Universidad Nacional Autónoma de México

Posgrado en Ciencia e Ingeniería de la Computación

ALGORITMOS GENÉTICOS PARA LA GENERACIÓN DE COMPORTAMIENTOS EN ROBOTS MÓVILES

T E S I S

que para obtener el grado de:

Maestro en Ingeniería
(Computación)

P R E S E N T A

Efrén Arturo Morales Arredondo

Director: Dr. Jesús Savage Carmona

México D.F. 2009

Dedicatoria

A la U.N.A.M., mi casa desde hace años.
A Yasmine por su apoyo, amor y compañia a lo largo de la maestría.
A mi mamá y hermanos que fueron, son y serán una parte muy importante en mi vida .

Agradecimientos

Me gustaría agradecer particularmente a:

A la Universidad Nacional Autónoma de México, por darme la oportunidad de pertenecer a ella.

A CONACYT por el apoyo económico recibido durante mis estudios.

Yasmine por apoyarme incondicionalmente a lo largo de los estudios de posgrado.

Al Dr. Jesús Savage por su atención y paciencia a lo largo del desarrollo de este trabajo.

Al personal del posgrado en computación.

A los amigos hecho a lo largo del periodo de maestría.

Agradezco también a las personas que sirvieron de apoyo dentro del posgrado como personal del IIMAS, profesores, sinodales y a todos los que me quieren y me apoyan de una u otra forma.

Gracias a todos.

:)

Resumen

El problema a resolver en este trabajo consiste en mejorar el comportamiento presentado por los robots móviles en algún ambiente. Los principales objetivos son: utilizar los conceptos de algoritmos genéticos, aplicarlos para generar aprendizaje y comportamientos en los robots y también para la solución de diversos problemas en regiones conocidas por los mismos (ambientes); se hizo uso de diferentes ramas del conocimiento para el análisis y desarrollo del problema.

En el presente trabajo se hace un desarrollo e implementación de diferentes comportamientos robóticos utilizando del concepto de algoritmos genéticos; primero se realiza la evolución de diferentes constantes que sirven para modificar el comportamiento de los robots embebidos en un mundo (el cual contiene obstáculos) en donde, basados en el análisis del mundo, se mueven por medio de los campos potenciales. Como segunda parte del trabajo, se evolucionan las máquinas de estado que sirven como comportamiento auxiliar al robot, esto se usa cuando este se bloquea o se traba en el mundo, en este caso también se hace uso del concepto de algoritmos genéticos evolucionando las máquinas de estados.

Los valores de las constantes obtenidos son para utilizarlos en el robot real y obtener un mejor comportamiento en los diferentes mundos que sean presentados. Las evaluaciones generacionales del sistema se realizaron por medio del simulador Virbot, el cual se encuentra desarrollado en C y es utilizado para probar diferentes sistemas que posteriormente son implementados en robots reales.

Los resultados obtenidos en la implementación de los algoritmos genéticos usando campos potenciales tuvieron un adecuado desarrollo, fueron obtenidos valores que sirven para que el robot presente un adecuado comportamiento en el ambiente dado, aunque se requiere un considerable procesamiento computacional, debido a que se requieren evaluar muchos individuo evaluando uno a uno por cada generación. En conclusión, el objetivo planteado al inicio fue cumplido y los resultados han probado el correcto funcionamiento del desarrollo realizado en el ambiente.

Índice general

Índice de figuras

Capítulo 1

Introducción

Desde sus orígenes la robótica ha estudiado y generado diferentes comportamientos en los robots móviles, algunos de estos han sido de los que son conocidos como *reactivos*, es decir que se generan una vez que algún tipo de sensor es activado. Por ejemplo: sensores infrarrojos, de ultrasonido, de contacto, o por medio de cámaras que detectan marcas para señalar obstáculos o acciones de acuerdo a esas lecturas. En este comportamiento no es necesario generar mundos abstractos del mundo porque el robot usa dichos componentes para reaccionar al entorno y por otro lado tenemos los comportamientos que implementan *máquinas de estado*, los cuales están en función de varias variables y lecturas las cuales pueden ser como en el caso del comportamiento reactivo: sensorial o por medidores.

El desarrollo del **cerebro robótico** implica introducir en él diferentes tipos de comportamientos del robot para que tome las decisiones necesarias de a cuerdo a los diferentes tipos de lecturas ya sea para evadir obstáculos o tomar una ruta especifica. Normalmente estos comportamientos son probados en diferentes simuladores y a veces el modelado del problema se realiza con *condiciones ideales*, es decir, en donde

los errores por deslizamiento no existen, en donde el robot podría consumir solo el combustible ideal, en donde no aparecen obstáculos no comprendidos en el problema.

1.1. Algoritmos genéticos para generación de comportamientos

Como anteriormente se ha mencionado en los cerebros robóticos son introducidos diferentes tipos de comportamientos que ejecutará el robot de acuerdo a ciertas condiciones. Ahora como parte del trabajo se esta sugiriendo el uso de algoritmos genéticos para definir los comportamientos de robots móviles autónomos, pero primero hay que entender que son los algoritmos genéticos, estos tienen como origen las teorías evolutivas de las especies, Charles Darwin el ícono en esta área realizó amplios estudios científicos en este campo, todas las especies sin excepción tienen como objetivo la preservación de sus genes y esto se logra por medio de la selección natural, es aquí cuando la naturaleza elige a los individuos con mejores características de adaptación al entorno y por ello una mejor preparación para sobrevivir; Existen múltiples ejemplos en los cuales se aplican estos conceptos, cualquier ser vivo se rige por estas ”leyes de sobrevivencia”, a manera de ejemplo en el desierto más seco del planeta el Atacama en Chile, en donde una lluvia de 1 **mm** puede tener lugar mínimo cada 5 años (al menos en los últimos años así se ha comportado) en este sitio aunque pareciera que no puede preservar vida en el, muchas especies se ha adaptado de una forma sorprendente para vivir en el, para esto muchas generaciones tuvieron que perecer y los padres de los individuos actuales sobrevivir, adaptarse al ambiente; esta adaptación al pasar a las nuevas generaciones es lo que se conoce como evolución, con esto al llevarse a cabo la reproducción de las especies, se guarda en los genes del descendiente esta información, la cual es de muy importante para la supervivencia del nuevo individuo en su ambiente.

Estos conceptos fueron utilizados por un gran número de investigadores a mediados del siglo pasado, pero fue John Holland en 1975 que volvió popular los algoritmos genéticos por medio de su libro ” *Adaptation in Natural and Artificial Systems* ”[1], el en su trabajo, propone una forma de seleccionar individuos y cruzarlos en la actualidad

[1] [11] Holland

se tienen un gran número de propuestas para realizar esto, David Golberg [2] retomó el algoritmo de Holland y lo popularizo llamándolo Algoritmo genético simple, en el cual se codifican los códigos genéticos en una representación computacional binaria.

1.2. Definición del problema

Como se mencionó los algoritmos genéticos simulan el proceso de evolución natural, para esto tenemos los siguientes puntos usados en la generación del algoritmo genético:

Codificación del dominio. en la naturaleza las características de cada individuo se encuentran determinadas de forma microscópica por las proteínas que contienen y producen; los aminoácidos que las conforman contienen el material genético que de forma particular son almacenadas en las celulas. (con ello la naturaleza codifica todas las posibles soluciones de crear un individuo óptimo en un conjunto de bases las cuales son productoras de proteinas que conforman un individuo con características particulares), mapeándolo a todas las posibles secuencias de nucleótidos, así para el algoritmo genético primero necesitamos saber en qué espacio podemos encontrar las diferentes soluciones al problema que se pretende resolver. Como el algoritmo opera sobre códigos genéticos (sobre los genotipos de cada individuo) se codifica de forma binaria generándose una secuencia de unos y ceros, los cuales contienen codificación del individuo y su espacio de soluciones, cuando tenemos esta codificación y tenemos la forma de obtener de un individuo a su código y de forma inversa, se debe tener un punto inicial, cada población estará compuesta por un número dado de individuos los cuales serán convertidos en la población a evaluar, a este primer grupo de individuos se les denomina población inicial y esta será la entrada para el algoritmo genético a partir del cual serán generadas nuevas poblaciones.

Evaluación de la población. En la naturaleza se tienen diferentes tipos de individuos algunos son más propensos a enfermedades, algunos más débiles, otros más fuertes en algunas características particulares, estas indican las diferencias entre cada uno de los individuos de la población, y claro estas diferencias siempre van relacionadas a la población a la cual pertenece dicho individuo. Si se tiene una medida que califique el

[2] [5] Golberg

desempeño de alguna manera de cada individuo, esta estará en base al desempeño del resto de la población. En los algoritmos genéticos es necesario establecer alguna medida de desempeño de la población de que contiene un conjunto de individuos. Para identificar a los individuos que de una manera o de otra tengan las características que mejoran el desempeño de la población es necesario calificarlos; De esta manera se conocen las mejores cualidades que tienen del resto y marcar al mismo tiempo a los peores elementos del conjunto. En la biología se utiliza un término (fitness) la cual indica el grado de adaptación de cada individuo con relación al resto, en los algoritmos genéticos esta es la llamada función de evaluación y es en realidad la que más poder de cómputo requiere en la mayoría de las veces.

Selección. Cuando los individuos son calificados se deben seleccionar los mejores de la población ya que son los que se adaptaron de la mejor forma al medio. Seleccionar a estos individuos incrementa las posibilidades de que los descendientes resulten también individuos con una buena calificación. para realizar la selección se debe tener un parámetro de calificación en el cual sólo pasaran los individuos que se encuentren al menos sobre ese parámetro, así de esta forma podremos asegurar que estamos seleccionando los mejores individuos de la población.

Cruzamiento. En este proceso se realiza la combinación de características de los individuos, el resultado es un individuo hibrido con cualidades de los padres, así de esta forma los descendientes tienen las mejores características en promedio de los individuos seleccionados y con ello el material genético (en el caso de los algoritmos genéticos son bits los que representan las cualidades de los individuos) tendrá la mejor adaptación al dominio con respecto a la población, existen muchos mecanismos de cruzamiento, en la naturaleza por ejemplo en el ser humano el cruzamiento puede verse casi inmediatamente por medio de los genes dominantes y los recesivos, los primeros son como su nombre lo indica dominan en las características que prevalecen en el individuo.

Mutación. Como en la naturaleza debe existir en el algoritmo un factor de alteración con una muy baja probabilidad, así algunos individuos de un algoritmo son alterados; esto permite que el algoritmo no se quede en un mínimo local y que pudiera no llegar a encontrar la solución buscada, la mutación nos sirve para poder incrementar

el espacio de soluciones para la resolución del problema.

1.3. Objetivos Generales

Utilizar los conceptos de algoritmos genéticos, para generar aprendizaje y comportamientos en los robots, también para la solución de diversos problemas en regiones conocidas por los mismos; se hará uso de diferentes ramas del conocimiento para el análisis y desarrollo del problema.

1.3.1. Objetivos particulares

- Aplicar los conceptos de algoritmos genéticos y campos potenciales, para generar un sistema que obtenga los mejores valores para las constantes para el movimiento del robot, de forma eficiente y dinámica para cualquier ambiente.
- Agregar al sistema existente VirBot el sistema a crear y que sea de fácil integración.
- Obtener los mejores valores para las constantes para el desplazamiento del robot en diferentes ambientes.

1.4. Estructura de la tesis

A continuación se presenta la estructura que formará el presente trabajo:

Capítulo 2 Comportamientos. En este capítulo se explicarán de forma breve el análisis del problema, así como algunos antecedentes y conceptos básicos a utilizar.

Capítulo 3 Comportamientos utilizando algoritmos genéticos En este capítulo se revisarán diferentes conceptos relacionados a los algoritmos genéticos, estrategias evolutivas y otras formas de abordar problemas haciendo uso básicamente de algoritmos genéticos

Capítulo 4 Mejoramiento de comportamientos. Se realizará la implementación de un algoritmo genético para realizar mejoras de comportamiento a robots móviles.

Capítulo 5 Análisis de resultados. En este capítulo se realizará un análisis de los resultados obtenidos en el capítulo anterior.

Conclusiones y trabajo futuro .

Capítulo 2

Comportamientos Robóticos

2.1. Robots y comportamientos

Comportamiento. Def. Manera de comportarse.[1].
Comportar. Def. Portarse, conducirse.[2]

La creación de robots siempre tiene un objetivo, estos son hechos para cumplir alguna función específica tal y como lo indica la definición de la RIA[3] que lo define como: ”dispositivo reprogramable, multifuncional, manipulador diseñado para mover materiales, partes, herramientas” o ”son dispositivos especializados variables, con movimientos programados para realizar una gran variedad de tareas”. [4]

[1]Diccionario de la Real Academia de la Lengua Española[18]
[2]Diccionario de la Real Academia de la Lengua Española[18]
[3]Robotic Industry Association
[4]Jablonski y Posey 1985

El diseño de robots siempre tienen un objetivo ya sean para usarse en tierra en agua o en el aire, y hasta en el espacio por lo cual depende de muchos factores el diseño de su comportamiento,para que fueron o van a ser creados. Por ejemplo podemos pensar en un robot de navegación, quizás queremos que este sea introducido en algún lugar el cual sería peligroso para que entrara alguna persona o simplemente por las dimensiones nadie podría ingresar, entonces el robot primero debe prepararse para que pueda desplazarse sin problemas por el lugar y al mismo tiempo prepararlo para reaccionar en caso de encontrar algún obstáculo y pueda ser capaz de evadirlo, moverlo o arrastrarlo según sean las especificaciones del mismo.

Supongamos un robot independiente y es necesario introducirlo a un sistema de pequeños ductos, el esbozo del mapa en varios estados se indica en la figura 2.1 y los ductos tienen la forma indicada, el robot cuenta con 3 sensores uno a la izquierda, otro a la derecha y uno más al frente, los movimientos permitidos para el mismo son:

- Vuelta a la izquierda
- Vuelta a la derecha
- Avanza
- Retrocede

Ahora definimos el comportamiento del robot en base al estímulo que puede recibir por medio de los tres sensores, entonces tendríamos el siguiente patrón de movimiento si los sensores izquierdo, derecho y central se representan por medio de Si, Sd y Sc respectivamente:

```
IF Sc == 0{
   AVANZA N PASOS
}ELSE{
   IF Si == 0{
      GIRO IZQUIERDA 90°
   } ELSE IF Sd == 0{
```

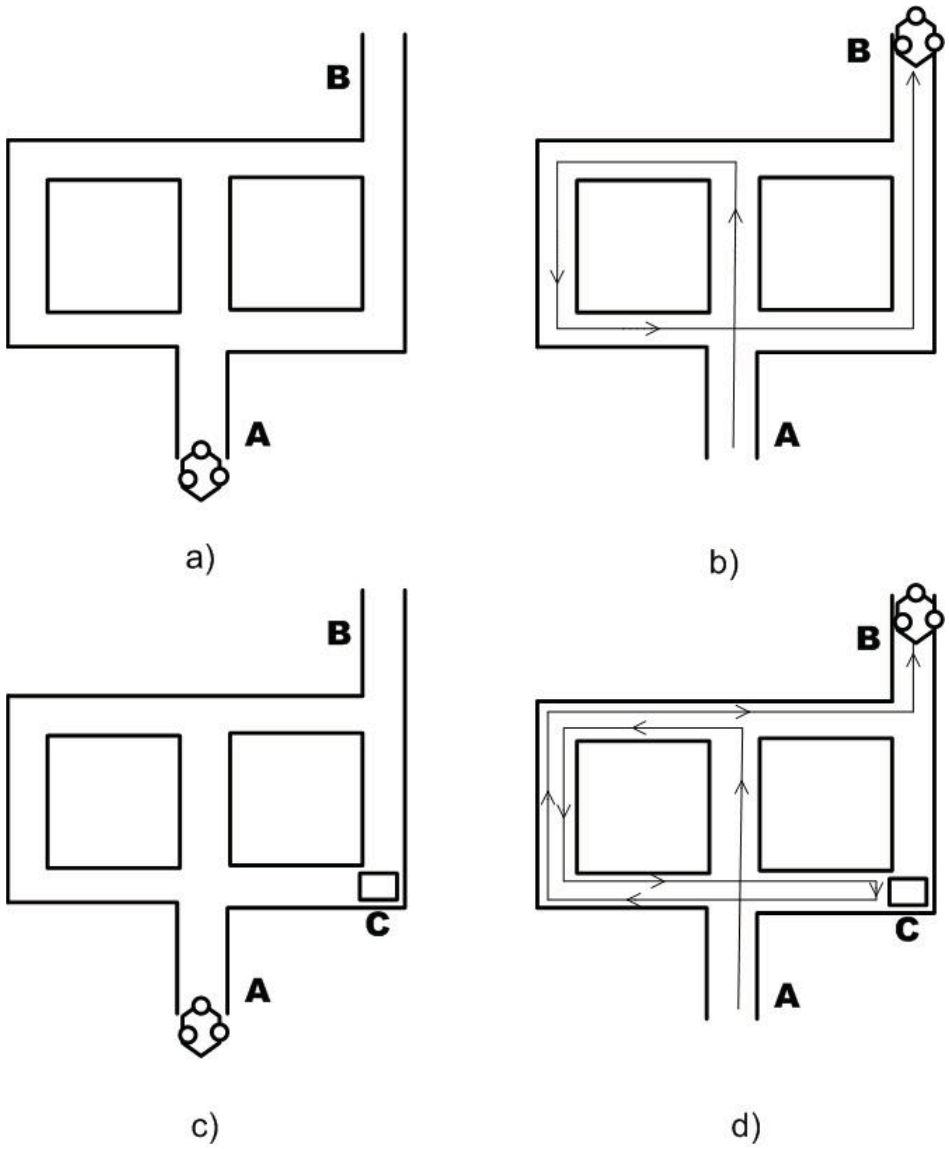

Figura 2.1: Ejemplos de trayectoria de un robot con tres sensores. Con origen en el punto A y como destino el punto B. a) El robot se encuentra en la posición inicial. b) Trayectoria del robot desde A hasta B. c) Un obstáculo es agregado en el trayecto del robot en la posición C. d) Modificación en el comportamiento del robot cuando encuentra cerrado el camino.

```
      GIRO DERECHA 90º
   }ELSE{
      RETROCEDE N PASOS
   }
}
```

Si deseamos que el robot tenga como posición inicial el punto A y como destino el punto B como se muestra en la figura 2.1 a, el robot presentaría un comportamiento generado a partir de las lecturas de los sensores y por medio de las evaluaciones del pseudocódigo anterior se desplazaría siguiendo la ruta que se puede apreciarse en la figura 2.1 b alcanzando el punto destino pero, ¿que sucedería en caso de que el robot encontrara un obstáculo representado por el punto C? esto se indica en la imagen 2.1 c. En este caso los 3 sensores serían activados dando como resultado que el robot retrocediera pero si no es una distancia lo suficientemente grande el robot ya no podría salir de ese sitio porque quizás los pasos para retroceder no serían suficientes para lograr evadir el obstáculo, pero si modificamos un poco ese comportamiento y lo dejamos como se indica a continuación:

```
IF Sc == 0{
   AVANZA N PASOS
}ELSE{
   IF Si == 0{
      GIRO IZQUIERDA 90º
   } ELSE IF Sd == 0{
      GIRO DERECHA 90º
   }ELSE{
      GIRO DERECHA 90º
   }
}
```

El robot al llegar al punto C giraría y presentaría el desplazamiento indicado en la imagen 2.1 d. Este es un ejemplo de un comportamiento robótico muy simple, pero descriptivo de la complejidad de generar comportamientos en robots. El comportamiento de este robot es meramente *reactivo*, es decir, que básicamente reacciona

basado en los impulsos que recibe por medio de los sensores. Este tipo de comportamiento es más rápido, pero presenta un nivel muy bajo de inteligencia y el cómputo de datos es muy sencillo como puede apreciarse en el pseudocódigo líneas arriba; pero si deseamos que el robot presente aprendizaje y sea más inteligente, (tenga un comportamiento*deliberativo*), este implica una mayor complejidad y dependiendo de las situaciones un procesamiento computacional más complejo. En la figura 2.2 tenemos una tabla en donde podemos analizar la comparación entre un sistema reactivo y uno deliberativo, gráficamente se puede ver claramente la diferencia, el modelo reactivo es un modelo de percepción y acción, de forma típica tienen el control de los motores en base a las percepciones (en el ejemplo anterior por medio de los sensores).

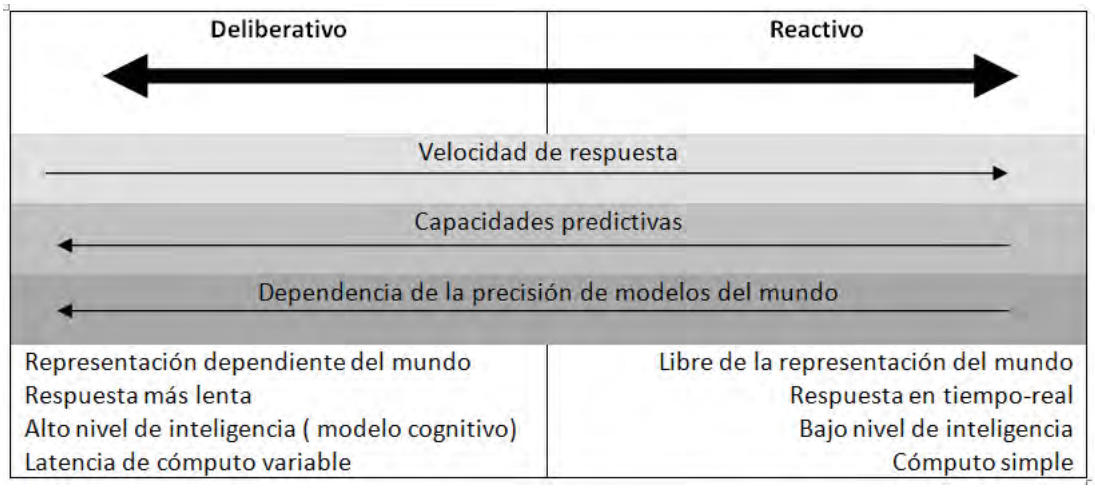

Figura 2.2: Comparación básica de comportamientos

2.2. La naturaleza en la robótica

Una fuente de inspiración del hombre para la generación de tecnología a lo largo de su historia ha sido la misma naturaleza y de forma más precisa los animales. El desarrollo de los aviones hoy en día no es concebible sin pensar en aves, en el campo de la robótica no es la excepción a esta regla, el desarrollo de comportamientos robóticos tiene sus orígenes en el modus-vivendi de diferentes especies. La ciencia

que se encarga de estudiar el comportamiento de los seres vivos tiene sus origenes en la *la etología*, cualquier individuo del reino animal presenta un comportamiento, el que este exista aun en el ser más pequeño es prueba suficiente de que existe inteligencia.

