción y supervivencia de los individuos más aptos (Gomez, 2010).

2.2.3.1 Cromosoma

Los algoritmos genéticos trabajan manipulando cromosomas, que son estructuras que codifican las distintas soluciones de un determinado problema. La forma en que se codifican los cromosomas es dependiente de cada problema en particular , y suele variar de problema en problema .Un cromosoma esta compuesto por un conjunto de genes, cada gen representa la característica en particular en un individuo (Pajares, 2006).

Es importante que se destaque el genotipo y el fenotipo. El genotipo es el conjunto de genes de un individuo ,la descripción genética del individuo. El fenotipo es la forma en que se expresa el genotipo ,como resultado de la interacción con su entorno.

2.2.3.2 Evaluación y Aptitud

Función de evaluación indica que tan buena es la solución obtenida tras la aplicación de un conjunto de parámetros, independientes de la aplicación de esta función sobre otro conjunto de parámetros (Pajares, 2006).

Función de Aptitud siempre esta definida con respecto al resto de los individuos de la población .Indica que tan buena(o mala)es una solución comparada con el resto de las soluciones obtenidas hasta el momento (Carbonell, 2000).

En el Algoritmo genético canónico, la función de adaptación se define como:

$$Fi = \frac{f_i}{\bar{f}}$$

$$\bar{f} = \frac{\sum_{i=0}^{N} fi}{N}$$

2.2.3.3 Población

Los algoritmos genéticos trabajan sobre una población de individuos, donde cada uno de los cuales representa una posible solución .La población es un concepto muy impórtate en los algoritmos genéticos por que con la población es la donde itera el algoritmo hasta conseguir la solución optima (Fogel, 1992).

2.2.3.4 Selección

2.2.3.4.1 Selección por ruleta

El primer paso para la aplicación de este operador consiste en obtener aleatoriamente un numero R de la siguiente manera:

$$\boldsymbol{R = Random(0, \sum_{i=1}^{N} Fi)}$$

2.2.3.4.1 Selección proporcional

El número de copias de copias del individuo i que pasaran a la siguiente población se calcula en base a la siguiente ecuación:

$$Ci = N * \frac{fi}{\sum_{i=1}^{N} Fi}$$

2.2.3.4.2 Selección por Ranking

Esta selección asigna una probabilidad de selección proporcional a la aptitud del individuo. Se asigna a un individuo en una probabilidad de selección igual a:

$$pi = \frac{N - i}{\sum_{i=1}^{N} Fi}$$

2.2.3.4.3 Cruce

Este operador de cruza asigna aleatoriamente los pesos .Cada hijo tiene una probabilidad de 0.5 de recibir los genes de si padre y por ende recibirlos de su madre (M.Darell, 1994) .

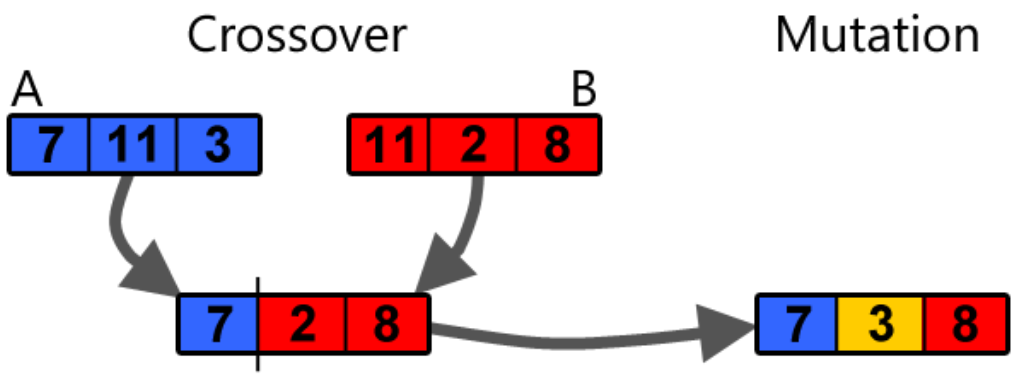


Figura 2.2.3.4.3 Cruce y Mutacion Fuente: *Neural Network Synthesis Using Cellular Encoding And The Genetic Algorithm*. M. Darell (1994)

Tendremos una tasa de cruce la cual será igual a 0.4 esto representa que cruzaremos mas del 40% de la nueva neurona

2.2.3.4.4 Mutación

En este caso la probabilidad de mutación permanece constante a lo largo de las distintas generaciones , como lo muestra la ecuación :

$$Pm(t) = Pm(0) = P_m^{max} = P_m^{min}$$

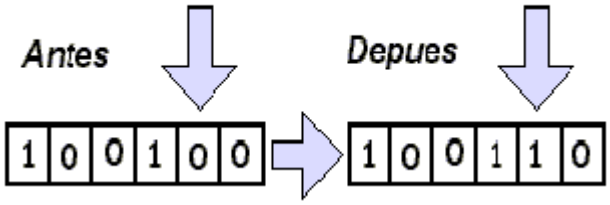


Figura 2.2.3.4.4Ejemplo Mutación fuente:Propia

Nuestra tasa de mutación será igual a 0.3 representa que el 30% será mutado

CAPÍTULO III DISEÑO METODOLÓGICO

3.1 INTRODUCCIÓN

En este capítulo se dará énfasis a la construcción y diseño del agente inteligente y el escenario de simulación 2D en el cual se desenvolverá el agente inteligente, de manera que pueda demostrarse o refutarse la hipótesis formulada en el capítulo I a su vez se apoyara en el capítulo II Marco Teórico en el cual se desarrollaron los criterios necesarios para el desarrollo de la investigación. Para esto se utilizará la metodología de prototipado o modelo de prototipos o metodología de desarrollo de prototipos, combinado con la metodología de agentes inteligentes Prometheus

3.1.1 Metodología de Prototipado

En esta metodología se tiene fases que se ajustan más que todo al diseño de las interfaces y diseño del escenario para construcción del prototipo (Marques, 1994). Las fases de la metodología son:

- Desarrollo del modelo: Análisis, Diseño y construcción, evaluación y modificación.
- Diseño y desarrollo final: Refinamiento y Evaluación
- Entrega: Producto, mantenimientos posteriores

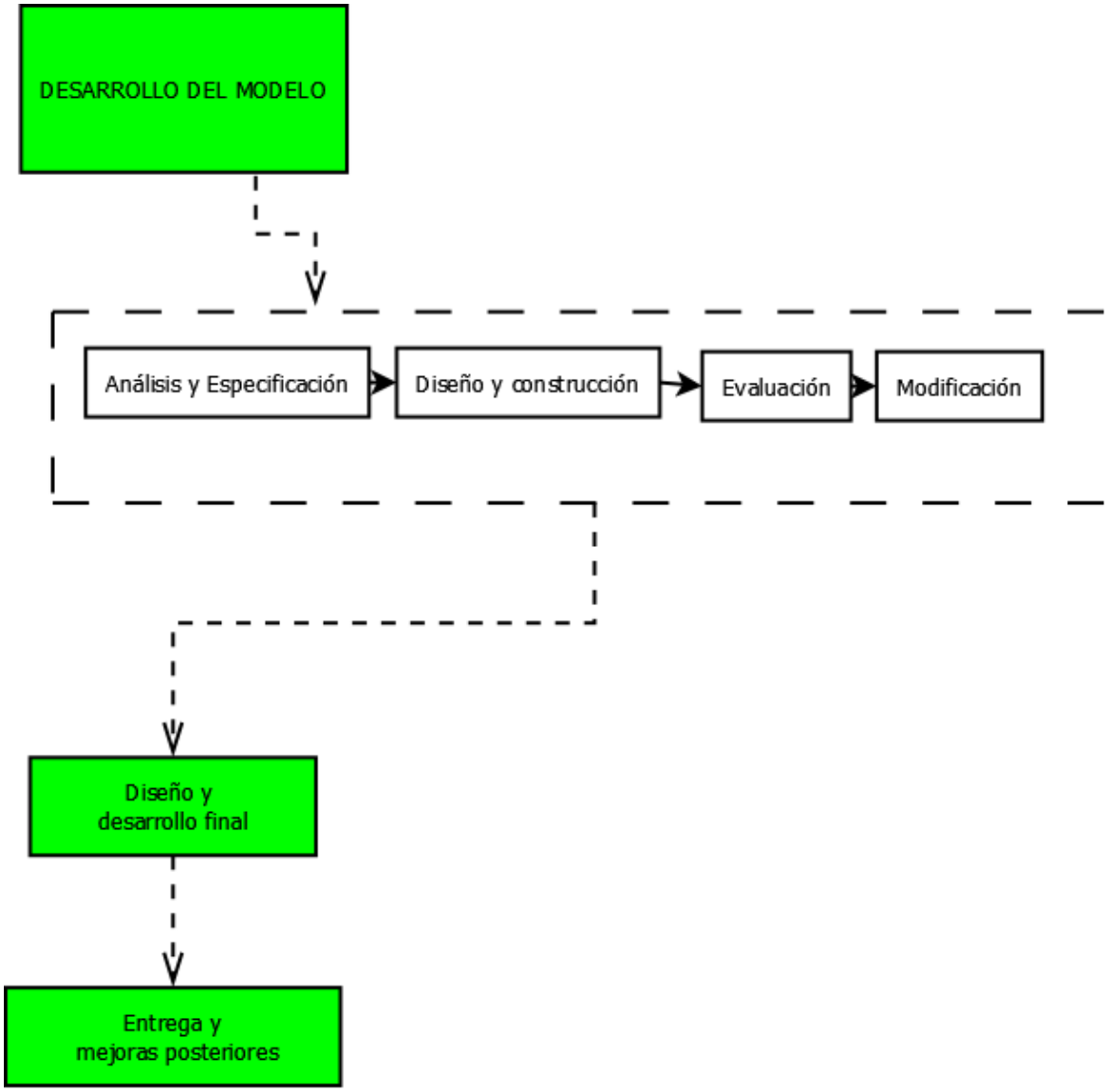


Figura 3.1.1 Flujo Metodología Prototipado Fuente :Propia

3.1.2 Metodología Prometheus.

Esta metodología se utilizará para el desarrollo del agente inteligente y entrenamiento evolutivo del agente inteligente. Las fases de esta metodología son: Especialización del sistema, diseño de la arquitectura y diseño detallado (SILVESTRE, 2012).

3.2 Metodología de Prototipado combinado con la Metodología Prometheus

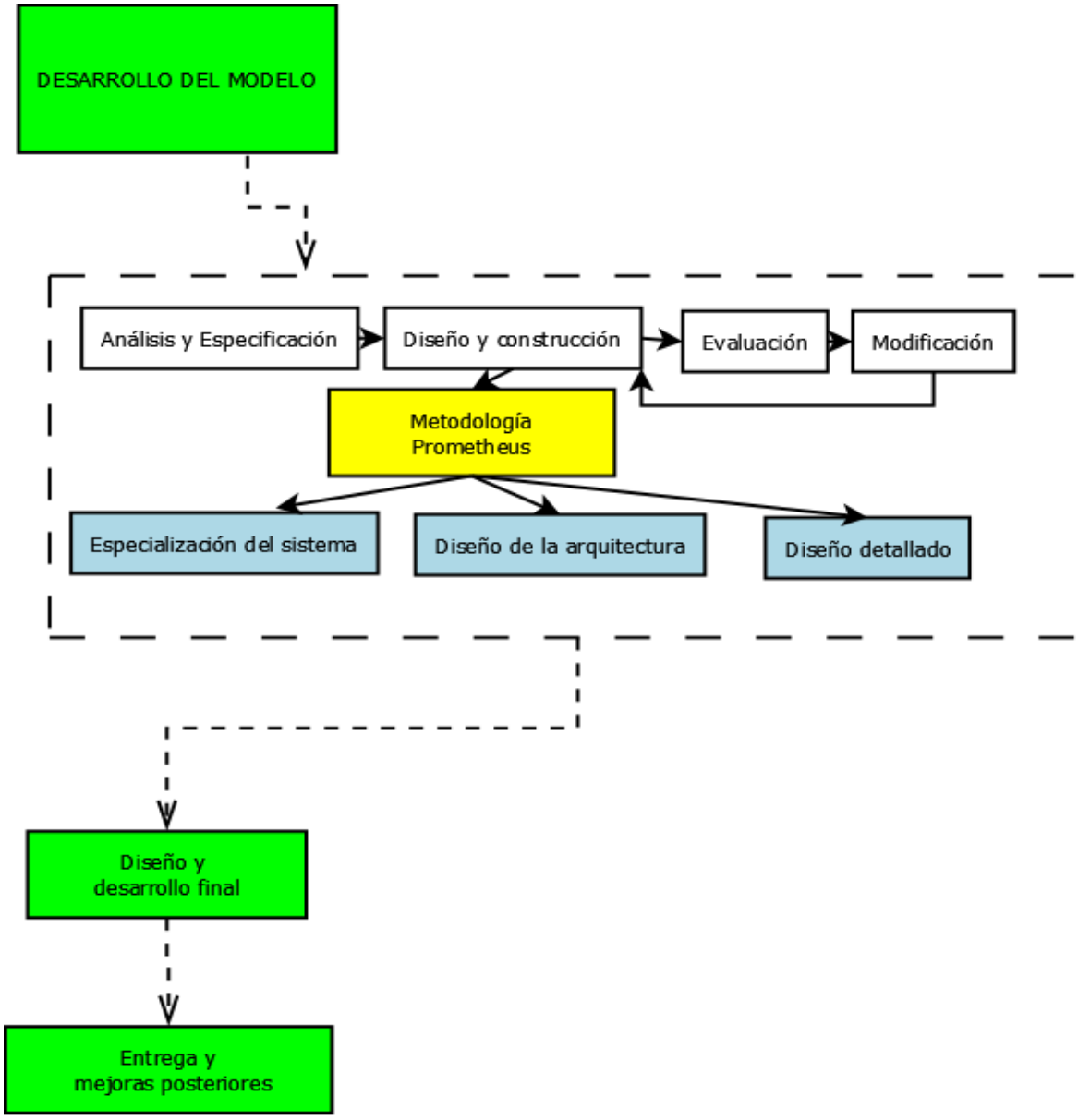


Figura 3.2 Metodología de Prototipado combinado con la Metodología Prometheus Fuente:Propia

3.2.1 Desarrollo Metodología Prototipado

3.2.1.1 Fase de Desarrollo de Modelo

3.2.1.1.1 Análisis y Especificación

Para realizar la comprobación o la refutación de la hipótesis se tomarán principalmente los resultados del prototipo de agente inteligente y su desenvolvimiento en su escenario.

Escenario 2D.- El escenario 2D se realizará a partir de un fondo de pantalla gris inicialmente se creará un suelo que tenga gravedad y fricción para que el agente inteligente pueda sostenerse sobre el a continuación unas capturas del escenario.

