DISEÑO Y DESARROLLO DE ALGORITMOS Y SISTEMAS DE CONTROL POR CLONACIÓN ARTITIFICIAL DE UN SENSOR DE VISCOSIDAD

BY

ING. WILMAN ALONSO PINEDA MUÑOZ

BY

ING. JAIRO AMADOR NIÑO

A Dissertation submitted to the Graduate School
in partial fulfillment of the requirements
for the Degree

Maestría en Ciencias Computacionales

Major Subject: Clonación Artificial

Minor Subject: Ingeniería

UNIVERSIDAD AUTÓNOMA DE BUCARAMANGA

INSTITUTO TECNOLÓGICO DE ESTUDIOS SUPERIORES

TEC-Monterrey - Méjico

Universidad Virtual

Bucaramanga

Abril de 2006

ÍNDICE GENERAL

CAPÍTULO 1

INTRODUCCIÓN

El presente documento contiene la investigación sobre una metodología científicamente fundamentada, para la clonación artificial de un sensor de viscosidad basada en técnicas de inteligencia artificial, titulada: "DISEÑO Y DESARROLLO DE ALGORITMOS Y SISTEMAS DE CONTROL POR CLONACIÓN ARTIFICIAL DE UN SENSOR DE VISCOSIDAD". Para el cumplimiento de este objetivo se aplican los pasos del método científico de la siguiente manera: análisis de la situación actual, planteamiento de hipótesis, uso del método de prueba de la hipótesis, recolección de datos y diseño del experimento, aplicación de la metodología sobre los datos y conclusión sobre la hipótesis.

Esta investigación se presenta entregando el estado del arte en los primeros cinco capítulos y los siguientes se refieren a la investigación propiamente dicha. El primer capítulo trata los sistemas genéticos naturales, el segundo trata la teoría de algoritmos genéticos; el tercero muestra la fundamentación matemática para la comprensión de los temas expuestos; el cuarto se refiere a la Instrumentación y Sensórica, el quinto describe la metodología de clonación artificial, el sexto presenta la solución propuesta, la prueba experimental y los resultados obtenidos en cumplimiento de la hipótesis objeto de la investigación, finalmente el capítulo

séptimo presenta las conclusiones y recomendaciones.

Se sientan las bases para la clonación artificial de un sensor de viscosidad de fenol, haciendo uso de la técnica posibilística que se encuentra implícita en la teoría de la lógica difusa de Zadeth [15] , de la metodología general de clonación propuesta por el director del trabajo de grado y la metodología utilizada por los autores del trabajo, abriendo así, un nuevo campo para la investigación en esta área.

CAPÍTULO 2

LA NATURALEZA COMO OPTIMIZADORA[1]

"Aunque el ingenio humano puede lograr infinidad de inventos, nunca ideará ninguno mejor, más sencillo y directo que los que hace la naturaleza, ya que en sus inventos no falta nada y nada es superfluo", escribió Leonardo Da Vinci (Cuaderno de Notas, la vida y estructura de las cosas, el embrión). Un siglo después, Johannes Goldschmidth, mejor conocido por su nombre latino Fabriciusab Aquapendente (quien se disputara con Galileo el descubrimiento de los defectos solares), afirmaría, anticipándose a Darwin: "La naturaleza perpetúa aquello que resulta mejor". Todos los seres vivos que habitamos en este planeta somos, de alguna manera, obras casi perfectas. Al percatarse de la vasta complejidad del ser, de cómo el cuerpo es una enredada madeja de relaciones delicadamente establecidas entre los distintos órganos, no podemos más que maravillarnos y otro tanto ocurre cuando percibimos al resto de los seres vivos de la misma manera. Recientemente, un grupo de ingenieros que diseñaba un submarino se percató de que el pingüino posee una forma tal, que al fluir el agua a su alrededor, prácticamente no genera turbulencias en las que se pierda la energía invertida por el ave al nadar y al mismo

[1]ESTE TEXTO ES UNA ADAPTACIÓN DE EXTRACTOS DEL LIBRO ALGORITMOS GENÉ TICOS DE ANGEL KURY Y JOSÉ GALAVIZ A SER PUBLICADO POR EL FONDO DE CULTURA ECONÓMICA. MAYO DE 2000

tiempo, posee un volumen cercano al máximo, lo que permite al pingüino acumular gran cantidad de grasa que lo aísla del frío exterior y constituye su reserva de energía. Se podrían citar miles de ejemplos más: "[...] cuanto más informados e iluminados estemos acerca de las obras de Dios, más inclinados estaremos a encontrarlas excelentes y totalmente conformes a cuanto se hubiera podido desear", diría Leibniz (Discours de Metaphysique). La naturaleza (o Dios, como prefiere Leibniz) genera seres óptimos, seres perfectamente adaptados a su entorno, cons-tituido por una infinidad de circunstancias: temperatura, presión atmosférica, precipitación, velocidad del viento, altitud, nivel de insolación, depredadores, ali-mentos, etc. En función de su entorno el pingüino es un animal perfecto para vivir en las soledades antárticas; en el Amazonas perecería. Su adaptación es perfecta porque, a lo largo de miles de generaciones se ha perfeccionado a pequeños saltos infinitesimales. Lo que vemos hoy en día es el resultado acumulado de miles de experimentos exitosos que han ido refinando paulatinamente alguna creación primigenia. Los experimentos fallidos, probablemente algunos órdenes de magnitud más numerosos que los exitosos, no los vemos. Los individuos resultantes perecieron compitiendo al lado de otros más aptos para sobrevivir. "La selección natural obra solamente mediante la conservación y acumulación de pequeñas modificaciones heredadas, provechosas todas al ser conservado.", escribió Darwin y dio nombre a este proceso (El origen de las especies, selección natural). Estas "modificaciones heredadas", señaladas por Darwin como las generadoras de organismos mejores, son llamadas

mutaciones hoy en día y constituyen el motor de la evolución. Un organismo mutante ha sufrido una modificación que lo hace diferente al resto de sus congéneres. Esta modificación puede ser un inconveniente para él (la falta de algún miembro útil de su cuerpo, por ejemplo), pero puede ocurrir también que le confiera alguna cualidad que le permita sobrevivir más fácilmente que al resto de los individuos de su especie. Este organismo tendrá mayor probabilidad de reproducirse y heredará a sus descendientes la característica que le dio ventaja. Con el tiempo, gracias a la competencia, los organismos que en un principio eran raros se volverán comunes a costa de la desaparición del "modelo anterior". Se habrá dado entonces un paso en el proceso evolutivo.

Esta cualidad del proceso natural de la evolución (generar organismos óptimos sobre los que influyen infinidad de variables), llamó la atención de algunos científicos en las décadas de los cincuenta y sesenta. Un alemán, de apellido Rechenberg, introdujo en 1965 lo que denominó evolutions strategie, o estrategias evolutivas, como un método para optimizar funciones de varias variables que definían dispositivos tales como perfiles de alas de avión. En 1966 los investigadores Fogel, Owens y Walsh se dieron a la tarea de dejar evolucionar máquinas de estados finitos sometiendo sus diagramas de transición a cambios aleatorios (mutaciones), creando con ello lo que se conoce como programación evolutiva. También en los sesenta, John Holland, junto con algunos de sus colegas y alumnos de la Universidad de Michigan, desarrolló lo que se conoce como algoritmos genéticos (AGs).

Sin embargo, el objetivo de Holland no era inventar un método de optimización basado en los mecanismos de la evolución natural, sino elaborar un estudio formal acerca del fenómeno de adaptación en sistemas naturales y artificiales, es decir, definir un modelo para la adaptación. El algoritmo genético definido por Holland en su trabajo [7] es un intento de abstraer las características escenciales del proceso evolutivo tal como se observa en la naturaleza, con vistas a importarlo a sistemas de cómputo.

En la actualidad, los AGs son preferentemente utilizados como métodos de búsqueda de soluciones óptimas que simulan la evolución natural y han sido usados con éxito en la solución de problemas de optimización combinatoria, optimización de funciones reales y como mecanismos de aprendizaje de máquina (machine learning). Esto último les ha ganado un lugar en el campo de la inteligencia artificial.

Discusión de la utilidad del aporte del autor de este artículo para esta investigación: El apartado anterior es una motivación al lector, para que se comprendan los orígenes y la estrapolación que se hace del comportamieto de la naturaleza, para representarla mediante computación evolutiva.

2.1. ANALOGÍA CON LA BIOLOGÍA

Discusión de la utilidad del aporte del autor de este artículo para esta investigación: En esta sección se presenta lo que se sabe acerca de los mecanismos de

la herencia y la terminología utilizada en esta área. El objetivo es evidenciar la analogía entre los algoritmos genéticos y aquello que pretenden simular, lo cual se relaciona directamente con el propósito de ésta investigación.

[13]Cada individuo de cada una de las especies que habitan en el planeta poseen ciertas características que lo identifican. Si se habla de seres humanos, cada uno posee cierta estatura, cierto color de ojos y de cabello y cierto tipo sanguíneo, entre otras muchas. Estas características "externas", aunque algunas de ellas no se puedan ver, como el tipo sanguíneo, constituyen lo que se denomina el fenotipo de un individuo. Cada una de estas características es igual a la correspondiente de alguno de los antecesores del individuo, es decir, son dadas por herencia, o por lo menos se dada con cierta predisposición a ella (como la diabetes, por ejemplo). El fenotipo es resultado de la interacción del medio ambiente en que se desarrolla un individuo y la herencia que éste recibe de sus ancestros. La herencia impone ciertos límites o predisposiciones que, al sumarse con el medio, generan el fenotipo. A veces el medio no importa, por ejemplo, no puede intervenir en el color de ojos, pero en otras influye de manera determinante. Si se posee cierta predisposición a padecer enfermedades cardiovasculares pero se tiene una excelente condición aeróbica desde pequeño, posiblemente éstas nunca se padezcan. El fenotipo de cada individuo está determinado por las proteínas que produce, y esto a su vez está definido en la información genética de cada una de sus células. La información acerca de cuáles proteínas se producirán está contenida en los cromosomas del

individuo. En cada célula somática (aquellas que constituyen el organismo) existen dos juegos de cromosomas que definen las mismas características; un juego es aportación del padre del individuo y el otro lo es de la madre.