Muchos animales desde los orígenes de la robótica han servido para " copiar "modelos de comportamientos con una gran similitud a sistemas robóticos, de hecho el análisis ha ido más allá de lo que estudia la etología como lo expresa Arkin "Para robots basados en comportamientos discuten que hay mucho más que puede ser obtenido para la robótica a través del estudio de *neurociencias*, *psicología* y *etología*, los comportamientos necesitan decidir como hacer uso de los resultados de estas disciplinas"[5] Para entender perfectamente el funcionamiento del cerebro aun falta mucho, se puede decir que su conocimiento aun es superficial. Actualmente el modelado computacional del cerebro básicamente tiene dos vertientes, el *esquema teórico* y las *redes neuronales*. En el esquema teórico el comportamiento se expresa de forma modular, este modelo filosófico tiene sus orígenes en el siglo XIX con el filósofo y científico Immanuel Kant[6] mientras que en las redes neuronales se definieron algunos esquemas para entender y agrupar percepción sensorial en el proceso de conocimiento o experiencia. En el siglo XX sirvió como puente abstracto entre el cerebro y la mente, Piaget (1971) [7] usó la teoría del esquema como mecanismo para expresar modelos de memoria y aprendizaje. Muchos modelos neuro-computacionales han sido desarrollados utilizando esas teorías, mientras que los modelos computacionales para redes neuronales, son referidos más como sistemas de conexiones. Aquí fue introducido el concepto de perceptron, el cual tiene como entradas un vector de entradas binarias, en donde cada una de estas entradas tiene un peso específico (peso sináptico), el resultado es sumado y sujeto a una operación para determinar la salida la cual es enviada a otra célula de la red, ver figura 2.3.

2.3. Etología como base de comportamiento en robots

La etología como se señaló anteriormente se encarga del estudio del comportamiento de animales en su medio ambiente(No olvidar que el ser humano se encuentra dentro de este conjunto), Konrad Lorenz[8] quien desde joven tenía interés

[5] Arkin [1]
[6] Kant [14]
[7] Piaget [15]
[8] Kant [16]

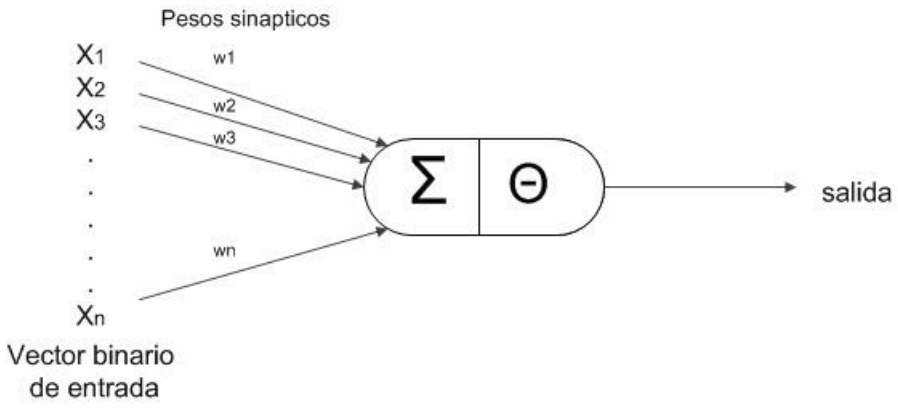

Figura 2.3: Perceptrón

en la evolución junto a Nikolaas Tinbergen y Karl R. von Frisch obtienen el premio nobel en el año de 1973 por estudios de comportamiento en el reino animal, actualmente los estudios se enfocan en los siguientes niveles:

- Causal. Son respuestas generadas por estímulos tanto externos como internos y cuando ya no se presentan estos deja de presentarse dicho comportamiento.
- Desarrollo. Este se encarga de discriminar en el periodo de vida de una especie cuando aparece y el porque de un comportamiento.
- Evolución. Este nivel es el encargado de estudiar los beneficios que se van guardando en los genes de las especies, que comportamientos generan algún tipo de beneficio a la especie a lo largo de varias generaciones.

Estas son algunas premisas que han servido de base para pensar en el desarrollo de comportamientos robóticos inmersos en un ambiente específico. En la imagen 2.4 podemos ver a BigDog[9] un robot de carga muy similar a una mula, este robot ha sido probado en diversos ambientes y el comportamiento y la forma de caminar es de alguna manera similar a algún animal de carga. Parte del comportamiento del bigdog es el de poder responder ante algunas situaciones que pudieran generar algún tipo de caída (alguna fuerza externa o quizás por algún defecto del suelo en que camina); este presenta una buena respuesta ante este tipo de situaciones, corrigiendo el

[9]Página del bigdog [7]

movimiento. Por otro lado tenemos a ASIMO un robot creado por la empresa Honda, este robot ha sufrido mejorías y desde su creación se busca que presente un comportamiento muy cercano al del ser humano. Desde los primer modelo (modelo E0) el cual presentaba solo un robot caminante hasta el actual ASIMO el cual ya presenta interacción con seres humanos y algunos comportamientos humanos dentro de ambientes específicos, son ejemplos de la utilización de la etología en la robótica. Resulta interesante ver como la misma naturaleza ha influenciado la robótica,¿y porque no habría de hacerlo? el ser humano trata de emular comportamientos y la física de algunas especies que mejor prueba que los mismos animales, han pasado miles de siglos, miles de generaciones han evolucionado para que existan las especies actuales, ellos, adaptados a su medio son dignos representantes de que defectos han sido superados y que los individuos débiles o no adaptados al medio han tenido que perecer.

Figura 2.4: Ejemplos de comportamientos de seres vivos en la robótica

2.4. Comportamientos robóticos

Como se puede ver en la figura 2.2 los modelos de comportamiento robóticos los podemos dividir en dos, los modelos deliberativos y reactivos. Precisamente uno de los problemas en robótica es llevar a cabo el control de los diferentes mecanismos que componen los robots, la coordinación de los diferentes sensores y componentes, los tiempos de reacción de acuerdo a las lecturas de los mismos y la forma en que se van a interpretar estas lecturas de forma sincronizada, es una de las cuestiones más complicadas en la robótica. Analicemos entonces ambos tipos de comportamiento, en la sección 2.1 revisamos un ejemplo específico de un comportamiento reactivo de que básicamente tenemos los siguientes comportamientos:

2.4.1. Comportamientos reactivos

El control reactivo en realidad es un sistema que presentará una rápida respuesta entre la percepción y la acción, esto es para obtener una respuesta rápida ante las percepciones de los diferentes sensores en mundos poco estructurados y desconocidos, Arkin[10] define una serie de características para definir los comportamientos reactivos:

- *Los comportamientos sirven como base en la construcción de bloques para acciones de robots*, un comportamiento de este tipo básicamente consiste en una simple relación sensor-motor, las lecturas de los sensores son suficientes para obtener la información necesaria para la respuesta en los motores.
- *La respuesta sensorial es más importante que la representación de algún conocimiento*. Los sistemas puramente reactivos, reaccionan directamente a lo percibido por medio de los sensores, evitan la necesidad de algún tipo de abstracción de conocimiento; para ser más preciso, lo que se percibe es lo que se obtiene. Este es particularmente utilizado para ambientes dinámicos o en territorios hostiles. Construir modelos abstractos del modelo del mundo consume tiempo y al mismo es propenso de errores.
- *Modelos de comportamiento animal*. Como se mencionó anteriormente, la naturaleza ha proporcionado la prueba de que muchas de las tareas que queremos

[10] Arkin [1]

que realicen los robots han sido copiadas de la misma naturaleza, Además las ciencias biológicas, como las neurociencias, etología, y psicología han servido de bases para mecanismos y modelos aplicables a los robots.

- *Los sistemas son modulares desde una perspectiva de software*Esto permite a los diseñadores de sistemas de robots agregar nuevos comportamientos a los comportamientos reactivos, sin rehacer o eliminar el anterior; esto es bastante útil para generar comportamientos más complejos.

Máquinas de estados finitos

Los estados de aceptación de las máquinas finitas son muy útiles para agregar secuencias de comportamientos, hacen explícita la respuesta de comportamiento de acuerdo al estado en el que se pudiera encontrar el robot. Los modelos son comportamientos del sistema con un número limitado de respuestas dependiendo de los estados en los que se encuentre. Una máquina de estados consiste de 4 elementos:

- Estados que definen el comportamiento y que al mismo tiempo producen acciones.
- Transiciones que son movimientos de un estado a otro.
- Reglas o condiciones que deben presentarse en cada una de las transiciones.
- Eventos de entrada que son generados los cuales pueden desencadenar reglas o transiciones de estado.

La máquina finita de estados debe tener de inicio un estado inicial, en este se presentarán las condiciones iniciales del sistema; un estado actual y en el mismo debe conocerse cual fue el estado anterior, tiene entradas que sirven para conocer cual será el siguiente estado. Por medio de las tablas de estados, en la figura 2.5 podemos ver una representación gráfica de una máquina finita de estados como puede visualizarse los estados normalmente son representados por círculos. En caso de ser un estado inicial se apunta una flecha sin un nodo origen y en caso del estado o estados finales se representan por medio de un par de círculos concéntricos. Las transiciones son representadas por medio de flechas con origen en un nodo y como destino otro

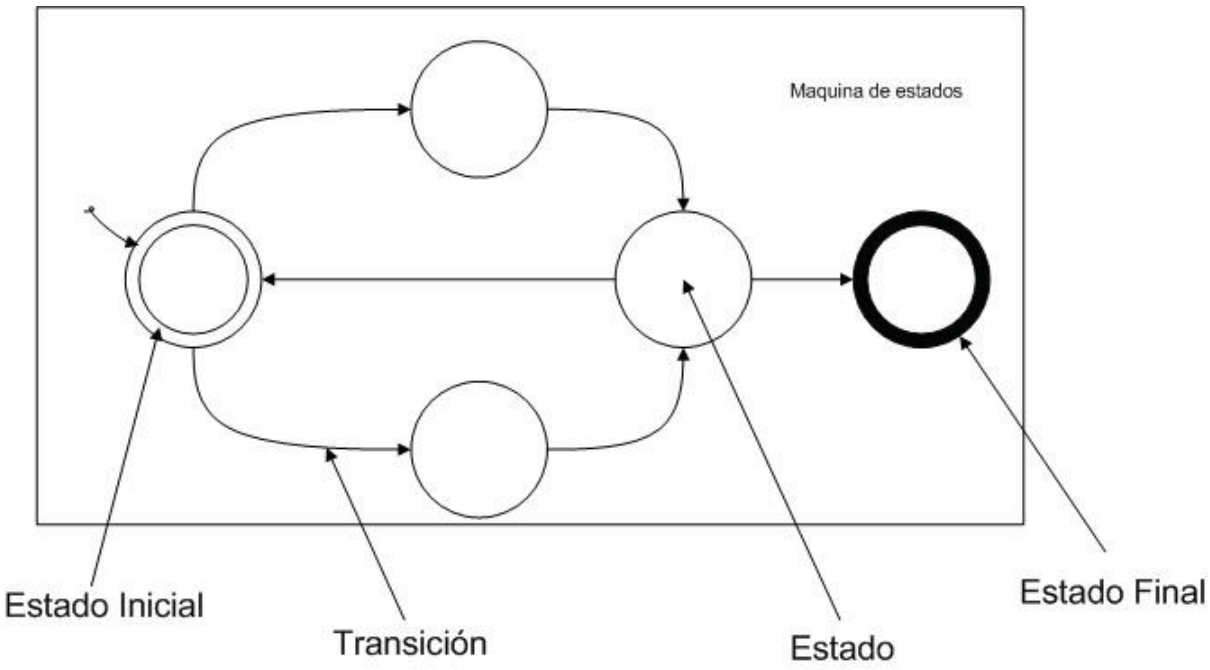

Figura 2.5: Ejemplo de máquina de estados finitos.

nodo o el mismo, dependiendo de la operación o evaluación de las variables.

¿Pero en robótica como aplicar Máquinas de estados finitos ? Conceptualmente es sencillo y muy descriptivo; como fue mencionado los estados de la máquina serán las configuraciones o respuestas en los diferentes sistemas que componen el robot y las transiciones condicionales estarán basadas en las lecturas de sensores. A manera de ejemplo pondremos una representación de máquina de estados de Whade[11]. Utiliza una máquina de estados continua en estados y transiciones, cada uno de los estados representa una acción en particular (los torques de dos motores), las transiciones son de la forma A < *Si* < B donde A y B son números reales entre 0 y 7 los cuales representan el mínimo y máximo censado (en este ejemplo) y *Si* es la lectura del sensor i. Las transición de condiciones son muestreadas en un orden dado, por ello los sensores de mayor importancia son los que son considerados primero. Para cada transición hay un estado al cual salta si se cumplen las condiciones, en caso contrario permanece en

[11] Whade [4]

el estado actual. El estado inicial siempre es el estado 1. En la figura 2.6 En el estado 1, las torques del motor izquierdo y derecho son de 0.5 y 0.3 respectivamente; la máquina de estados salta al estado 2 en caso de que el sensor tenga una lectura entre 1.0 y 1.3 y en caso de que presente un lectura entre 3.1 y 4.7.

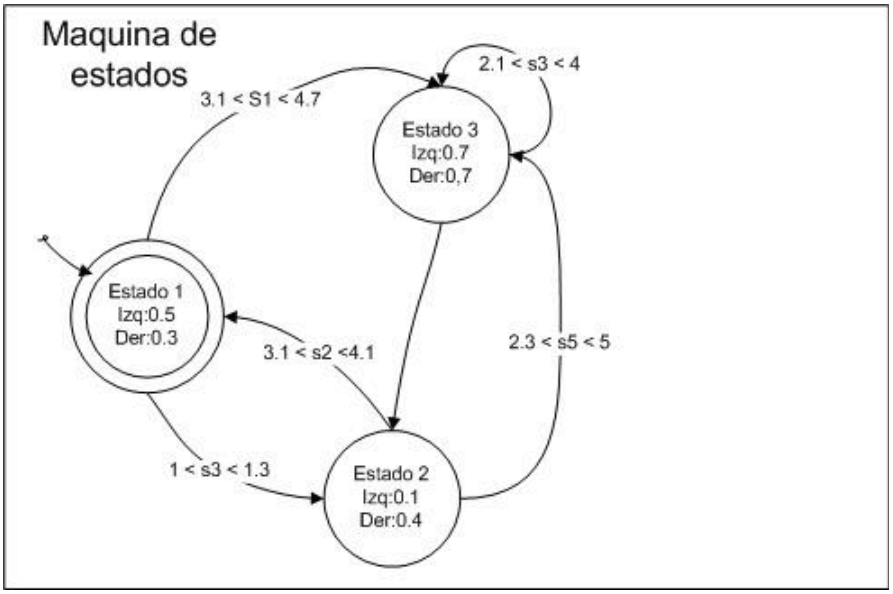

Figura 2.6: Ejemplo de Máquina finita de estados 3-estados.

2.4.2. Comportamientos deliberativos

Aunque es mucho más común el desarrollo de comportamientos reactivos, también cada vez es más necesario desarrollar comportamientos más inteligentes, que decidan y se comporten de algún modo (que presenten aprendizaje) es por ello que este tipo de comportamientos aunque resulten en un mayor consumo de procesamiento y también presenten una respuesta más lenta, pero el tratamiento de información temporal, en donde el robot debe ejecutar diferentes acciones dependiendo de la información procesada o de experiencias anteriores, el presentar algún razonamiento para ejecutar alguna acción predictiva en el futuro. Supongamos que un robot no conoce el entorno, pero que por medio de un censado inicial puede encontrar diferentes objetos dentro de un mundo; esta información puede ser manejada para que el robot aprenda donde

hay paredes y objetos, justamente aquí es en donde entra el desarrollo de la *inteligencia artificial* la cual se utiliza para producir máquinas que automaticen tareas con comportamiento inteligente. Este estudio se ha convertido en un disciplina de ingeniería enfocada a solucionar problemas de la vida real así como aplicaciones de software juegos de estrategia. El concepto de IA es aún demasiado difuso. Contextualizando, y teniendo en cuenta un punto de vista científico podríamos englobar a esta ciencia como la encargada de imitar a una persona, y no por fuera en su cuerpo, sino imitar al cerebro, en todas las funciones posibles, existentes en el humano o inventadas sobre el desarrollo de la máquina inteligente. Así, aplicando literalmente la definición de Inteligencia Artificial, no cabe otra posibilidad que pensar en máquinas inteligentes, es decir, sin emociones que obstaculicen encontrar la mejor solución a un problema dado.

Debemos pensar en dispositivos artificiales capaces de concluir miles de premisas a partir de otras premisas dadas, sin que ningún tipo de emoción tenga opción de sobrevivir. En esta línea, hay que saber que ya existen sistemas inteligentes, capaces de tomar decisiones acertadas. Sin embargo, el concepto de Inteligencia Artificial que la mayoría de las personas podemos hacernos es muy distinto al descrito anteriormente, pues lo que realmente nos gustaría, lo que produciría curiosidad experimental, es un dispositivo con emociones. Y aquí es donde finaliza el campo científico y comienza todo un conjunto de teorías, avisos, discusiones y toda forma expresiva de conflictos intelectuales entre semejantes por predecir lo que podría o no ocurrir en caso de existir máquinas emocionales. Aquí finaliza la ciencia porque, hoy por hoy, no existe red neuronal, o sistema , capaz de desarrollar emociones. Pero estos no son años de estos temas, simplemente por el avance tecnológico que vivimos en el presente espacio-tiempo. Además habría que añadir el concepto de heurística. Dicho concepto se basa en la búsqueda de un camino a seguir para llegar a una solución, la heurística son las decisiones tomadas en dicho camino y las razones de ello. Así si pretendemos ”implementar”la inteligencia nos basaremos en un conocimiento previo, pues hasta el momento la Inteligencia Artificial se basa todo en conocimiento humano, no conocimiento desarrollado, y llegaremos al objetivo deseado.

El Aprendizaje de máquinas es un amplio campo de la inteligencia artificial en el cual las computadoras ”aprenden”. En esta área existen tres grandes tipos de aprendizaje:

- Inductivo. Algunas veces es conocido como lógica inductiva, es un proceso de razonamiento en donde la premisa de un argumento supone la solución pero no la asegura, es usando para atribuir propiedades o tipos basados en identificadores; por ejemplo, un número de observaciones o experiencias. Existen también a su vez diferentes tipos de razonamiento inductivo como: la generalización en donde se concluye de acuerdo a la población; por ejemplo tenemos un conjunto de elementos de un mismo color y ponemos en algún lugar un elemento de otro color, si tenemos un robot con un sistema de visión quizás de acuerdo a una percepción no adecuada de la población que ahí no se tienen elementos diferentes a la generalidad.

  Otro tipo de aprendizaje inductivo es la inferencia casual, se basa en obtener una conclusión basada en las condiciones de causa y efecto, premisas que pueden indicar una relación entre ellas, pero ambos factores deben estar confirmados para establecer la relación exacta de la relación causal y existen un buen número de aprendizajes inductivos.

- Deductivo. En este tipo de razonamiento obtenemos una solución basada en premisas conocidas en este caso si las premisas son verdaderas la conclusión también lo será. El desarrollo de este aprendizaje tiene como base las reglas de inferencia.

- Abductivo.El concepto de aprendizaje abductivo puede ser definido como un caso discriminante, el cual contiene también características de aprendizaje de un problema, combina características de aprendizaje y discriminante mediante la implicacion abductiva[12], este se refiere a un aprendizaje basado en " teorías incompletas" mediante suposiciones verdaderas con otras que no han sido comprobadas, esto resulta contrario a la lógica clásica.

El mayor enfoque de las investigaciones de aprendizaje es extraer información de datos de forma automática, por medio de métodos estadísticos. El aprendizaje de maquinas esta estrechamente relacionado a la minería de datos y estadística, el aprendizaje tiene un gran campo de desarrollo de aplicaciones como pueden ser procesamiento de lenguaje natural, reconocimiento de patrones sintáctico, máquinas de búsqueda, visión por computadora y del desarrollo de Sistemas Basados en el conocimiento o sistemas expertos (SEs)[13] entre muchos otros.

[12]Lavrac [12]

[13]S. Marcellin [10] Un Sistema Experto (SE) es un sistema basado en computadora que integra bases

2.5. Sistemas de comportamientos híbridos

Los sistemas híbridos nos permiten combinar estos dos principales comportamientos, tanto el reactivo como el deliberativo logrando esto obtenemos robots más cercanos a la reacción del mismo ser humano. Los comportamientos híbridos también tienen que hacer uso de los que se conoce como métodos competitivos y se refiere a cuando tenemos dos o mas comportamientos activos, en donde cada uno tiene su propia respuesta además de ser independientes, este permite resolver algún posible conflicto en la decisión de cual comportamiento ejecutar, a este ente abstracto se le conoce como árbitro y requiere de una función que en base a algún criterio decide el comportamiento dominante para alguna acción determinada, existen varios métodos de decisión que pueden incidir en un selector de comportamiento (arbitro), en este ejemplo por prioridad como se puede ver en la figura 2.7. Y como su nombre lo indica la selección del comportamiento será por medio de la prioridad que tengan cada uno de ellos[14]. Otro tipo de arbitraje el cual es el método de selección de acción ver figura 2.8[15] en donde el árbitro selecciona un comportamiento simple, en este tipo de arbitraje robótico, los comportamientos compiten unos con otros a través del uso de niveles de activación, los cuales son regidos por la información obtenida por medio de los sensores. Otro método es el democrático en donde cada comportamiento va obteniendo un cierto número de ”votos”por cada una de las acciones que realice, obviamente el comportamiento con más votos gana, en caso de que se presente algún empate el árbitro decide cual comportamiento es el ganador y así se ejecuta la acción del comportamiento victorioso.

Lyons y Hendriks (1992 y 1995) presentaron una arquitectura Planeación-Reacción y la propuesta de arquitectura es presentar un planeador de acciones como mecanismo para modificar continuamente la ejecución de un sistema reactivo. La arquitectura general se presenta en la figura 2.9, el planeador es un árbitro de ejecución que adapta el comportamiento bajo el sistema, en un ambiente cambiante, donde el robot

de datos, memorias, mecanismos de razonamiento, algoritmos, heurísticas, para adquirir, generar y almacenar conocimientos inicialmente adquiridos a través de varios expertos humanos dentro de un dominio especifico llamado ”nube ”.