Personaje para implementar el Agente Inteligente. -En este personaje se implementará el agente inteligente para que pueda evolucionar su comportamiento en el escenario2D.

Opciones de la interface. -Tendrá en la pantalla de inicio la opción de iniciar el entrenamiento y salir del entrenamiento dentro del escenario se podrá verificar con una tabla los datos del agente inteligente evolutivo y también se podrá verificar la evolución y se podrá acelerar y desacelerar el entrenamiento.

Adaptación de las redes neuronales y algoritmo genético.-Se compilara un archivo dll el cual tendrá programando la red neuronal y el algoritmo genético y se podrá utilizar en el motor de videojuegos GodotEngine que tiene un lenguaje de programación combinado entre Python y JavaScript.

3.2.1.1.2 Diseño y construcción.

Dentro de esta etapa se hará también el desarrollo de la metodología Prometheus para la implementación del agente inteligente evolutivo con la red neuronal y el algoritmo genético.

3.2.2 Desarrollo de la metodología prometheus.-

Se va desarrollar sus fases que son la especialización del sistema, diseño de la arquitectura y diseño detallado.

3.2.2.1 Especialización del sistema

Actores y Elementos del sistema. -Para el desarrollo del agente inteligente según la metodología es necesario identificar los actores, pero ya que el agente se desenvolverá en un ambiente de simulación 2D también será necesario identificar los elementos del escenario. A continuación, se mostrará en la siguiente tabla los actores y elementos.

Actor ó Elemento	Características
Actor Agente Inteligente Personaje Evolutivo Inicial	Este se caracteriza de ser el agente inteligente que va a entrenar en el entorno y que va lograr desplazarse con naturalidad y sensación de evolución en el escenario. Cabe decir que estará conformado por la red neuronal
Actor Agente Inteligente Padre	Esta generación estará comprendida de varios personajes evolutivos pero cada

	personaje será independiente en su función de aceptación fitnes con el cual se decidirá si sera el padre de la siguiente generación por lo cual cada personaje actúa con una red neuronal y en base a esta si ninguno cumple la función de aceptación se desechara pero en base a su función de aceptación se construirá una nueva generación de personajes evolucionados al anterior generación
Actor Agente Inteligente mas apto	Este será el que tenga un resultado mas alto en su función fitnes el cual podrá ser parte del cruce de la siguiente generación
Elementos del Escenario	Los elementos serán un piso donde se simulara las fuerzas físicas que acompañarían en todo momento la simulación (gravedad,friccion,trayectoria) También tendrá un tabla de datos que mostrara en detalle la información del entrenamiento del agente evolutivo

Escenario. - En este proceso de la metodología de desarrollo para agentes inteligentes se detallarán los escenarios que participarán en el sistema, por lo cual se mostrara cada escenario

del sistema, con una nota que detallara cada escenario del sistema en los cuales el usuario final ingresara los datos al sistema.

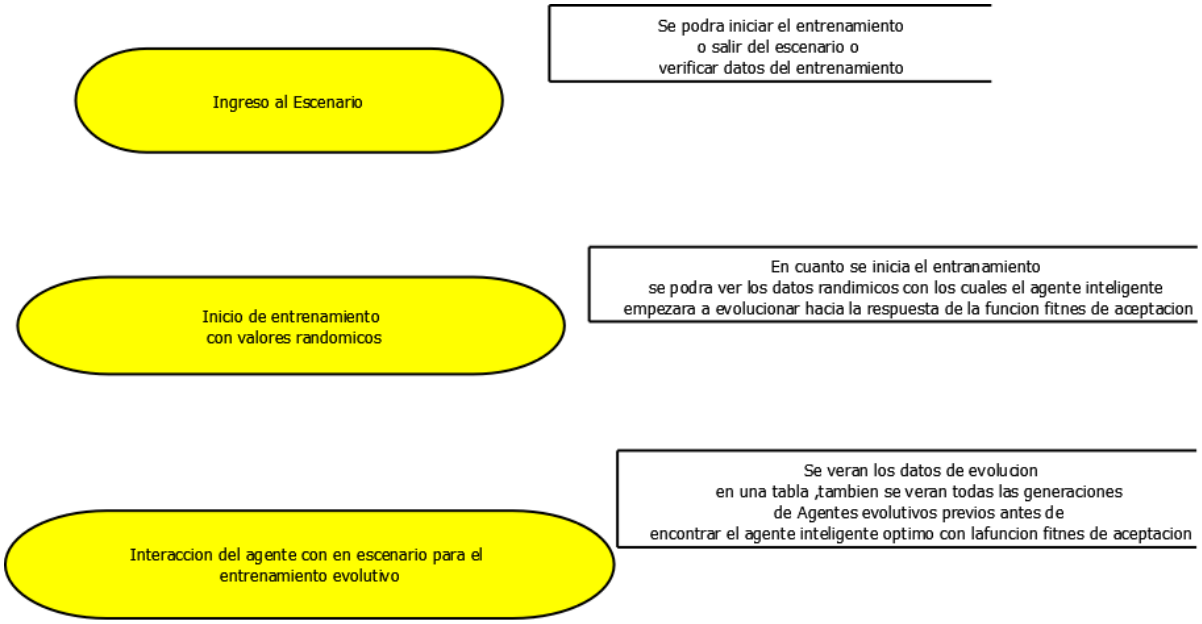


Figura 3.2.2.1 Diagrama de Escenarios Fuente: Propia

Definición de metas. - Estas metas y submetas que perseguirán los agentes inteligentes para poder comprobar o refutar la hipótesis planteada en esta investigación serán descritas en la siguiente tabla.

Tipo de meta	**Descripción**
Meta Principal	Lograr que el agente evolutivo iguale o supere por sí mismo evolucionando, la función fitnes de aceptación
Sub metas	• Poder entrenar a la red neuronal con un algoritmo genético

	• Implementar la red neuronal entrenada en el agente evolutivo para obtención de datos • Crear una nueva generación de personajes evolutivos cuando alguna se haya eliminado • Eliminar una generación cuando todos los Agentes evolutivos hayan fallado en su desplazamiento

3.2.2.2 Diseño y Arquitectura

En esta etapa se cubren tres aspectos: Determinar el tipo de agente en el escenario, Describir la interacción entre los agentes, Diseñar la estructura total del sistema.

Determinar el tipo de agente en el escenario

Dentro de el escenario ya identificamos a 3 agentes los cuales son: Agente inicial, Agente padres, Agente con mejor aptitud.

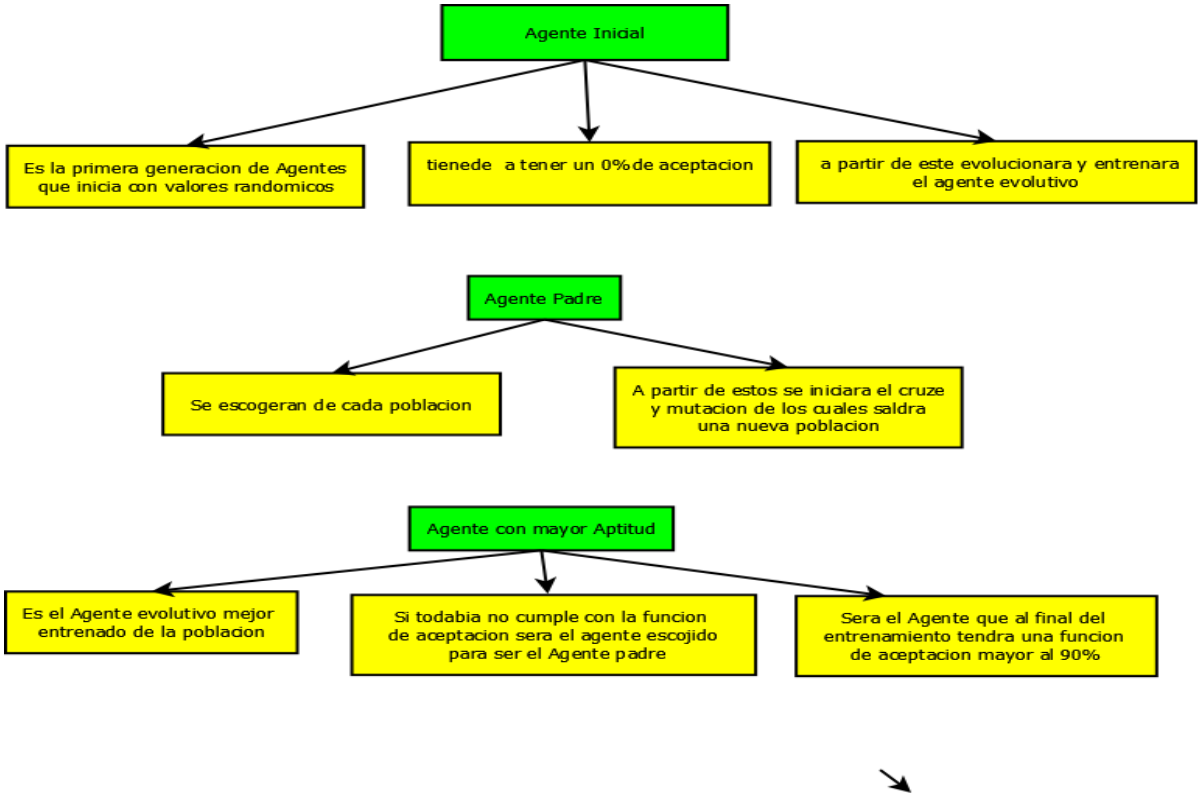


Figura 3.2.2.2 Diagrama Agente -rol Fuente: Elaboración Propia

3.2.2.3 Diseño detallado

3.2.2.3.1Algoritmo Genético para el entrenamiento de la red neuronal. -

La técnica evolutiva propuesta consiste en la adaptación del algoritmo genético. Se diseñara el algoritmo genético capaz de entrenar una red neuronal que lograra que el agente inteligente pueda desempeñarse evolutivamente en un escenario.

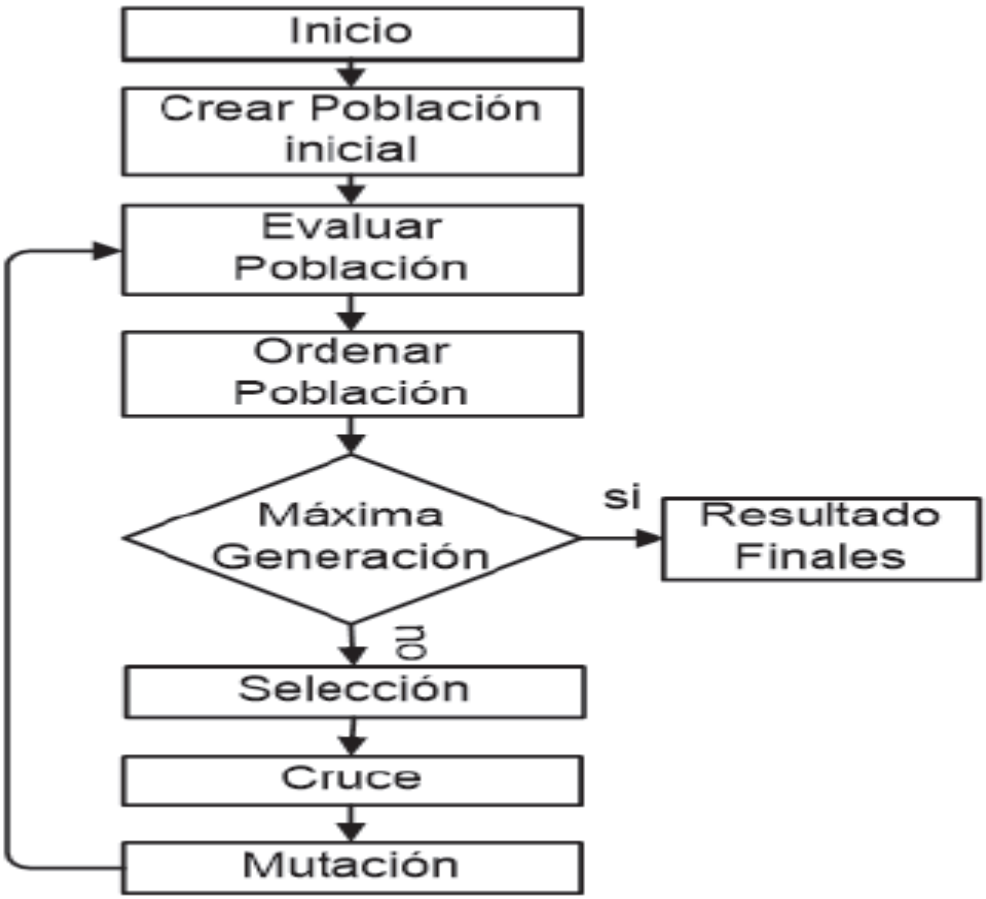


Figura 3.2.2.3.1 Diagrama de Flujo Algoritmo Genético Fuente: Elaboración Propia

3.2.2.3.2 Codificación del Cromosoma.

Cada uno de los diferentes individuos creados de manera aleatoria estará codificada por un vector que tendrá espacios binario , también estarán expresados en la red neuronal con n-entradas y 1 salida. Las entradas estarán definidas por la cantidad de movimientos que agá el agente se definirá la salida como cuanto de distancia recorrido, pero se representara con una probabilidad entre 0 y 1.

3.2.2.3.3 Genotipo y Fenotipo.

Su genotipo estará representado con sus atributos como el vector binario ,una dirección definida físicamente por el motor y la longitud de movimiento que hizo su fenotipo será una probabilidad según la longitud de movimiento

3.2.2.3.4 Evolución y Aptitud.

Como evaluaremos la aptitud será con la ecuación:

$$f_i = \frac{1}{e_i}$$

f_i será el valor de la evaluación, i el individuo actual,$e_i$es el error cuadrático de la red neuronal. Así se buscará favorecer al individuo que tenga menor error para su posterior reproducción.

Después de esto se calculará las aptitudes de los individuos con la siguiente ecuación:

$$F_i = 2 * \frac{f_{min}}{fi}$$

Donde F_i será la aptitud del individuo, f_i será el valor de la evaluación,f_{min} será el valor mínimo obtenido de las generaciones anteriores la actual de esta manera se le asignara mayor probabilidad a las redes neuronales que tengan mejor aptitud, así pueden mejorar en las siguientes generaciones

3.2.2.3.5 Selección por Ranking.

Para la selección realizaremos las siguientes operaciones:

$$P_i = \frac{N - i}{\sum_{i=0}^{N} i}$$

Luego se selecciona un R aleatorio de la siguiente manera:

$$R = Random\ (0, \sum_{j=1}^{N} Pi)$$

Por ultimo se elige al individuo tal que:

$$\sum_{i=1}^{j} Pi \leq R \leq \sum_{i=1}^{j+1} Pi$$

Este individuo pasa a la siguiente generación. El procedimiento se repite tantas veces como sea necesario hasta completar la siguiente generación

3.2.2.3.6 Operación de cruce.

La operación de cruce se hará directamente al fenotipo para poder realizar el algoritmo genético. En lugar de cruces individuales como seria con solo el algoritmo genético se realizará cruces de neuronas completas para poder seguir entrenando con el algoritmo genético.