Un ser humano posee 23 pares de cromosomas. En términos sencillos, un cromosoma es una larga molécula de ADN (Ácido Desoxirribonucleico), formada por cuatro distintos compuestos más simples llamados bases o nucleótidos: adenina (A), guanina (G), citosina (C) y timina (T). Cada subcadena de tres nucleótidos codifica un aminoácido diferente que al unirse con los generados por otros tercetos, formará una proteína. A las subcadenas de tres nucleótidos se les llama codones y al conjunto de nucleótidos que codifican una proteína completa se les llama genes. El valor que posee un gen determinado se denomina alelo. Las células que poseen dos juegos de cromosomas se denominan diploides, ya que en éstas los cromosomas se encuentran en pares; también los genes deben estar en pares homólogos. Si ambos tienen el mismo alelo se dice que son homocigos, si no, son heterocigos, y en este último caso sólo uno de los alelos se manifestará en el fenotipo (éste se denomina dominante, su homólogo que no se manifiesta, se llama recesivo). El conjunto de todos los cromosomas, es decir, toda la información genética de un individuo se llama genoma y el conjunto de genes contenidos en el genoma genotipo. Es éste el que determina, en buena medida, el fenotipo del individuo. Hay unas células especiales llamadas gametos, que intervienen en la reproducción (los espermatozoides y los óvulos humanos pertenecen a esta categoría). Los gametos

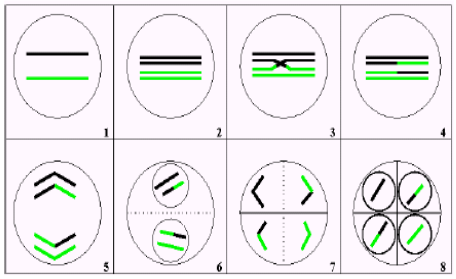


Figura 2.1: Reproducción celular por meiosis

no se reproducen por mitosis como el resto de las células, el proceso de división se llama en este caso meiosis. En la mitosis las células producidas son diploides, mientras que en la meiosis el resultado, los gametos, son haploides, sólo tienen un juego de cromosomas. Partiendo de una sola célula diploide el proceso meiótico es como sigue (figura 1):

1. Se duplica el número de cromosomas en la célula, esto es, se hace una copia de cada cromosoma. Al final quedan dos juegos correspondientes al padre y dos a la madre.

2. Se cruzan un juego de cromosomas del padre con uno de la madre, formándose dos juegos de cromosomas híbridos. El resultado es un juego de cromosomas puros del padre, un juego puro de la madre y dos juegos de cromosomas híbridos.

3. Se divide la célula dos veces y al final del proceso quedan cuatro células haploides: una con cromosomas puros del padre, una con cromosomas puros de la madre y dos con cromosomas híbridos.

Figura 2.2: El proceso de cruzamiento de dos cromosomas

En el paso dos del proceso de meiosis se mezclan las características del padre y la madre. Para el cruzamiento de dos cromosomas se forman entre ellos puntos de ruptura y unión de las cadenas de ADN. Estos puntos, llamados quiasmas, cortan el cromosoma en segmentos llamados cromátidas y unen cromátidas complementarias de dos distintos cromosomas. Al final cada cromosoma que participó en la cruza queda constituido por segmentos que ya poseía y por otros que eran de su análogo (figura 2).

En el paso uno del proceso se deben replicar los cromosomas existentes. Hay una enzima encargada de copiarlos, ésta es la ADN-polimerasa. La molécula de ADN tiene forma de una doble hélice, como una escalera de caracol. La enzima abre por en medio los "escalones" de esta hélice y ensambla en cada mitad los nucleótidos que debe ensamblar (figura 3). Ocasionalmente esta enzima comete

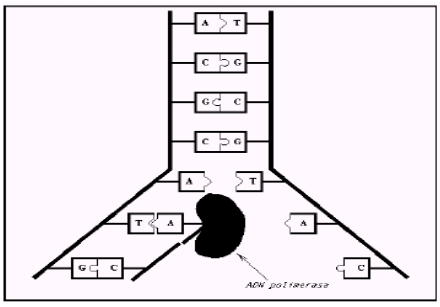


Figura 2.3: Proceso de replicación de la molécula de ADN

un error, que puede ser causado por radiaciones energéticas externas o sustancias extrañas. La alteración de la molécula de ADN original constituye una mutación que puede manifestarse en el fenotipo y hacer al individuo diferente del resto de sus congéneres. Es muy poco probable que cambiar al azar un trozo de información que la naturaleza ha refinado cuidadosamente a lo largo de millones de años resulte en algo bueno. Por lo general las mutaciones son desfavorables, incluso letales, para el organismo mutante. Pero ocasionalmente pueden no serlo y conferirle a dicho organismo alguna ventaja que le permita sobrevivir más fácilmente en su medio. Esta característica será transmitida a sus descendientes y un pequeño paso evolutivo se habrá dado.

Al principio existen dos pares de cromosomas, un par del padre y otro de la madre; entre ellos se forman puntos de ruptura y unión (quiasmas) que definen segmentos llamados cromátidas. Se unen cromátidas complementarios para formar dos cromosomas híbridos.

CAPÍTULO 3

ALGORITMOS GENÉTICOS[1]

Discusión de la utilidad del aporte del autor de este artículo para esta investigación: Se mencionó ya que los algoritmos genéticos simulan el proceso de evolución natural. En esta sección se aclara la manera como se lleva a cabo esta simulación. Este capítulo muestra el estudio exhaustivo que realizaron los autores sobre los diferentes tipos de algoritmos genéticos que permiten encontrar soluciones en problemas similares al abordado en la investigación. La codificación del dominio permitío determinar los parámetros en conjunto, se llegó a la conclusión de que no es conveniente trabajar con los parámetros de manera individual, como cadenas de bis de longitud finita, es así que los cromosomas diseñados tienen estructuras de árboles n-arios; Como se refleja en el capítulo de la solución propuesta, los autores proponen una técnica para la aplicación de los operadores genéticos, que en últimas permiten clonar el sensor real. Se estudiaron diferentes métodos de optimización y al final del capítulo se presenta una comparación entre ellos, los autores concluyen que para el proceso de clonación se debe utilizar la programación genética, técnica que por su aporte a la investigación, se explica en el

[1]ESTE TEXTO ES UNA ADAPTACIÓN DE EXTRACTOS DEL LIBRO .ALGORITMOS GENÉTICOS"DE ANGEL KURY Y JOSÉ GALAVIZ A SER PUBLICADO POR EL FONDO DE CULTURA ECONÓMICA. MAYO DE 2000.

capítulo dedicado a la solución propuesta.

3.1. CODIFICACIÓN DEL DOMINIO

En la naturaleza las características de los seres vivos, incluso aquéllas que los hacen óptimos para habitar en su medio, están determinadas por las proteínas que producen. A su vez, como hemos visto, estas proteínas (o más bien, los aminoácidos que las forman) se codifican en el material genético contenido en cada una de las células del individuo. Así pues, la naturaleza ha mapeado cada posible solución al problema de crear un individuo óptimo en una secuencia particular de bases que producirá ciertas proteínas, ha codificado el dominio del problema (todos los posibles individuos) mapeándolo al conjunto de todas las posibles secuencias de nucleótidos.

Así, para un algoritmo genético lo primero que se requiere es determinar en qué espacio se encuentran las posibles soluciones al problema que se pretende resolver. En caso de tener un problema de optimización de una función cuyo dominio es un subconjunto de los números reales, entonces este subconjunto es al que se refiere. Pero el algoritmo opera sobre "códigos genéticos", sobre genotipos que se deberán mapear al espacio de soluciones. Es decir, es necesario codificar de alguna manera el dominio del problema para obtener estructuras manejables que puedan ser manipuladas por el AG. Cada una de estas estructuras constituye el equivalente al genotipo de un individuo en términos biológicos. El elemento del dominio del

problema al que se mapea este genotipo es el análogo al fenotipo. Es frecuente que el código de los elementos del dominio del problema utilice un alfabeto binario (0's y 1's). Una vez que se ha definido la manera de codificar los elementos del dominio del problema y se conoce la forma de pasar de un elemento a su código y viceversa, es necesario fijar un punto de partida. Los algoritmos genéticos manipulan conjuntos de códigos en generaciones sucesivas. Nuevamente haciendo una analogía, manipulan poblaciones de códigos. En éstas un código puede aparecer más de una vez. El algoritmo se encargará de favorecer la aparición en la población de códigos que correspondan a elementos del dominio que estén próximos a resolver el problema. En resumen, el algoritmo recibirá como entrada una población de códigos y a partir de ésta generará nuevas poblaciones, donde algunos códigos desaparecerán mientras que otros, que se mapean en mejores soluciones posibles, aparecen con más frecuencia hasta que se encuentra una satisfactoria o hasta que se cumple alguna otra condición de terminación. A lo largo del texto, los códigos en una población, es decir, los elementos de ésta serán llamados individuos y a los códigos en general, ya no en el contexto exclusivo de una población, se les denominará indistintamente cromosomas, genotipo, genoma o código genético, por analogía con los términos biológicos de donde surgen.

3.1.1. Evaluación de la población.

En la naturaleza hay individuos más hábiles que otros para sobrevivir. En una manada de gacelas hay unas más rápidas que otras, hay algunas enfermas o propensas a enfermar, hay algunas más débiles que otras. Todas las características mencionadas señalan alguna diferencia entre los individuos. Además, esta diferencia es relativa, es decir, siempre está referida al resto de la población de gacelas.

Se dice: "ésta es más rápida que el resto de la población" o "aquélla es más saludable que el promedio de sus congéneres". De alguna manera, siempre se relaciona al individuo con la población a la que pertenece. Si se considera cada una de estas características como medidas del desempeño de cada individuo, se está hablando de que el desempeño de cada individuo de la población está en función del desempeño de sus congéneres. Por ejemplo, ¿qué tan veloz debe ser una gacela para evitar ser cazada por un cheetah? Si es más rápida que el cheetah está salvada. Pero lograr eso es difícil, casi no habría gacelas. Más bien la respuesta correcta es relacionar al individuo con el resto de su población y decir: debe ser más rápida que la gacela más lenta, de este modo el depredador perseguirá a otra (claro que después de la primera cacería la medida de desempeño cambia porque lo ha hecho la población misma). Al igual que en la naturaleza, en los algoritmos genéticos es necesario establecer algún criterio que permita decidir cuáles de las soluciones propuestas en una población son mejores respecto del resto de las propuestas y

cuáles no lo son. Es necesario establecer, para cada individuo, una medida de desempeño relativa a la población a la que pertenece. Para determinar cuáles de estos individuos corresponden a buenas propuestas de solución y cuáles no, es necesario calificarlos de alguna manera. Cada individuo de cada generación de un algoritmo genético recibe una calificación o, para usar el término biológico, una medida de su grado de adaptación (fitness). Éste es un número real no negativo tanto más grande cuanto mejor sea la solución propuesta por dicho individuo. El objetivo de este número es que permita distinguir propuestas de solución buenas de aquéllas que no lo son. Si el problema a resolver consiste en maximizar una función, entonces la calificación asignada a un individuo determinado debe indicar qué tan alto es el valor de la función en el elemento de su dominio codificado por el individuo. Si, en cambio, el problema es determinar la ruta más corta entre dos puntos, la calificación deberá ser tanto más alta cuanto más corto sea el camino codificado en el individuo que esté siendo calificado.