Con un Sistema Experto, se pueden dar recomendaciones y/o tomar acciones en las áreas de análisis, diseño, diagnóstico, planeación y control o dar solución a problemas o aplicar técnicas de enseñanza o en general recomendar, actuar y explicar las acciones que hay que tomar en actividades en las cuales normalmente, se requiere del conocimiento o saber de expertos humanos dentro de una nube especifica.

[14] Arkin [1]

[15] Arkin [1]

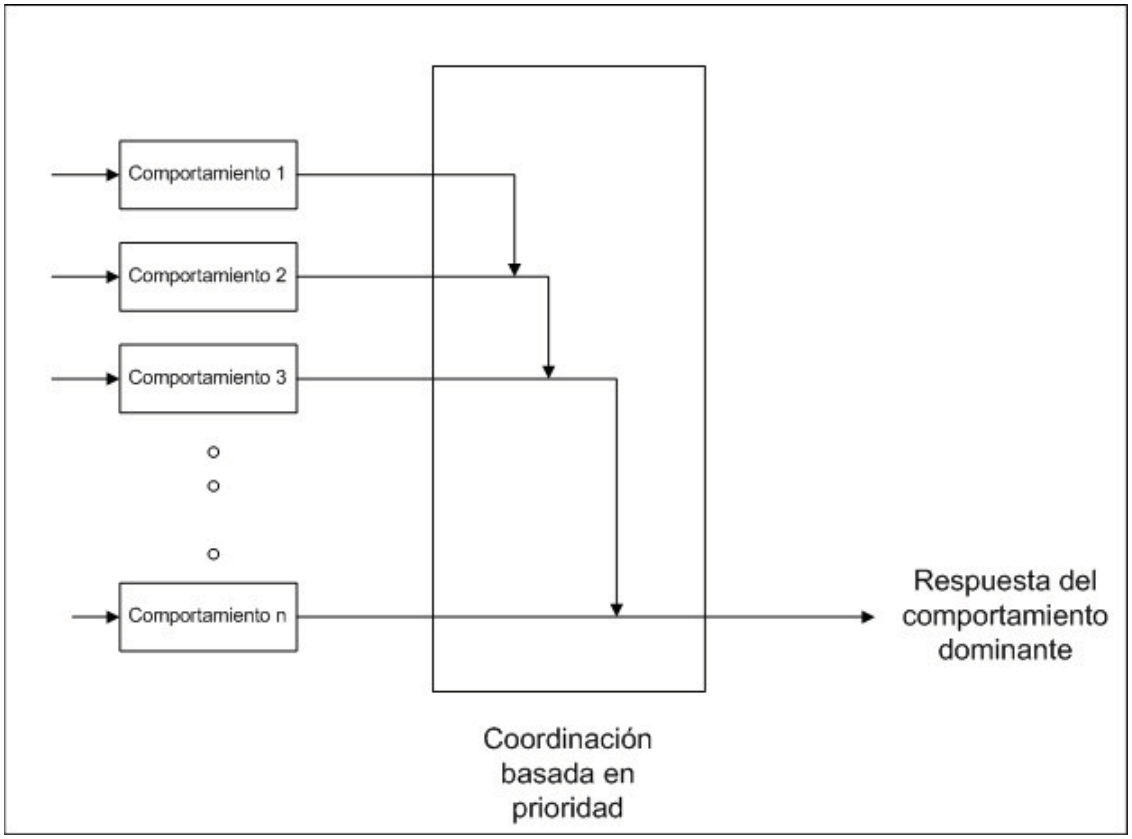

Figura 2.7: El comportamiento resultante será el dominante

tiene objetivos, este implementa un componente reactivo que puede presentarse en cualquier momento para hacer más eficiente la respuesta.

Ahora pensemos en una configuración básica en donde tenemos a un robot, este tiene conocimiento previo del mundo en el cual va a ser inmerso, además de presentar una arquitectura híbrida, pensemos en tres diferentes pruebas, primero en donde es colocado el inicio del mapa en un punto **A** con alguna orientación en el mundo, también es colocado un objeto extra, el cual tiene su centroide en en la posición (x1, y1). Para la segunda prueba se coloca otro objeto con en la posición (x2, y2). Para la tercer prueba se dejan ambos objetos, el robot tendrá las siguientes características:

- Conocimiento del mundo.

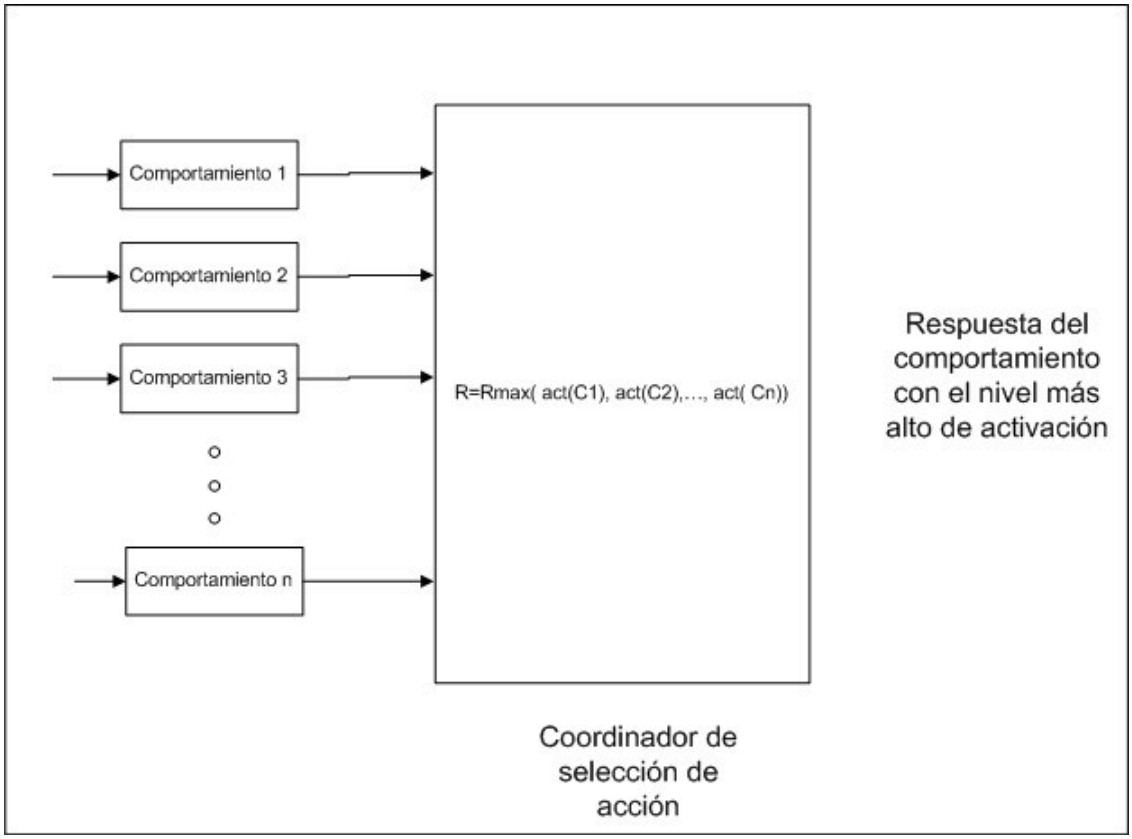

Figura 2.8: El comportamiento resultante será basado en el resultado de la evaluación

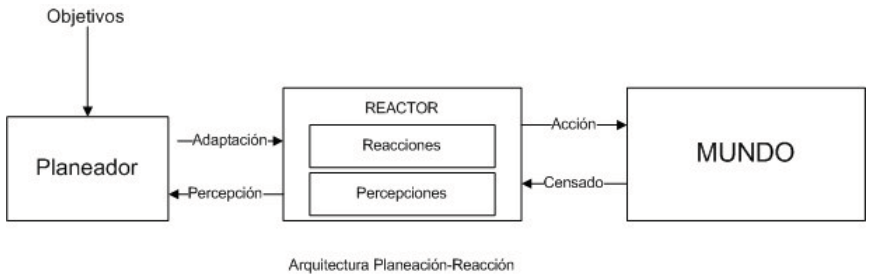

Figura 2.9: Arquitectura Planeación-acción

- El robot es de tipo cilíndrico y cuenta con 16 sensores distribuidos a la misma altura como se muestra en la figura 2.10 uniformemente alrededor.

- El robot deberá moverse del punto A al Punto B.
- El robot tiene la capacidad de aprender, es decir si encuentra un obstáculo nuevo será agregado a la estructura de su mundo.
- El proceso se repite 3 veces posicionándolo en el mismo lugar inicial y con la misma orientación.

Figura 2.10: Robot con 16 sensores.

El robot utiliza una arquitectura híbrida, en caso de que el objeto se presente en su camino guardará la posición del contacto y con ello el reconocimiento del objeto aproximadamente en (x1, y1). En el caso 2, nuevamente si el objeto detectara por medio de los sensores que existe un obstáculo nuevo tendría que guardar el lugar aproximado para reconocer que existe ahí un obstáculo, para la tercer lectura pueden presentarse diferentes situaciones:

- El robot evade correctamente los nuevos obstáculos y alcanza la posición destino (ideal)
- El robot choca en otra posición con uno o los dos obstáculos. En este caso el robot deberia reconocer nuevos puntos y agregarlos a la estructura del mapa.

En el peor de los casos el robot en una n-ésima prueba debería ser capaz de reconocer la posición de dichos objetos y evadirlos. Este es un ejemplo sencillo de la combinación de los tipos de comportamientos descritos.

Campos potenciales

Los campos potenciales son un ejemplo de comportamientos híbridos, estos presentan comportamientos reactivos de acuerdo al mundo, pero también presentan comportamientos deliberativos debido a que los cálculos de los vectores de atracción y repulsión se calculan en base a los objetos del mundo y posiciones del robot dentro del mundo. La idea básica dentro del comportamiento de robots resulta de ver estos como una reacción a estímulos externos que posiblemente son recibidos por medio de diferentes sensores electrónicos. Este concepto, de forma genérica podría verse en el diagrama de la figura 2.11.

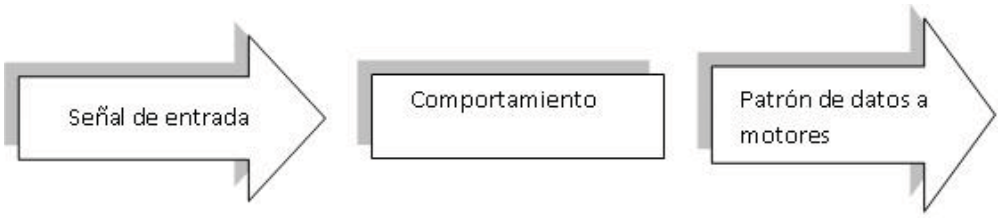

Figura 2.11: Diagrama general de bloques Entrada-Respuesta

La idea de campos potenciales en robótica tiene el mismo concepto manejado en electromagnetismo en donde se estudia a una partícula en un campo magnético, en este caso la partícula es el robot que se desplaza a través de campos de fuerzas generados por objetos que fungen como obstáculos los cuales actúan como repulsores al robot, el comportamiento del robot estará en base a dichos campos generados por estos diferentes obstáculos en el camino (figura 2.13).

Ahora el robot necesita tener un objetivo hacia el cual dirigirse, aquí es donde aparecen los campos atractivos. Para que el robot tenga una dirección a la cual dirigirse necesita tener este objetivo. Una forma de ver estos campos potenciales es como un conjunto de vectores en 2 dimensiones, lo primero es obtener la distancia y el ángulo entre el objetivo y la posición actual del robot. La distancia puede calcularse como la distancia entre dos puntos:

Definiendo el punto inicial como las coordenadas (x, y), mientras que para nuestro punto destino tendremos las coordenadas x_D, y_D al ser una representación de un par de puntos en el plano (ver figura 2.12). Podemos obtener la distancia entre los dos

puntos como:

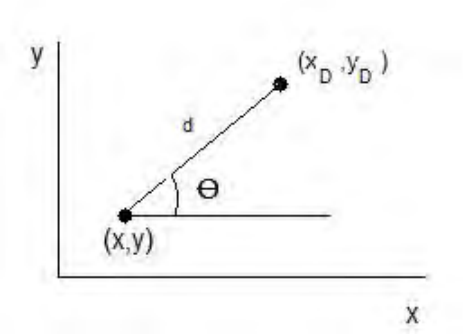

Figura 2.12: Representación del punto origen y destino en un plano

$$d = \sqrt{(x_D - x)^2 + (y_D - y)^2} \tag{2.1}$$

Donde (X_D, Y_D) es el punto destino al cual quiere llegar el robot.
Y para encontrar el ángulo con respecto a x tenemos:

$$\Theta = tan^{-1}\left(\frac{y_D - y}{x_D - x}\right) \tag{2.2}$$

Una vez teniendo esto, se deben sumar las fuerzas de atracción a las de repulsión para obtener la fuerza resultante (los campos potenciales se componen de fuerzas atractivas y fuerzas repulsivas que actúan sobre el robot) los vectores de fuerzas repusivos o atractivos son representados por q. Sumando ambas fuerzas obtenemos una fuerza resultante $\bar{F}$.

$$\bar{F}(\bar{q}) = \bar{F}_{atr}(\bar{q}) + \sum_{i=1}^{n}\left(\bar{F}_{rep}(\bar{q}_i)\right) \tag{2.3}$$

Las fuerza de atracción se calcula mediante la siguiente ecuación:

$$\bar{F}_{atr}(\bar{q}) = \begin{cases} \varepsilon_i(\bar{q} - \bar{q}_{dest}), \text{ si } \left|\bar{q} - \bar{q}_{dest}\right| \leq d_1 \\ \frac{\varepsilon_i(\bar{q} - \bar{q}_{dest})}{\left|\bar{q} - \bar{q}_{dest}\right|}, \text{ si } \left|\bar{q} - \bar{q}_{dest}\right| > d_1 \end{cases} \tag{2.4}$$

Y la fuerza de repulsión se calcula mediante:

$$\bar{F}_{rep}(\bar{q}) = \begin{cases} -\mu(\frac{1}{|\bar{q}-\bar{q}_{obst}|} - \frac{1}{d_0})(\frac{1}{|\bar{q}-\bar{q}_{obst}|^2})(\frac{\bar{q}-\bar{q}_{obst}}{|\bar{q}-\bar{q}_{obst}|}), \text{ si } \left|\bar{q}-\bar{q}_{obst}\right| \leq d_0 \\ 0 \text{ , si } \left|\bar{q}-\bar{q}_{obst}\right| > d_0 \end{cases} \tag{2.5}$$

De esta forma la posición siguiente del robot estará dado por:

$$\bar{q}_{i+1} = \bar{q}_i + \delta \bar{f}(\bar{q}_i) \tag{2.6}$$

En cada paso se realiza el cálculo de fuerzas. En la figura la ruta del robot esta representada por la línea gruesa y el círculo superior representa el destino en tanto que el circulo inferior representa un obstáculo (figura 2.13).

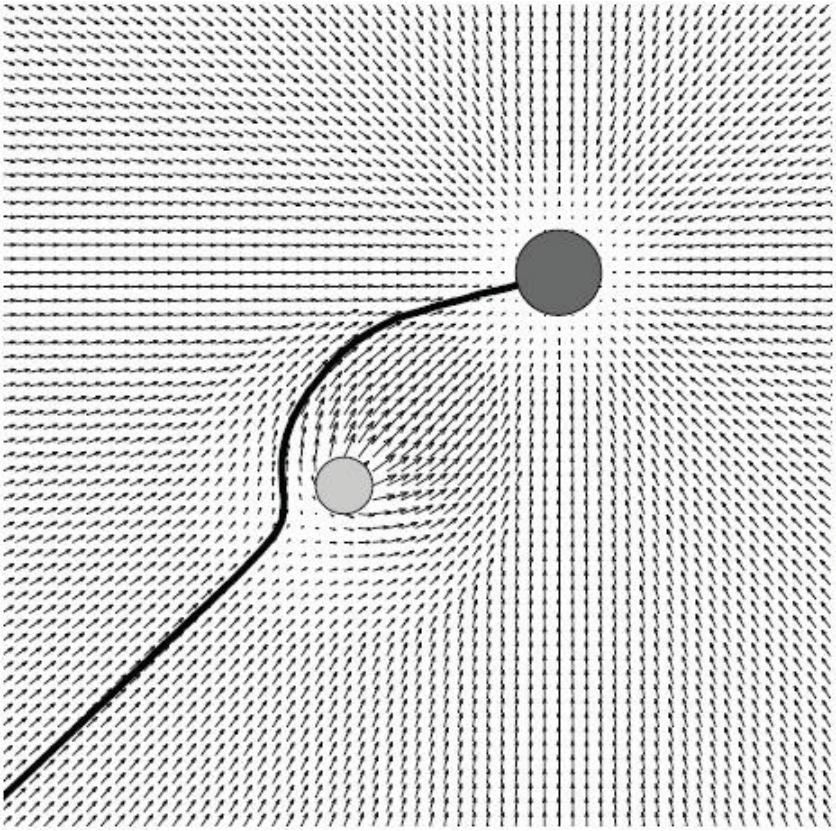

Figura 2.13: Campo vectorial con fuerzas atractivas y repulsivas para generar la ruta del robot

Capítulo 3

Comportamientos usando algoritmos genéticos

3.1. Algoritmos evolutivos

El estudio de los algoritmos evolutivos ha mejorado en varias versiones del algoritmo simple, pero todas tienen como base los mismos principios generales pero con sutiles diferencias unos de otros. Los más importantes son:

- Algoritmos genéticos.
- Programación genética.
- Estrategias evolutivas.
- Programación evolutiva.

Además dentro de cada categoría existen diferentes versiones, con ello podemos darnos cuenta de la popularidad que ha tenido esta técnica para la resolución de un buen número de problemas de muchos tipos.

3.2. Algoritmos genéticos

En los algoritmos genéticos las variables son codificadas como cadenas de genes, normalmente consisten en números binarios o decimales discretos. Inicialmente se genera una población que consiste en estas cadenas, Holland[1] en su libro *Adaptation in natural and artificial systems* realiza una propuesta en donde son seleccionados algunos individuos y estos son cruzados, este trabajo es retomado por Goldberg [2] en su publicación: *Genetic algorithms in search* donde propone una manera relativamente simple de seleccionar individuos y de realizar la cruza de forma metódica llamándolo Algoritmo genético simple (AGS); el algoritmo simple es el siguiente:

1. Decidir cuales son los parámetros del problema a resolver.
2. Generar aleatoriamente una población inicial de n posibles soluciones, dependiendo del problema a resolver.
3. Calificar cada posible solución de la población actual.
4. Seleccionar dos individuos de la población actual con una probabilidad proporcional a su calificación (Método de la ruleta)[3]
5. ejecutar la cruza con una probabilidad de 50 %.
    - Si en el experimento tenemos que realizar la cruza (ver figura 3.1) obtenemos los nuevos individuos para formar dos híbridos, estos serán llamados nuevos individuos.
    - Si en el experimento no realizamos la cruza dichos individuos pasan a ser los nuevos individuos.

[1] Holland [11]
[2] Goldberg [13]
[3] Este método es por medio de la suma de calificaciones de todos los individuos de la población y esta suma es considerada al 100 % de una circunferencia, entonces cada individuo tiene un valor de probabilidad igual a su calificación dividido entre dicha suma.

6. Por cada bit de cada nuevo individuo realizar la mutación con una probabilidad de mutación Pm.
    - Si el experimento indica que debemos realizar la mutación entonces cambiar el bit en turno por su complemento.
    - En caso contrario no se realiza la modificación del bit.
7. Incluir los nuevos individuos en una nueva población.
8. Si la nueva población ya tiene n individuos será renombrada a población actual. y se regresa al paso 3, a menos que sea cumplida alguna condición de terminación.
9. Si no regresar a paso 4.

En el paso 8 se hace referencia a una condición de paro, esta puede ser definida de muchas formas, ya sea fijando un número máximo de generaciones, o que cierto número de individuos de una generación dada tengan un *fitness* aceptable, es decir si se decide que 85 individuos de una generación de 100 (85 %) tienen un fitness aceptable o ideal se termina el proceso, inclusive pueden ser otras evaluaciones, eso depende del problema o de la decisión de quien implemente el algoritmo.

Cuando se realiza la cruza de dos individuos, es de diferentes formas, en esta es necesario mezclar los cromosomas de ambos individuos para crear híbridos, uno de los más importantes es por medio de la cruza en un punto (ver figura 3.1). Este tipo de algoritmo tiene las siguientes características:

1. Tiene un número fijo de individuos por generación.
2. Selección proporcional.
3. Cruzamiento en un punto, todos los individuos tienen la misma probabilidad de cambiar.
4. Mutación uniforme, todo el cromosoma tiene la misma probabilidad de mutación.
5. Selección no elitista, individuos son copiados de una generación a otra sin selección aleatoria.

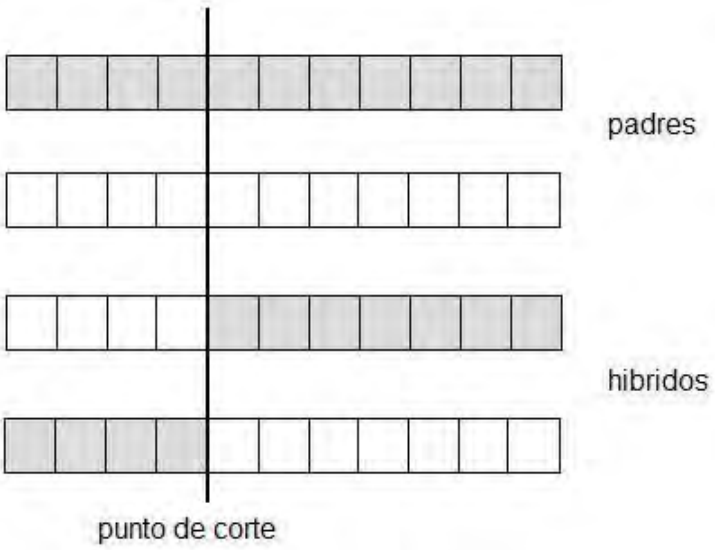

Figura 3.1: Cruzamiento en un punto

Existen muchas variaciones al algoritmo original, pero básicamente los pasos anteriores son los fundamentales. Las variedades en los diferentes algoritmos son en cuanto a la selección de los individuos, a veces son seleccionados sólo los mejores individuos para generar un híbrido, otras veces se seleccionan el mejor y peor individuo para tener un híbrido con las mejores y las peores características. En fin, son muchas las modificaciones que se han realizado al algoritmo original y estas son hechas de acuerdo a los requerimientos de su implementación.