3.2.2.3.7 Cruce Multipunto

Se trata de una generalización del cruce de 1 punto. En vez de cortar por un único punto los cromosomas de los padres como en el caso anterior se realizan dos cortes. Deberá tenerse en

cuenta que ninguno de estos puntos de corte coincida con el extremo de los cromosomas para garantizar que se originen tres segmentos. Para generar la descendencia se escoge el segmento central de uno de los padres y los segmentos laterales del otro padre.

Figura 3.2.2.3.7 Cruce Multipunto Fuente: Elaboración Propia

3.2.2.3.8 Mutación.

La mutación de un individuo provoca que alguno de sus genes, generalmente uno sólo, varíe su valor de forma aleatoria. Aunque se pueden seleccionar los individuos directamente de la población actual y mutarlos antes de introducirlos en la nueva población, la mutación se suele utilizar de manera conjunta con el operador de cruce. Primeramente, se seleccionan dos individuos de la población para realizar el cruce. Si el cruce tiene éxito entonces uno de los descendientes, o ambos, se muta con cierta probabilidad Pm. Se imita de esta manera el comportamiento que se da en la naturaleza, pues cuando se genera la descendencia siempre se

produce algún tipo de error, por lo general sin mayor trascendencia, en el paso de la carga genética de padres a hijos.

La probabilidad de mutación es muy baja, generalmente menor al 1%. Esto se debe sobre todo a que los individuos suelen tener un ajuste menor después de mutados. Sin embargo se realizan mutaciones para garantizar que ningún punto del espacio de búsqueda tenga una probabilidad nula de ser examinado.

Tal y como se ha comentado, la mutación más usual es el reemplazo aleatorio. Este consiste en variar aleatoriamente un gen de un cromosoma. Si se trabaja con codificaciones binarias consistirá simplemente en negar un bit. También es posible realizar la mutación intercambiando los valores de dos alelos del cromosoma. Con otro tipo de codificaciones no binarias existen otras opciones:

- Incrementar o decrementar a un gen una pequeña cantidad generada aleatoriamente.
- Multiplicar un gen por un valor aleatorio próximo a 1.

Aunque no es lo más común, existen implementaciones de Algoritmos Genéticos en las que no todos los individuos tienen los cromosomas de la misma longitud. Esto implica que no todos ellos codifican el mismo conjunto de variables. En este caso existen mutaciones adicionales como puede ser añadir un nuevo gen o eliminar uno ya existente.

Figura 3.2.2.3.8 Mutacion por complemento Fuente:Elaboracion Propia

3.3 Diseño y desarrollo final

3.3.1 Interface de inicio.-

La interface de inicio se muestra a continuación.

Figura 3.3.1 Interface de Inicio Fuente:Elaboracion Propia

3.3.2 Escenarios 2D.-

El escenario 2D se realizara a partir de un fondo de pantalla gris inicialmente se creara un suelo que tenga gravedad y fricción para que el agente inteligente pueda sostenerse sobre el a continuación unas capturas del escenario.

Figura 3.3.2 Plataforma 2D para desplazamiento del agente inteligente pista de conducir

Fuente:Elaboracion Propia

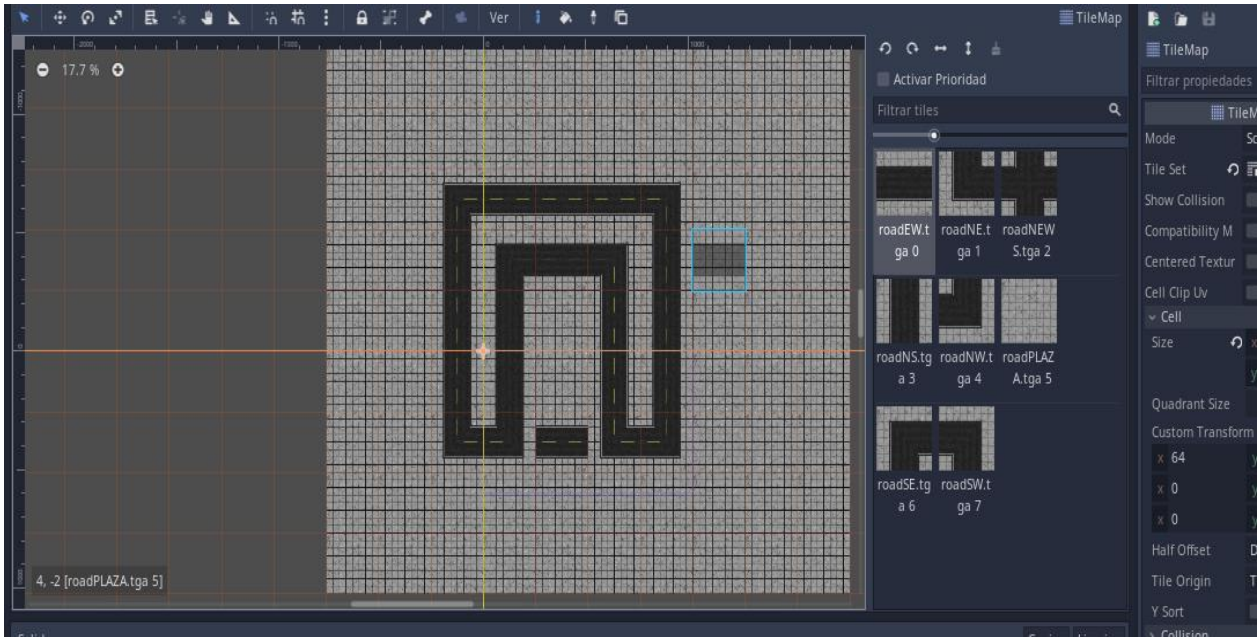


Figura 3.3.2.1 TileMaps Plataforma 2D para desplazamiento del agente inteligente pista de conducir Fuente:Elaboracion Propia

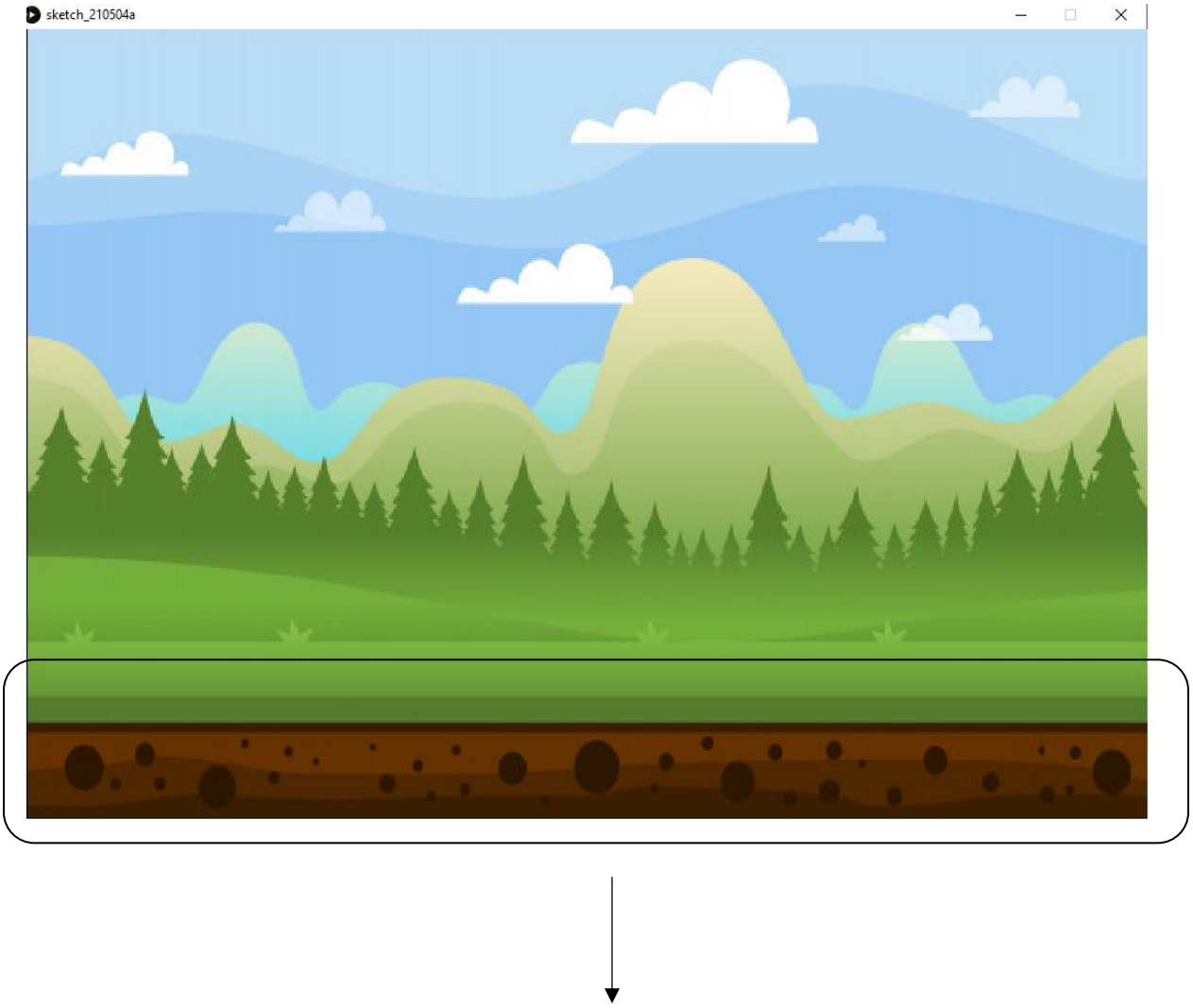


Figura 3.3.2.2 Plataforma 2D para desplazamiento del agente inteligente pista de conducir Fuente:Elaboracion Propia

Esta parte contiene un Static Body para simular el suelo y nos ayuda que podemos poner la fricción directamente con un suavizado mas realista directamente sin programar una fricción del ambiente

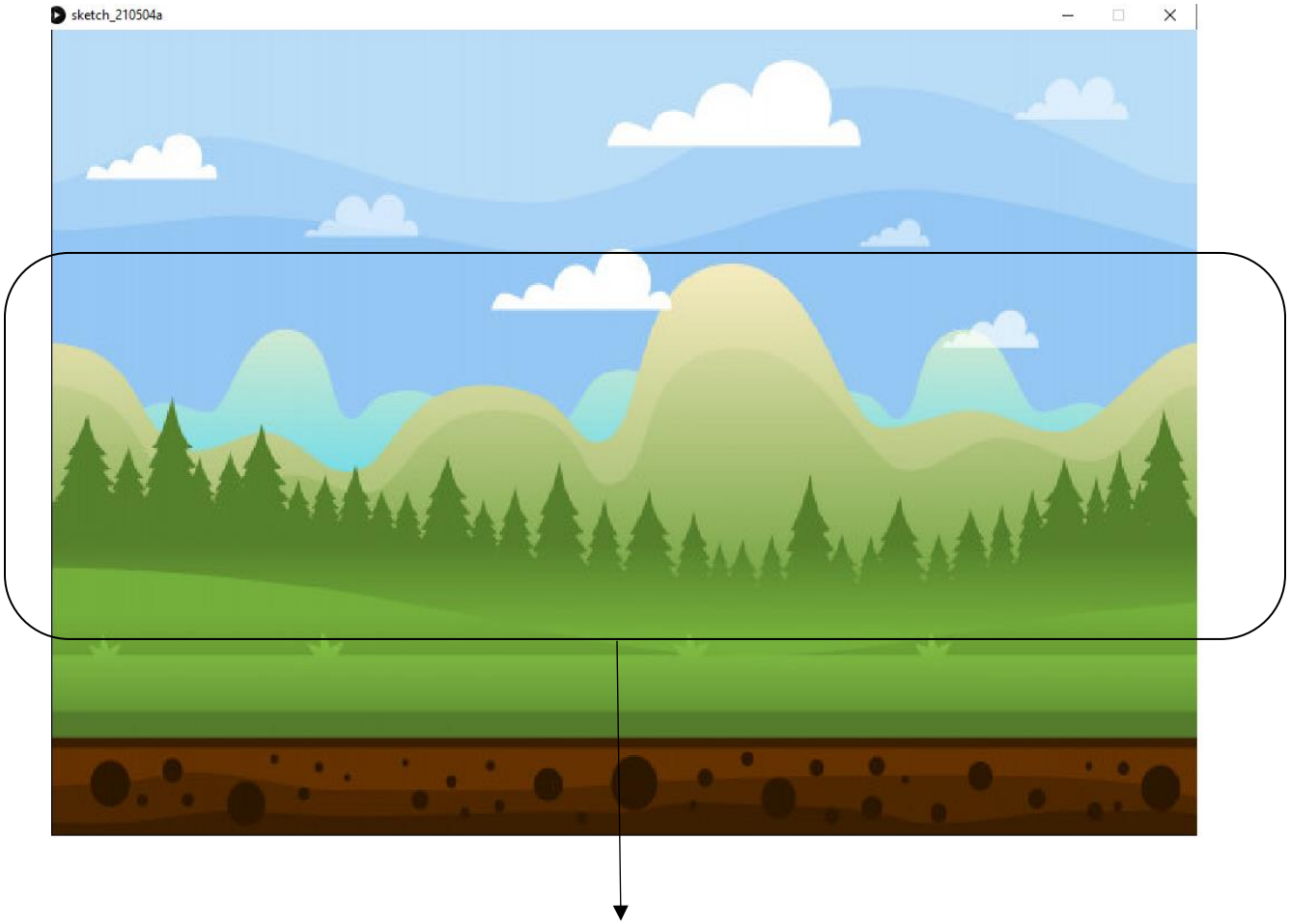


Figura 3.3.2.3 Plataforma 2D para desplazamiento del agente inteligente pista de conducir

Fuente:Elaboracion Propia

Las montañas y el paisaje están hechos en 2D y se irán repitiendo como un carrusel por que esta hecho a partir de un dibujo estático como Sprite y repetidos con un paralax para el efecto carrusel.

3.3.3 Diseño de la apariencia del agente.-

El agente inteligente será un diseño que básicamente tendrá cabeza brazos y piernas.