Evidentemente, al hablar de que a cada individuo de la población se le signa una y sólo una calificación, se está hablando de una función que se denomina función de adaptación, cuya evaluación puede no ser sencilla y es, de hecho, lo que en la mayoría de los casos consume más tiempo en la ejecución de un algoritmo genético. Hay que tener en cuenta que se evalúa una vez en cada individuo de cada generación. Si un AG es ejecutado con una población de tamaño 100 durante 100 generaciones, la función es evaluada 10,000 veces. Además, puede darse el caso

de que la función de evaluación no tenga una regla de correspondencia explícita, esto es, una expresión algebraica, y puede ocurrir incluso que la función cambie de generación en generación.

3.1.2. Selección

Una vez calificados todos los individuos de una generación, el algoritmo debe, al igual que lo hacen la naturaleza y el hombre, seleccionar a los individuos más calificados, mejor adaptados al medio, para que tengan mayor oportunidad de reproducción. De esta forma se incrementa la probabilidad de tener individuos "buenos" (con alta calificación) en el futuro. Si de una determinada generación de individuos se seleccionaran sólo aquellos con una calificación mayor o igual que cierto número c para pasarlos a la siguiente generación, es claro que en ésta la calificación promedio superará c y por tanto al promedio de la generación anterior. La selección ocasiona que haya más individuos buenos, explota el conocimiento que se ha obtenido hasta el momento, procurando elegir lo mejor que se haya encontrado, elevando así el nivel de adaptación de toda la población. Más adelante en el texto se verá qué tan importante es esta explotación. En principio podría parecer que es conveniente tener una estrategia de selección estricta para que mejore rápidamente la población y converja el algoritmo, es decir, que la población se acumule alrededor de un genotipo óptimo. Esto no es cierto. Lo que ocurrirá es que la población se acumulará rápidamente alrededor de algún individuo que

sea bueno, comparativamente con el resto de los individuos considerados a lo largo de la ejecución del algoritmo, pero este individuo puede no ser el mejor posible. A esto se le suele llamar convergencia prematura. No se puede asegurar pero sí procurar que lo anterior no ocurra. Además de la explotación es necesario que exista exploración. El AG debe, no sólo seleccionar de entre lo mejor que ha encontrado, sino procurar encontrar mejores individuos. A esto se dedican los operadores que serán descritos a continuación, los que aseguran que en todo momento exista cierto grado de variedad en la población, procurando con ello que no se "vicie".

En la estrategia de selección normalmente se incluye un elemento extra que sirve de "ancla". Si sólo se hace selección forzando que sea más probable elegir al mejor individuo de la población pero sin asegurarlo, es posible que este individuo se pierda y no forme parte de la siguiente generación. Para evitar lo anterior se fuerza la selección de los mejores n individuos de la generación para pasar intactos a la siguiente. A esta estrategia se le denomina elitismo y puede ser generalizada especificando que permanezcan en la población los n mejores individuos de las pasadas k generaciones.

3.1.3. Cruzamiento.

Durante la meiosis ocurre el proceso de producción de gametos. El código genético de los padres de un individuo se mezcla para producir gametos cuyo

contenido genético es híbrido, es decir, una mezcla. De esta manera es posible que un individuo herede a sus descendientes las características mezcladas de sus propios padres, por ejemplo: el color de ojos del padre y el de cabello de la madre o, para aprovechar el ejemplo mencionado antes, es posible que una gacela herede la velocidad de su abuelo paterno y la fuerza de su abuela paterna, la salud de su abuelo materno y la agudeza visual de su abuela materna. Si estas características le confirieron a sus ancestros una alta aptitud de sobrevivencia, entonces este individuo será, con alta probabilidad, un individuo exitoso en su manada. La cruza de los códigos genéticos de individuos exitosos favorece la aparición de nuevos individuos que hereden de sus ancestros características deseables.

En el contexto de los algoritmos genéticos reproducirse significa que, dados dos individuos seleccionados en función de su grado de adaptación, éstos pasen a formar parte de la siguiente generación o, al menos, mezclen sus códigos genéticos para generar hijos que posean un código híbrido. Es decir, los códigos genéticos de los individuos se cruzan. Existen muchos mecanismos de cruzamiento.

En esta sección sólo se presenta uno de ellos pero todos tienen por objeto que el código de un individuo A y el de uno B, previamente seleccionados, se mezclen, es decir, se fragmenten y recombinen para formar nuevos individuos con la esperanza de que éstos hereden de sus progenitores las características deseables. El mecanismo de cruzamiento más común es el llamado cruzamiento de un punto, el cual se describe con detalle más adelante. Por ahora considérese como una

analogía directa del proceso de cruzamiento descrito en la sección anterior, a propósito de la generación de gametos, con la formación de un único quiasma entre los cromosomas (individuos). A este quiasma es al que se le denominará más adelante punto de corte.

3.1.4. Mutación

Algunas veces, muy pocas de hecho, la ADN-polimerasa (la enzima encargada de replicar el código genético), se equivoca y produce una mutación, una alteración accidental en el código genético de los seres vivos.

Ocasionalmente algunos elementos del código de ciertos individuos de un algoritmo genético se alteran a propósito. Éstos se seleccionan aleatoriamente en lo que constituye el símil de una mutación. El objetivo es generar nuevos individuos, que exploren regiones del dominio del problema que probablemente no se han visitado aún. Esta exploración no presupone conocimiento alguno, no es sesgada. Aleatoriamente se buscan nuevas soluciones posibles que quizá superen las encontradas hasta el momento. Esta es una de las características que hacen aplicables los algoritmos genéticos a gran variedad de problemas: no presuponer conocimiento previo acerca del problema a resolver ni de su dominio, no sólo en la mutación sino en el proceso total. De hecho, el problema a resolver sólo determina la función de evaluación y la manera de codificar las soluciones posibles (la semántica de los códigos genéticos de los individuos). El resto de los subprocesos

que constituyen el algoritmo son independientes y universalmente aplicables.

3.1.5. El algoritmo genético simple (AGS)

En su trabajo, Holland propone una manera de seleccionar individuos y de cruzarlos. Actualmente existen muchas otras propuestas, pero las de Holland constituyen aún hoy la base de muchos desarrollos teóricos y prácticos en el área. Goldberg [6] retomó el algoritmo de Holland y lo popularizó llamándolo algoritmo genético simple (AGS). En éste se considera que los códigos genéticos están en binario. Explicado con detalle, el proceso de un AGS es:

1. Decidir cómo codificar el dominio del problema.

2. Generar un conjunto aleatorio (población inicial) de N posibles soluciones codificadas al problema. A ésta se le llamará la población actual.

3. Calificar cada posible solución (individuo) de la población actual.

4. Seleccionar dos individuos de la población actual con una probabilidad proporcional a su calificación.

5. Lanzar una moneda al aire (con probabilidad pc cae cara).

6. Si cayó cara mezclar los códigos de los dos individuos seleccionados para formar dos híbridos, a los que llamaremos nuevos individuos.

7. Si cayó cruz llamamos a los individuos seleccionados nuevos individuos.

8. Por cada bit de cada nuevo individuo lanzar otra moneda al aire (con probabilidad pm cae cara).

9. Si cae cara cambiar el bit en turno por su complemento.

10. Si cae cruz el bit permanece inalterado.

11. Incluir a los dos nuevos individuos en una nueva población.

12. Si la nueva población tiene ya N individuos, llamarla población actual y regresar al paso 3, a menos que se cumpla alguna condición de terminación.

13. Si no, regresar al paso 4.

En el algoritmo se utiliza el término "lanzar una moneda al aire" para decir un experimento de Bernoulli (aquel en el que pueden ocurrir exclusivamente dos eventos posibles, uno con probabilidad p y otro con probabilidad 1-p). Es decir, el lanzamiento de una moneda "extraña" en la que no necesariamente ambas caras son equiprobables.

La condición de terminación, a la que se hace referencia en el paso 12, puede definirse de muchas maneras. Se puede fijar un número máximo de generaciones que se pretende ejecutar el algoritmo, o puede decidirse hacer alto cuando la mayoría de la población, digamos el 85 %, tenga una calificación que esté dentro de 0.6 desviaciones estándar de la media. En fin, opciones hay muchas. Generalmente depende del problema o de las preferencias personales la decisión acerca de cuándo es conveniente detenerse.

En el paso 4 se menciona que hay que seleccionar dos individuos con probabilidad proporcional a su calificación. Este tipo de selección proporcional es también llamado de "ruleta" (roulette wheel selection) por lo siguiente: supóngase que

se suman las calificaciones de todos los individuos de la población y esta suma es considerada el 100 % de una circunferencia. Luego, a cada individuo se le asigna el trozo que le corresponde de ésta según su aportación a la suma de las calificaciones. Es decir, si la calificación de un individuo es xi entonces le corresponde un segmento de circunferencia dado por la simple regla de tres:

$$s = 2\pi \frac{X_i}{\sum_j x_j} \tag{3.1}$$

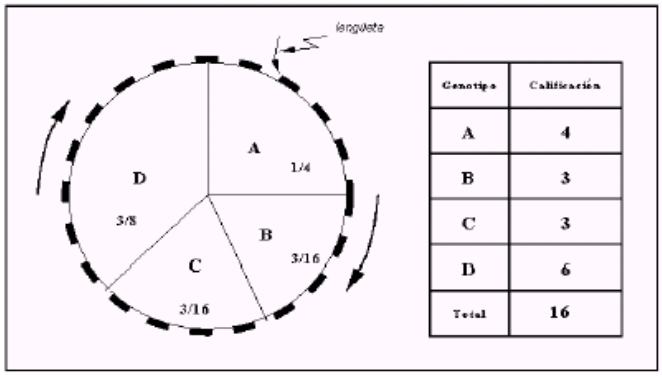
¿Qué ocurrirá entonces si se considera ésta como una ruleta y se coloca una lengüeta que roce el borde de ella? (figura 4). La probabilidad de que dicha lengüeta quede en el arco correspondiente al individuo de calificación xi, cuando la rueda se detenga tras realizar algunos giros, es:

$$p(j) = \frac{x_i}{\sum_j x_j} \tag{3.2}$$

lo que es proporcional a la calificación (xi) del individuo.