3.3. Programación genética

En la programación genética las poblaciones son programas de computadora, generación a generación realiza estocásticamente transformaciones en ellos para que se obtengan como resultado nuevos programas y obviamente mejores a los originales. En procesos aleatorios como es normal pueden obtener resultados no deseados en ocasiones, pero la programación genética es en esencia aleatorio. El programa genético básicamente realiza la evaluación del nuevo programa obtenido probándolo, entonces el comportamiento es comparado con alguno ideal. Las operaciones genéticas pri-

marias para crear nuevos programas de los existentes son:

- Cruza. la creación de programas hijos combinando aleatoriamente partes elegidas de dos programas padres seleccionados.
- Mutación. La creación de nuevos programas hijos aleatoriamente y realizando modificaciones también aleatoriamente a partes del programa seleccionado.

Los pasos básicos son los siguientes:

1. Aleatoriamente generar una población inicial de programas de primitivas disponibles.
2. repetir.
   - Ejecutar cada programa y evaluar su fitness.
   - Seleccionar uno o dos programas de la población con una probabilidad basada en su fitness para participar las operaciones genéticas.
   - Crear nuevos individuos de programas aplicando operaciones con probabilidades específicas.
3. Hasta que la solución más aceptable sea encontrada o sea cumplida alguna condición de paro.
4. regresar el mejor individuo.

En este caso el operador de cruza difiere de lo que se utiliza en el algoritmo genético, en el se selecciona de forma aleatoria el punto de cruza, mientras que en la programación genética localidades son intercambiadas, también el operador de mutación puede cambiar en los operadores.

3.4. Estrategias evolutivas

Las estrategias evolutivas han resultado de combinaciones entre si, en sus orígenes las estrategias evolutivas utilizaban una codificación de números reales y los nuevos individuos se formaban haciendo uso del operador de mutación, considerando por simplicidad una función de maximización dado por la función g = g(x_1, x_2, ... x_n), la

estrategia evolutiva opera de la siguiente forma: la población consiste en un solo individuo el cual se representa por medio de un par de vectores $\{(x_1,x_2,.....x_n),(\sigma_1,\sigma_2,.....\sigma_n)\}$, donde $(x_1, x_2, ... x_n)$ son las variables del problema y $(\sigma_1,\sigma_2,.....\sigma_n)$ el vector de varianza el cual se usa para formar nuevos individuos. Un individuo es evaluado computando $g(x_1, x_2, ... x_n)$, entonces el vector $\{(x_1,x_2,.....x_n),(\sigma_1,\sigma_2,.....\sigma_n)\}$ es mutado de acuerdo a

$$x_i \text{ -> } x_i = x'_i + N(0,\sigma_i),\ i=1,2,....,n$$

donde $N(0,\sigma_i)$ es una función aleatoria gaussiana con valor esperado 0 y desviación estándar σ_i. Cuando se encuentra el vector $(x'_1, x'_2, ... x'_n)$ los nuevos individuos se encuentran evaluando $g(x'_1, x'_2, ... x'_n)$ si el nuevo valor es mejor que el anterior entonces el individuo anterior es reemplazado, en caso contrario se genera una nueva mutación.

3.5. Programación evolutiva

Esta fue desarrollada en los 60's con la generación de las máquinas inteligentes, inicialmente se generaron haciendo uso de máquinas de estados finitos, pero con el paso de los años fue evolucionando o actualizandose para usarse como método de solución para problemas complejos.

En la programación evolutiva se genera una población aleatoria de máquinas de estados finitos, usualmente con algunas limitantes en los tamaños de las máquinas de estados. La tarea principal de la programación evolutiva es proporcionar la mejor acción posible dada la información de eventos previos; los eventos consisten en la representación de símbolos de entrada de un alfabeto dado. Como respuesta a una entrada la Máquina de estados finitos produce una salida, la cual puede ser por ejemplo un movimiento de un robot. El desempeño de las máquinas de estados es evaluado y nuevas máquinas de estado son generadas por medio de la selección y la mutación, el operador de mutación puede tener diferentes efectos: puede cambiar el estado inicial, las condiciones de transición, y también el número de estados; el programa evolutivo es reactivo a alguna señal del exterior, y es debido a esto que se genera la mejor respuesta.

3.6. Métodos para aprendizaje y comportamientos

El aprendizaje indica que los comportamientos serán modificados en un proceso de tiempo finito en donde momento a momento el estado actual n debería presentar un mejor comportamiento que el comportamiento n-1 y así sucesivamente. En ocasiones esa escala de tiempo no es muy pequeña, esto depende de la función de evaluación y de las características del problema. Como se ha indicado anteriormente el uso del concepto de algoritmos evolutivos tiene como referencia central la noción de población, que significa que tenemos un conjunto de individuos del mismo tipo (en donde estos pueden tener parejas y tener descendencia); generalmente el número de individuos generación a generación es constante y tienen un tiempo de vida, como en el caso de los entes biológicos: nacen, se reproducen y mueren, cada elemento de una población es conocido como *individuo*, dependiendo de la cruza y mutación, de cada generación será tan diferente una de las otras.

Una de las ideas centrales en la teoría evolutiva es la del cambio por *herencia* gradual, las nuevas generaciones deberán presentar una mejor adaptación de acuerdo a los ambientes en donde sean evaluados los nuevos individuos; el concepto de herencia (donde nuevas propiedades pueden ser transmitidas a las siguientes generaciones cuando se reproducen), pero, ¿como puede ser que un individuo herede información de otro individuo? la respuesta es por medio del genoma, en los animales consiste en muchos cromosomas que contienen genes, estas son las unidades de la herencia y tendrá codificada la información necesaria Cada gen contiene una secuencia de bases en cromosomas (o moléculas de ADN), estas pueden ser denotadas denotadas por medio de A,C,G y T la información genética se almacena como una secuencia de estas bases, en el tiempo de vida de un individuo el ADN es leído por enzima para producir bloques de proteínas.

Cada uno de los genes tiene un propósito por ejemplo algunos son los encargados del color de piel, de ojos, color y tipo de cabello, estos son conocidos como alelos, durante su desarrollo la información almacenada en los genes es decodificada resultando en un individuo que posee los rasgos codificados del genoma, a los individuos con todos los rasgos se le conoce como fenotipo correspondiente del genotipo.

Otros dos conceptos centrales en la evolución tenemos el fitness (aptitud) y la selección para llevar a cabo la reproducción; estos conceptos resultan bastante sencillos de entender. Normalmente un individuo mejor adaptado a su ambiente es debido a que es más fuerte o mas inteligente que los demás y también tiene más oportunidades

de reproducirse dando como resultado más individuos con sus características. La reproducción es el centro de la evolución, los cromosomas de dos son combinados obteniendo unos genes de un padre y otros del otro proporcionando las características aprendidas al nuevo o nuevos individuos, y como no todo es tan perfecto existen errores, es aquí cuando se presenta la mutación de los nuevos individuos los cuales contienen nueva información para el proceso evolutivo.

Para resumir, la evolución es un proceso que actúa en poblaciones de individuos, la información se almacena en los individuos dentro de los cromosomas, los cuales son agrupaciones de genes, los individuos mejores adaptados son los que tienen una aptitud más alta y a su vez son los que presentan más posibilidades de cruza.En la reproducción los individuos heredan a sus descendientes las cualidades aprendidas o heredadas y para agregar un elemento aleatorio tenemos el caso de la mutación la cual proporciona nuevas características al proceso evolutivo, El proceso evolutivo de ninguna manera es un método aleatorio de búsqueda, las mutaciones son aleatorias, pero la selección no lo es, a esto diferentes investigadores del área biológica lo llaman *evolución dirigida*.

Capítulo 4

Desarrollo de comportamientos

Como fue mencionado anteriormente la idea principal del problema es llevar a cabo la generación de individuos con ciertas características las cuales serán evaluadas uno a uno en cada generación para obtener los individuos mejor adaptados a un ambiente dado. De inicio los individuos estarán representados por un grupo de cromosomas los cuales serán indicativos del comportamiento y mediante la minimización de una función de evaluación ya sea por medio de un cierto límite de generaciones o por medio de alguna evaluación interna se determinarán las condiciones de paro del proceso evolutivo.

4.1. Campos potenciales

Como condiciones iniciales tendremos las siguientes características:

- Haremos uso del simulador de robots llamado Virbot[1] para evaluar gráficamente el comportamiento del robot.
- Tendremos conocimiento del mundo (con esto podemos generar los campos potenciales obteniendo las fuerzas repulsivas que pudieran existir en el ambiente tales como paredes u obstáculos). También conociendo el mundo se generarán de forma aleatoria algunos puntos origen y destino, los cuales no pueden ser creados dentro de los obstáculos.
- El simulador en base a las lecturas de los diferentes sensores conocerá en todo momento la posición actual del robot.
- Proporcionaremos al sistema el número de generaciones y el número de individuos por generación, así como un par de valores que representan un valor mínimo y un valor máximo para la creación de valores representativos de las constantes.

Las poblaciones deberán contener un cromosoma o representación del mismo que de las siguientes constantes: $d0$, $d1$, $\varepsilon 1$, $\varepsilon 2$ y η (definidas en el capitulo 2). Estas constantes serán evaluadas y al mismo tiempo serán calificadas de acuerdo a la distancia alcanzada por el robot desde su posición inicial a su destino. Esta calificación será su calificación o *fitness* de cada individuo, además de que también tendremos el número por cada generación, el fitness será primordial para aplicar a alguno de las variantes al algoritmo genético y será la variable que indique si el individuo continua en la siguiente generación o es desechado. El algoritmo básicamente será el siguiente:

1. Proporcionar los datos de entrada:
   - Número de individuos.
   - Número de generaciones.
   - Posición inicial.
   - Posición destino.
   - Intervalo de valores mínimo y máximo.
   - Probabilidad de cruza Pc.

[1] Savage [17]

- Probabilidad de mutación Pm.

2. Inicializar variables del simulador.
3. Generar la población inicial.
4. Evaluar individuo a individuo su aptitud en el ambiente:
   - evaluar las constantes del individuo actual.
     - Evaluar las constantes en el individuo hasta que llegue a la posición destino o se cumpla cierto número de pasos.
     - Obtener la distancia del punto final (alcanzado por el robot) al punto destino esta será su calificación de fitness o de aptitud.
   - Seleccionar a los individuos por medio del método de la ruleta.
   - aplicar cruza en caso de que aplique de acuerdo a cierta probabilidad.
   - aplicar mutación de acuerdo a cierta probabilidad.
   - obtener la nueva generación de individuos.

   El proceso se ilustra en la figura 4.1.

El programa internamente hará uso de un archivo de generaciones en donde se tendrá la representación real del cromosoma de cada uno de los individuos y su representación de las constantes *d0*, *d1*, $\varepsilon 1$, $\varepsilon 2$ y η a manera de ejemplo la siguiente tabla:

d0	d1	$\varepsilon 1$	$\varepsilon 2$	η	aux	paso	fitness
43.496265	5.990670	36.866016	28.263643	31.898247	11.604525	0	10000.000000
15.210112	32.634804	5.146708	6.492622	15.471784	18.630983	0	10000.000000
38.229530	39.941837	26.030890	35.776100	35.262886	9.813074	0	10000.000000
30.461653	27.899569	4.650660	18.726416	10.577366	0.128688	0	10000.00000
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

La primer generación en todos sus individuos comienza con una distancia muy grande 10,000.00, pero conforme avanzan las generaciones este fitness va reduciéndose hasta alcanzar una valor aceptable. El diagrama de componentes de la figura 4.2 muestra la interacción del sistema con el generador de población inicial.

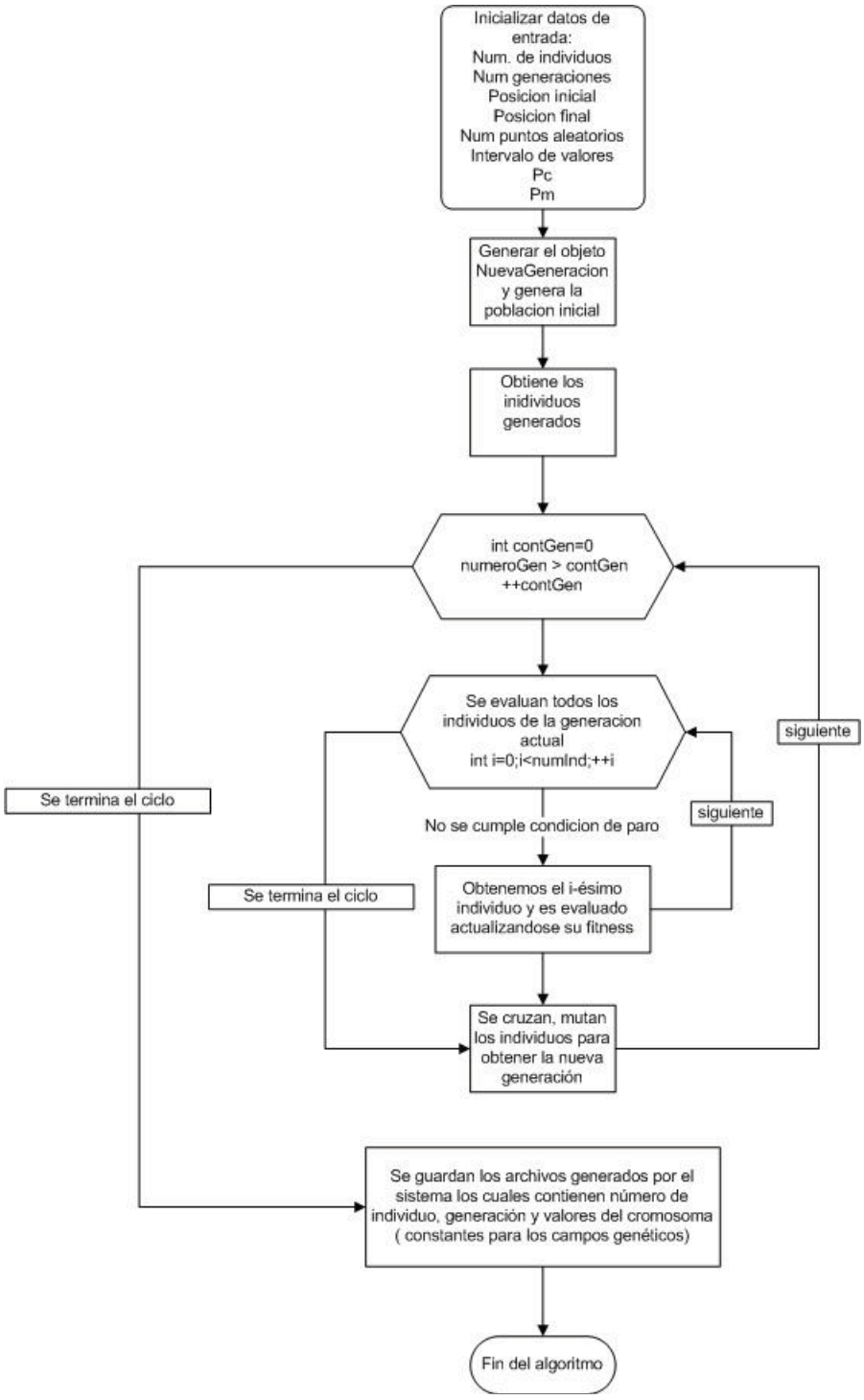

Figura 4.1: Diagrama de flujo del algoritmo genético.

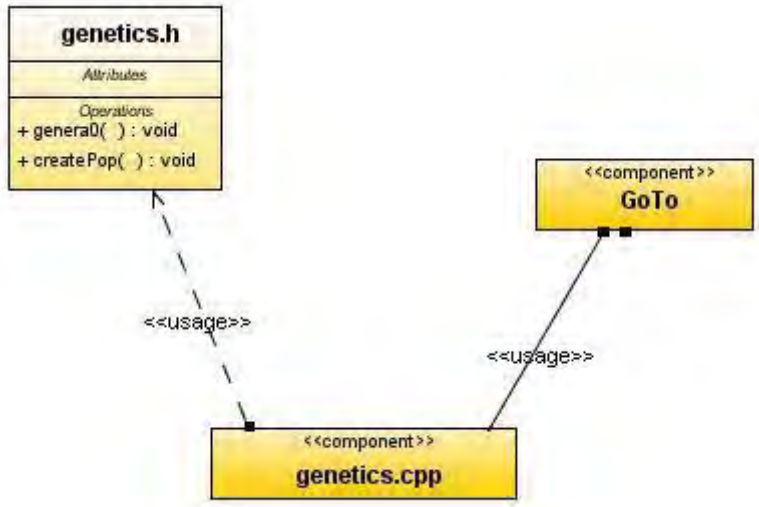

Figura 4.2: Diagrama de componentes para el generador de población inicial.

El sistema hará uso de las funciones contenidas en *genetics.h* generando el archivo indicado anteriormente.

Para el proceso iterativo en el cual se realizará una evaluación de cada individuo para calificarlo y de acuerdo a la posición inicial del robot y su posición final se calculará la distancia en donde quede el robot; esta distancia será la aptitud de ese individuo o su *fitness*. Así los mejores individuos serán aquellos que tengan un *fitness* mas pequeño (lo cual sería indicativo de que el robot con esos valores se aproximó más al destino). Nuevamente se hace uso del objeto generado de la clase NuevaGeneracion la cual es la encargada de los procesos de evolución de los individuos. En la imagen puede apreciarse de forma general la arquitectura del sistema de genéticos (figura 4.3). La clase Cromosoma contiene, tres constructores uno de ellos sin parámetros, otro que recibe como parámetro un valor entero y otro que recibe un valor flotante. Esta clase contiene cuatro diferentes métodos, getFloatFromInt que nos sirve para obtener el valor de la representación flotante utilizando la notación descrita en la figura 4.4 del valor que recibe como parámetro y de forma análoga el método getIntFromFloat nos regresa el valor entero del flotante recibido como parámetro. Por otro lado si se hace uso de los constructores que reciben un parámetro podemos hacer uso de los métodos getIntValue y getFloatValue, los cuales nos regresan el mismo número pero con la

representación flotante indicada en la figura anterior.

NuevaGeneracion contiene algunos métodos que son usados por el programa, getNewInds nos sirve para obtener la nueva generación de individuos después de que se presentó una evolución, getIndividuos nos permite obtener la primer generación, generaR nos genera número aleatorios, createNewPop, genera la población inicial, crossover realiza la cruza de individuos, mutation muta con alguna probabilidad a los individuos, findBest nos regresa el mejor individuo.

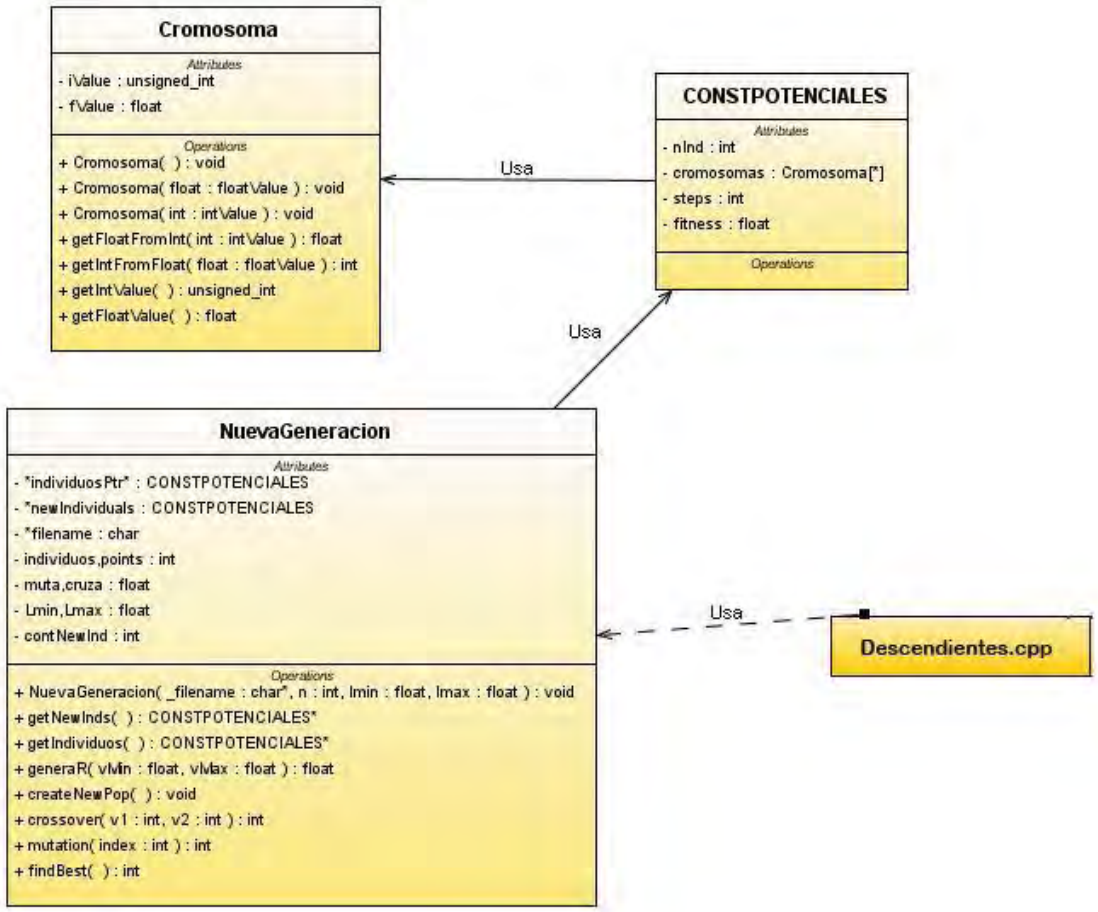

Figura 4.3: Diagrama de clases para el algoritmo genético

Como puede apreciarse la clase Descendientes hace uso de objetos creados a partir de NuevaGeneracion y esta a su vez utiliza de la estructura CONSPOTENCIALES que contiene un arreglo objetos de tipo Cromosoma, el uso de cada uno de estos componentes es detallado a continuación:

Cromosoma Tiene la función de realizar la codificación de las constantes haciendo uso de una notación de punto flotante variable, es decir que nosotros podemos tener una representación de un valor flotante y para el sistema manejaremos valores enteros. Una vez que nosotros busquemos obtener el valor flotante nuevamente podemos obtenerlo sin problemas por medio de su método. La codificación del cromosoma (representa el valor de las constantes) se trabajó representado por una variable entera sin signo, la cual tendrá 8 bits para la fracción y 24 para la representación entera; de esta forma los números máximos a ser representados en esta notación son 2^{-8} y 2^{24} respectivamente, (ver figura 4.4). Esta codificación puede ser variable (se pueden modificar los valores de la parte entera y la parte flotante del cromosoma, para el sistema se manejó que de los bits 8 al 31 tuvieramos la representación de la parte entera del valor, mientras que para la parte flotante la tenemos del bit cero al bit 7.