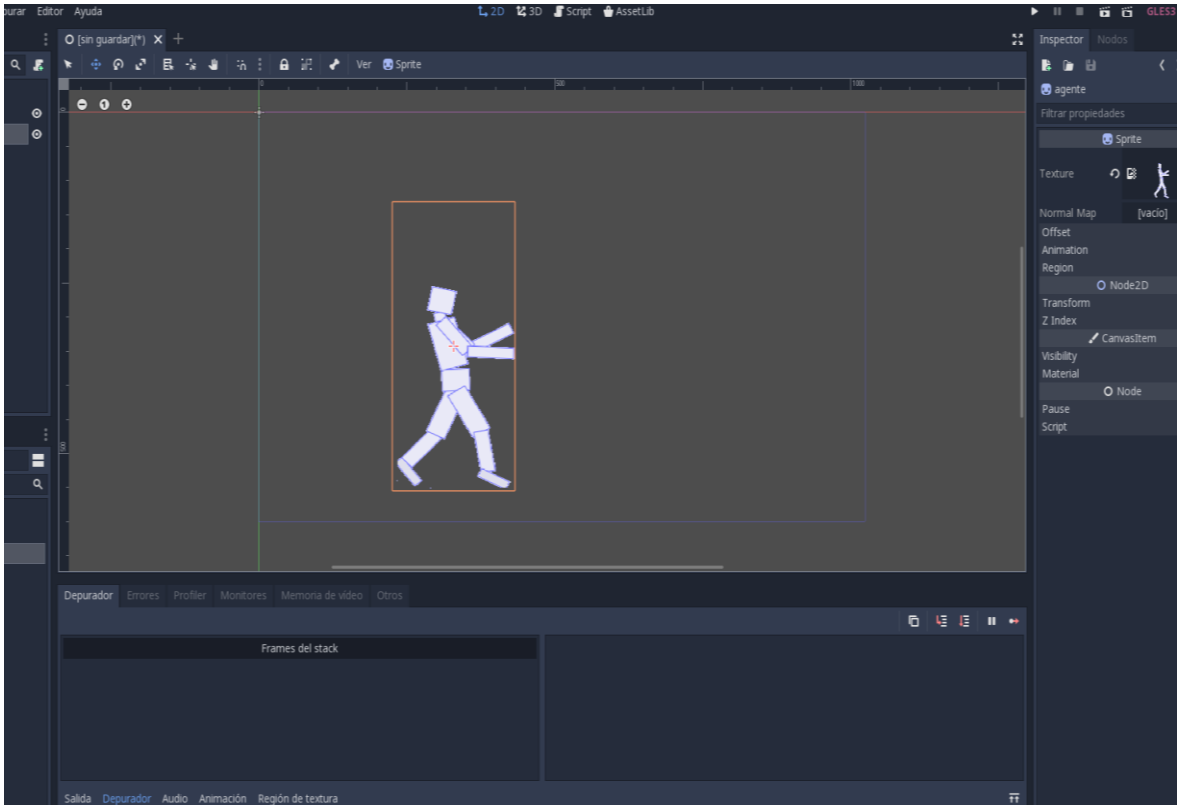


Figura 3.3.3 agente inteligente que se desplazara en el escenario Fuente:Elaboracion Propia

El agente inteligente aplicado a un auto en 2D visto desde arriba.

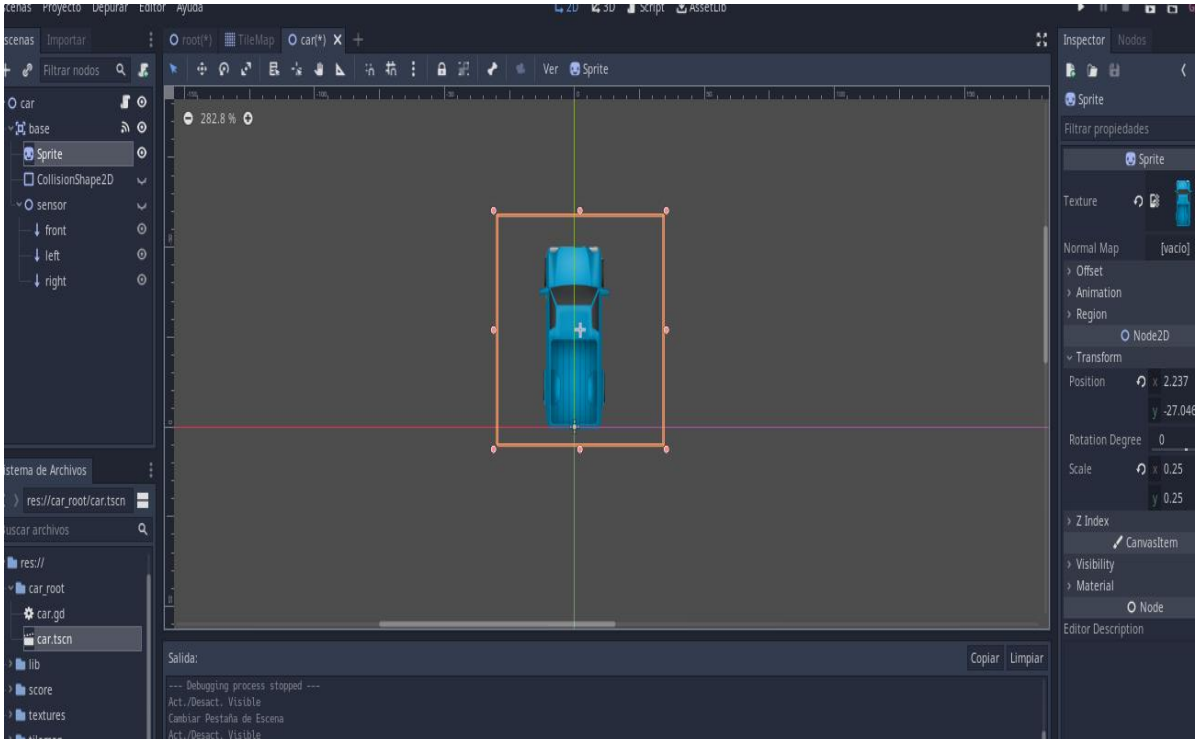


Figura 3.3.3.1 agente inteligente auto que se desplazara en el escenario Fuente: ElaboraciónPropia

Diseño y animaciones en Godot 3.2 agente inteligente

3.4 Entrega de Datos

3.4.1 Tabla para mostrar estadísticas. se muestra la mutación en porcentaje.-

Tabla 2 Estadísticas a mostrar en la interface
Fuente: Elaboración Propia

Probabilidad de mutación genética(Mut.rate)	%5
Campeones para el cruce	2
Tipo de Cruce	Multipunto
Velocidad de la simulación(Speed)	20 fps
Variar velocidad	J-K-L
Generación	Sera la generación actual de la población

3.4.2 Ejecución del agente inteligente en escenario 2D.-

Se ejecutará en un ordenador con procesador i3,8GB de ram y 500 GB de disco duro.

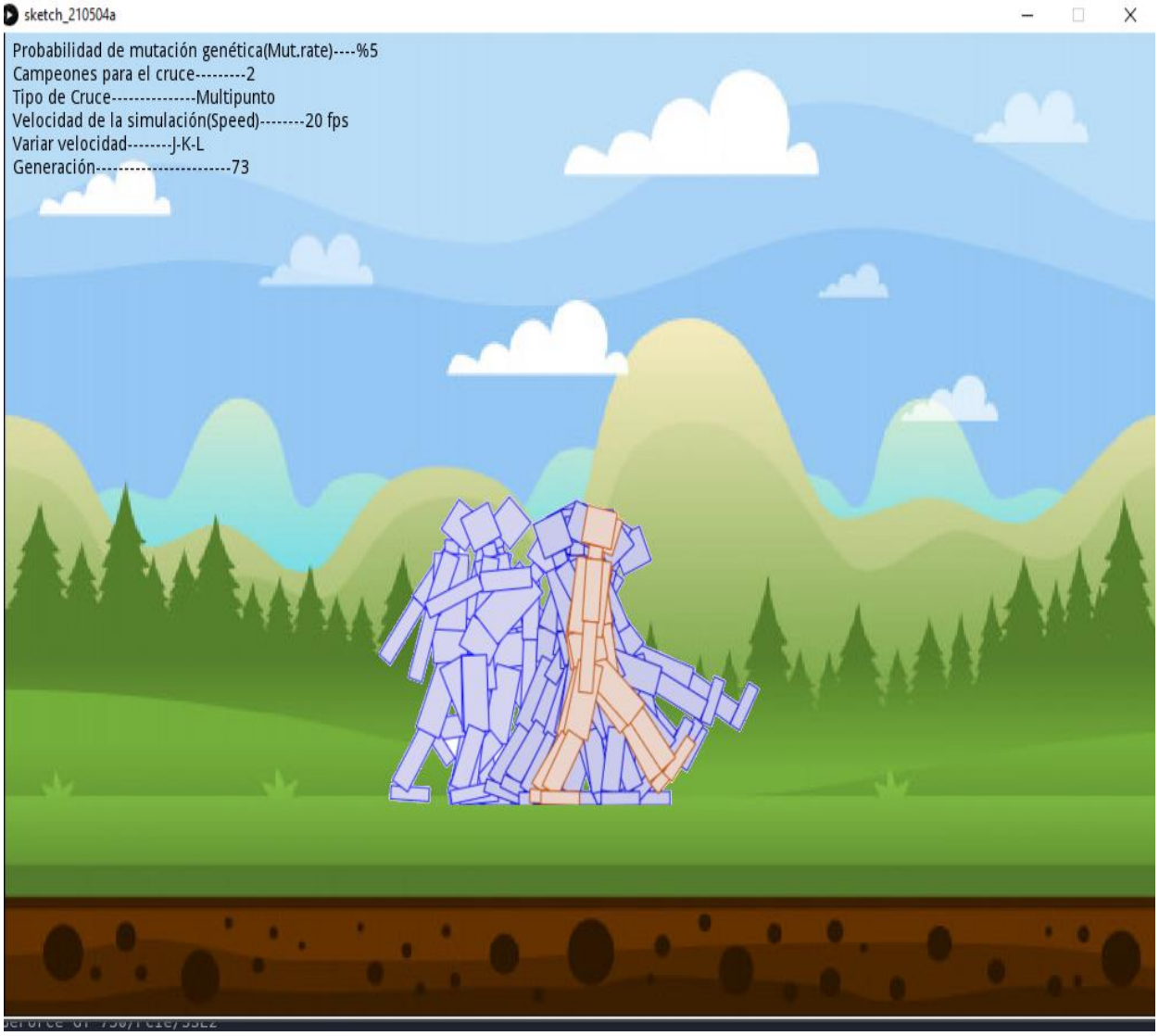


Figura 3.4.2 agente inteligente desplazándose en la plataforma 2D Fuente:Elaboracion Propia

Los 2 de color naranja son los agentes con mas Ranking después de 73 generaciones

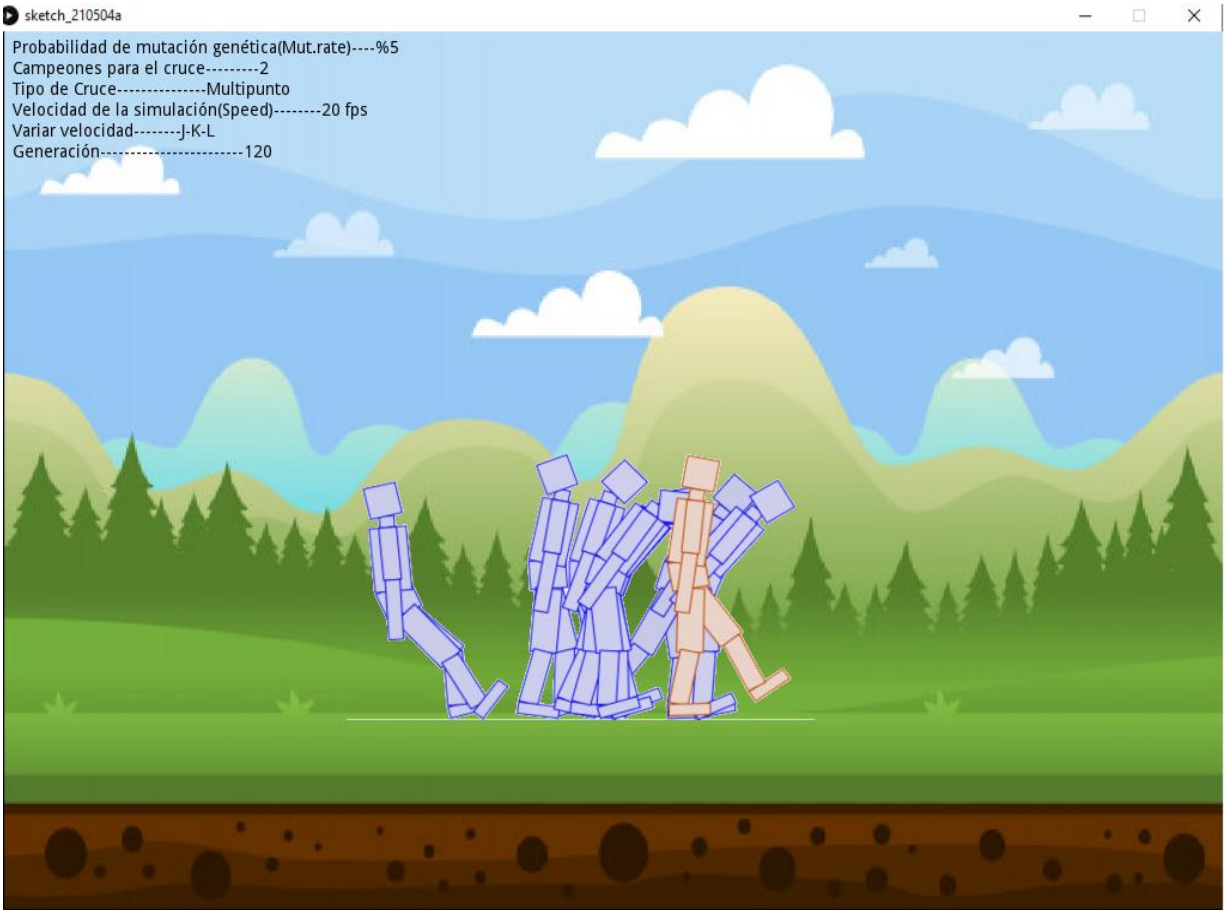


Figura 3.4.2.1 agente inteligente desplazándose en la plataforma 2D Fuente:Elaboracion Propia

Los agentes después de 120 generaciones máxima velocidad en fps que es igual 20 y una mutacion del 5%