Aún queda por aclarar cómo es que se mezclan los códigos de dos individuos para formar dos híbridos. En general hay muchas maneras de hacerlo. Pueden ocurrir muchas formas, sin embargo, la que se usa en el AGS y una de las más populares, es el cruzamiento de un punto (1-point crossover) ya mencionado.

En este tipo de cruzamiento, dados dos individuos se elige aleatoriamente un punto de corte entre dos bits cualesquiera del cromosoma. Esto define segmentos izquierdos y derechos en cada genotipo. Se procede entonces a intercambiar los segmentos derechos (o los izquierdos, indistintamente) de cada individuo. De esto



Genotipo	Calificación
A	4
B	3
C	3
D	6
Total	16

Figura 3.1: Selección proporcional o por ruleta

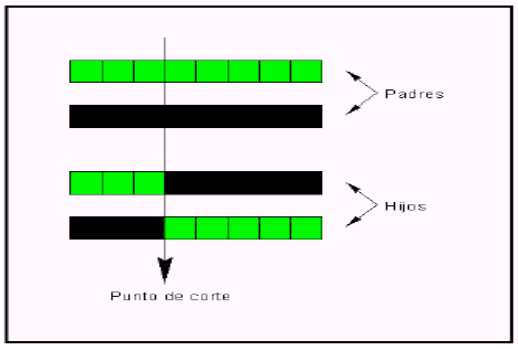


Figura 3.2: Cruzamiento de un punto (1-point crossover)

resultan, al igual que en el caso del cruzamiento de cromosomas, dos híbridos (figura 5).

A lo largo del texto se hará referencia frecuentemente al algoritmo genético simple como aquel caracterizado por tener:

1. Tamaño de población fijo en todas las generaciones.

2. Selección proporcional (de ruleta).

3. Cruzamiento de un punto. La probabilidad de cruza se mantiene fija para

todas las generaciones y todas las parejas.

4. Mutación uniforme (todas las posiciones de las cadenas genéticas tienen la misma probabilidad de ser cambiadas). La probabilidad de mutación permanece fija para todas las generaciones y todas las posiciones de los individuos.

5. Selección no elitista. Esto es, no se copian individuos de una generación a otra sin pasar por el proceso de selección aleatoria (en este caso, proporcional).

3.2. OTROS MÉTODOS DE OPTIMIZACIÓN

En esta sección se presentan algunos métodos de optimización, con el objetivo de tener un panorama de las distintas alternativas que existen para el uso de los algoritmos genéticos. Algunos de los métodos presentados efectúan la búsqueda de puntos óptimos mediante estrategias que, en buena medida, dependen de eventos aleatorios. A este tipo de estrategias se les conoce como heurísticas.(Los algoritmos genéticos operan de esta manera.)

3.2.1. Métodos tradicionales.

Las técnicas que aquí se muestran han sido seleccionadas porque son representativas de los dos grandes grupos de métodos de optimización: los directos (que no requieren más que los valores de la función objetivo), como simplex, y los de ascenso (que, además, requieren los valores de la o las derivadas de la función), como el de Newton y el de ascenso de máxima pendiente. Existen otros muchos

métodos dentro de esta clasificación que son bastante más útiles que los que se muestran aquí. Sin embargo, el objetivo de esta sección es presentar, con propósitos didácticos, un panorama representativo de los métodos de optimización no heurísticos.

3.2.1.1. Método de Newton

En los cursos elementales de cálculo diferencial se les enseña a los alumnos que, bajo ciertas condiciones, es posible encontrar los puntos del dominio de una función f:A ■B (A,B ∈ R), donde ésta alcanza sus valores extremos (máximos y mínimos) encontrando las soluciones de la ecuación:

$$f\prime(x) = 0 \tag{3.3}$$

donde f ' denota la derivada de f. Resolver (1) analíticamente puede no ser fácil, así que en ocasiones es necesario encontrar soluciones numéricas aproximadas. Antes que cualquier otra cosa se especificarán las condiciones necesarias para que funcione el método que aquí se describe. Se supondrá que: f es continua y derivable en su dominio al igual que su derivada f ', la doble derivada f " es distinta de cero en todo su dominio y existen ciertos puntos donde f ' tiene un valor negativo y otros donde tiene un valor positivo. Sean a y b dos puntos del dominio de f tales que a < b y f ' (a) y f ' (b) tienen signos opuestos, como f ' es continua entonces, por el teorema del valor intermedio, existe una h ■entre a y b tal que

f ' (h) = 0.

Sean: x0=b, l la recta tangente en (x0,f ' (x0)) a la curva descrita por f ', x1 el punto donde l corta el eje x y q ■el ángulo en que lo corta (figura 7). Es evidente que:

$x_1 = x_0 - (x_0 - x_1)$

además: $\frac{f'(x)}{x_0 - x_1} = \tan(\theta) f''(x_0)$

por lo tanto: $x_0 - x_1 = \frac{f'(x)}{f''(x_0)}$

es decir: $x_1 = x_0 - \frac{f'(x)}{f''(x_0)}$

En general, iterando el proceso:

$x_r = x_{r-1} - \frac{f'(x_{r-1})}{f''(x_{r-1})}$

Los valores que se obtienen de [2] conforme crece r son cada vez más próximos a algún valor $\xi \in A$, donde f ' (ξ)=0. Si la función f ' tiene varios puntos donde se anula, entonces para encontrar los demás será necesario probar con otros valores para x0. El lector interesado en profundizar en este método puede consultar [1].

3.2.1.2. Búsqueda de Fibonacci.

Este método sirve para buscar el máximo o mínimo de funciones de R■R. El objetivo es acotar el punto donde la función alcanza su valor extremo por intervalos cada vez más pequeños, de esta forma al final se obtiene un intervalo (x, x+e) en cuyo interior está el punto buscado. Supóngase que se quiere localizar el punto donde una función alcanza su máximo y que es posible hacer hasta n

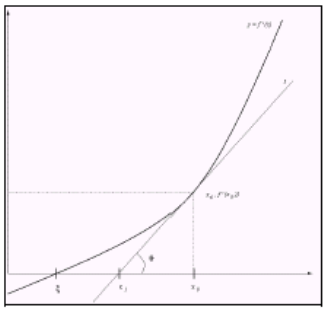


Figura 3.3: Ilustración del método de Newton para hallar los puntos donde f' se anula. x0 es el punto inicial, a partir de éste se encuentra x1.

evaluaciones de la función. La pregunta es: ¿De qué manera es posible determinar el valor de los puntos de muestra de la función, de tal forma que al final se tenga el intervalo de incertidumbre más pequeño posible? Supóngase ahora que se poseen tres puntos donde es conocido el valor de la función x1 < x2 < x3 y sean: L= x2-x1 y R= x3-x2 donde L>R, tal y como se muestra en la figura 8. Se desea determinar el punto x4 tal que se obtenga el intervalo de incertidumbre más pequeño posible. Si x4 es elegido en (x1, x2) entonces: Si f(x4) > f(x2) el nuevo intervalo de incertidumbre será (x1, x2) de longitud x2- x1=L. Si f(x4) < f(x2) el nuevo intervalo de incertidumbre será (x4, x3) de longitud x3- x4.

Dado que no es posible determinar a priori cuál de estas dos cosas ocurrirá, es necesario elegir x4 de forma tal que se minimicen al mismo tiempo x2-x1 y x3-x4. Esto se logra haciéndolas iguales, es decir, colocando x4 en una posición simétrica con respecto a x2.

En general, en cualquier nivel de iteración n>3 del algoritmo, si se denomina

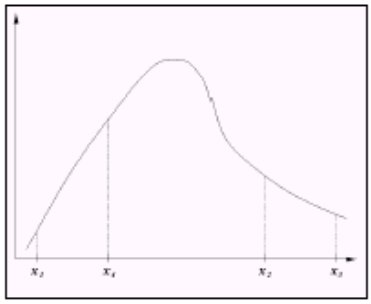


Figura 3.4: Ilustración del método de búsqueda de Fibonacci.

Ln la longitud del intervalo de incertidumbre, se tiene que: Ln-2 = Ln-1 + Ln. Además, si en ese nivel ocurre que xn-xn-1 = e, ■entonces Ln-1 = 2Ln-ε■ (véase figura 9), de donde:

$L_{n-1} = 2L_{n-\varepsilon}$

$L_{n-2} = L_{n-1} + L_n = 3L_{n-\varepsilon}$

$L_{n-3} = L_{n-2} + L_{n-1} = 5L_{n-2\varepsilon}$

$L_{n-3} = L_{n-3} + L_{n-2} = 8L_{n-3\varepsilon}$ etc.

La sucesión de números de Fibonacci es: F0=1, F1=1 y Fk =Fk-1 + Fk-2 para =2,3,... Por lo que generalizando, es posible escribir:

$L_{n-j} = F_{j+1} - F_{j-1\varepsilon} \qquad j = 1, 2, ..., n-1$

Esta es la razón del nombre búsqueda de Fibonacci.

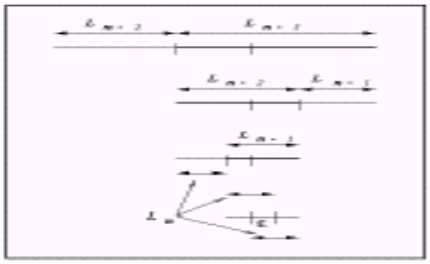


Figura 3.5: Relación entre las longitudes de los intervalos en la búsqueda de Fibonacci. Con líneas gruesas se han denotado los intervalos de incertidumbre.

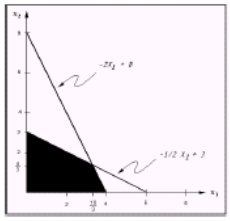


Figura 3.6: Dominio restringido del problema de programación lineal presentado como ejemplo.