Figura 4.4: Representación del cromosoma parte entera (color azul) y la representación de la parte fraccionaria (color gris)

CONSTPOTENCIALES Contiene un arreglo de objetos cromosoma (que a su vez representan las constantes por cada individuo, como ya se ha indicado anteriormente), tiene un valor representativo del número del individuo, así como su *fitness* (aptitud) además del número de pasos por individuo.

NuevaGeneracion Esta es el corazón del algoritmo genético, esta contiene un arreglo dinámico de estructuras de tipo CONSTPOTENCIALES, nombre del archivo de generaciones (indicado anteriormente), así como los valores de mutación, cruza número de individuos, los limites inferior y superior para los valores que son generados de las constantes, entre otros. Tiene también los métodos necesarios para ejecutar los procesos de los algoritmos genéticos, como pueden ser un *getter* para obtener el apun-

tador a la posición inicial del arreglo de individuos,el método para generar una nueva población evaluando el fitness y utilizando los métodos de cruza de individuos y mutación (de acuerdo a los valores guardados por las variables de clase). También tiene métodos auxiliares para la impresión en un archivo de la población actual. Una vez que analizamos de forma genérica el diagrama de componentes, procederemos al proceso de ejecución del sistema:

Inicialmente debemos elegir la posición que servirá como origen del robot y la posición destino; esto podemos hacerlo de dos formas una pasándolo directamente como parámetro al programa GoTo_GENETICS generado a partir del diagrama de componentes mostrado en la figura 4.3. Además podemos hacer uso del entorno gráfico del Virbot (este internamente hace el llamado al programa GoTo_GENETICS, es más intuitivo su uso). Para obtener dichos puntos gráficamente, con el botón izquierdo podemos generar la posición inicial, y con el botón derecho podemos generar la posición destino. Por ejemplo en la figura 4.5 tenemos la posición inicial indicada por la letra A, y la posición final o letra B, los valores de las coordenadas de acuerdo a dicho mapa son datos que sirven de entrada al sistema.

Para dicho ejemplo los valores (x, y) de la posición inicial son A(111,57) y para la posición destino B(244,555) (como en muchos sistemas gráficos la posición inicial del canvas esta en la parte superior izquierda). Estos, junto al mapa y el tipo de sensor son datos de entrada al sistema, una vez que tenemos estas condiciones ya podemos generar nuestra población inicial. Una vez generado el archivo de la población inicial podemos generar la instancia de la clase la clase NuevaGeneracion la cual recibe 4 parámetros, la ruta en la cual se almacenaran los archivos generados por el sistema, el número de individuos y los parámetros mínimo y máximo de las constantes manejadas por el individuo. La instanciación de NuevaGeneracion se realiza de la mediante:

```
NuevaGeneracion nuevaGeneracion(PATH_GEN, numIndividuos, vMin,vMax );
```

Los nuevos individuos serán almacenados en un arreglo dinámico de tipo CONSTPOTENCIALES este es una estructura que maneja los valores de las constantes así como el fitness del individuo, y una vez que fue generada la población inicial (guardada en un archivo) se procede a obtener dichos individuos.

```
CONSTPOTENCIALES *individuosPtr= nuevaGeneracion.getIndividuos();
```

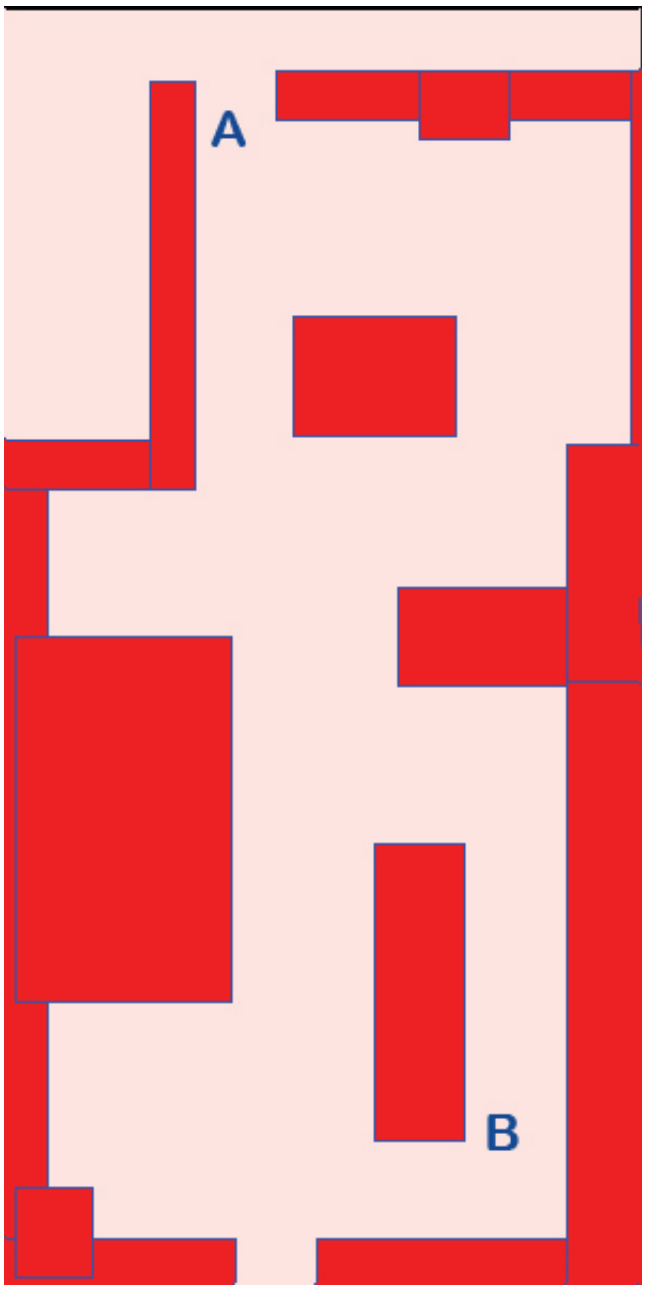

Figura 4.5: Mapa de ejemplo la posición inicial indicada por la letra A y la posición final etiquetada por B.

individuosPtr es un apuntador a un arreglo de objetos de tipo CONSTPOTENCIALES, estos contienen todas las constantes a evaluar, cada unas de estas constantes estarán contenidas en un arreglo de objetos interno de tipo Cromosoma, el cual es el encargado de realizar la codificación de tipos (como se mencionó anteriormente).

Estos individuos son evaluados uno a uno usando los valores de las constantes generadas como entrada. Dichas constantes afectan directamente el comportamiento del robot.

El siguiente paso fue generar a los individuos (primera generación), después evaluar esos individuos y quedarse con los que cumplieron la meta (llegaron al punto final desde el inicial utilizando campos potenciales). Posteriormente se vuelven a generar nuevos individuos de acuerdo a alguna probabilidad (esa probabilidad nos indica si el individuo se vuelve a generar); después también necesitamos evaluar si el individuo nuevamente de acuerdo a alguna probabilidad presenta algún tipo de mutación o se queda tal y como se encuentra. Este proceso se hace de acuerdo a unas constantes utilizadas en el sistema; este se detiene hasta que se obtienen todas las generaciones y un archivo es generado el cual contiene los mejores individuos por generación aquellos cuyo fitness es muy cercano a cero (lo cual nos indica que alcanzaron el objetivo).

El resultado de este proceso lo vemos a continuación en la imagen 4.6:

En el Virbot se puede analizar el comportamiento que presenta individuo a individuo después de que un alguno es generado, se podemos analizar su comportamiento, un individuo que quizás no logro su objetivo o como en este caso alguno u algunos de ellos que alcanzaron la posición destino, el sistema genera archivos con extensión raw, los cuales contienen la posición del robot paso a paso así como las lecturas del sensor dependiendo el tipo seleccionado, así como su posición en el mapa; con estos datos es posible obtener las rutas que siguió el robot utilizando las constantes del individuo i-ésimo, para evaluar gráficamente el comportamiento generado.

4.1.1. Mejorando los comportamientos en el mapa

Para tener una versión mejorada de los individuos generados a partir de una posición inicial y una destino se agregó también la posibilidad de generar puntos aleatorios en el mapa. Estos puntos aleatorios deben ser generados en lugares del mapa que no sean

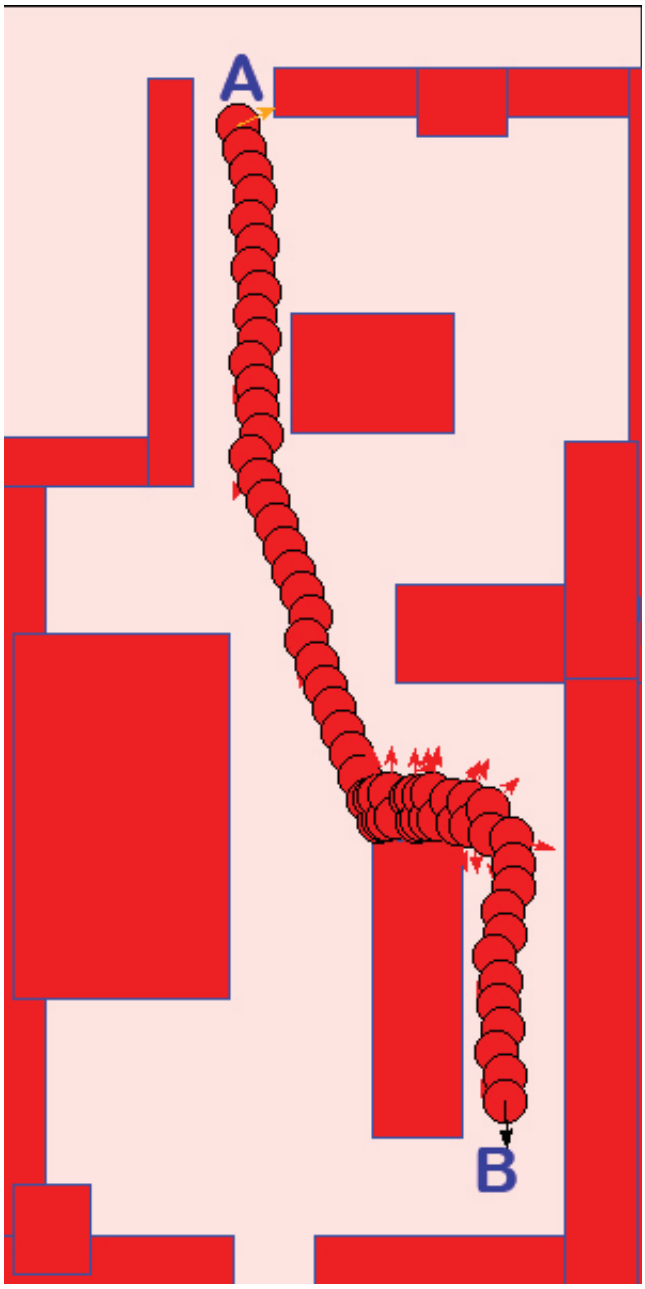

Figura 4.6: Comportamiento en el simulador del robot con un individuo exitoso, logrando pasar del punto A, al punto destino B, resultado obtenido por el algoritmo genético

paredes u obstáculos, por lo que se generó una clase Map la cual tiene los métodos necesarios para generar puntos aleatorios. Estos puntos son evaluados internamente por un método privado que revisa si el punto generado se encuentra dentro de algún obstáculo en caso de que sea este el caso vuelve a generar el punto y así hasta que encuentra un punto que no pertenezca a objetos dentro del ambiente, una vez que se obtienen se usan como posición inicial y posición final. En el diagrama de flujo de la figura 4.7, pueden apreciarse las modificaciones al algoritmo original.

4.2. Máquinas de estados

Otro análisis de comportamiento a analizar es el de máquina de estados; la principal idea es evolucionar la máquina de estados. Inicialmente la idea básica de la propuesta es representar las máquinas de estados como listas, normalmente tenemos la siguiente representación presentada en en la figura 4.8, donde tenemos un árbol de estados, pero de acuerdo a los valores que se obtienen en las bifurcaciones siempre tenemos en realidad una ruta a seguir, por lo que el árbol podría verse como una lista de estados a ejecutar de forma consecutiva (ver figura 4.9). Estos estados serán usados por el sistema, y por medio de listas ligadas obtendremos los estados consecutivos a ejecutar. Como en el caso de los algoritmos genéticos también necesitamos ciertas condiciones iniciales, estas son nuevamente la posición inicial, posición destino además del mapa y tipo de sensor, nuevamente necesitamos valores para limitar al número de individuos, generaciones y en este caso el número máximo de estados.

Cuando se elige una ruta en una máquina de estados esta puede verse como un listado siempre tendrá un estado inicial (raíz) y un nodo terminal (hoja) por lo que si vemos el caso mostrado en el ejemplo 4.9 podríamos ver que siempre y en todos los casos donde tenemos máquinas de estados, se cumple esta condición, esto podemos manejarlo para hacer uso de esta característica y resolver el problema manejando los estados como una cadena de estados consecutivos.

Una vez que tenemos los estados representados como listas ligadas en donde cada estado representa una acción que deberá realizar el robot, este estado en caso de no ser uno final o inicial, tendrá siempre un estado anterior y uno posterior. Inicialmente debemos tener estas listas consecutivas y con propiedades individuales, es decir, se agregó que cada estado deberá contener una *magnitud* y una *dirección*, así como su número de estado y el número del estado siguiente y anterior (esto último sólo es

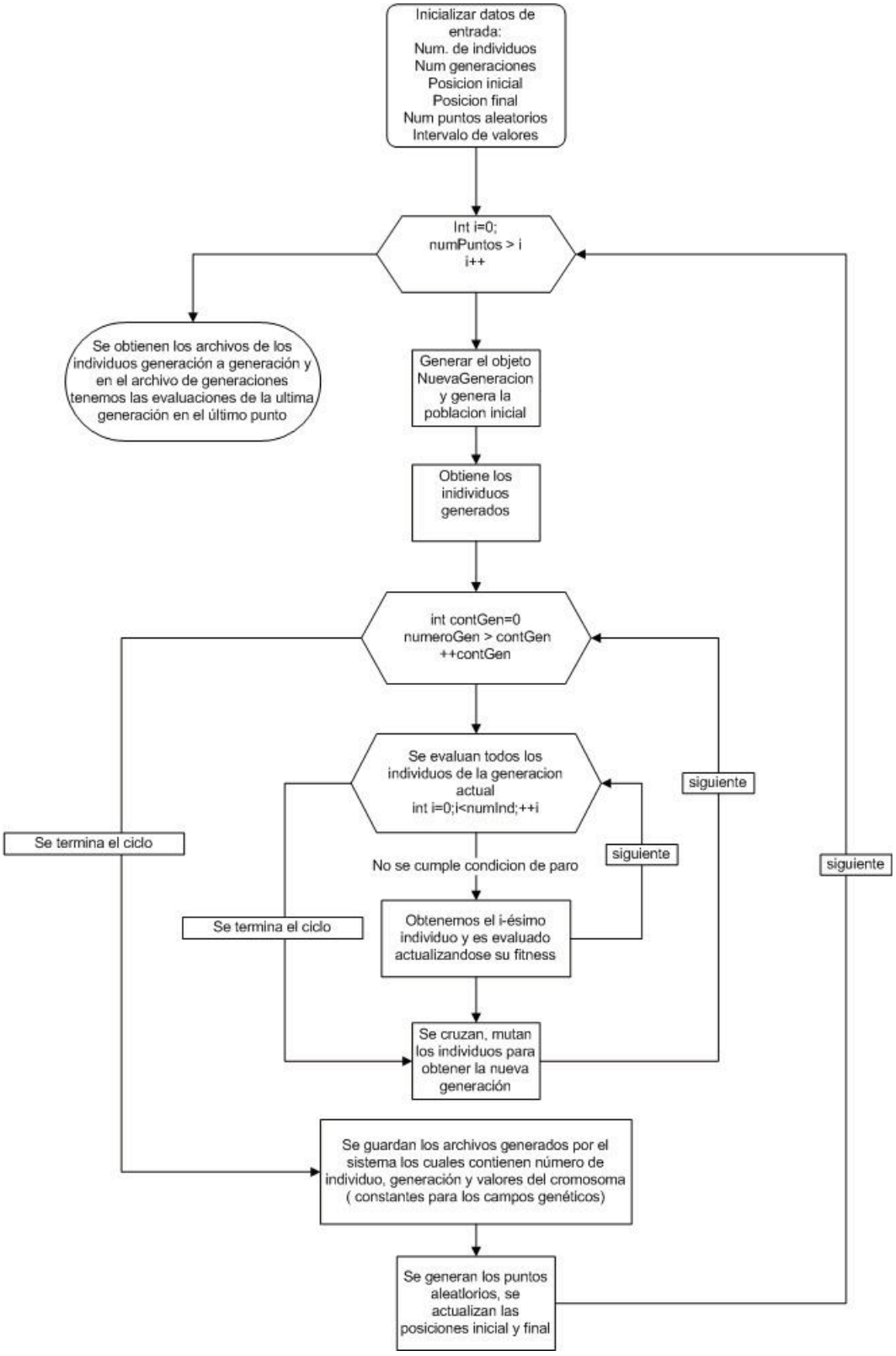

Figura 4.7: Diagrama de flujo del algoritmo genético modificado

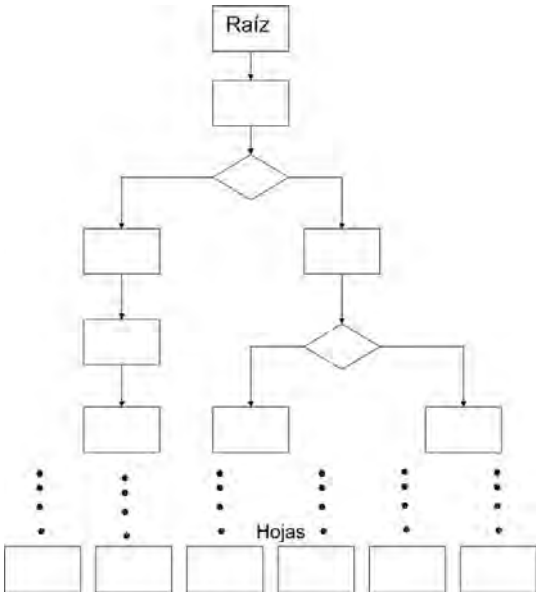

Figura 4.8: Máquina de estados genérica.

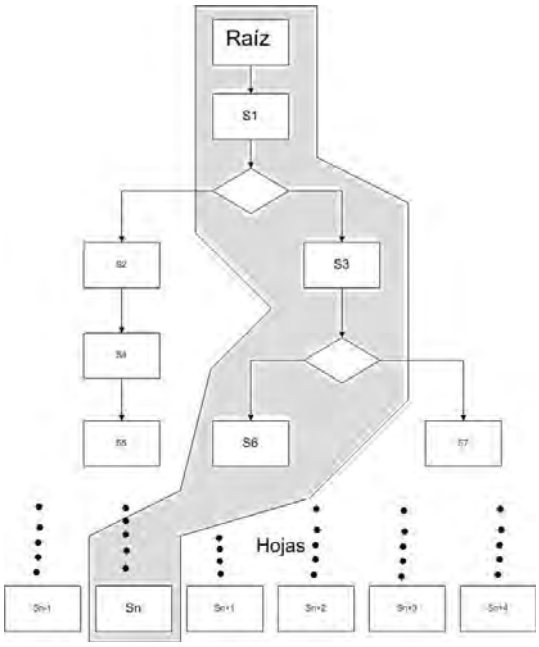

Figura 4.9: Ruta de estados seguido por un camino cualquiera visto como lista.

necesario cuando se desea hacer una cruza de listas). Una vez que tenemos esto necesitamos la característica de cada individuo, cada uno debe tener un parámetro de aptitud (fitness) la cual nos indicará si esa máquina de estados es buena o no. También necesitamos un número de individuo para saber cual es el individuo actual y que posición tiene. Además del listado indicado líneas atrás, el diagrama de clases mostrado en la figura 4.10 podemos ver la arquitectura genérica del sistema.

Las clases State y StateMachineInd son *beans*[2], la clase LinkedListGenetics contiene una buena cantidad de métodos, tiene dos constructores, getNumInds() nos regresa el número de individuos, getInds() nos regresa la lista de individuos (de tipo StateMachineInd). El método imprimeEstadosIndividuos(), imprime los estados de individuos a línea de comandos, el método initGenerations() genera la población inicial, generaDatoAleatorio(..) nos permite generar números aleatorios utilizando los parámetros que recibe como límite inferior y límite superior, addStateInitial(..) y addStateFinal(..) nos permite agregar estados al inicio y al final de la lista respectivamente, deleteSubList() recibe como parámetro una lista y le recorta a la misma algunos elementos indicados por los parámetros que sirven como límite inferior y superior, regresando la lista restante, addSubList(..) nos permite agregar una lista a otra desde cierta posición indicado por el parámetro, imprimeListGenetics() nos permite imprimir los estados a linea de comandos para observar el comportamiento de los mismos. Los métodos mutation(..) y cruzaListas(..) permite hacer las mutación del algoritmo y la cruza de los estados (de acuerdo a cierta probabilidad), y ejecutaEvoluciónNietzsche() realiza la evolución de la generación actual.