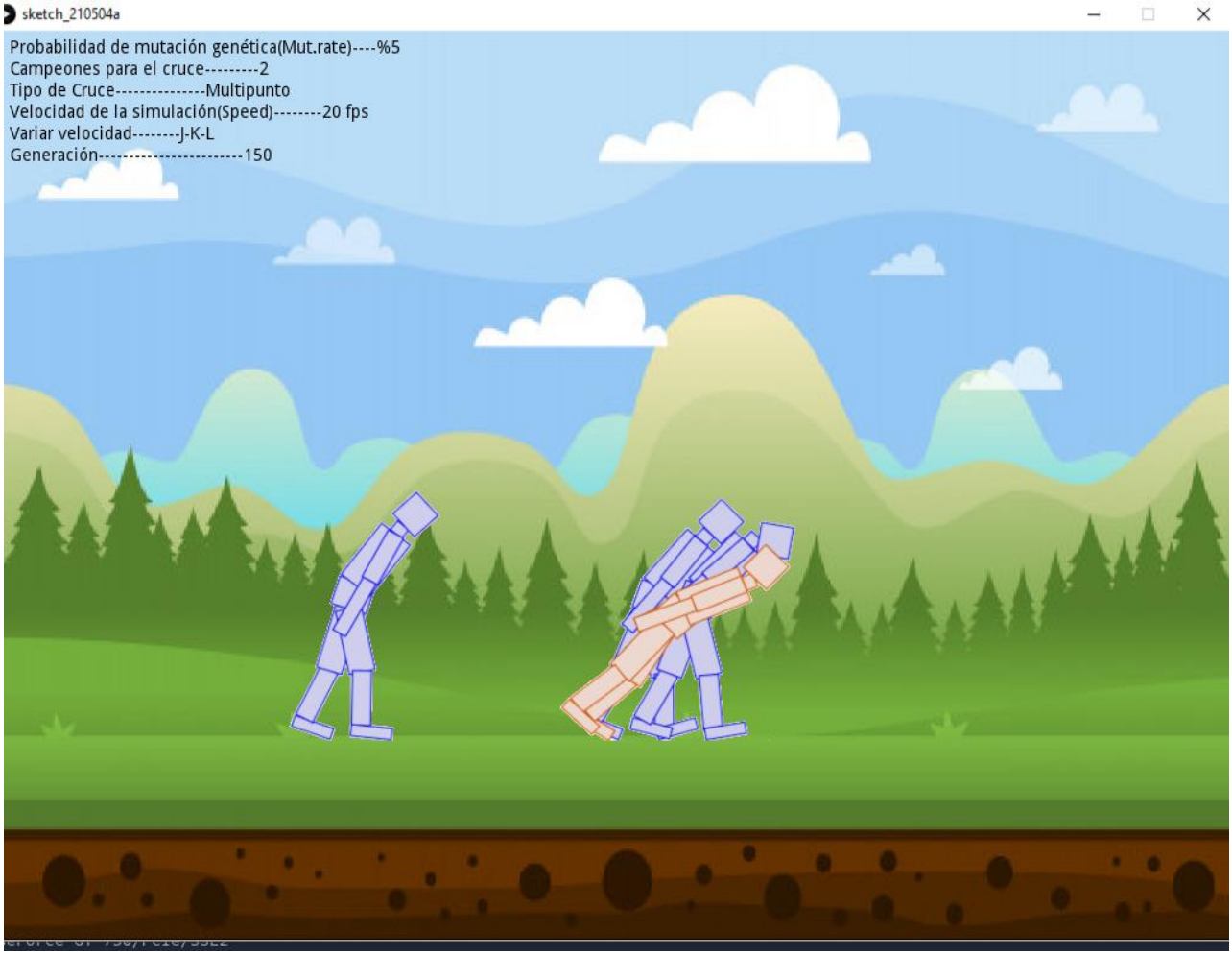


Figura 3.4.2.2 agente inteligente desplazándose en la plataforma 2D Fuente:Elaboracion Propia

Los agentes después de 150 generaciones máxima velocidad en fps que es igual 20 y una mutación del 5%

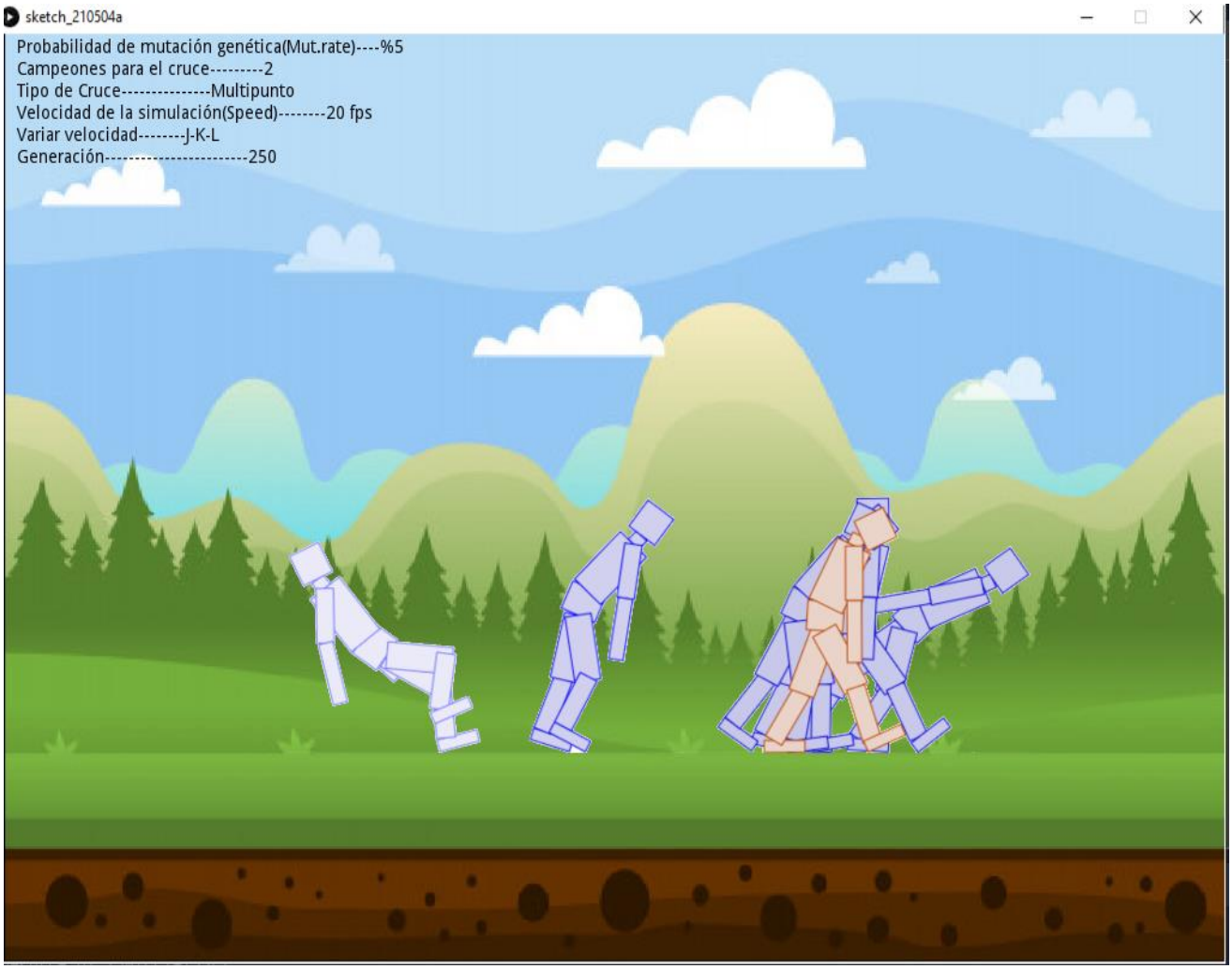


Figura 3.4.2.3 agente inteligente desplazándose en la plataforma 2D Fuente: Elaboración Propia

Los agentes después de 150 generaciones máxima velocidad en fps que es igual 20 y una mutación del 5%

Figura 3.4.2.4 agente inteligente auto desplazándose en la plataforma 2D Fuente: Elaboración Propia

Figura 3.4.2.4 agente inteligente auto desplazándose en la plataforma 2D Fuente:ElaboraciónPropia

Figura 3.4.2.4 agente inteligente auto desplazándose en la plataforma 2D Fuente:ElaboraciónPropia

Figura 3.4.2.5 agente inteligente auto desplazándose en la plataforma 2D Fuente:ElaboraciónPropia

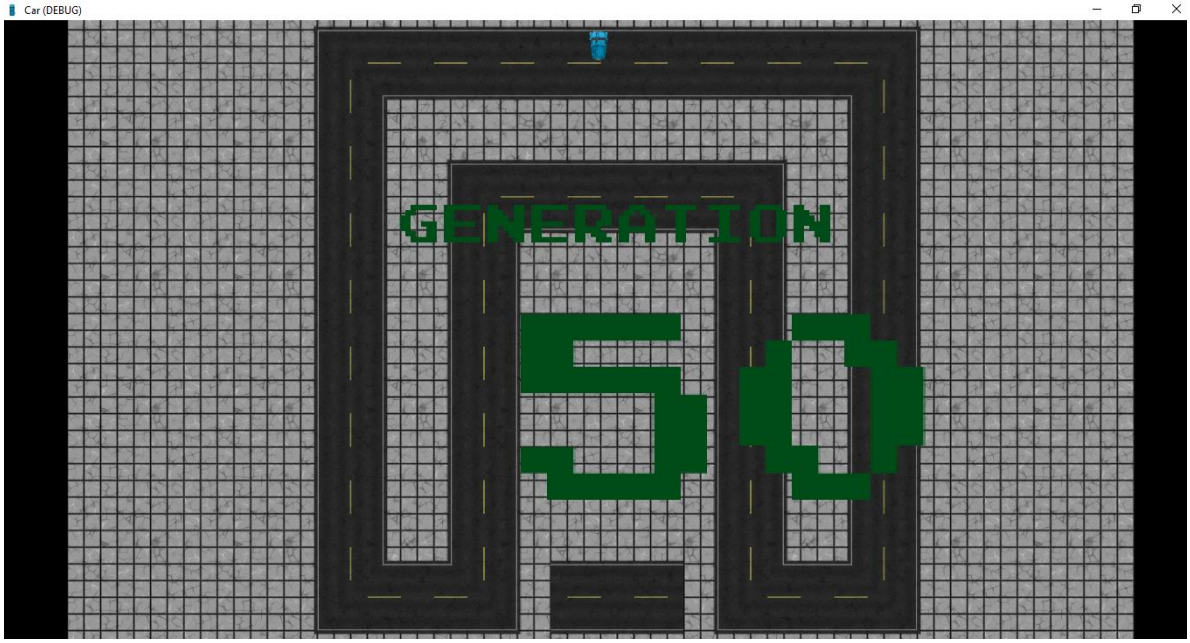


Figura 3.4.2.6 agente inteligente auto desplazándose en la plataforma 2D Fuente:ElaboraciónPropia

CAPÍTULO IV EVALUACIÓN DE RESULTADOS

4.1 RECOLECCIÓN DE DATOS.

En la siguiente tabla se mostrará los datos por generaciones.

Las columnas más importantes la Generación y el porcentaje de aceptación. Tomar en cuenta que el dato de cada generación es el del agente con aceptación más alta, el más apto

Tabla 3 .muestra las generaciones de 1 -35

Generación	Tipo de Cruce	Mutación	Velocidad	Porcentaje de Aceptación
1	Multipunto	0.05	20	8
2	Multipunto	0.05	20	5
3	Multipunto	0.05	20	9
4	Multipunto	0.05	20	9
5	Multipunto	0.05	20	6
6	Multipunto	0.05	20	7
7	Multipunto	0.05	20	7
8	Multipunto	0.05	20	8
9	Multipunto	0.05	20	9
10	Multipunto	0.05	20	6

11	Multipunto	0.05	20	6
12	Multipunto	0.05	20	6
13	Multipunto	0.05	20	6
14	Multipunto	0.05	20	8
15	Multipunto	0.05	20	9
16	Multipunto	0.05	20	5
17	Multipunto	0.05	20	7
18	Multipunto	0.05	20	8
19	Multipunto	0.05	20	8
20	Multipunto	0.05	20	5
21	Multipunto	0.05	20	7
22	Multipunto	0.05	20	9
23	Multipunto	0.05	20	9
24	Multipunto	0.05	20	8
25	Multipunto	0.05	20	7
26	Multipunto	0.05	20	9
27	Multipunto	0.05	20	9
28	Multipunto	0.05	20	8
29	Multipunto	0.05	20	6
30	Multipunto	0.05	20	8
31	Multipunto	0.05	20	9
32	Multipunto	0.05	20	5

33	Multipunto	0.05	20	8
34	Multipunto	0.05	20	9
35	Multipunto	0.05	20	9

Después de 35 generaciones de padres todavía no se llega al valor esperado de más de 90% de aceptación para el agente inteligente para esto se espera más generaciones.

Tabla 4 Muestra las generaciones del 101-135 Fuente: Propia

Generación	Tipo de Cruce	Mutación	Velocidad	Porcentaje de Aceptación
101	Multipunto	0.05	20	16
102	Multipunto	0.05	20	24
103	Multipunto	0.05	20	26
104	Multipunto	0.05	20	23
105	Multipunto	0.05	20	17
106	Multipunto	0.05	20	18
107	Multipunto	0.05	20	30
108	Multipunto	0.05	20	21
109	Multipunto	0.05	20	21
110	Multipunto	0.05	20	16
111	Multipunto	0.05	20	28
112	Multipunto	0.05	20	19
113	Multipunto	0.05	20	17

114	Multipunto	0.05	20	30
115	Multipunto	0.05	20	17
116	Multipunto	0.05	20	27
117	Multipunto	0.05	20	28
118	Multipunto	0.05	20	18
119	Multipunto	0.05	20	19
120	Multipunto	0.05	20	18
121	Multipunto	0.05	20	17
122	Multipunto	0.05	20	15
123	Multipunto	0.05	20	29
124	Multipunto	0.05	20	24
125	Multipunto	0.05	20	28
126	Multipunto	0.05	20	24
127	Multipunto	0.05	20	30
128	Multipunto	0.05	20	24
129	Multipunto	0.05	20	17
130	Multipunto	0.05	20	28
131	Multipunto	0.05	20	23
132	Multipunto	0.05	20	15
133	Multipunto	0.05	20	28
134	Multipunto	0.05	20	18
135	Multipunto	0.05	20	29

Después de 135 generaciones de padres todavía no se llega al valor esperado de más de 90% de aceptación para el agente inteligente, pero se ve y se percibe que ahora tiene porcentajes más cercanos al requerido.