3.2.2. Métodos heurísticos

3.2.2.1. Búsqueda tabú.

Este método de búsqueda utiliza cierta memoria o conocimiento acerca del dominio para encontrar una buena solución al problema que se pretende resolver, o una solución, al menos, cercana a la óptima. El método se deriva del trabajo de Glover, y desde su creación ha recibido numerosas contribuciones y se han desarrollado algunas variantes.

Sea X el dominio de búsqueda relacionado con algún problema y sea x∈■X

una posible solución. Para buscar puntos mejores en X se define, de alguna manera conveniente, la vecindad de un punto cualquiera en X. De esta manera es posible buscar soluciones mejores que x entre sus vecinos, a los que se denotará con N(x). Conforme pasa el tiempo y se va explorando más el dominio, será conveniente excluir algunos puntos que se sabe no son buenos (para no buscaren vano). De manera que la vecindad de un punto x cambia con el tiempo, depende de la historia (H) del proceso de búsqueda. Así que es conveniente denotar con N(x) sólo a la vecindad inicial de x, y con N(H,x) la vecindad de x como función de la historia. Algunos puntos que estaban en N(X) y que, al paso del tiempo, resulten no ser buenos candidatos de solución, serán marcados como tabú y no podrán ser alcanzados desde x, es decir, no serán elementos de N(H,x). También, conforme transcurre el tiempo la función que evalúa la conveniencia de cada solución posible se puede modificar y, en general, se denota con b(H,x).

Al principio de la ejecución del algoritmo es deseable identificar regiones donde se obtienen buenas soluciones y regiones donde se obtienen malas. Esto es posible si se posee suficiente diversidad, es decir, si se procura la exploración. En etapas tardías del algoritmo es mejor hacer una búsqueda más intensiva dentro de las regiones buenas para aproximarse al óptimo. Esto es, en general, válido para varias heurísticas. Cuando es costoso determinar qué elementos están en N(H,x) se selecciona una muestra representativa de la vecindad $NC(H,x) \subseteq N(H,x)$. En la búsqueda tabú la aleatoriedad es, si no ignorada, por lo menos minimizada y es

empleada de manera muy restringida en el entendimiento de que la búsqueda ha de ser un proceso sistemático. Por supuesto, existe una variante llamada búsqueda tabú probabilística (probabilistic tabu search) en la que, a partir de un elemento x, se selecciona aleatoriamente el vecino que debe ser visitado a continuación.

En general el algoritmo de búsqueda tabú es:

PROCESS (TABÚ)

begin

x = selecc(X)

x* = x

H = ■

max = b(H,x)

calcular (NC(H,x))

elegir x' tal que b(H, x') = $max\{b(H,y)|y \in$ ■$NC(H,x)\}$

repeat

x = x'

if (b(H,x) > b(x*))

then

max = b(H,x)

x* = x

endif

actualizar(H)

actualizar(NC(H,x))

elegir x' tal que b(H, x') = $max\{b(H,y)|y \in \blacksquare NC(H,x)\}$

until (cond. termina)

return (x*)

end

3.2.2.2. Recocido simulado.

Al igual que el método descrito en la subsección anterior, el recocido simulado (simulated annealing) es heurístico, y está basado en una analogía con el proceso físico seguido para obtener sólidos con configuraciones de energía mínima. Fue propuesto por primera vez por Metropolis [10] y usado en optimización combinatoria por Kirkpatrick [8]. Probablemente se sepa que los sólidos cristalinos son los materiales más duros y que esto se debe a que su estructura molecular es muy simétrica, posee mínima energía. En Japón, desde hace milenios los maestros armeros elaboran los famosos sables de guerrero samurai siguiendo un proceso cuyos detalles se transmiten de generación en generación. Descrito superficialmente, este proceso consiste en calentar al blanco vivo una hoja de acero, doblarla sobre sí misma y golpearla con el martillo hasta que se integren las dos mitades y luego enfriarla sumergiéndola en agua. Estos pasos: calentar, doblar, golpear y enfriar se repiten decenas de veces y el resultado es una fina hoja de acero con el filo de una navaja de afeitar, pero con una dureza extraordinaria

que la hace difícil de mellar. Esto es porque el acero de la hoja, a través de los múltiples calentamientos y enfriamientos, adquiere una configuración molecular cuasicristalina, con una muy baja energía.

Este mismo proceso es el que se sigue en la física de materia condensada para obtener sólidos con estados de mínima energía. Se calienta el sólido hasta hacerlo líquido y luego se le enfría muy lentamente para que las partículas de éste se acomoden en su estado de mínima energía o en alguno cercano a éste. Si el enfriamiento no se hace cuidadosamente es más probable que no se alcance el estado de energía mínima. En este caso hay que volver a calentarlo (en general menos que la vez anterior) y volver a enfriarlo con lentitud. El recocido simulado se adapta naturalmente a problemas de minimización. Evidentemente, el problema puede modificarse para convertirlo en uno de maximización, pero aquí se supondrá que se pretende resolver problemas de minimización. En términos muy generales, la simulación del proceso de recocido en el algoritmo es:

1. Sea Ei la energía actual. Se desplaza un poco una molécula del estado presente. Con esto se cambia su posición respecto a las demás y se altera la energía del sistema que ahora será Ei+1 = Ei + d. El valor de d ■es calculado.

2. Si d■< 0 entonces el nuevo estado se acepta y es el nuevo estado actual. Si d ■■0 entonces el nuevo estado es aceptado con cierta probabilidad dada por: $\varrho^{\left(\frac{-\delta}{K_B T}\right)}$, donde T es la temperatura y KB una constante física, conocida como constante de Boltzmann.

3. Si conforme transcurre el tiempo se disminuye lentamente la temperatura, entonces el sólido alcanza un equilibrio térmico en cada temperatura y la probabilidad de estar en el estado caracterizado por tener energía Ei es:

$$\frac{1}{Z(T)} \varrho^{\left(\frac{-E_i}{K_B T}\right)} \tag{3.4}$$

donde Z(T) es la función de partición definida como:

$Z(T) = \sum_i \varrho^{\left(\frac{-E_i}{K_B T}\right)}$

De I.4.I es fácil ver que es mayor la probabilidad de tener un bajo nivel de energía (Ei pequeña, cercana a cero) y que esto es más difícil mientras menor sea la temperatura. De allí que sea importante disminuirla poco a poco. El algoritmo descrito con más detalle es mostrado a continuación. En éste, X es el dominio de búsqueda, T0 es la temperatura inicial, N(i) es el conjunto de vecinos de i y E(i) es la energía de la solución propuesta i.

PROCESS (RECOCIDO)

begin

x = selecc(X)

elegir T0

T=T0

repeat

for (contador = 0) to L do

elegir j Î■N(i)

d■= E(j)-E(i)

if ((d■< 0) OR (aleatorio[0,1) < exp(-d/T)))

then

i=j

endif

endfor

reducir (T)

until (cond. termina)

return (i)

end

Nótese que, a diferencia de la búsqueda tabú, el estado actual del proceso en el recocido simulado depende sólo de su estado inmediato anterior. Esto ha hecho posible que se haga un análisis detallado de este método utilizando cadenas de Markov y, de hecho, ha sido probado teóricamente que el algoritmo converge asintóticamente a la solución global óptima. El lector interesado en profundizar puede encontrar un análisis detallado en [3] y [11].

3.2.3. Comparaciones.

Los métodos de optimización presentados aquí son sólo una muestra de todos los existentes. A la pregunta, ¿cuál de ellos es el bueno?; en optimización, al igual que en muchas otras situaciones de la vida cotidiana, no existe "el método". La

conveniencia de cada uno depende del problema que se pretende resolver. Si éste involucra una función continua y derivable de una sola variable indepen-diente, probablemente los métodos que se enseñan en los cursos de cálculo serán suficientes. En cambio, si el problema involucra una función muy complicada de decenas de variables independientes, de la que quizás no se conoce la regla de correspondencia, entonces muy probablemente (si no se tiene cierta predisposición al sufrimiento) se elija uno de los métodos heurísticos. Entre estos dos extremos hay una amplia variedad de problemas y métodos que se pueden utilizar. Hay mucho de donde escoger. La elección debe ir en proporción al problema. Coloquialmente, "no debe usarse un cañón para matar pulgas", pero tampoco se deben "cortar olmos con machete". Hay, eso sí, métodos aplicables a casi cualquier problema, como los heurísticos, entre los que se encuentran los algoritmos genéticos y otros métodos que exigen mucho del problema, como los basados en el cálculo diferencial. En contraste, el grado de certeza que se puede tener en los resultados va en proporción inversa a la aplicabilidad. Casi nunca se puede estar completamente seguro de que el resultado entregado por un algoritmo heurístico sea el máximo mínimo global o cuándo se debe detener el proceso de búsqueda.

En términos generales los métodos presentados aquí poseen las siguientes carac-terísticas:

· Newton: La función objetivo debe ser continua, dos veces derivable, con primera derivada continua y cuya segunda derivada no se anule.

· Búsqueda de Fibonacci: La función objetivo debe ser continua y suave sin saltos violentos).

· Ascenso de máxima pendiente: La función objetivo debe ser continua y derivable. Calcular el gradiente puede no ser fácil. El lector interesado en profundizar puede consultar [9] y [12]

· Simplex: La función a optimizar debe ser lineal así como sus restricciones. Al ejecutarlo en computadora la exactitud del resultado depende de la estabilidad numérica de la matriz asociada.El lector interesado en profundizar puede consultar [2]

· Métodos heurísticos: El problema general en estos métodos es determinar a priori los valores de los parámetros de operación del algoritmo, por ejemplo: cómo establecer la dependencia entre la vecindad y la historia en búsqueda tabú, cómo ir modificando con el tiempo la temperatura en recocido simulado o qué valor asignarle a la probabilidad de mutación en un algoritmo genético. El lector interesado en profundizar puede consultar [5]. La intención de esta sección ha sido presentar un panorama general de las alternativas a los algoritmos genéticos que existen en el área de la optimización, en el entendimiento de que no existe el mejor método de optimización en general.

Ante un problema se debe decidir, con base en sus características, cuál método será el más adecuado. No es válido tratar de resolver todos los problemas de una sola manera.

En el otro campo de aplicabilidad de los algoritmos genéticos, la inteligencia artificial, concretamente en el aprendizaje de máquina, ocurre algo similar. No todos los problemas se pueden resolver de la mejor manera con un sistema experto, con una red neuronal o con un algoritmo genético. Seguramente sería más fácil resolver algunos problemas si se utilizara alguna herramienta no usada hasta ahora, o mejor aún, una combinación de herramientas.