La clase principal es LinkedListGenetics la cual contiene un listado de StateMachineInd (cada uno de estos representa a un individuo con sus estados (clase State) y su propio fitness. Esta clase principal contiene los parámetros para el algoritmo genético, valor porcentual de mutación, de cruza, número de individuos, número máximo de estados; en los constructores se inicializan varios de estos parámetros, como puede verse en la figura 4.10 Esta clase tiene muchos métodos la primera que vamos a analizar es la de *initGenerations()*, este método genera los individuos iniciales, genera los estados de forma aleatoria y con los valores indicados en el constructor genera el número de individuos con un número de estados aleatorio (mínimo 2 hasta alcanzar el máximo). Tenemos un par de métodos generadores de valores algunos

[2]Un bean es una clase que no contiene reglas de negocio, sólo tiene métodos de acceso a sus atributos de clase

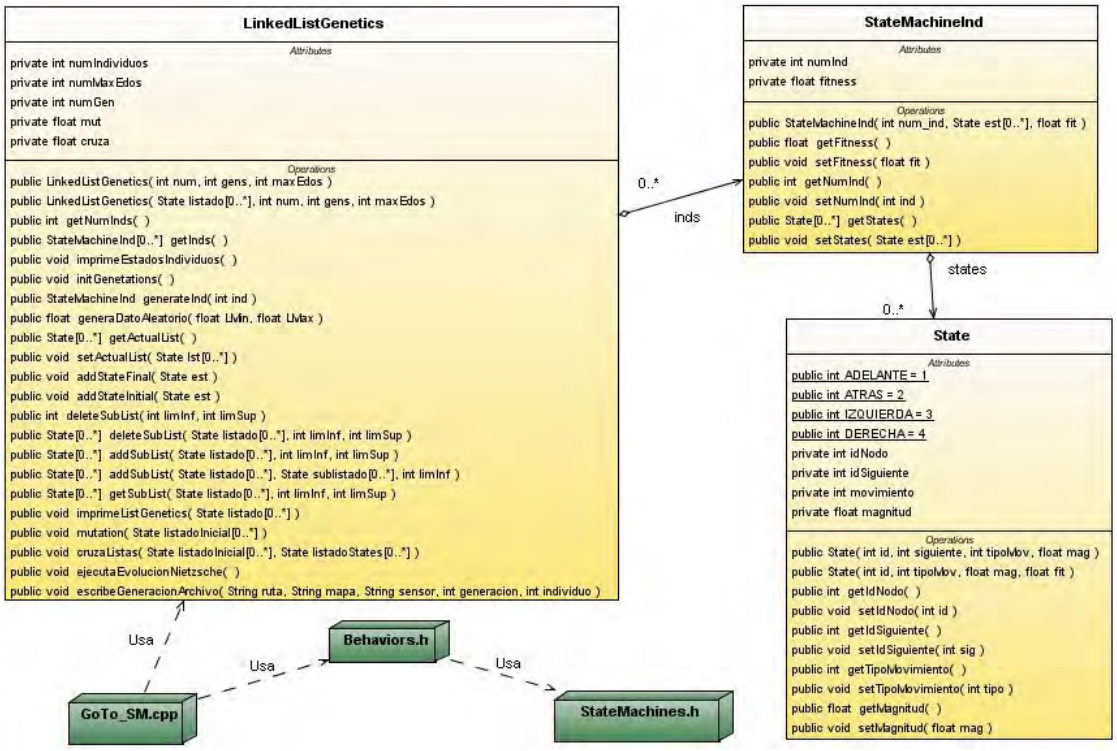

Figura 4.10: Diagrama de clases del sistema de máquinas de estado.

valores, son enteros (como el número de estados por individuo y el tipo de movimiento), otros flotantes como es la magnitud de cada estado; como en el caso del anterior algoritmo genético hacemos uso del constructor para generar un objeto en este caso de tipo LinkedListGenetics de la siguiente forma:

LinkedListGenetics sm(numIndividuals,numGens,estados);

y para generar la población inicial lo hacemos por medio del llamado al método: *sm.initGenetations( );* el diagrama de flujo de la figura 4.11 muestra la manera de generar cada individuo y cada uno de sus estados almacenados en un listado. Este listado de individuos será el utilizado posteriormente cuando se comience el proceso del algoritmo genético.

Una vez que tenemos generada la población inicial, obtenemos los individuos con sus máquinas de estado generadas aleatoriamente. Para obtener a los individuos generados lo hacemos por medio del método getInds() de la clase LinkedListGenetics y al igual que en el proceso de los algoritmos genéticos con campos potenciales ahora tendremos un ciclo que realizará la cruza, mutación y actualización del listado de individuos. Este proceso es muy similar al indicado en la figura 4.1, pero en este caso haremos una cruza elitista, o como lo indica Kuri [3] haremos un proceso en donde los mejores adaptados al ambiente serán los que se cruzaran con los mejores (método de Nietzsche). Una vez que han sido evaluados los individuos serán ordenados de acuerdo a su *fitness* (los que tengan el menor fitness serán los mejores individuos). Así el individuo que tenga el mejor fitness se podría cruzar con el individuo con el siguiente mejor fitness y así sucesivamente hasta obtener la cruza de los individuos. Los resultantes serán los representativos de la nueva generación y estos nuevamente serán evaluados y de acuerdo a su desempeño serán ordenados para ser cruzados. En el proceso pueden existir mutaciones de acuerdo a cierto valor porcentual; estos individuos son los que le permiten agregar al proceso nuevos valores y así ampliar el universo de soluciones al proceso.

El proceso de cruza de estados se puede apreciar en la figura 4.12 inicialmente tenemos dos individuos representativos de la generación actual, cada uno con su respectivo fitness y sus estados independientes uno del otro, cada individuo tendra la siguiente configuración:

- Número de estado.

[3] Kuri [6]

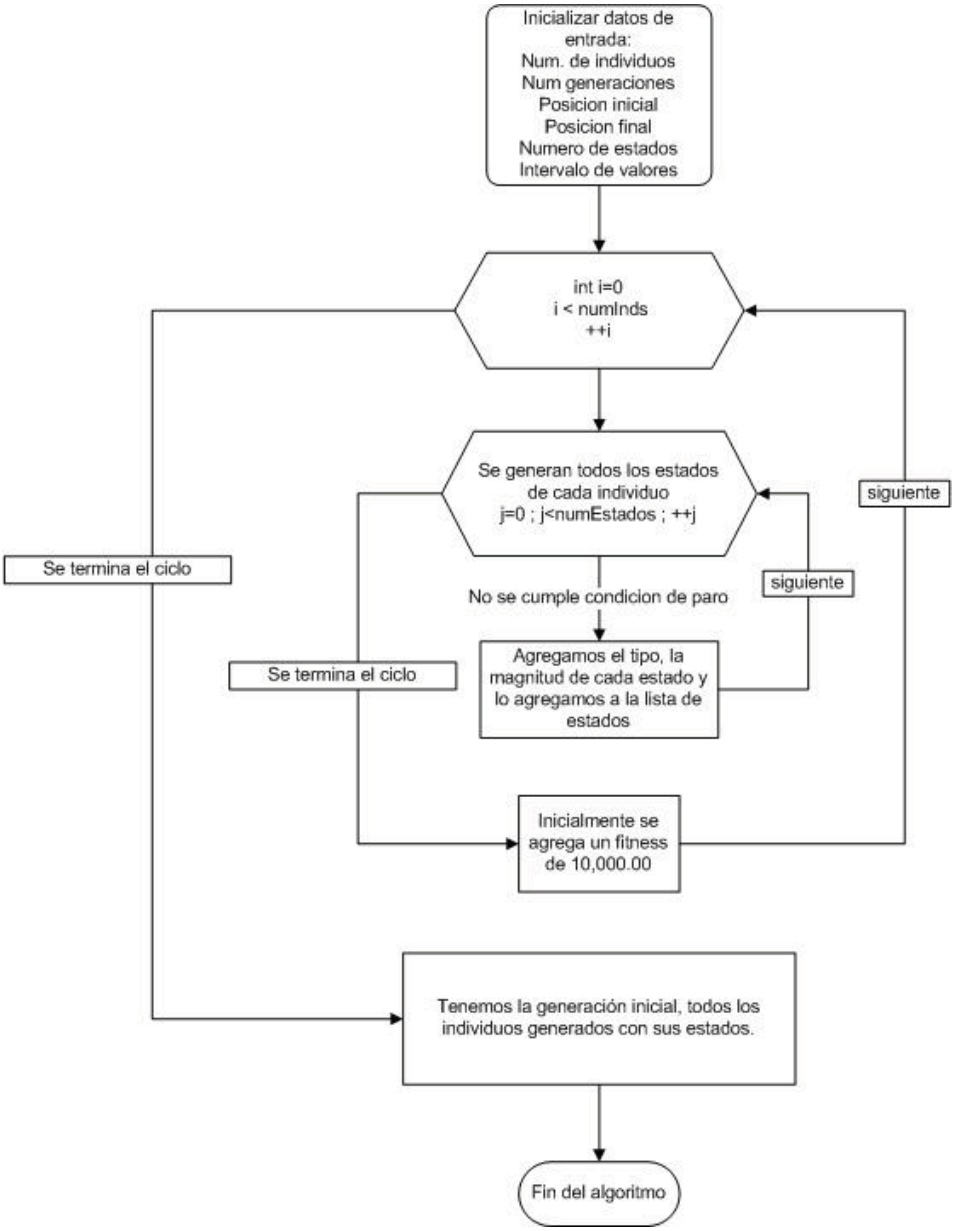

Figura 4.11: Diagrama de flujo para la creación de la generación inicial.

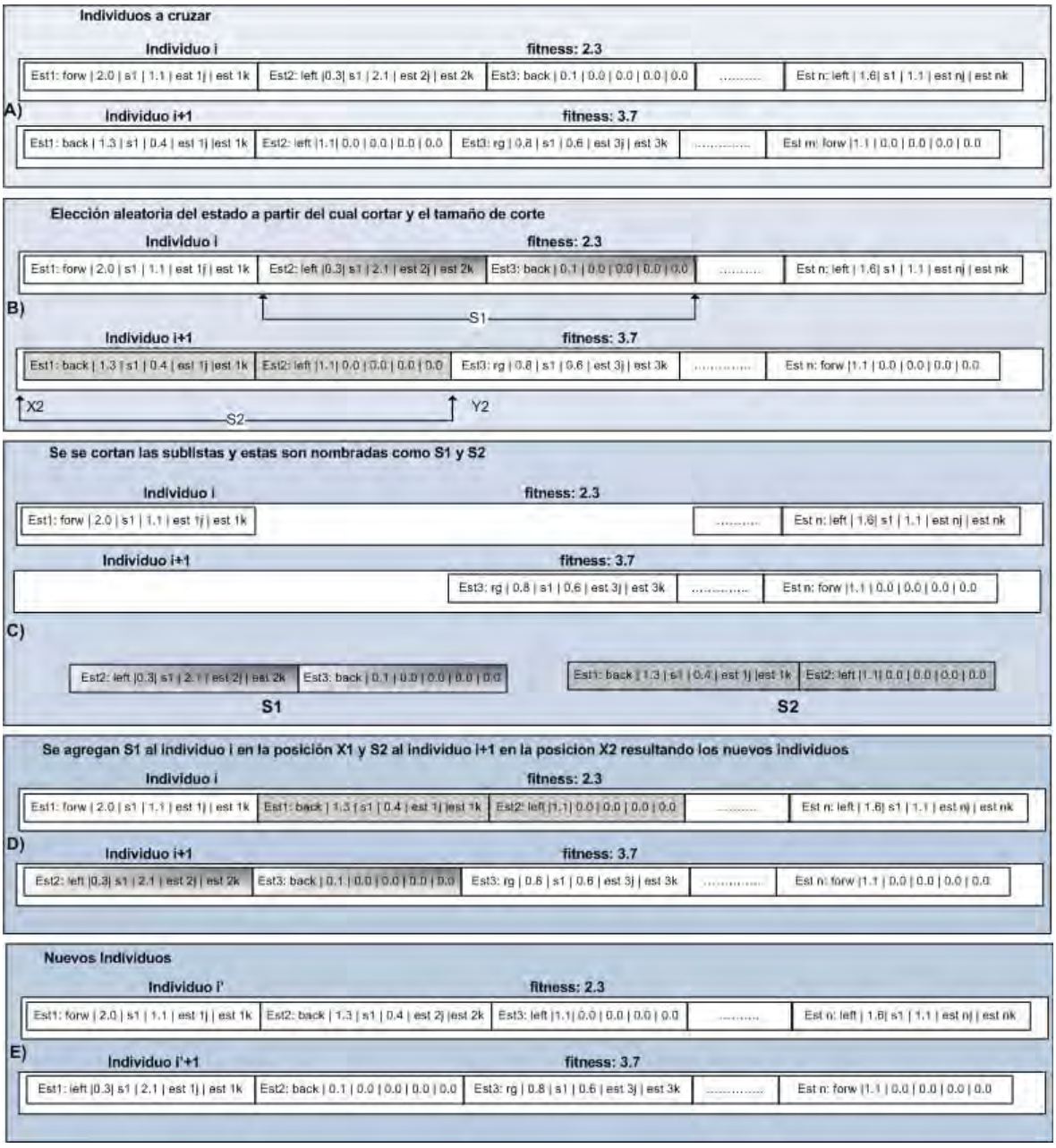

Figura 4.12: Proceso de cruza de máquinas de estado.

- Tipo de movimiento.
- Magnitud del movimiento.
- Sensor a evaluar (si lo contiene el estado).
- Constante de evaluación del sensor (si lo contiene el estado).
- Estado siguiente 1 (si contiene evaluacion del sensor el presente estado).
- Estado siguiente 2 (si contiene evaluacion del sensor el presente estado).

Los estados y el proceso de cruza puede ser apreciado en la figura 4.12 A) representándolos como individuo i y el individuo i+1, para elegir cuantos estados serán usados para la cruza se procederá de la siguiente forma:

- Para la lista más pequeña de estados de tamaño n_1, crear un número aleatorio desde 0 hasta $n_1 - 2$ identificar dicho número como X_1, después generar un número aleatorio desde X_1 hasta $n_1 - 1$ e identificarlo como Y_1.
- Obtener la distancia de X_1 a Y_1 llamandola d_1, por medio de: d_1=Y_1-X_1.
- Para la segunda lista de tamaño n_2, donde $n_2 \geq n_1$. Generar un número aleatorio desde 0 hasta $n_2 - d_1 - 1$ y marcarlo como X_2 y a partir de este obtener $Y_2 = X_2 + d_1$, para obtener la segunda posición.

Una vez que tenemos X_1,Y_1 y X_2,Y_2, obtenemos las sublistas del individuo i y de i+1 cada uno será etiquetado como S_1 y S_2 respectivamente figura 4.12 B), una vez que obtenemos estas sublistas actualizamos las ligas del estado anterior a X_1 al estado siguiente a Y_1 y las del estado anterior a X_2 con la siguiente a Y_2, ver figura 4.12 C), el siguiente paso es Agregar la sublista S_2 al individuo i a partir de la posición X_1 actualizando las listas y al individuo i+1 se le agrega la sublista S_1 a partir de X_2, obteniendo la cruza de las dos máquinas de estado, figura 4.12 D) por último actualizamos los estados y actualizamos las ligas del proceso obteniendo dos nuevos individuos con nuevos estados independientes uno del otro figura 4.12 E), los cuales se encuentran listos para evaluarlos y obtener su nuevo fitness. En la figura 4.12 la cruza es realizada por 2 individuos uno con un *fitness* de 2.3 y otro con uno de 3.7, del individuo 1 es generado X_1 en el estado 1 y X_2 en el estado 2 quedando una

$d_1 = 1$, con ello obtenemos s_1 conformado por lo que contienen los dos puntos de corte, y una vez que se obtiene d_1 se genera el punto aleatorio (siguiendo la regla indicada anteriormente) para obtener X_2 (punto inicial del segundo individuo, para este ejemplo) y una vez obtenido este punto obtenemos Y_2, después de esto se realiza el corte de estados en ambos individuos para obtener S_1 y S_2 después se realiza la sustitución de estados y se renombran los contenidos de acuerdo a la posición de cada máquina de estados.El proceso evolutivo continua iterativamente hasta que son completados cierto número de generaciones indicadas con los parámetros iniciales del proceso.

Capítulo 5

Análisis de resultados

En el presente análisis se usaron diferentes ambientes para el proceso evolutivo; esto resulta importante porque un robot no siempre se desenvuelve en el mismo mundo, normalmente se presentan muchos obstáculos. Por ello en todos y cada uno de estos se llevaron a cabo pruebas con cierto número de individuos y generaciones. El programa desarrollado permite generar ciertos archivos (revisar apéndice B). En dichos archivos se almacena la información evolutiva. Los ambientes utilizados para el análisis de datos son presentados en la figura 5.1. Como es obvio diferentes resultados son obtenidos por el sistema, todos presentan una diferente distribución de objetos: diferentes pasillos, algunos más amplios, otros más estrechos, algunos con objetos más grandes otros más pequeños. Es importante señalar que para el caso de los campos potenciales necesariamente se deben obtener las fuerzas repulsivas de los objetos esto fue hecho por medio del análisis del mundo que a su vez sirve como dato de entrada, la fuerza atractiva es representada por medio del punto destino, como puede verse el resultado de las fuerzas repulsivas siempre es dependiente del mundo.

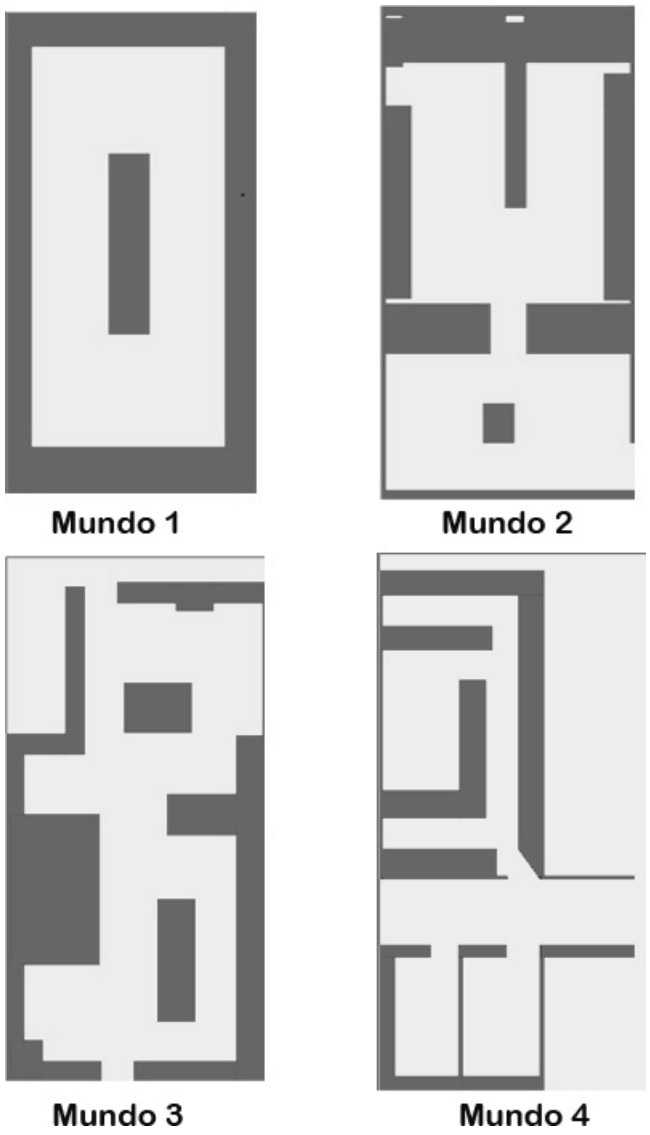

Figura 5.1: Diferentes mundos evaluados por el AG

5.1. Análisis de resultados usando campos potenciales

Ahora haremos un análisis de los ambientes indicados en la figura 5.1; serán mostrados diferentes comportamientos de los robots en cada uno de los mundos, además de que serán mostrados gráficamente todos los individuos que alcanzaron su destino por cada generación. Para este análisis se utilizaron 100 individuos por 100 generaciones y los resultados son los siguientes:

Inicialmente se utiliza el ambiente más sencillo indicado en la figura 5.1 (mundo 1); este consiste en un sólo obstáculo al centro y básicamente el comportamiento que puede presentar el robot es evadir dicho obstáculo rodeándolo. Cuando se estaban haciendo las pruebas de ejecución los obtenidos nos demuestran que a partir de un momento dado los resultados obtenidos por las constantes poco a poco se van haciendo homogéneas observando los siguientes por cada mundo:

5.1.1. Mundo 1

Como puede apreciarse en la imagen este es el mundo más sencillo de los indicados en la figura 5.1. Básicamente tenemos un ambiente rodeado de muros y al mismo tiempo con un sólo obstáculo al centro, estas serían las únicas fuerzas repulsivas que pueden actuar sobre la partícula, en este caso el robot. Si lo colocamos en la parte superior izquierda A (posición inicial) y la posición final en la inferior derecha podemos observar diferentes comportamientos del robot en el mundo, en cada uno de ellos hacen uso de un individuo en la evaluación, en la figura 5.2. Estos individuos erroneos van disminuyendo conforme avanza el número de generaciones, en dicha imagen aunque sólo vemos 5 representaciones de estos individuos, se puede apreciar que presentan comportamientos similares cada uno con pequeñas variaciones de movimiento en alguno de los pasos evaluados.