Tabla 5. muestra las generaciones del 200-234

Generación	Tipo de Cruce	Mutación	Velocidad	Porcentaje de Aceptación
200	Multipunto	0.05	20	41
201	Multipunto	0.05	20	41
202	Multipunto	0.05	20	44
203	Multipunto	0.05	20	42
204	Multipunto	0.05	20	41
205	Multipunto	0.05	20	40
206	Multipunto	0.05	20	40
207	Multipunto	0.05	20	43
208	Multipunto	0.05	20	40
209	Multipunto	0.05	20	48
210	Multipunto	0.05	20	45
211	Multipunto	0.05	20	40
212	Multipunto	0.05	20	42
213	Multipunto	0.05	20	40
214	Multipunto	0.05	20	45
215	Multipunto	0.05	20	45

216	Multipunto	0.05	20	44
217	Multipunto	0.05	20	40
218	Multipunto	0.05	20	42
219	Multipunto	0.05	20	46
220	Multipunto	0.05	20	44
221	Multipunto	0.05	20	44
222	Multipunto	0.05	20	45
223	Multipunto	0.05	20	44
224	Multipunto	0.05	20	48
225	Multipunto	0.05	20	49
226	Multipunto	0.05	20	42
227	Multipunto	0.05	20	47
228	Multipunto	0.05	20	42
229	Multipunto	0.05	20	47
230	Multipunto	0.05	20	44
231	Multipunto	0.05	20	41
232	Multipunto	0.05	20	47
233	Multipunto	0.05	20	46
234	Multipunto	0.05	20	44

Después de 234 generaciones de padres todavía no se llega al valor esperado de más de 90% de aceptación para el agente inteligente, pero se ve y se percibe que ahora tiene porcentajes más cercanos al requerido con el mas alto de 48%.

Tabla 6.muestra las generaciones 304-338

Generación	Tipo de Cruce	Mutación	Velocidad	Porcentaje de Aceptación
304	Multipunto	0.05	20	50
305	Multipunto	0.05	20	66
306	Multipunto	0.05	20	64
307	Multipunto	0.05	20	55
308	Multipunto	0.05	20	67
309	Multipunto	0.05	20	66
310	Multipunto	0.05	20	51
311	Multipunto	0.05	20	60
312	Multipunto	0.05	20	52
313	Multipunto	0.05	20	56
314	Multipunto	0.05	20	56
315	Multipunto	0.05	20	63
316	Multipunto	0.05	20	51
317	Multipunto	0.05	20	57
318	Multipunto	0.05	20	59
319	Multipunto	0.05	20	64
320	Multipunto	0.05	20	54
321	Multipunto	0.05	20	54
322	Multipunto	0.05	20	54
323	Multipunto	0.05	20	54

324	Multipunto	0.05	20	61
325	Multipunto	0.05	20	65
326	Multipunto	0.05	20	64
327	Multipunto	0.05	20	50
328	Multipunto	0.05	20	61
329	Multipunto	0.05	20	52
330	Multipunto	0.05	20	64
331	Multipunto	0.05	20	61
332	Multipunto	0.05	20	62
333	Multipunto	0.05	20	62
334	Multipunto	0.05	20	53
335	Multipunto	0.05	20	68
336	Multipunto	0.05	20	60
337	Multipunto	0.05	20	50
338	Multipunto	0.05	20	59

Después de 238 generaciones de padres todavía no se llega al valor esperado de más de 90% de aceptación para el agente inteligente, se percibe que ahora tiene porcentajes más cercanos al requerido con el mas alto de 67%.

Tabla 7. muestra las generaciones 381-415
Fuente:Elaboracion Propia

Generación	Tipo de Cruce	Mutación	Velocidad	Porcentaje de Aceptación
381	Multipunto	0.05	20	71
382	Multipunto	0.05	20	73
383	Multipunto	0.05	20	71
384	Multipunto	0.05	20	74
385	Multipunto	0.05	20	70
386	Multipunto	0.05	20	76
387	Multipunto	0.05	20	74
388	Multipunto	0.05	20	77
389	Multipunto	0.05	20	77
390	Multipunto	0.05	20	70
391	Multipunto	0.05	20	77
392	Multipunto	0.05	20	73
393	Multipunto	0.05	20	70
394	Multipunto	0.05	20	78
395	Multipunto	0.05	20	70
396	Multipunto	0.05	20	77
397	Multipunto	0.05	20	75
398	Multipunto	0.05	20	70
399	Multipunto	0.05	20	75

400	Multipunto	0.05	20	72
401	Multipunto	0.05	20	69
402	Multipunto	0.05	20	70
403	Multipunto	0.05	20	77
404	Multipunto	0.05	20	78
405	Multipunto	0.05	20	71
406	Multipunto	0.05	20	71
407	Multipunto	0.05	20	73
408	Multipunto	0.05	20	73
409	Multipunto	0.05	20	71
410	Multipunto	0.05	20	69
411	Multipunto	0.05	20	74
412	Multipunto	0.05	20	72
413	Multipunto	0.05	20	77
414	Multipunto	0.05	20	71
415	Multipunto	0.05	20	76

Después de 415 generaciones se puede notar un gran avance con el porcentaje mas alto de 78% de aceptación en este punto se podrán ver agentes que ya pueden pararse erguidos y ya podrán dar algunos pasos sin caer al suelo del escenario ,pero todavía no llegamos al agente mas apto.

Tabla 8 generaciones del 490-524

Fuente :Elaboracion Propia

Generación	Tipo de Cruce	Mutación	Velocidad	Porcentaje de Aceptación
490	Multipunto	0.05	20	89
491	Multipunto	0.05	20	86
492	Multipunto	0.05	20	83
493	Multipunto	0.05	20	83
494	Multipunto	0.05	20	87
495	Multipunto	0.05	20	89
496	Multipunto	0.05	20	84
497	Multipunto	0.05	20	81
498	Multipunto	0.05	20	84
499	Multipunto	0.05	20	88
500	Multipunto	0.05	20	81
501	Multipunto	0.05	20	84
502	Multipunto	0.05	20	89
503	Multipunto	0.05	20	84
504	Multipunto	0.05	20	83
505	Multipunto	0.05	20	88
506	Multipunto	0.05	20	85
507	Multipunto	0.05	20	84
508	Multipunto	0.05	20	89

509	Multipunto	0.05	20	84
510	Multipunto	0.05	20	86
511	Multipunto	0.05	20	85
512	Multipunto	0.05	20	86
513	Multipunto	0.05	20	84
514	Multipunto	0.05	20	80
515	Multipunto	0.05	20	86
516	Multipunto	0.05	20	83
517	Multipunto	0.05	20	82
518	Multipunto	0.05	20	81
519	Multipunto	0.05	20	83
520	Multipunto	0.05	20	87
521	Multipunto	0.05	20	84
522	Multipunto	0.05	20	89
523	Multipunto	0.05	20	81
524	Multipunto	0.05	20	89

Después de 525 generaciones se puede notar un gran avance con el porcentaje mas alto de 89% de aceptación en este punto se podrán ver agentes aparte de dar algunos pasos sin caer al suelo del escenario ya pueden dar caminatas mas largas es claro que nuestro porcentaje ya se acerca bastante al requerido bastara algunas generaciones para llegar al porcentaje propuesto .

Tabla 9Muestra las generaciones del 530-564

Fuente:Elaboracion propia

Generación	Tipo de Cruce	Mutación	Velocidad	Porcentaje de Aceptación
530	Multipunto	0.05	20	88
531	Multipunto	0.05	20	87
532	Multipunto	0.05	20	90
533	Multipunto	0.05	20	88
534	Multipunto	0.05	20	90
535	Multipunto	0.05	20	87
536	Multipunto	0.05	20	90
537	Multipunto	0.05	20	88
538	Multipunto	0.05	20	91
539	Multipunto	0.05	20	90
540	Multipunto	0.05	20	88
541	Multipunto	0.05	20	88
542	Multipunto	0.05	20	88
543	Multipunto	0.05	20	90
544	Multipunto	0.05	20	88
545	Multipunto	0.05	20	89
546	Multipunto	0.05	20	87
547	Multipunto	0.05	20	90
548	Multipunto	0.05	20	88

549	Multipunto	0.05	20	87
550	Multipunto	0.05	20	89
551	Multipunto	0.05	20	89
552	Multipunto	0.05	20	91
553	Multipunto	0.05	20	88
554	Multipunto	0.05	20	89
555	Multipunto	0.05	20	89
556	Multipunto	0.05	20	90
557	Multipunto	0.05	20	89
558	Multipunto	0.05	20	88
559	Multipunto	0.05	20	88
560	Multipunto	0.05	20	91
561	Multipunto	0.05	20	87
562	Multipunto	0.05	20	89
563	Multipunto	0.05	20	87
564	Multipunto	0.05	20	90

Ya en este punto se logro hasta 564 iteraciones de las cuales en la generación 538 ya se logro el porcentaje de 91% de aceptación el cual era el propuesto en este punto el agente puede desenvolverse de forma intuitiva al caminar y pararse.

4.2ANALISIS DE DATOS.

De acuerdo a les datos recogidos de las mas de 500 generaciones se puede interpretar de la siguiente manera:

4.3 Demostración de la Hipótesis.

Tabla 10 Interpretación de los datos evolutivas

Fuente:Elaboracion Propia

Intervalo de generaciones	Porcentajes de aceptación máximo	Análisis del agente
1-35	9%	En este comienzo el agente esta aprendiendo a poder pararse sin caer en el suelo
101-134	30%	En este punto el agente puede quedarse casi parado durante algunos segundos
200-234	48%	Aquí El Agente ya puede parase y intentar dar sus primeros pasos, pero todavía tiende a caer al suelo
304-338	67%	El agente además de pararse puede dar un par de pasos
381-415	78%	Aunque en este punto el agente ya puede pararse y

		dar sus primeros pasos tiende a caer al suelo después de algunos segundos
490-524	89%	El agente ya puede parase erguido y caminar de forma mas prolongada por lo que nos acercamos al Agente evolutivo requerido
530-564	91%	Con este agente ya se nota la evolución de aprender a caminar y desenvolverse atractivamente dentro del escenario durante un tiempo mucho mas prolongado

Para nuestra Hipótesis se Propuso una función Fitness o de aceptación del 91% a alcanzarse con el agente inteligente. Por la tabla de Análisis se puede demostrar que después de 538 la técnica de la red neuronal a base del algoritmo genético tiende al 91% de aceptación:

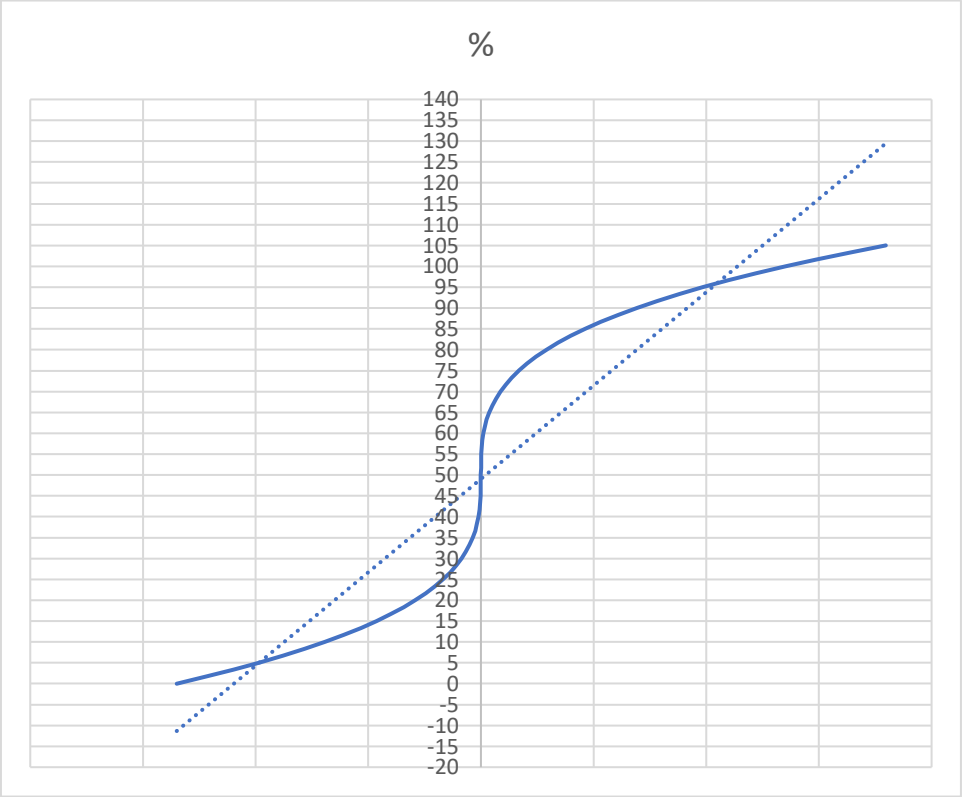


Figura4.3 Tendencia de Evolución Fuente:Elaboracion Propia

Según nuestra tabla de interpretación de datos se puede notar en la gráfica la evolución según su función de aceptación el porcentaje. A partir del 91% propuesto en la hipótesis se empieza a formar una recta la cual significa que el agente evoluciono hasta poder desplazarse de manera atractiva en el escenario 2D.

Por tanto, se escriben los siguientes datos a partir del resultado de la generación 538 y generaciones posteriores para ver que las demás generaciones tengan la misma función de aceptación o superior.

Para demostrar el entrenamiento de los agentes inteligentes tomaremos los siguientes 15 agentes inteligentes después de haber alcanzado función de Aceptación propuesta del 91%

Tabla 11. Generaciones posteriores de agentes inteligentes
Fuente: Elaboración Propia

#Generacion	Aceptación
538	95
539	93
540	95
541	90
542	91
543	91
544	92
545	90
546	94
547	91
548	94
549	94
550	91
551	92
552	93

Se tomaron en cuenta 15 generaciones posteriores a la primera generación que obtuvo el 91% de función fitness.

$$Promedio = 93{,}138889\%$$

Tabla 12. Generaciones posteriores de agentes inteligente, evaluando resultados para varianza

Fuente: Elaboración Propia

#Generacion	Aceptación	$\lvert x-u \rvert$	$\lvert x-u \rvert^2$
538	95	2	4
539	93	0	0
540	95	2	4
541	90	3	9
542	91	2	4
543	91	2	4
544	92	1	1
545	90	3	9
546	94	1	1
547	91	2	4
548	94	1	1
549	94	1	1
550	91	2	4
551	92	1	1

552	93	0	0

$\frac{\sum|x-\mu|^2}{N} = 3{,}1333$ varianza

$\sqrt[2]{\frac{\sum|x-\mu|^2}{N}} = 1{,}7701$ Desviación estándar

Se utilizará un $\alpha = 0{,}05$.