CAPÍTULO 4

FUNDAMENTOS MATEMÁTICOS

Discusión de la utilidad del aporte del autor de este artículo para los fundamentos matemáticos de la investigación:

Este capítulo se presenta para que el lector tenga las bases necesarias para la comprensión de la fundamentación matemática que utilizaron los autores, para proponer la estructura funcional del sensor, el cluster y el genoma, además, el diseño experimental utilizado. Se presenta el teorema del esquema, dada la importancia de éste modelo matemático propuesto por Holland, que ha dado un rigor científico al uso de los AGs como una técnica metaheurística.

Los conceptos del álgebra, probabilidad y estadística permiten argumentar el diseño experimental utilizado en la investigación y la geometría y la topología son las bases del diseño de la estructura funcional del sensor.

La sección de optimización se presenta debido a la semejanza con los algoritmos genéticos, en el sentido de que en la primera se cuenta con una área básica factible que contiene la solución óptima; y en la segunda se cuenta con una población que contiene una solución óptima a través de los individuos de la población.

4.1. TERMINOLOGÍA

4.1.1. Función de adaptación

Los algoritmos genéticos son mecanismos de búsqueda que operan sobre un conjunto de códigos posiblemente muy grande, pero finito.

Del dominio de búsqueda se toman iteradamente conjuntos de muestras y cada elemento de una muestra es calificado dependiendo de qué tan bien cumpla con los requisitos de aquello que es buscado. Luego son seleccionados los elementos, a los que también llamaremos individuos, que cumplan mejor con dichos requisitos. Éstos se multiplican en las siguientes muestras y son sometidos a ciertos operadores que emulan a los que funcionan en la naturaleza.

Uno de los elementos esenciales involucrados en este proceso es la calificación asignada a cada individuo, a la que también llamaremos grado de adaptación a lo largo del texto. Esta calificación es un número mayor o igual a cero, y en conjunto con la selección se convierte en una medida de la posibilidad de reproducción para cada individuo. Esto es, existe una función de adaptación que asocia a cada individuo de la muestra (población) con un número real no negativo. Mientras más grande sea el valor asignado a un individuo dado mayor será la probabilidad de éste de ser seleccionado para formar parte de la siguiente generación de muestras.

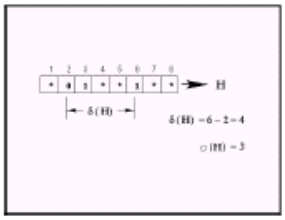


Figura 4.1: Orden y longitud de definición de un esquema.

4.1.2. Esquemas.

Se ha creado una notación para los conjuntos de cadenas binarias (cromosomas) mediante cadenas compuestas de tres símbolos, a saber: {0,1,*}. Considérese un conjunto de cadenas que poseen valores comunes en ciertas posiciones. Para construir la cadena que denota este conjunto, basta colocar en las posiciones donde las cadenas coinciden el valor explícito que tienen, y en las posiciones donde los valores no coinciden se coloca un *. De esta manera por ejemplo, 1*0* denota el conjunto {1000,1001,1100,1101}. A las cadenas compuestas de 1, 0 y * se les denomina esquemas. El número de posiciones con valor explícito de un esquema H se denomina orden del esquema y se denota por O(H). La distancia entre la primera y la última posición explícita se denomina longitud de definición (defining length) y se denota como d(H). En la figura 4.1.1 se ilustran esos conceptos.

4.2. EL TEOREMA DEL ESQUEMA[1]

Hasta la fecha, el modelo matemático más usual para describir el comportamiento de los algoritmos genéticos está basado en el teorema del esquema. Éste fue planteado originalmente por Holland [7] para el AGS, y se caracteriza por utilizar selección proporcional y cruzamiento de un punto, entre otras cosas. En esta sección se deducirá el teorema procurando ser un poco más general que en otras presentaciones, pues se considerarán tipos de cruzamiento distintos del tradicional como las de (6) u (8).

4.2.1. Selección

Supóngase que en la población de tamaño N de generación t de un algoritmo genético existen k representantes del esquema H y que f1, f2, ..., fk son los valores de la función de adaptación para los k representantes. Si se selecciona aleatoriamente un miembro de la población usando el mecanismo de la ruleta, la probabilidad de que éste sea el i-ésimo representante del esquema H es:

$p_i = \frac{f_i}{\sum_{j=1}^{N} f_j}$

Entonces, la probabilidad de seleccionar algún representante del esquema H es:

$$p_h = \frac{f_i}{\sum_{j=1}^{N} f_j} + \frac{f_2}{\sum_{j=1}^{N} f_j} + ... + \frac{fk}{\sum_{j=1}^{N} f_j} = \frac{f_1 + f_2 + ... + f_k}{\sum_{j=1}^{N} f_j} \tag{4.1}$$

[1] http://cursos.itam.mx/akuri/PUBLICA.CNS/1999/Programaci%F3n%20Evolutiva%20y%20AGs.pdf

Sean f (H) el valor de adaptación promedio de los representantes del esquema H y m(H,t) el número de representantes del esquema en la población de generación t (i.e., m(H,t)=k). Por definición

$$\bar{f}(H) = \frac{f_1 + f_2 + ... + f_k}{m(H,t)}$$

de donde:

$$f_1 + f_2 + ... + f_k = \bar{f}(\bar{H})m(H.t)$$

sustituyendo en (3):

$$p_h = m(H.t)\frac{\bar{f}(H)}{\sum_{j=1}^{N} f_j} \tag{4.2}$$

esto denota la probabilidad de seleccionar un representante del esquema H en la población de generación t. Sea $\bar{\bar{f}}$ el valor promedio de adaptación de la población de generación t

$$\bar{f} = \frac{\sum_{j=1}^{N} f_j}{N}$$

entonces, si se seleccionan N individuos de la población de generación t para la generación t+1, el valor esperado de individuos seleccionados del esquema H es:

$$m(H, t + sel) = Nm(H,t)\frac{\bar{\bar{f}}(H)}{\sum_{j=1}^{N} f_j} = m(H,t)\frac{\bar{\bar{f}}(H)}{\bar{\bar{f}}}$$

Hay que reiterar que las expresiones presentadas son válidas sólo para el tipo de selección proporcional o de ruleta (no son ciertas en general). Suponiendo que cada selección es un evento independiente.

4.2.2. Cruzamiento

Supóngase ahora que se aplica algún tipo de cruzamiento con cierta probabilidad pc sobre los individuos previamente seleccionados. Considérese lo que ocurriría con aquellas cadenas pertenecientes a un cierto esquema H. Algunas de estas cadenas se cruzarían con otras de forma tal que la cadena resultante ya no sería representante del esquema H, es decir, el esquema se rompería. Otras no serían seleccionadas para cruzarse y simplemente pasarían intactas a la si-guiente generación y habría otras más que originalmente no eran representantes del esquema y que al cruzarse generarían cadenas de H. El valor esperado de cadenas representantes de H que han sido seleccionadas y a las que no se les aplica cruzamiento es:

$$(1 - p_c)m(H, t)\frac{\overline{f}(H)}{\overline{f}} \tag{4.3}$$

(4)

Sea la probabilidad de ruptura del esquema H bajo el tipo de cruzamiento que esté siendo utilizado.2 El valor esperado del número de cadenas representantes de H que fueron seleccionadas y permanecen en el esquema después de aplicárseles cruzamiento es:

$$p_c \left(m(H, t)\frac{\overline{f}(H)}{\overline{f}}(1 - p_c) \right) \tag{4.4}$$

A la expresión (5) habría que hacerle una pequeña corrección para considerar aquellas cadenas que originalmente estaban fuera del esquema y que después de

cruzarlas generaron representantes de él. Sea g el número de cadenas ganadas por el esquema H mediante el mecanismo descrito, reescribiendo (5) se tiene:

$$p_c \left(m(H,t)\frac{\overline{f}(H)}{\overline{f}}(1-p_c) + g \right) \tag{4.5}$$

(6)

Resumiendo 4 y 6, el valor esperado del número de representantes del esquema H tras haber efectuado selección y cruzamiento es:

$m(H, t + sel + cru) = (1 - p_c)m(H,t) + p_c \left(m(H,t)\frac{\overline{f}(H)}{\overline{f}}(1-p_c) + g \right)$

si excluimos g:

$m(H, t + sel + cru) \geq (1 - p_c)m(H,t)\frac{\overline{f}(H)}{\overline{f}} + p_c \left(m(H,t)\frac{\overline{f}(H)}{\overline{f}}(1-p_c) \right)$

factorizando:

$m(H, t + sel + cru) \geq m(H,t)\frac{\overline{f}(H)}{\overline{f}} \left[(1-p_c) + p_c(1-p_r) \right]$

es decir:

$$m(H, t + sel + cru) \geq m(H,t)\frac{\overline{f}(H)}{\overline{f}}(1 - p_c p_r) \tag{4.6}$$

4.2.3. Mutación

Por último, hay que considerar el efecto de las mutaciones. Supóngase que una mutación se aplica con probabilidad p_m y que tiene el efecto de invertir un bit (cambiar un 1 por un 0 ó viceversa). Para que una cadena representante del esquema H permanezca en él tras una mutación, debe ocurrir que ninguno de

los bits definidos del esquema sea invertido. Recuérdese que el número de bits definidos del esquema es O(H). La probabilidad de que un bit no sea invertido por una mutación es $1-p_m$, así que la probabilidad de que ninguno de los bits definidos sea invertido, suponiendo que el invertir cada bit es un evento independiente, es:

$$\mu(p_m) = (1 - p_m)^{o(H)} \tag{4.7}$$

Añadiendo 4.2.7 a la expresión 4.2.6 se tiene:

$$m(H, t+1) \geq m(H, t)\frac{\overline{f}(H)}{\overline{f}}(1 - p_c p_r)\,(1 - p_m)^{o(H)} \tag{4.8}$$

A esta expresión se le conoce como el teorema del esquema, y existen diversas versiones de éste, todas ellas equivalentes. Goldberg, por ejemplo, prefiere desarrollar la expresión 4.2.7 en serie de MacLaurin (o bien mediante el teorema del binomio) para simplificar, dado que, en principio, pm es un número muy cercano a cero:

μ'$(p_m) = \frac{d(1-p_m)^{o(H)}}{dp_m} = -o(H)(1 - p_m)^{o(H)-1}$

de donde: $\mu(0) = 1$ y μ'$(0) = -o(H)$

Por lo tanto:

$$\mu(p_m) = ((1 - p_m)^{o(H)}) \approx 1 - o(H)p_m \tag{4.9}$$

Añadiendo 4.2.9 a 4.2.6:

$m(H, t+1) \geq m(H, t)\frac{\overline{f}(H)}{\overline{f}}(1 - p_c p_r)\,(1 - o(H)p_m)$

$$= m(H,t)\frac{\bar{f}(H)}{\bar{f}}(1 - p_c p_r - o(H)p_m + o(H)p_m p_c p_r)$$

$$\geq m(H,t)\frac{\bar{f}(H)}{\bar{f}}(1 - p_c p_r - o(H)p_m)$$

Finalmente Goldberg escribe:

$$m(H,t+1) \geq m(H,t)\frac{\bar{f}(H)}{\bar{f}}(1 - p_c p_r - o(H)p_m) \tag{4.10}$$

En otras versiones se encuentra todo dividido por el tamaño de la población, de tal forma que todo queda expresado en términos de probabilidad:

$$p(H,t+1) \geq p(H,t)\frac{\bar{f}(H)}{\bar{f}}(1 - p_c p_r)(1 - p_m)^{o(H)} \tag{4.11}$$

o bien:

$$p(H,t+1) \geq p(H,t)\frac{\bar{f}(H)}{\bar{f}}(1 - p_c p_r - o(H)p_m) \tag{4.12}$$

4.2.4. Tipos de cruzamiento[2]

Como se mencionó, existen diversos tipos de cruzamiento. En las expresiones 4.2.8 y 4.2.12 aparece una cierta probabilidad genérica pr que depende del tipo de cruzamiento que se use y de las características del esquema que se analice. Los cruzamientos más utilizados son tres, a saber:

a) Cruzamiento de un punto. Se elige un punto de corte y se intercambian los segmentos análogos de las dos cadenas ([1],[6],[12]).

[2] www.cipcusco.org.pe/meca_elect/ algoritmos%20geneticos.pdf

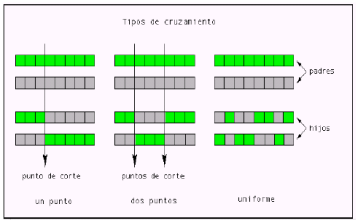


Figura 4.2: Tres tipos de cruzamiento

b) Cruzamiento de dos puntos. Se eligen dos puntos de corte y se intercambian los segmentos medios de ambas cadenas, se considera a los extremos de la cadena como sitios contiguos, i.e., como un anillo.([2],[12])

4.2.4.1. Cruzamiento de un punto.

Si las cadenas se cortan en un solo punto, un esquema se romperá. Cuando la cadena que lo representa es cortada en algún punto entre los bits fijos del esquema, es decir, si se corta en algún lugar de los abarcados por d(H).

En una cadena de longitud l existen l-1 posibles puntos de corte, así que la probabilidad de romper el esquema H con un corte es:

$p_r \leqslant \frac{\delta(H)}{l-1}$

El símbolo ■ ■se debe a que puede ocurrir que al cruzarse dos instancias de un esquema se generen, nuevamente, instancias de dicho esquema.

4.2.4.2. Cruzamiento de dos puntos

Existen exactamente maneras de elegir dos puntos de corte de una cadena de longitud l > 1 considerándola como unida en sus extremos. Supóngase que en esta cadena se encuentra un esquema de longitud definida d(H) y orden O(H). Si el orden del esquema es dos (o(H)=2), entonces H se rompe cuando uno de los puntos de corte cae dentro del segmento abarcado por d (H) y el otro cae fuera de ese segmento, en alguno de los l-d (H) lugares restantes. Es decir, la probabilidad de ruptura de un esquema de orden dos es:

$$P_{\delta(H),2} = \frac{\delta(H)(l-\delta(H))}{\binom{1}{2}} = \frac{2\delta(H)(l-\delta(H))}{l(l-1)} \tag{4.13}$$

En cambio, si se supone que el esquema tiene definidos todos los lugares abarcados por d (H), es decir, O(H)= d(H)+1, entonces el esquema también se rompe siempre que los dos puntos de corte caen dentro del segmento considerado por d (H), así que modificando 13 se obtiene:

4.2.4.3. Cruzamiento uniforme.

Sean A y B dos cadenas que se desean cruzar para formar una cadena C y una D. En este tipo de cruzamiento cada posición de bit de la cadena C es ordenada entre A y B. La que resulte elegida aportará su valor correspondiente a la misma posición para que sea colocado en C y la perdedora aportará su bit correspondiente a D. Supóngase que se desea preservar un esquema H del que es

representante alguna de las cadenas padre. Este esquema posee O(H) bits fijos. No importa a cuál de los hijos vaya a dar el primer bit fijo de H. Lo que sí importa es que los restantes O(H)-1 bits del esquema vayan a dar exactamente al mismo. Cada uno de estos bits tiene probabilidad 1/2 de quedar en la misma cadena que el primero, así que la probabilidad de que los O(H) bits del esquema sean copiados a la misma cadena resultante es: $\left(\frac{1}{2}\right)^{o(H)-1}$Por lo tanto, la probabilidad de romper el esquema H es: $P_{\delta(H),o(H)} = 1 - \left(\frac{1}{2}\right)^{o(H)-1}$

Esto en el peor de los casos, cuando los padres no coinciden en ninguna de las posiciones de interés para el esquema. En general:

$p_r \leqslant 1 - \left(\frac{1}{2}\right)^{o(H)-1}$

4.3. CRÍTICAS

Se han hecho diversas críticas al teorema del esquema [12], las más importantes son:

a) Sólo es una cota inferior, es decir, no es exacto.

b) No es muy útil para predecir a largo plazo el comportamiento de un algoritmo genético.

c) Sólo considera los efectos destructivos de los operadores genéticos y no los efectos constructivos.

d) Es muy particular. Está hecho para un AGS con selección proporcional (de ruleta), cruzamiento de un punto y probabilidad de mutación uniforme. Sin

embargo, el teorema del esquema ha sido durante años uno de los pocos intentos formales para modelar los algoritmos genéticos y ha servido de punto de partida para otros modelos más recientes. En la presentación hecha aquí se ha procurado ser más general, en la expresión que se presenta no aparece involucrado el tipo de cruzamiento.

4.4. ALGEBRA

4.4.1. El producto cartesiano

Si establecemos el producto cartesiano de un conjunto no vacío A consigo mismo, podemos observar que se tienen las siguientes parejas "curiosas" como se muestra en el siguiente ejemplo:

A = {a,b,c}

A X A = {(a,a),(a,b)(a,c),(b,a),(b,b),(b.c),(c,a),(c,b),(c,c)}

Observe que en el conjunto se encuentran parejas como estas: (a,a), (b,b), (c,c) que presentan la curiosidad que el primer elemento de la pareja es igual al segundo elemento. Es decir, cada elemento de la pareja es copia o reflejo del otro.

En este mismo conjunto se presenta otra curiosidad, cada vez que encontremos parejas de la forma (x, y); también encontramos otras parejas en el mismo conjunto que son de la forma (y, x). Así por ejemplo, se tienen las parejas (a, b) y (b, a); (a, c) y (c, a); (b, c) y (c, b).

Una tercera curiosidad que se presenta en este conjunto es la siguiente: cada

vez que se encuentren dos parejas como estas (x, y) e (y, z) en el conjunto, también se encuentra la pareja (x, z). Intuitivamente, da la impresión que el elemento y, común en ambas parejas y ubicado en estos sitios estratégicos sirve de "transmisor" entre x y z..

Teniendo en cuenta el anterior ejemplo, si establecemos relación entre los elementos de un mismo conjunto, esta relación puede tener alguna de las siguientes propiedades:

4.4.1.1. Propiedades de las relaciones

1. Una relación R definida en un conjunto A no vacío se llama reflexiva si para cada elemento x en A, se tiene que x R x, es decir la pareja (x, x) está en la relación R.

2. Una relación R definida en un conjunto A no vacío se llama simétrica si siempre que se tenga x R y también se tiene y R x, es decir si la pareja (x, y) está en la relación entonces la pareja (y, x) también está en la relación.

3. Una relación R definida en un conjunto A no vacío se llama transitiva si siempre que se tenga x R y e y R z también se tiene que x R z, es decir, si las parejas (x, y) e (y, z) están en la relación, entonces la pareja (x, z) también está en la relación.

4.4.1.2. Relación de equivalencia

Toda relación que es a la vez reflexiva, simétrica y transitiva, se llama de equivalencia.

· CLASES DE EQUIVALENCIA

Sea R una relación de equivalencia definida sobre un conjunto A, sea a en A. El conjunto formado por todos los elementos x en A que están relacionados con a se llama la clase de equivalencia de a y lo notamos [a], es decir:

[a] = { x E A l a R x}

Observe que [a] es un subconjunto de A.

· PROPIEDADES DE LAS CLASES DE EQUIVALENCIA

Sea R una relación de equivalencia definida sobre un conjunto A, entonces:

1. Para cada a de A C A.

2. [a] =[b] si y sólo si a R b.

3. Si [a] C [b] entonces [a] = [b]

4.4.1.3. Particiones

Una colección de subconjuntos A1, A2,... An de un conjunto A se dice que es una partición de A si:

i) Ninguno de los subconjuntos es vacío.

ii) Los subconjuntos son dos a dos disyuntos.

iii) La unión de todos ellos es A.

Toda relación de equivalencia definida sobre un conjunto induce una partición sobre el mismo.

4.4.1.4. Otras clases de relaciones

Si una relación cumple la siguiente propiedad: Siempre que tenga x R y e y R x es porque x = y. Esta propiedad que cumplen algunas relaciones se le conoce con el nombre de antisimétrica.

· RELACION DE ORDEN

Una relación R definida sobre un conjunto A se dice que define un orden total sí:

i) R es reflexiva

ii) R es antisimétrica

iii) R es transitiva

iv) Para todo x, y que pertenecen a A se debe tener que x R y ó y R x.

· RELACION DE ORDEN PARCIAL

Una relación R sobre un conjunto A se dice que establece un orden parcial sobre A si:

i) R es reflexiva

ii) R es antisimétrica

iii) R es transitiva

Pero no necesariamente R está definida para todos los elementos de A.