Como se ha mencionado anteriormente a medida que avanzan las generaciones la aptitud de los individuos se hace más pequeña (esto es correcto porque este es un problema de minimización). Obteniendo la evaluación de la aptitud de acuerdo a la distancia que alcanza el individuo con respecto a la posición destino o deseada. Inicialmente tenemos pocos individuos que llegan al destino, pero generación a generación estos individuos se hacen más comunes y los individuos que no alcanzan la posición destino van de poco a poco desapareciendo (En el caso de que la prueba sea

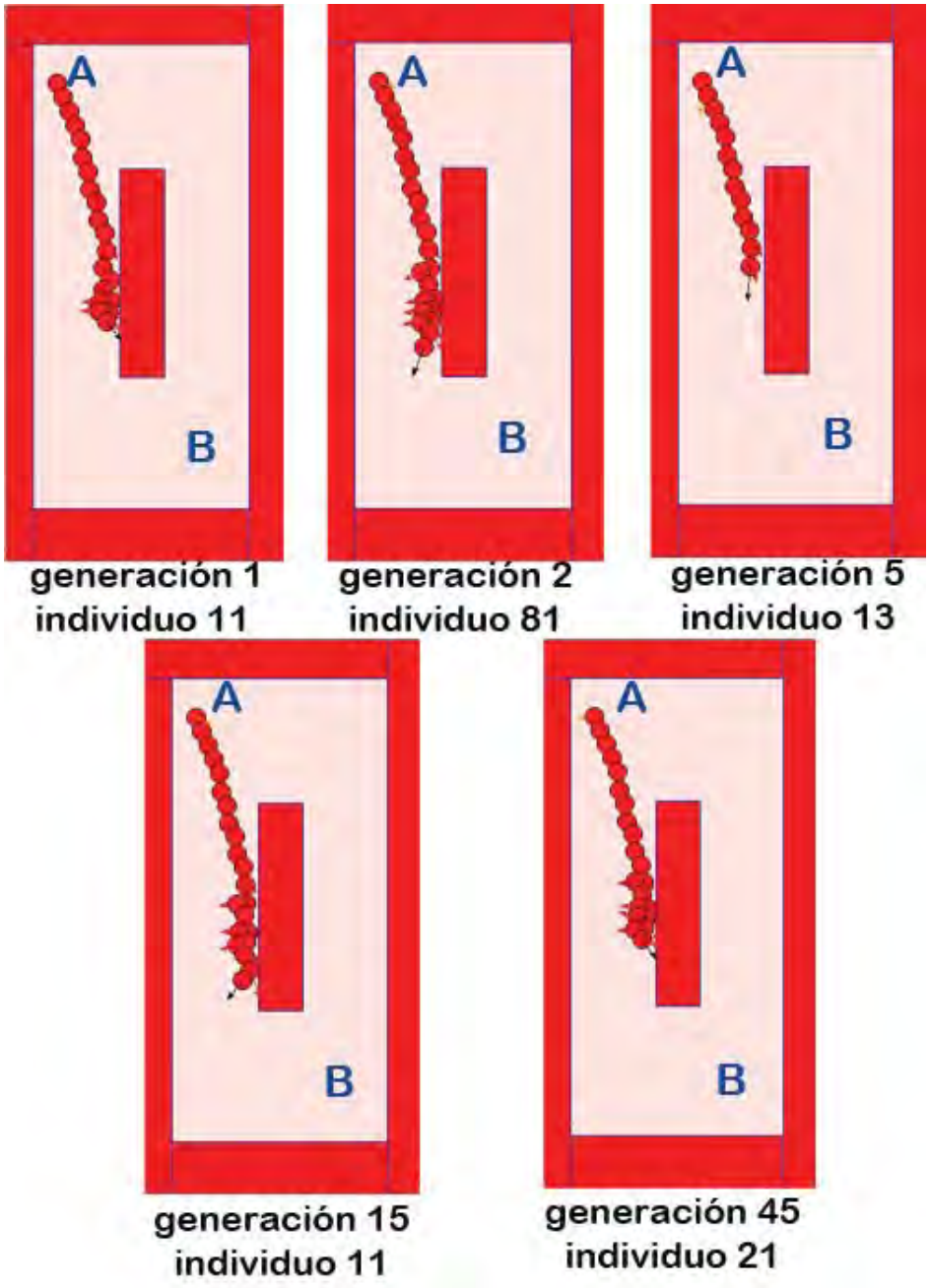

Figura 5.2: Individuos erroneos Mundo 1 el robot debe ir del punto A al punto B

correcta). En la figura 5.3 pueden apreciarse diferentes individuos que alcanzan la posición destino.

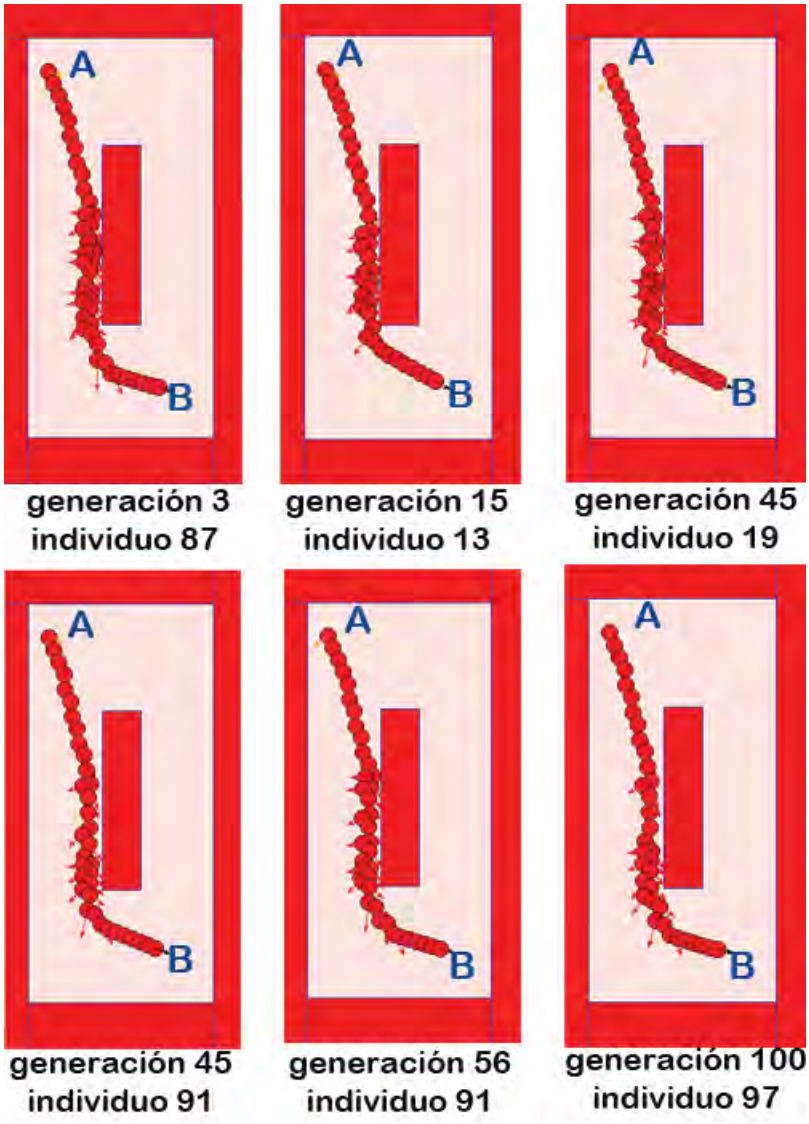

Figura 5.3: Individuos que alcanzan el destino Mundo 1, El robot se desplaza del punto A al punto destino B

En la figura 5.4 Puede apreciarse que generación a generación los individuos se aproximan más y más a la posición destino indicada como el punto B. También pueden apreciarse picos, estos indican que se presentan mutaciones las cuales incrementan (o disminuyen) dicha distancia, pero al tener un problema de minimización inmediatamente que una distancia se incrementa convierte a dicho individuo en uno poco apto para sobrevivir y heredar sus propiedades. Así de esta manera podemos ver en la gráfica que nuevamente se estabiliza para acercarse más al punto destino. Cabe destacar que inicialmente se usa un número mucho mayor para inicializar la primer generación (10000), pero para que quede más claro en el gráfico se escaló a dicho valor.

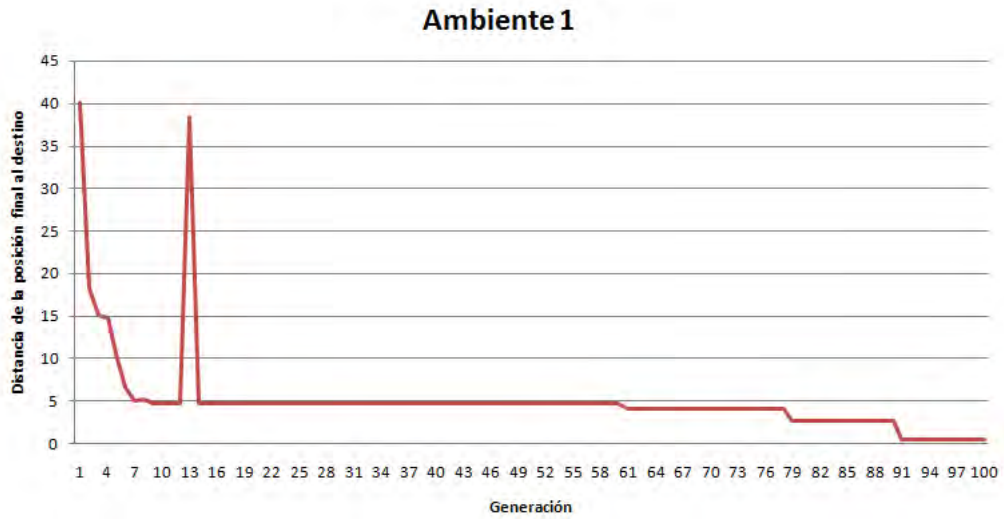

Figura 5.4: Gráfico de distancia de la posición final al punto destino por generación

5.1.2. Mundo 2

El mundo 2 se presenta como un ambiente un poco más complejo en donde existen variaciones tanto en su forma como en los obstáculos. Se generan pasillos que pueden ser recorridos por el robot para alcanzar algún objetivo. En la figura 5.5 se incluyen un par de ejemplos en donde tenemos diferentes posiciones iniciales y finales. Los puntos iniciales se encuentran identificados por medio de la etiqueta A mientras que para los puntos finales usamos la letra B; estos ejemplos nos sirven para darnos cuenta cual es el recorrido del robot y cual es el punto que necesita alcanzar (punto B).

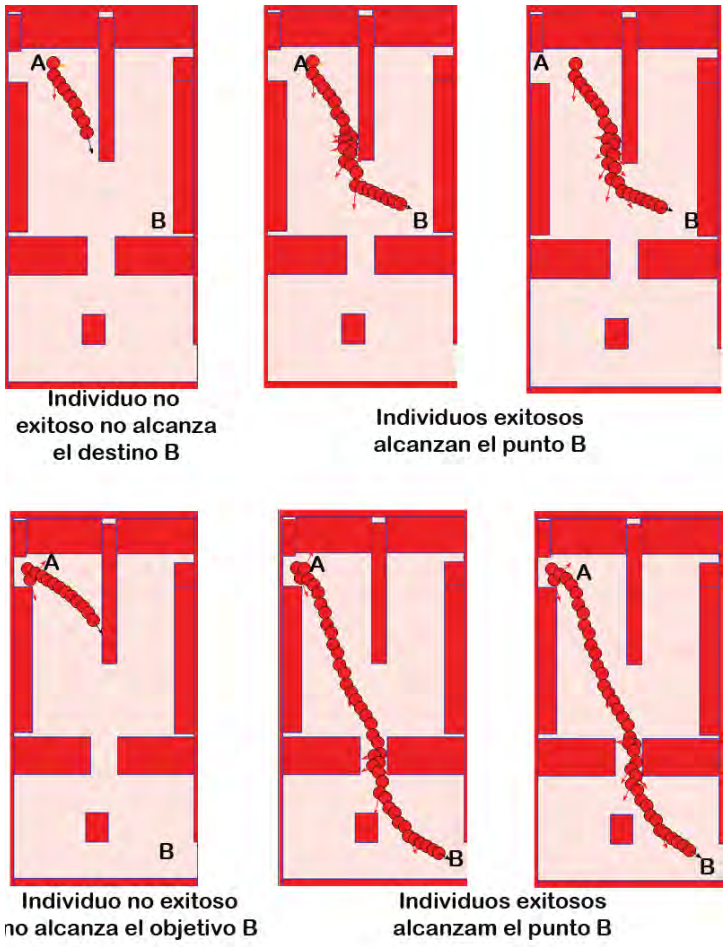

Figura 5.5: Ejemplos del mundo 2, La posición origen etiquetada como A y la posición destino B

En la figura 5.5 la parte superior izquierda podemos darnos cuenta que el robot no alcanza la posición destino, en tanto que en las dos imágenes a su derecha; si logran alcanzar el objetivo B siguiendo la ruta indicada por el simulador. Puede apreciarse claramente que los comportamientos en los tres casos son diferentes, tanto en posición, ruta de desplazamiento y orientación. En este ejemplo el "robot" se queda atrapado en el obstáculo encontrado en el trayecto al punto destino, mientras que los otros dos

casos que resultan exitosos, evaden el obstáculo para lograr el objetivo de alcanzar el punto destino B.

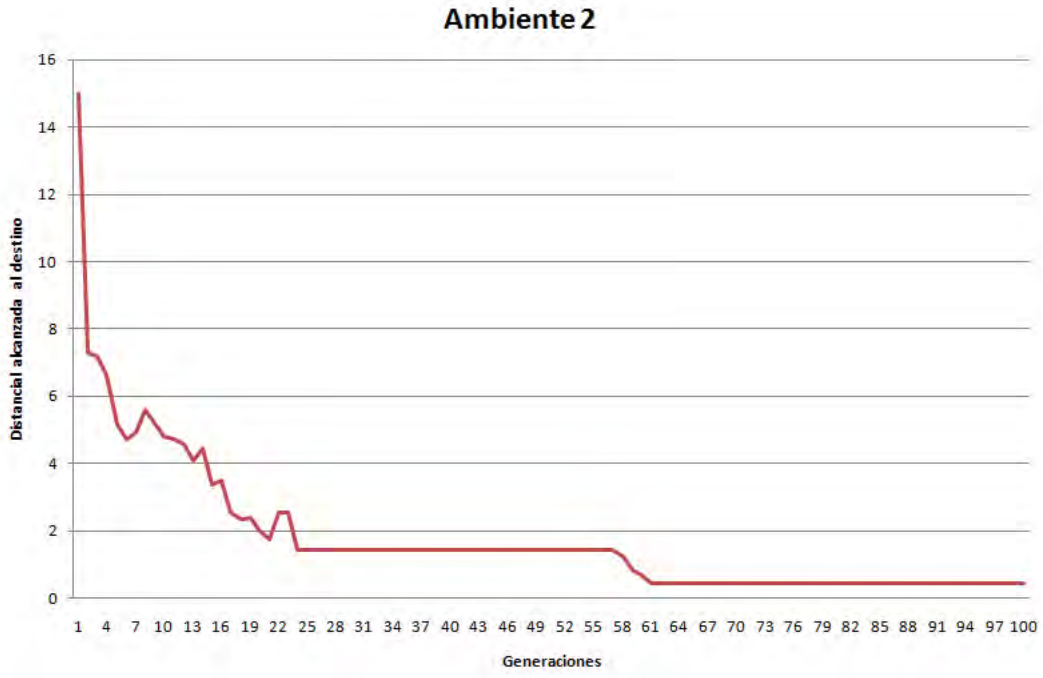

Figura 5.6: Gráfico de distancia de la posición final al punto destino por generación

En el gráfico de la figura 5.6, puede apreciarse nuevamente que generación a generación el algoritmo genético se acerca a la posición destino marcada como B. Nuevamente se presentan picos en la gráfica esto como se indicó en el ambiente 1, es debido a las mutaciones que se presentan, y como alejan al robot de su posición final al destino B. A partir de la generación 61 comienza a estabilizarse llegando a 0.451757 de la posición destino.

5.1.3. Mundo 3

El mundo 3 es un mundo que resulta menos homogéneo que los dos anteriores, este ya presenta características más irregulares que los dos anteriores. Inicialmente presenta obstáculos de diferentes tamaños, pasillos menos simétricos y como en los anteriores.

También inicialmente se necesita obtener los vectores de repulsión. Una vez hecho esto se inicializan los datos necesarios para que al final ejecutemos el algoritmo genético.

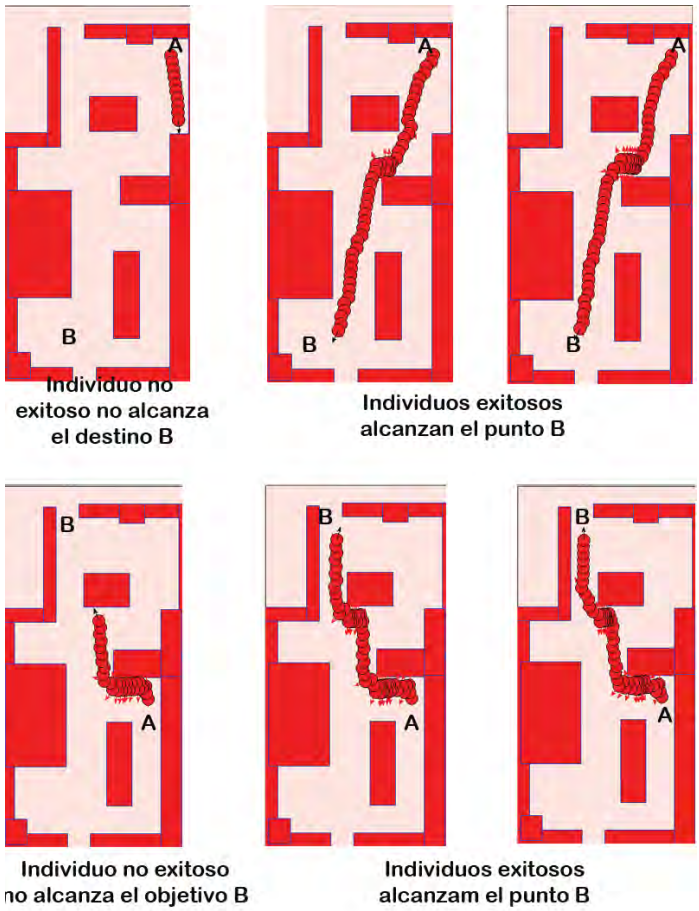

Figura 5.7: Ejemplos del mundo 3, el robot se debe desplazar del punto origen A al punto destino B

Como puede apreciarse en la imagen 5.7 tenemos del lado izquierdo 2 individuos que no resultan exitosos al intentar pasar del punto A al destino (representado por el punto B). En el primer caso el robot se queda trabado en el primer obstáculo encontrado, en el segundo, cuando es evaluando otro individuo; podemos ver que el robot logra de forma exitosa evadir el obstáculo que ha localizado, pero al llegar

al segundo obstáculo se traba y no alcanza el punto destino B. En la parte superior central y derecha podemos ver el comportamiento del robot evaluando dos diferentes individuos que alcanzan el destino y al mismo tiempo se puede apreciar en ambos casos la diferencia de comportamiento del robot (en estos ejemplos alcanzan el punto destino). En la parte inferior del mundo, para el caso siguiente, podemos ver que el comportamiento es muy similar (ambos comportamientos son muy parecidos) pero no son iguales y la forma de evadir ambos obstáculos es representada de forma diferente, puede apreciarse la diferencia de ángulo que se presenta en ambos ejemplos.

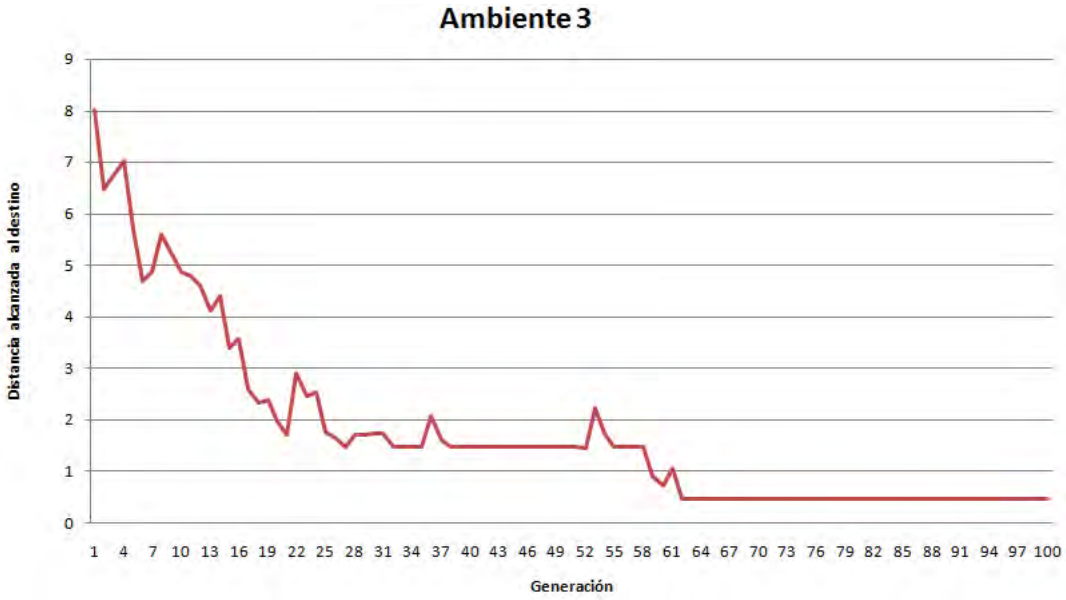

Figura 5.8: Gráfico de distancia de la posición final al punto destino por generación del mundo 3

En la figura 5.8 podemos ver un comportamiento diferente en cuanto a los dos ejemplos anteriores. En este caso tenemos que se presentan más variaciones al inicio para después de algunas generaciones; estabilizarse acercándose al punto B teniendo una distancia a ese punto menor a uno.

5.1.4. Mundo 4

En la figura 5.9 tenemos un mundo más complejo, con pasillos más estrechos y podemos ver que por este espacio logra moverse sin muchos problemas el robot (como puede apreciarse en las imágenes mostradas). El robot muchasveces logra alcanzar la posición destino, pero en otros se queda trabado en paredes u obstáculos del ambiente (Esto seguramente sucede cuando se queda en algún mínimo local). En la imagen superior izquierda puede apreciarse que aunque el robot parece tener un comportamiento bueno para alcanzar la posición final B, este se queda trabado en la salida de la habitación, en tanto que los otros dos ejemplos en la parte superior logran alcanzar la posición destino, en otro ejemplo del mismo mundo en la parte inferior izquierda tenemos el mismo problema pero con otra situación; nuevamente el robot no llega a la nueva posición destino B y se queda trabado en la salida de esa zona. Un ejemplo de individuo exitoso en ese mismo ejemplo lo tenemos a la derecha de este último.

El comportamiento del conjunto de individuos se muestra en la figura 5.10 desde tempranas generaciones se obtuvieron individuos que mejoraban la distancia al destino, presentando nuevamente los picos relacionados con las mutaciones presentadas en los individuos en algunas generaciones. Nuevamente se presenta una estabilidad con los individuos finales.

Haciendo un análisis general del comportamiento del algoritmo genético en los ambientes anteriores podemos ver que se necesita cierto número de generaciones para resolver el problema de minimización. En cierto número de generaciones se obtienen las constantes que mejoran a las nuevas generaciones, pero por otro lado con la mutación que aparece con cierto porcentaje en muchas ocasiones mejora dichos individuos, pero en otras empeora un tanto los resultados desplegados en los gráficos, presentando picos que indican cierta inestabilidad, pero generaciones adelante, se eliminan dichas anomalias, siendo reemplazadas por individuos con características más aptas para ser utilizadas como propuesta de solución.