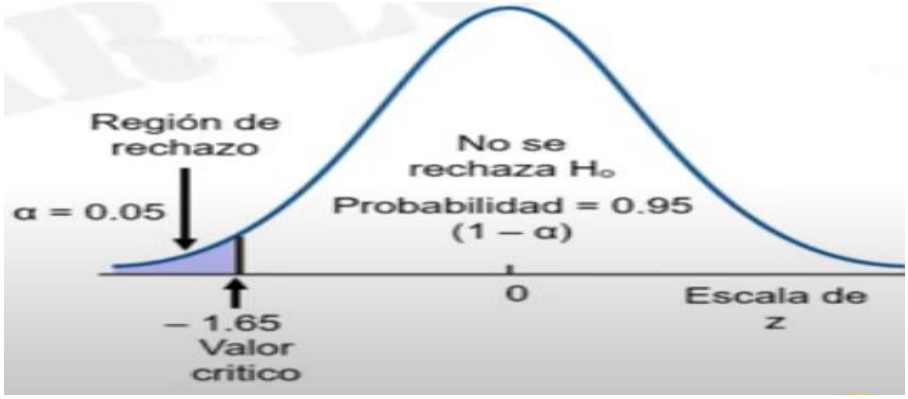
Hipótesis nula $H_0 \geq 91\%$ nuestro porcentaje de aceptación propuesto,

Hipótesis alterna $H_1 < 91\%$ Significa que el agente inteligente no cumple el porcentaje de evolución propuesto para poder desplazarse.

$$Z = \frac{\bar{x} - \mu}{\frac{\sigma}{\sqrt{N}}} = 4{,}3760$$

Sacando punto crítico.

$$Probabilidad = 1 - 0{,}05 = 0{,}95$$



Punto crítico es = -1,65.

Como $Z = 4{,}3760 > -1{,}65$ entonces se acepta H_0 por tanto con el valor de función de aceptación del 91% se puede entrenar el agente inteligente evolutivo tal que las siguientes generaciones pueden desplazarse en la plataforma con una aceptación de 91% o superior.

Con esto se puede demuestra la hipótesis planteada de que El agente se puede desplazar con un comportamiento evolutivo atractivo con una función fitness del 91%

4.4 Resuelve el problema

Si se pudo mejorar a partir de la red neuronal entrenada con el algoritmo genético se pudo mejorar el desempeño del agente inteligente, obteniendo una sensación atractiva y evolutiva sin tener que usar una máquina de estados que si se tratara de realizar de la misma manera se encontrarían con una tendencia de estados infinitos dependiendo la complejidad del escenario de simulación aparte de tener que usar computadoras de alto rendimiento para que puedan soportar el entrenamiento de tantos estados

CAPÍTULO V CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

Después del diseño e implementación del agente inteligente evolutivo, se llega a las siguientes conclusiones.

- Usando como herramienta de simulación GODOT ENGINE se pudo diseñar un ambiente de desplazamiento óptimo para que un agente inteligente evolutivo pueda entrenar
- Se ha realizado el escenario 2D correspondiente al género plataforma para el desempeño evolutivo del agente inteligente.
- La red neuronal diseñada fue entrenada con el algoritmo genético para desempañarse según los estímulos del ambiente del escenario.
- Se evaluó el agente inteligente y luego se hizo la prueba de hipótesis con un valor de confianza del 95% y se pudo demostrar que la técnica evolutiva de agente inteligente entrenado con redes neuronales y algoritmo genético tiene una sensación evolutiva en la plataforma 2D.

5.2 RECOMENDACIONES

Finalizando la investigación se puede recomendar que a causa de diferentes avances de tecnología que existen esta técnica de entrenamiento evolutivo puede ser optimizado con y adaptado con varios tipos de lenguajes de programación, pero la premisa de entrenamiento será la misma.

Para posteriores investigaciones se dará las siguientes recomendaciones:

- Se recomienda tener una investigación profunda en metauristica y programación evolutiva para comprender el objetivo al que se pretende llegar con la evolución.
- Este agente inteligente puede adaptarse incluso a ambientes 3D solo hará falta rediseñar las redes neuronales.
- Si se cambia el escenario a 3D también se podrá extraer el agente inteligente evolutivo entrenado para poder instalarlo incluso a placas Arduino y otras placas industriales en otras palabras se podrán instalar a incluso robots y estos podrán desempañarse igual al escenario de simulación.

Bibliografía

Adamaris. (2014). *Verion 1.Internal Technical Report.Automation and Robotics.* Chicago: Internal Technical Report.

Arthut, L. S. (1990). *the machine comerce IBM.* New York: IBM work.

Barado, S. &. (2013). *sistemas expertos: fundamentos, metodologías y aplicaciones.* Buenos aires: palermo.

Bueno. (2014). *EL PROGRAMA DE MEJORA DE LA INTELIGENCIA P.A.T.* Madrid: Departamento de Psicología Evolutiva y de la Educación.

Carbonell. (2000). *Inteligencia Artificial Aplicada a la instrucción asistida por computador.* Mexico: J.R.

Carlos Soto, M. (2002). *Teoría de sistemas expertos". Universidad Nacional Mayor de San Marcos. Lima, Perú.* Lima, Perú: http://sisbib.unmsm.edu.pe/bibvirtualdata/Tesis/Basic/carlos_sm/cap1.pdf .

Fogel. (1992). *evolutionary computing.* Chicago: Internal Technical Report.

Garcia. (2007). *Inteligencia Artificial.* New York: IMT.

Gomez. (2010). *Agente pedagógico.* Madrid: Universidad Complutense Madrid .

Hubel, D. (2005). *Brain and visual perception: the story of a 25-year collaboration.* Venice: Brain and visual.

James, A. (2010). *Metodo de prototipados.* Mexico: percepción visual.

Koury, J. M. (2020). *Cómo y por qué llegamos a esta pandemia.* Madrid-España.

LAGE. (2009). *Modelo de Sistemas Tutor Inteligente distribuido para educación a distancia.* Buenos Aires: Universidad Tecnológica Nacional.

León, Q. (2012). *Sistemas expertos y sus aplicaciones.* Pachuca: Universidad Autónoma del estado de Hidalgo.

León, Q. T. (2007). *Sistemas expertos y sus aplicaciones.* Universidad Autónoma del estado de Hidalgo. Pachuca, México.

Linden, A. (2003). *smart agents.* New york: current smart agents and neural networks.

M.Darell. (1994). *Neuronal Network Synthesis Using Cellular Encoding and The Genetic Algorithm.* Chicago: Network Synthesis Using Cellular Encoding.

Marques. (1994). *Metodología del software educativo.* Barcelona: Barcelona Estel.

McCarthy. (1956). *Inteligencia artificial.* Dartmouth.

McCulloch. (1943). *Biologi Artifitial.* Chicago: Pitts.

Mnangagwa, E. (2020, marzo 18). El coronavirus es la obra de dios para castigar a los países que nos han impuesto sanciones. p. 5.

Obrador, A. (2020). . *Ogranizacion Panamericana de la Salud*, 12-13.

Padgam, E. (2002). *Prometheus agile.* Venice: NEtwork Work.

Pajares. (2006). *Inteligencia Artificial e Ingeniería del Conocimiento.* México D.f.: Alfa omega Grupo Editor S.A.

Schmidhuber. (2015). *Deep Learning in Neural Networks: An Overview.* California: Neural Networks 61.

SILVESTRE, J. (2012). *Agentes Inteligentes.* Caracas: Universidad Central de Venezuela.

Turban. (1995). *Decision Support and Expert Systems (4ta edición).* Pasadena: Prentice-Hall.

Turban, E. (1995). *Decision Support and Expert Systems (4ta edición).* EE.UU.

Whitley, D. (2008). *Algoritmos geneticos y redes neuronales.*

Wilson, E. (2005). *BackPropagation learning.*

Zilouchian, A. (2010). *Fundamentals of Neural Network.* Philadelphia: Using Soft Computing Methodologies.

ANEXOS

ANEXO A

La informática evolutiva

Las primeras ideas, incluso antes del descubrimiento del ADN, vinieron de Von Neumann, uno de los mayores científicos de este siglo. Von Neumann afirmó que la vida debía de estar apoyada por un código que a la vez describiera como se puede construir un ser vivo, y tal que ese ser creado fuera capaz de autorreproducirse, por tanto, un autómata o máquina autorreproductiva tendría que ser capaz, aparte de contener las instrucciones para hacerlo, de copiar tales instrucciones a su descendencia, y así continuar con este proceso.

Sin embargo, no fue hasta mediados de los años cincuenta, cuando el rompecabezas de la evolución se había prácticamente completado, cuando Box comenzó a pensar en imitarla, para mejorar procesos industriales. La técnica de Box, denominada EVOP (Operación

Evolucionaria), consistía en elegir una serie de variables que regían un proceso industrial. Sobre esas variables se creaban pequeñas variaciones que formaban un hipercubo, variando el valor de las variables una cantidad fija.

Se probaba entonces con cada una de las esquinas del hipercubo durante un tiempo, y al final del periodo de pruebas, un comité humano decidía sobre la calidad del resultado. Es decir, se estaba aplicando mutación y selección a los valores de las variables, con el objeto de mejorar la calidad del proceso. Este procedimiento se aplicó con éxito a algunas industrias químicas.

Un poco más adelante, en 1958, Friedberg y sus colaboradores pensaron en mejorar usando técnicas evolutivas la operación de un programa. Para ello diseñaron un código máquina de 14 bits (2 para el código de operación, y 6 para los datos y/o instrucciones); cada programa, tenía 64 instrucciones. Un programa llamado Herman, ejecutaba los programas creados, y otro programa, el profesor, le mandaba a Herman ejecutar otros programas y ver si los programas ejecutados habían realizado su tarea o no. La tarea consistía en leer unas entradas, situadas en una posición de memoria, y debían depositar el resultado en otra posición de memoria, que era examinada al terminarse de ejecutar la última instrucción.

Para hacer evolucionar los programas, Friedberg hizo que en cada posición de memoria hubiera dos alternativas; para cambiar un programa, alternaba las dos instrucciones (que eran una especie de aleles), o bien reemplazaba una de las dos instrucciones con una totalmente aleatoria.

En realidad, lo que estaba haciendo es usar mutación para generar nuevos programas; al parecer, no tuvo más éxito que si hubiera buscado aleatoriamente un programa que hiciera la misma tarea. El problema es que la mutación sola, sin ayuda de la selección, hace que la búsqueda sea prácticamente una búsqueda aleatoria.

Más o menos simultáneamente, Bremmerman trató de usar la evolución para entender los procesos de pensamiento creativo y aprendizaje, y empezó a considerar la evolución como un proceso de aprendizaje. Para resolver un problema, codificaba las variables del problema en una cadena binaria de ceros y unos, y sometía la cadena a mutación, cambiando un bit de cada vez; de esta forma, estableció que la tasa ideal de mutación debía de ser tal que se cambiara un bit cada vez. Bremmerman trató de resolver problemas de minimización de funciones, aunque no está muy claro qué tipo de selección usó, y el tamaño y tipo de la población. En todo caso, se llegaba a un punto, la trampa de Bremmerman, en el cual la solución no mejoraba, y luego en intentos sucesivos trató de añadir entrecruzamiento entre soluciones, pero tampoco obtuvo buenos resultados.

Una vez más, el simple uso de operadores que creen diversidad no es suficiente para dirigir la búsqueda genética hacia la solución correcta, y corresponde a un concepto de la evolución darwiniano clásico: por mutación, se puede mejorar a un individuo; en realidad, la evolución actúa al nivel de población, es decir que actúa de manera global.

El primer uso de procedimientos evolutivos en inteligencia artificial se debe a Reed, Toombs y Baricelli, que trataron de hacer evolucionar un tahúr que jugaba a un juego de cartas simplificado. Las estrategias de juego consistían en una serie de 4 probabilidades de apuesta alta o baja con una mano alta o baja, con cuatro parámetros de mutación asociados. Se mantenía una población de 50 individuos, y aparte de la mutación, había intercambio de probabilidades entre dos padres. Es de suponer que los perdedores se eliminaban de la población (tirándolos por la borda). Aparte de, probablemente, crear buenas estrategias, llegaron a la conclusión de que el entrecruzamiento no aportaba mucho a la búsqueda.

Los intentos posteriores, ya realizados en los años 60, ya corresponden a los algoritmos evolutivos modernos, y se han seguido investigando hasta nuestros días. Algunos de ellos son simultáneos a los algoritmos genéticos, pero se desarrollaron sin conocimiento unos de otros. Uno de ellos, la programación evolutiva de Fogel, se inició como un intento de usar la evolución para crear máquinas inteligentes, que pudieran prever su entorno y reaccionar adecuadamente a él. Para simular una máquina pensante, se utilizó un autómata celular. Un autómata celular es un conjunto de estados y reglas de transición entre ellos, de forma que, al recibir una entrada, cambia o no de estado y produce una salida.

Fogel trataba de hacer aprender a estos autómatas a encontrar regularidades en los símbolos que se le iban enviando. Como método de aprendizaje, usó un algoritmo evolutivo: una población de diferentes autómatas competía para hallar la mejor solución, es decir, predecir cual iba a ser el siguiente símbolo de la secuencia con un mínimo de errores; los peores 50%

eran eliminados cada generación, y sustituidos por otros autómatas resultantes de una mutación de los existentes.

De esta forma, se lograron hacer evolucionar autómatas que predecían algunos números primos (por ejemplo, uno, cuando se le daban los números más altos, respondía siempre que no era primo; la mayoría de los números mayores de 100 son no primos). En cualquier caso, estos primeros experimentos demostraron el potencial de la evolución como método de búsqueda de soluciones novedosas.

Más o menos a mediados de los años 60, Rechenberg y Schwefel describieron las estrategias de evolución. Las estrategias de evolución son métodos de optimización paramétricos, que trabajan sobre poblaciones de cromosomas compuestos por números reales. Hay diversos tipos de estrategias de evolución, que se verán más adelante; pero en la más común, se crean nuevos individuos de la población añadiendo un vector de mutación a los cromosomas existentes en la población, en cada generación, se elimina un porcentaje de la población (especificado por los parámetros), y los restantes generan la población total, mediante mutación y crossover. La magnitud del vector mutación se calcula adaptativamente.

John Holland desde pequeño, se preguntaba cómo logra la naturaleza, crear seres cada vez más perfectos. Lo curioso era que todo se lleva a cabo basándose en interacciones locales entre individuos, y entre estos y lo que les rodea. No sabía la respuesta, pero tenía una cierta idea de como hallarla: tratando de hacer pequeños modelos de la naturaleza, que tuvieran

alguna de sus características, y ver cómo funcionaban, para luego extrapolar sus conclusiones a la totalidad. De hecho, ya de pequeño hacía simulaciones de batallas célebres con todos sus elementos: copiaba mapas y los cubría luego de pequeños ejércitos que se enfrentaban entre sí.

En los años 50 entró en contacto con los primeros ordenadores, donde pudo llevar a cabo algunas de sus ideas, aunque no se encontró con un ambiente intelectual fértil para propagarlas. Fue a principios de los 60, en la Universidad de Michigan en Ann Arbor, donde, dentro del grupo lógica de computadoras, sus ideas comenzaron a desarrollarse y a dar frutos. Además, fue leyendo un libro escrito por un biólogo evolucionista llamado, R. A. Fisher, titulado "La teoría genética de la selección natural", como comenzó a descubrir los medios de llevar a cabo sus propósitos de comprensión de la naturaleza.