5.2. Análisis de eficiencia.

Para el análisis de eficiencia del algoritmo se realizaron treinta diferentes pruebas con los mismos datos de entrada (se ejecutó la prueba para 100 generaciones y 100

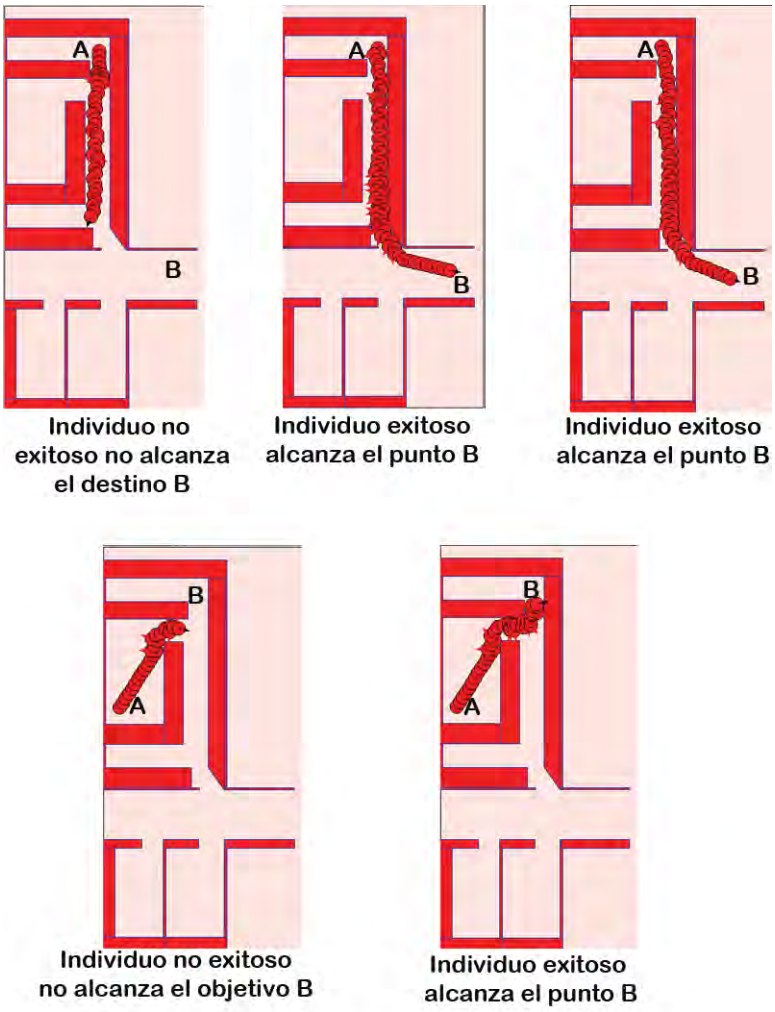

Figura 5.9: Ejemplos del mundo 4 el robot debe desplazarse de la posición A a la posición B

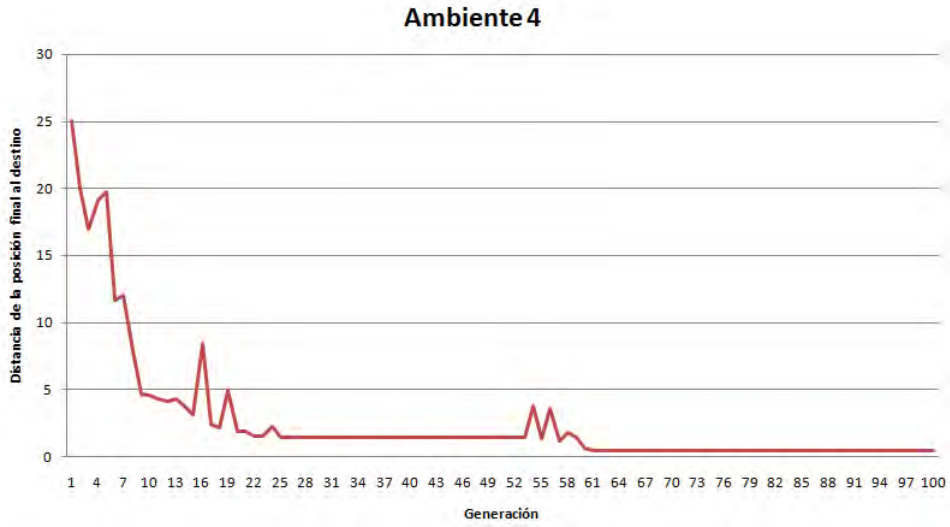

Figura 5.10: Gráfico de distancia de la posición final al punto destino por generación del mundo 4

individuos por generación). Analizándose el mundo 1 y el 4, de estos se obtuvieron los resultados mostrados en el cuadro 5.1, en cada una de las pruebas se obtuvieron el número de individuos que alcanzaron el objetivo y se dividió entre el número de individuos resultado del producto de 100 individuos por 100 generaciones (eficiencia del 100 %). Así el promedio de todos los casos es de 31.33 % para la el mundo 1, es decir, de cada 100 individuos alrededor del 31 % son individuos que alcanzaron el destino, para el mundo 4 el porcentaje de eficiencia subió al 47.9723333 % casi la mitad de los individuos del total de las generaciones alcanzaron el objetivo planteado, pero como puede apreciarse en el cuadro señalado anteriormente (cuadro 5.1). En el análisis de este caso existieron muchos casos en los cuales no se logro una solución al problema de minimización propuesto, por lo que promedian cero en su análisis, en este caso se puede notar que el algoritmo genético no es capaz de salir de algún mínimo local y generación a generación no consigue moverse a pesar de la mutación propuesta (.01 %). En las imágenes de la figura 5.11 y la figura 5.12 muestran el comportamiento de la eficiencia de cada una de las pruebas, el tope máximo era del 100 % para el caso de 10000 individuos.

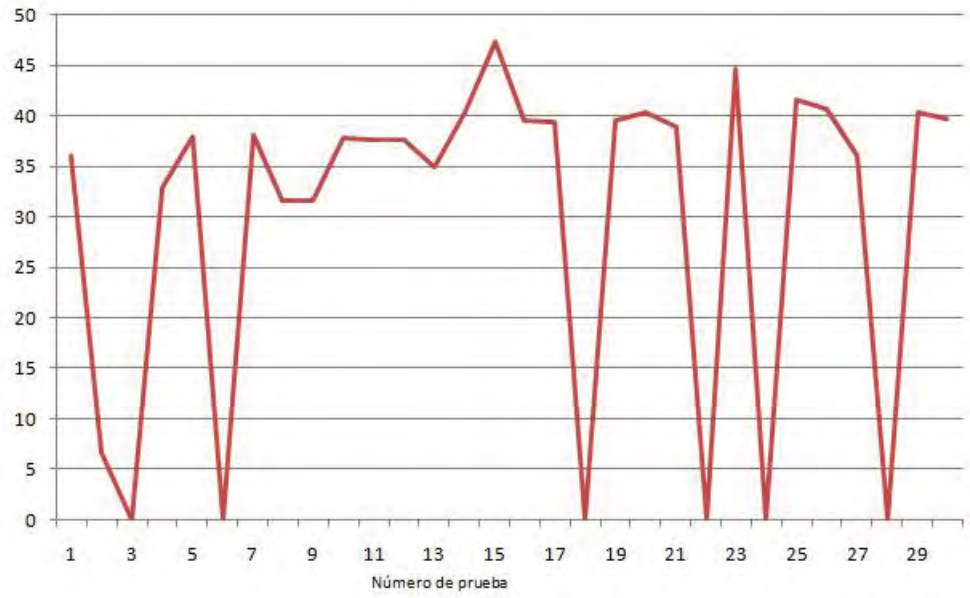

Figura 5.11: Análisis del mundo 1

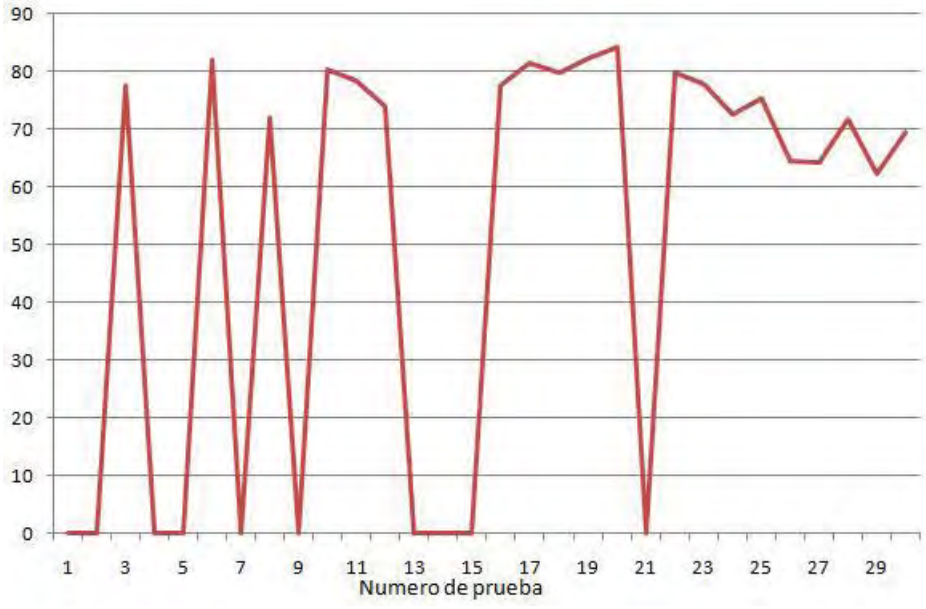

Figura 5.12: Análisis del mundo 2

Número de Prueba	Eficiencia presentada en el mundo 1	Eficiencia presentada en el mundo 2
1	84.37	0
2	6.7	0
3	0	77.57
4	32.81	0
5	38.04	0
6	0	82.08
7	38.06	0
8	31.64	72.11
9	31.63	0
10	37.85	80.47
11	37.62	78.44
12	37.63	74.06
13	34.95	0
14	40.35	0
15	47.38	0
16	39.57	77.71
17	39.43	81.58
18	0	79.73
19	39.58	82.21
20	40.34	84.37
21	38.99	0
22	0	79.91
23	44.62	77.98
24	0	72.45
25	41.67	75.47
26	40.62	64.6
27	36.1	64.37
28	0	71.86
29	40.43	62.2
30	39.76	69.42
prom:	31.338	47.97233333

Cuadro 5.1: Porcentaje de eficiencia por prueba para el mundo 1

Conclusiones y trabajo futuro

En la presente tesis se han abordado diferentes temas, inicialmente entendimos que significan los comportamientos en los robots, cuales son sus implicaciones, además de conocer cuales son sus deficiencias. Por otro lado se presentaron diferentes caminos utilizando como base los algoritmos genéticos, desde el algoritmo genético simple hasta otros que presentan ciertas variaciones que quizás ayudan a ampliar el rango de posibles resultados (esto es dependiente del tipo de problema).

Inicialmente el objetivo de la tesis fue el resolver un importante problema en la robótica el comportamiento del robot en un ambiente dado. En este trabajo se analizaron diferentes mapas, diferentes entornos que sirvieron para probar el sistema realizado y adaptado para funcionar con algunas herramientas ya existentes como el caso del Virbot la cual es una herramienta de simulación usada desde hace años para diferentes desarrollos y aplicaciones en el comportamiento robótico.

El sistema inicialmente ha sido desarrollado en el lenguaje de programación C++, el desarrollo de las clases y librerías auxiliares fueron creadas para poder interactuar con los sistemas ya existentes indicados en las líneas superiores. El desarrollo del sistema tenía como objetivo tener un sistema sencillo de utilizar, implementar y hacerlo dinámico. El sistema se modifica de acuerdo a los parámetros que le sean introducidos.

El sistema tuvo variaciones a lo largo del desarrollo, el análisis del *fitness* y el com-

portamiento del robot, obligaron a realizar modificaciones en la función de obtención de la aptitud por individuo, además de que también resultó interesante analizar el comportamiento de la aplicación del algoritmo genético usando diferentes puntos en un mismo mapa. En realidad no había mucha variación en el comportamiento del algoritmo, las constantes una vez que eran obtenidas siempre oscilaban entre los mismos valores.

La aplicación del algoritmo genético aplicado al comportamiento de robots resultó exitoso, en la mayoría de los casos, y muchas veces desde las primeras generaciones pudimos apreciar candidatos que se acercaban bastante al destino, estos individuos al evolucionar, heredaban valores a las generaciones siguientes; llegando así a alguna condición de paro y obtener al final la solución buscada.

Como trabajo futuro pueden combinarse otros comportamientos, puede ser evaluado cuando el robot se localice en algún mínimo local y que la mutación no sea suficiente agregar algún discriminante que pudiera usarse para aplicar otro tipo de comportamiento como las maquinas de estados, algo más que puede aplicarse al sistema es utilizar redes neuronales, observar y evaluar el comportamiento del robot, hacer comparaciones de rendimiento en tiempos y por medio de estos mejorar el comportamiento del robot y quizás usar hilos para la evaluación de los individuos acelerando así la velocidad de procesamiento haciéndolo mucho más eficiente.

Apéndice A

Algoritmo Evolutivo

```
while(MAX_GENERATIONS>countGen){
  countGen++;
  for(int i=0; i< numIndividuos;++i){

      xmv=xo;
      ymv=yo;
      thetamv=theta;
     //printf("xmv %f ymv %f thetamv %f\n",xmv,ymv,thetamv);

      num_sonars=NUM_SONARS;
      flg=0;
      coord_dest[1].xc=xd;
      coord_dest[1].yc=yd;
      printf("\norigin x %f y: %f Final destination x %f y %f \n",
```

```
                                xmv, ymv, coord_dest[1].xc,coord_dest[1].yc);

distance1 = MAX_DISTANCE;
sprintf(file_obs,"%sgen-%d-Indv-%d-sonar_%s_map_0.raw2",
                                PATH,countGen,individuosPtr[i].nInd,env);

if((fpw=fopen(file_obs,"w")) == NULL){
    printf("File %s can not be open\n",file_obs);
    return(0);
}

while(flg == 0){

  num_obs++;
  // Navegation using only odometry
  paralell_behaviors(fpw,num_sonars,&num_obs,coord_dest,&individuosPtr[i]);

  // it checks if the robot reached its node destination
  coordinates_robot("Ryu",&coord_robot);
  theta=coord_robot.anglec;

  fprintf(fpw,"%f %f",coord_robot.xc,coord_robot.yc );
  printf(" %f %f ",coord_robot.xc,coord_robot.yc );

  if( (distance1=get_distance(coord_robot,coord_dest[1])) < EPS_DEST){
       printf(" reached distance destination %f\n",distance1);
       flg= 1;
  }


  if(individuosPtr[i].steps > LIMIT_SIM){
      //      fclose(fpw);
      break;
  }
}
```

```
        individuosPtr[i].fitness=distance1;
        individuosPtr[i].steps=individuosPtr[i].steps++;
        printf("####### gen-%d-Indv-%d \n",countGen, individuosPtr[i].nInd );
        fclose(fpw);
      }


      if(MAX_GENERATIONS < ( countGen)){
         nuevaGeneracion.createNewPop();
         individuosPtr = nuevaGeneracion.getNewInds();
         nuevaGeneracion.imprimeIndividuos();
         nuevaGeneracion.escribeNuevaGeneracion();
         break;
      }


}
```

Apéndice B

Formato de archivos generados y utilizados

Para el desarrollo del sistema se escriben diferentes archivos para identificar los diferentes individuos evaluados generación a generación VirBot utiliza archivos que contienen los movimientos del robot, las constantes y datos de cada individuo, los archivos generados son los siguientes:

- Archivo de comportamiento del robot por individuo por generación.
- Archivo de generaciones archivo que almacena temporalmente los datos de todos los individuos, este archivo es sobreescrito generación a generación.
- Archivo de Individuos exitosos.
- Archivo de número de individuos exitosos por generación.

B.1. Archivo de comportamiento

Este archivo contiene las lecturas de los sensores, así como la posición y destino del robot, el formato del nombre de estos archivos es el siguiente:

gen_[número de generación]_Indv_[número de individuo]_[sensor]_[ambiente]_map_[punto].raw

por ejemplo si el mapa usado es uno llamado china y evaluamos el punto 1 con sonar y con estos datos al individuo 36 de la generación 16 el nombre del archivo que contiene las posiciones del robot queda de la siguiente forma:

gen_16_Ind_36_sonar_china_map_1.raw

El archivo contiene una serie de observaciones de los sensores y la posición del robot así como el destino, el archivo queda de la siguiente forma:

```
(obs 153511 0 29.438564 115.440338 3.981898 9.926579 8.195905
8.018638 9.223317 10000.000000 10000.000000 7.529415 3.554850
5.096378 12.306586 40.970596 47.125793 21.182646 17.489492 49.838039
15.593234 ) (obs 153512 0 28.860811 113.008011 4.479179 17.674259
10.046354 7.852740 7.365808 8.096370 10.815669 10000.000000
10000.000000 6.040816 8.069733 15.764552 41.167168 45.250210
19.801174 15.477621 108.081429 ) (obs 153513 0 28.389208 110.552895
4.522610 18.292902 9.807457 7.477327 6.889351 7.436739 9.679404
10000.000000 10000.000000 8.578738 11.165799 19.656910 41.611706
44.917931 17.158478 13.081855 10000.000000 ) (obs 153514 0 27.918959
108.097519 4.523162 17.067472 9.143122 6.968562 6.419114 6.927561
9.013795 10000.000000 10000.000000 11.226749 14.607672 21.586082
42.082123 45.415363 13.670680 11.768793 10000.000000 ) (obs 153515 0
27.449921 105.641914 4.523655 15.839549 8.479269 6.460708 5.950086
6.420073 8.351103 15.267579 10000.000000 13.874147 18.047178
27.862043 42.551327 19.012857 10.178013 12.958658 10000.000000 )
```

B.2. Archivo de generaciones

Este archivo contiene los datos de las constantes del individuo actual, el número de pasos procesados así como su aptitud (fitness), el nombre del archivo es **generaciones_0** y este contiene un encabezado indicando el orden de las constantes evaluadas: D0-D1-E1-E2-ETA en la siguiente línea tenemos los datos del primer individuo y la última línea del archivo contiene los datos del n-ésimo individuo y los valores de las constantes, a manera de ejemplo tenemos lo siguiente:

```
D0-D1-E1-E2-ETA
 5686.380859 4928.386719 6702.432617 7529.710449 4928.386719 4928.386719 1000 18.040026
 5686.380859 4928.386719 6702.432617 7529.710449 4928.386719 4928.386719 1000 18.040026
 6702.432617 4928.386719 5686.380859 7529.710449 5686.380859 5686.380859 1000 18.040026
 6824.016113 4928.386719 5686.380859 4928.386719 4928.386719 7529.710449 1000 18.040026
 ..........
```

B.3. Archivo de Individuos exitosos

Este archivo contiene los individuos que alcanzaron el objetivo, contienen los datos del individuo a que generación pertenece, los valores de las constantes así como su fitness (o distancia al punto destino), este archivo tiene el siguiente formato:

Inds_[sensor]_[ambiente]_[punto].ind

por ejemplo para el mismo mapa china con el sonar como tipo de sensor y el punto cero tendríamos el siguiente archivo:

Inds_sonar_china_0.ind

El cual dentro tiene una linea de encabezado indicando el individuo, la generación y los valores de las constantes así como el valor del fitness en ese momento.

```
ind gen d0 d1  e1  e2  eta

 0   87  2551.873535 3348.983887 4633.826172 7537.037109 6810.177734 4.559546
 1   87  2551.873535 3348.983887 4633.826172 7537.037109 6810.177734 4.559546
```

```
4   87  1980.363037 2551.873535 3348.983887 6810.177734 1980.363037 4.559546
5   87  4633.826172 2551.873535 2073.137695 7537.037109 4633.826172 4.559546
7   87  1980.363037 1980.363037 4633.826172 4065.339600 1980.363037 4.559546
8   87  3348.983887 2551.873535 3348.983887 6810.177734 7537.037109 4.559546
9   87  1980.363037 4633.826172 3348.983887 6810.177734 1980.363037 4.559546
10  87  2551.873535 2551.873535 2073.137695 1980.363037 2551.873535 4.559546
11  87  6810.177734 2551.873535 2169.492432 2073.137695 7537.037109 4.559546
14  87  4633.826172 1980.363037 2169.492432 5560.397461 7537.037109 4.559546
15  87  2551.873535 3348.983887 2551.873535 7537.037109 5560.397461 4.559546

..........
```

B.4. Archivo de Individuos exitosos por generación

Este tipo de archivos sirve para contabilizar el número de individuos que alcanzaron el objetivo por generación, este archivo tiene como finalidad obtener los datos estadísticos por generacion y el nombre del archivo tiene el siguiente formato:

numIndSuc_[sensor]_[ambiente]_[punto].ind

Este contiene los datos por generación de la siguiente forma:

```
Gen:    3   Inds:   45
Gen:    4   Inds:   32
Gen:    5   Inds:   55
Gen:    6   Inds:   40
Gen:    7   Inds:   45
Gen:    8   Inds:   43
Gen:    9   Inds:   50

..........
```

Bibliografía

[1] ARKIN, Ronald C. · Behavior-Based Robotics · Cambridge, U.S.A. The MIT Press · 1998

[2] LATOMBE, Jean-Claude · Robot Motion · Boston Planning Kluwer Academic Publisher · 1991

[3] MULLER, Jorg P. · The Design of Intelligent Agents · Berlin Springer · 1996

[4] WAHDE, Mattias · Evolving complex behaviors on autonomous robots · 7th Mechatronics Forum International Conf. · 1998

[5] GOLDBERG, David · The design of innovation lessons from and for competent genetics algorithms · Kluwer Academic publishers · 2002

[6] KURI Morales, Angel · Algoritmos Geneticos · Fondo de cultura económica ·2002

[7] http://www.bostondynamics.com/content/sec.php?section=BigDog ·

[8] http://world.honda.com/ASIMO/ ·

[9] http://www.inl.gov/adaptiverobotics/behaviorbasedrobotics/ ·

[10] MARCELLIN Jacques, Sergio · Notas del Curso:" Construcción de Sistemas Expertos ", Posgrado en Ciencia e Ingeniería de la Computación, UNAM · 2008

[11] HOLLAND, John Henry · Adaptation in Natural and Artificial Systems · Cambridge, massachusetts MIT · 1992

[12] LAVRAC,Nada · Lecture Notes in Artificial Inteligence 1297, Inductive logic programming · Ellis Horwood · 1994

[13] GOLDBERG, David E. · Genetic algorithms in search · Addison-wesley · 1989

[14] KANT, Immanuel · Sobre pedagogía · Editorial de la Universidad Nacional de Córdoba coedición con Encuentro Grupo Editor · 2009

[15] PIAGET, J., INHELDER · Memoria e Inteligencia · El Ateneo, Argentina · 1978

[16] LORENZ Konrad Zacharias · Fundamentos de la etología · Paidos · 1986

[17] SAVAGE, Jesús ,BILLINGHURST, Mark · The Virbot: A virtual reality mobile robot driven with multimodal commands · Expert Systems with Applications 15 pp 413-419 · (1998)

[18] http://www.rae.es/rae.html · Diccionario de la Real Academia de la Lengua Española *Online* edición 22 ·