De ese libro aprendió que la evolución era una forma de adaptación más potente que el simple aprendizaje, y tomó la decisión de aplicar estas ideas para desarrollar programas bien adaptados para un fin determinado.

En esa universidad, Holland impartía un curso titulado: "Teoría de sistemas adaptativos", dentro de este curso, y con una participación activa por parte de sus estudiantes, fue donde se crearon las ideas que más tarde se convertirían en los algoritmos genéticos.

Por tanto, cuando Holland se enfrentó a los algoritmos genéticos, los objetivos de su investigación fueron dos:

La imitación de los procesos adaptativos de los sistemas naturales, y

El diseño sistemas artificiales (normalmente programas) que retengan los mecanismos importantes de los sistemas naturales.

Unos 15 años más adelante, David Goldberg, que en la actualidad es el maestro de los algoritmos genéticos, conoció a Holland, y se convirtió en su estudiante. Goldberg era un ingeniero industrial trabajando en diseño de pipelines, y fue uno de los primeros que trató de aplicar los algoritmos genéticos a problemas industriales. Aunque Holland trató de disuadirle, porque pensaba que el problema era excesivamente complicado como para aplicarle algoritmos genéticos, Goldberg consiguió lo que quería, escribiendo un algoritmo genético en un ordenador personal Apple II. Estas y otras aplicaciones creadas por estudiantes de Holland convirtieron a los algoritmos genéticos en un campo con base suficiente aceptado para celebrar la primera conferencia en 1985, ICGA´85, la cual se sigue celebrando bianualmente.

Ubicación del problema

El problema consiste en desarrollar un algoritmo genético para la simulación del crecimiento de una población, basándose en los índices de natalidad, mortalidad, y las migraciones, y luego observar su comportamiento a través del tiempo.

Además se ha de optimizar el crecimiento de una población, a través del tiempo, mediante algoritmos genéticos, y finalmente se realizará una selección de un modelo más complejo, basándose en el paradigma de los algoritmos genéticos.

Limitaciones y alcance del tema.

Las técnicas de simulación convencionales, son de gran utilidad; pero aplicando los algoritmos genéticos, se puede realizar la simulación del crecimiento de una población demográfica, aplicando el proceso de selección natural y otros operadores genéticos, donde cada individuo forma parte de la población, y por lo tanto, pertenece a la solución.

Objetivos: generales y específicos.

El objetivo general es desarrollar un modelo de crecimiento poblacional, utilizando el paradigma de los algoritmos genéticos como una herramienta robusta para la simulación y la optimización y se tratará de estudiar el comportamiento de una población mediante un modelo basado en el proceso de la evolución.

Como objetivos específicos se tienen los siguientes:

Observar el comportamiento de los índices poblacionales, así como la posibilidad de una tendencia, o convergencia en un modelo desarrollado con algoritmos genéticos.

Realizar variaciones en los algoritmos genéticos de para adaptarlos al problema planteado.

Implantar una aplicación robusta que aplique los algoritmos genéticos para modelar el crecimiento de una población, y sus interacciones.

Comparar los modelos de crecimiento poblacional convencionales, con un modelo basado en algoritmos genéticos.

Ejemplos de aplicaciones realizadas solo con algoritmos geneticos:

Los siguientes son ejemplos de aplicaciones que he realizado mientras comprendía el funcionamiento de los algoritmos genéticos, y planteaba como utilizarlos en mi aplicación dirigida a los modelos de crecimiento poblacional.

El problema del vendedor viajero (TSP)

Es una pequeña aplicación realizada en Visual C++ 6.0 que optimiza rutas. Esta aplicación lee un archivo de texto ubicado en el mismo directorio que la aplicación se encuentra. En esta aplicación, se utilizó una representación de los cromosomas por medio de permutaciones. Además debe configurarse como parámetros del algoritmo, el tamaño de la población, el tamaño del torneo, el número de iteraciones y la semilla aleatoria. Estos son explicados en detalle en el capítulo II, para una mejor comprensión de los algoritmos genéticos.

Punto máximo en una función normal estándar

Es una pequeña aplicación que me permitió comprender el algoritmo genético básico representando los cromosomas mediante cadenas binarias. Está desarrollado completamente en Visual C++ 6.0 en el modo ventana de diálogo con MFC, y pide el ingreso del número de iteraciones, tamaño de la población, intervalo de acción, probabilidad de crossover, probabilidad de mutación, y probabilidad de convergencia.

Algoritmos genéticos – modelo poblacional

Es la principal aplicación que realicé, con la que determino el crecimiento de una población heterogénea, basada en el número de hijos, edad límite de vida, intervalo de edades de reproducción, y la configuración del algoritmo genérico que presenta algunas opciones que luego son explicadas detalladamente.

Este es ámbito al que nos vamos a dedicar en este proyecto, en este caso se aprovecha la selección natural, el cruce y la mutación para simular a través de generaciones, y observar las medidas de desempeño de los índices poblacionales que nos interesan, tales como tasa de natalidad, tasa de mortalidad y las tasas de migraciones, así como la edad promedio de la población y la distribución por sexos.

Ejemplos de otras aplicaciones que pueden realizarse:

Planificación de multitarea

Una aplicación de los algoritmos genéticos en un problema de asociación óptima de procesos y procesadores. Un objetivo es disminuir el costo que se deriva de la comunicación entre procesos en un ordenador paralelo de memoria distribuida. La Planificación de multitarea también puede ser relacionada con problemas de robótica.

Biología molecular y física o química

Existe una hipótesis de trabajo que sustentaría el uso de técnicas basadas en poblaciones debido a una estructura de función objetivo. En problemas relacionados con estructura molecular existen experimentos que dan indicios claros sobre la validez de esa hipótesis. Así que es fácil observar que son cada vez más numerosas las aplicaciones de algoritmos genéticos en este campo.

Ingeniería en construcciones

Los Algoritmos genéticos se han ganado la aceptación en un gran número de problemas de ingeniería, estos problemas están basados en la optimización de recursos. Una aplicación es la optimización discreta de estructuras.

Búsqueda en bases de dados

Así como existen los métodos de búsqueda convencionales en bases de datos, los algoritmos genéticos pueden facilitar dichas búsquedas, dando resultados muy eficientes.

Geofísica

Para un problema llamado de Inversión de la forma de la ola sísmica, generalmente se utiliza la estadística Bayesiana, a dicho problema se asocia una información que es apriori sobre un modelo. Surgen así funciones que requieren métodos de optimización global, donde es de mucho provecho el uso de los algoritmos genéticos.

Compresión de Dados

La compresión de dados en general, es la compresión de imágenes sólidas en particular. Esta aplicación consiste en encontrar un método que utiliza los Algoritmos genéticos para encontrar

un sistema de funciones locales iteradas (LIFS) para la codificación de imágenes. Produciendo como resultado final una imagen con calidad similar a la utilización de un método convencional de compresión fractal, con un tiempo 30% menor.

Optimización de rutas

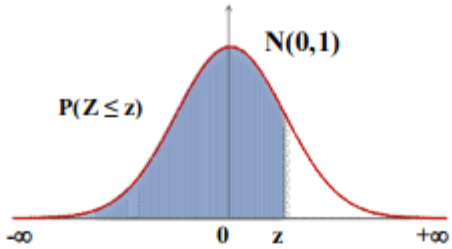
Cuando queremos optimizar rutas, existen muchas técnicas de programación que nos llevan a una respuesta adecuada. Podemos aplicar también los algoritmos genéticos para hallar una ruta óptima.

Además existen aplicaciones de los algoritmos genéticos muy generales, que pueden ser adaptadas a cualquier necesidad, siempre que se tenga cuidado al considerar la correcta representación de sus elementos.

Entre estos algoritmos encontramos la aplicación GaUI 1.0, la cual está desarrollada completamente en un lenguaje de cuarto nivel, y presenta muchas etapas de los algoritmos genéticos para la resolución de diferentes problemas comunes.

ANEXO B Tabla de distribución normal

FUNCIÓN DE DISTRIBUCIÓN NORMAL N(0,1)



z	0,00	0,01	0,02	0,03	0,04	0,05	0,06	0,07	0,08	0,09
0,0	0,5000	0,5040	0,5080	0,5120	0,5160	0,5199	0,5239	0,5279	0,5319	0,5359
0,1	0,5398	0,5438	0,5478	0,5517	0,5557	0,5596	0,5636	0,5675	0,5714	0,5753
0,2	0,5793	0,5832	0,5871	0,5910	0,5948	0,5987	0,6026	0,6064	0,6103	0,6141
0,3	0,6179	0,6217	0,6255	0,6293	0,6331	0,6368	0,6406	0,6443	0,6480	0,6517
0,4	0,6554	0,6591	0,6628	0,6664	0,6700	0,6736	0,6772	0,6808	0,6844	0,6879
0,5	0,6915	0,6950	0,6985	0,7019	0,7054	0,7088	0,7123	0,7157	0,7190	0,7224
0,6	0,7257	0,7291	0,7324	0,7357	0,7389	0,7422	0,7454	0,7486	0,7517	0,7549
0,7	0,7580	0,7611	0,7642	0,7673	0,7704	0,7734	0,7764	0,7794	0,7823	0,7852
0,8	0,7881	0,7910	0,7939	0,7967	0,7995	0,8023	0,8051	0,8078	0,8106	0,8133
0,9	0,8159	0,8186	0,8212	0,8238	0,8264	0,8289	0,8315	0,8340	0,8365	0,8389
1,0	0,8413	0,8438	0,8461	0,8485	0,8508	0,8531	0,8554	0,8577	0,8599	0,8621
1,1	0,8643	0,8665	0,8686	0,8708	0,8729	0,8749	0,8770	0,8790	0,8810	0,8830
1,2	0,8849	0,8869	0,8888	0,8907	0,8925	0,8944	0,8962	0,8980	0,8997	0,9015
1,3	0,9032	0,9049	0,9066	0,9082	0,9099	0,9115	0,9131	0,9147	0,9162	0,9177
1,4	0,9192	0,9207	0,9222	0,9236	0,9251	0,9265	0,9279	0,9292	0,9306	0,9319
1,5	0,9332	0,9345	0,9357	0,9370	0,9382	0,9394	0,9406	0,9418	0,9429	0,9441
1,6	0,9452	0,9463	0,9474	0,9484	0,9495	0,9505	0,9515	0,9525	0,9535	0,9545
1,7	0,9554	0,9564	0,9573	0,9582	0,9591	0,9599	0,9608	0,9616	0,9625	0,9633
1,8	0,9641	0,9649	0,9656	0,9664	0,9671	0,9678	0,9686	0,9693	0,9699	0,9706
1,9	0,9713	0,9719	0,9726	0,9732	0,9738	0,9744	0,9750	0,9756	0,9761	0,9767
2,0	0,9772	0,9778	0,9783	0,9788	0,9793	0,9798	0,9803	0,9808	0,9812	0,9817
2,1	0,9821	0,9826	0,9830	0,9834	0,9838	0,9842	0,9846	0,9850	0,9854	0,9857
2,2	0,9861	0,9864	0,9868	0,9871	0,9875	0,9878	0,9881	0,9884	0,9887	0,9890
2,3	0,9893	0,9896	0,9898	0,9901	0,9904	0,9906	0,9909	0,9911	0,9913	0,9916
2,4	0,9918	0,9920	0,9922	0,9925	0,9927	0,9929	0,9931	0,9932	0,9934	0,9936
2,5	0,9938	0,9940	0,9941	0,9943	0,9945	0,9946	0,9948	0,9949	0,9951	0,9952
2,6	0,9953	0,9955	0,9956	0,9957	0,9959	0,9960	0,9961	0,9962	0,9963	0,9964
2,7	0,99653	0,99664	0,99674	0,99683	0,99693	0,99702	0,99711	0,99720	0,99728	0,99736
2,8	0,99744	0,99752	0,99760	0,99767	0,99774	0,99781	0,99788	0,99795	0,99801	0,99807
2,9	0,99813	0,99819	0,99825	0,99831	0,99836	0,99841	0,99846	0,99851	0,99856	0,99861
3,0	0,99865	0,99869	0,99874	0,99878	0,99882	0,99886	0,99889	0,99893	0,99896	0,99900
3,1	0,99903	0,99906	0,99910	0,99913	0,99916	0,99918	0,99921	0,99924	0,99926	0,99929
3,2	0,99931	0,99934	0,99936	0,99938	0,99940	0,99942	0,99944	0,99946	0,99948	0,99950
3,3	0,99952	0,99953	0,99955	0,99957	0,99958	0,99960	0,99961	0,99962	0,99964	0,99965
3,4	0,99966	0,99968	0,99969	0,99970	0,99971	0,99972	0,99973	0,99974	0,99975	0,99976
3,5	0,99977	0,99978	0,99978	0,99979	0,99980	0,99981	0,99981	0,99982	0,99983	0,99983
3,6	0,99984	0,99985	0,99985	0,99986	0,99986	0,99987	0,99987	0,99988	0,99988	0,99989
3,7	0,99989	0,99990	0,99990	0,99990	0,99991	0,99991	0,99992	0,99992	0,99992	0,99992
3,8	0,99993	0,99993	0,99993	0,99994	0,99994	0,99994	0,99994	0,99995	0,99995	0,99995
3,9	0,99995	0,99995	0,99996	0,99996	0,99996	0,99996	0,99996	0,99996	0,99997	0,99997
4,0	0,99997	0,99997	0,99997	0,99997	0,99997	0,99997	0,99998	0,99998	0,99998	0,99998

Nota: En el interior de la tabla se da la probabilidad de que la variable aleatoria Z, con distribución N(0,1), esté por debajo del valor z.

DOCUMENTOS

La Paz,15 Junio del 2021

Señor

Ph.D. José María Tapia Baltazar

DIRECTOR

CARRERA DE INFORMATICA

FAC. CIENCIAS PURAS Y NATURALES

UNIVERSIDAD MAYOR DE SAN ANDRES

Presente. -

Ref.- Aval para la defensa de Tesis de Grado

De mi mayor consideración:

Mediante la presente, me dirijo a su autoridad, en calidad de Tutor Metodológico para informar que luego de haber realizado el seguimiento de la Tesis/Proyecto de Grado titulado: : "TECNICA EVOLUTIVA DE AGENTE INTELIGENTE CON REDES NEURONALES Y ALGORITMO GENETICO EN UNA PLATAFORMA 2D", presentado por el Univ.:JOSE GONZALO OLIVAREZ BOLIVAR, con C.I.8347981 L.P.,para optar al titulo de Licenciado en Informática mención Ingeniería en Sistemas Informáticos.

En este sentido, presento mi conformidad y aval respectivo para la defensa publica de la Tesis de Grado de acuerdo a Reglamento vigente en la Universidad Mayor de San Andrés.

Sin otro particular, me despido con las consideraciones más distinguidas.

Atentamente.

M.Sc Grover Alex Rodriguez Ramírez

TUTOR