4.5. PROBABILIDADES Y ESTADÍSTICA

4.5.1. Eventos aleatorios y probabilidad

Un evento aleatorio es un evento el cual tiene una oportunidad de ocurrir, y la probabilidad es una medida numérica de esa oportunidad. La probabilidad es un número entre 0 y 1, ambos inclusive; los valores más altos indican una mayor oportunidad. Se escribe P(A) para denotar la probabilidad de que un evento A ocurra; P(A+B+...) para la probabilidad de que al menos uno de los eventos A, B,... ocurra; P(AB...) para la probabilidad de que todos los eventos A, B,...ocurran; y P(A|B) para denotar la probabilidad de que el evento A ocurra cuando se conoce que el evento B ocurrió, P(A|B) se llama probabilidad condicional de A dado B. Los axiomas más importantes que gobiernan la probabilidad son:

P(A+B+...) $\leq$ P(A)+P(B)+...

y P(AB) = P(A|B)*P(B).

Si solamente uno de los eventos A, B,... puede ocurrir, ellos se llaman excluyentes. Si al menos uno de los eventos A, B,... tiene que ocurrir ellos se llaman completos.

Una función de probabilidad es una función que asigna a cada suceso A en C (colección de todos los eventos) un número P(A), llamado probabilidad del suceso A, la cual cumple los axiomas siguientes:

i) P(A)$\geq$0 para todo A en C,

ii) P(∞)=1 donde ∞ denota al suceso seguro,

iii) Para toda sucesión numerable de sucesos disjuntos A1, A2, ..., An,...

$$P\left(\sum_{n=1}^{\infty} An\right) = \sum_{n=1}^{\infty} P(An)$$

4.5.2. Procesos estocásticos

La teoría de los procesos estocásticos (algunos autores utilizan las expresiones proceso afortunado o proceso aleatorio) se define generalmente como la parte dinámica de la teoría de las probabilidades, en la cual se estudia un conjunto de variables aleatorias (llamado proceso estocástico) desde el punto de vista de su interdependencia y su comportamiento límite. Se observa un proceso estocástico siempre que se examina un proceso que se desarrolla en el tiempo de manera controlada por leyes probabilísticas. Un ejemplo de proceso estocástico es el recorrido de una partícula en movimiento browniano (una partícula de aproximadamente una diezmilésima de diámetro sumergida en un líquido o gas presenta movimientos irregulares incesantes, este movimiento se conoce como browniano en honor al botánico ingles Robert Brown que descubrió el fenómeno en 1827).

Un proceso estocástico es una familia de variables aleatorias $\{X(t), t \in T\}$ clasificada mediante un parámetro t que varia en un conjunto índice T. No se define ninguna restricción sobre la naturaleza de T. Sin embargo, existen dos casos importantes:

- $T=\{+1,-1, +2,-2,...\}$ o $T=\{0,1,2,...\}$, en cuyo caso se dice que el proceso estocástico es un proceso de parámetro discreto.

- T={t: -$\infty \leq t \leq \infty$} o T={t: t≥0}, en cuyo caso se dice que el proceso estocástico es un proceso de parámetro continuo.

4.5.3. Generación de números aleatorios

El método congruencial se utiliza comúnmente como método para generar números aleatorios en la computadora. Existen tres variantes para el método congruencial, el multiplicativo, el aditivo y el mixto.

El método congruencial aditivo supone k valores iniciales, donde k es un entero positivo y calcula una secuencia de números para la relación congruencial siguiente:

$n_{i+1} = n_i + n_{i-k} (\mathrm{mod}\, m)$

El método congruencial multiplicativo genera una secuencia de números enteros no negativos cada uno menor que m por la relación:

$n_{i+1} = an_i (\mathrm{mod}\, m)$

El método mixto usa como expresión:

$n_{i+1} = an_i + c (\mathrm{mod}\, m)$

donde a, c y m son enteros no negativos.

4.5.4. Vectores

El conjunto de todas las n-uplas de números reales, denotado por Ân , se llama un n-espacio. En particular una n-upla en Ân,

U= (U1, U2,...,Un) , Ui∈ ℜ, i=1,...,n

se denomina punto o vector; los números reales Ui se llaman las componentes (o coordenadas) del vector U.

Es muy usual utilizar modelos de redes que utilizan vectores binarios, o sea, vectores en que todas las componentes toman sólo dos valores, generalmente en los conjuntos {0,1} o {+1,-1}. Por esta razón, se requiere frecuentemente convertir vectores de componentes no binarios a vectores binarios; seguidamente se presenta un ejemplo de esto. Supóngase que se desea clasificar un producto según los rasgos

COLOR={azúl, verde, negro, rojo}

TAMAÑO={pequeño, mediano, grande}

PESO={ligero, medio, pesado}

los datos sobre un producto específico forman el patrón de entrada a la red, el cual se define como un vector, por ejemplo el patrón P={verde, mediano, ligero}. Para convertir este vector a uno binario se puede proceder de la forma siguiente. Se define un vector de dimensión $N = \sum_{i=1}^{M} D_i$ donde M es la cantidad de rasgos y Di es la cantidad de elementos del dominio de cada rasgo; en el ejemplo N=4+3+3=10. Cada una de las N componentes se hace corresponder con un elemento del dominio de cada rasgo, esta componente tomara valor 0 ó 1 en dependencia de si el rasgo toma o no ese valor. El vector binario correspondiente al vector P es P'={0 1 0 0 0 1 0 1 0 0}.

Algunas operaciones sobre vectores son las siguientes.

El producto de un número real k por el vector U, denotado por kU, es el vector

que se obtiene multiplicando cada componente de U por el valor k:

kU=(kU1, kU2,...,kUn).

El producto interno o producto escalar de los vectores U y V, denotado por U*V, es el escalar que se obtiene multiplicando las componentes correspondientes de los vectores y sumando luego los productos resultantes:

U*V= U1*V1 + ... + Un*Vn.

Los vectores U y V se llaman ortogonales (o perpendiculares) si su producto interno es igual a 0, U*V=0.

La norma o longitud del vector U, denotada por ||U||, se define como la raíz cuadrada no negativa de :

||U||=$\sqrt[2]{U * U} = \sqrt{U_1^2 + ... + U_n^2}$

Si un vector V tiene norma 1 (||V||=1), o sea, si V*V=1, entonces se dice que V es un vector unitario o que está normalizado. Obsérvese que todo vector V distinto de 0 se puede normalizar tomando U= V/ ||V||.

El ángulo entre dos vectores no nulos U y V se define por $\cos\theta = \frac{U*V}{||U||||V||}$. Obsérvese que si U*V=0, entonces q=90, lo cual está de acuerdo con la definición de ortogonalidad.

4.5.4.1. Conjuntos ortonormales e Independencia lineal

Se dice que un conjunto {Ui} de vectores es ortogonal si sus elementos diferentes son ortogonales, o sea, si Ui*Uj=0 para i^{1}j. En particular, el conjunto {Ui}

se dice que es ortonormal si es ortogonal y cada Ui tiene norma igual a 1, o sea, si

$$\mathrm{U}_i * U_j = \partial_{ij} = \left\{ \begin{array}{l} 0 \text{ para i} \neq \text{j} \\ 1 \text{ para i=j} \end{array} \right\}$$

Siempre es posible obtener un conjunto ortonormal a partir de un conjunto ortogonal de vectores diferentes de 0 normalizando cada vector.

Los vectores U1,U2,...,Um en Ân se llaman linealmente dependientes si existen escalares k1,.., km, no todos nulos, tales que

k1U1 + k2U2 + ... +kmUm=0.

En caso contrario se llaman linealmente independientes.

Dados los vectores U1,...,Um, cualquier vector de la forma

a1U1 + a2U2 + ... + amUm

se llama combinación lineal de los vectores U1,...,Um.

Algunas observaciones de interés son:

- Todo conjunto ortonormal es linealmente independiente.

- Los vectores U1,...,Um distintos de 0 son linealmente dependientes, si y solo si, uno de ellos, por ejemplo Ui, es una combinación lineal de los vectores anteriores a él,

Ui= k1U1 + k2U2 + ... + ku-1Ui-1.

- El conjunto {U1,...,Um} se llama dependiente o independiente según sean los vectores U1,..., Um linealmente dependientes o independientes.

- Si dos de los vectores U1,...,Um son iguales entonces los vectores son linealmente dependientes.

- Dos vectores U1 y U2 son linealmente dependientes, si y solo si, uno de ellos es múltiplo del otro.

- Si el conjunto {U1,...,Um} es linealmente independiente entonces cualquier reordenamiento {Ui1,...,Uim} también lo es.

- Sea V un espacio vectorial de dimensión finita N, entonces cualquier conjunto de N+1 o más vectores es linealmente dependiente.

4.5.4.2. Distancia y Métrica

La distancia entre dos vectores U y V (la distancia entre los puntos que ellos representan) se denota por d(U,V) y se define por $d(U,V) = \sqrt[2]{(u_1 - v_1)^2 + ... + (u_n - v_n)^2}$.obsérvese que d(U,V)=||U-V||.

El concepto de distancia entre vectores es empleado en el campo de las redes neuronales para indicar una medida que es inversamente proporcional al grado de similaridad entre los patrones que ellos representan. Además de la distancia anterior, conocida como distancia euclidiana (L2 Distance), existen otras definiciones entre las que están:

Distancia de Hamming (L1 Distance):

Dh(U,V)= número de componentes diferentes entre U y V.

Distancia absoluta o Manhattan: $d_{L1}(U,V) = \sum_{i=1}^{N} |U_i - V_i|$

Distancia Producto interior (inner product): $d_{L2}(U,V) = \|U\|_{L2}^2 - 2U * V + \|V\|_{L2}^2$

Dirección del coseno: $d(U,V) = \cos\theta = \frac{U*V}{\|U\|\|V\|}$

Nótese que cuando se emplea el concepto de distancia como una medida de semejanza entre dos patrones y se usan expresiones como las antes dadas no se tienen en cuenta factores como que no todos los rasgos tienen igual importancia, ni que dos valores diferentes pueden ser equivalentes. Dos distancias que consideran esto son:

Distancia Euclidiana Pesada: $d(U.V) = \sqrt[2]{w_{1(x_1-y_1)^2+\ldots+w_n(x_n-y_n)^2}}$

Distancia de Mahalanobis: $d(U,V) = \sum_{i=1}^{N}\sum_{j=1}^{N} w_{ij}(u_i - v_i)(u_j - v_j)$, donde Wi y Wij representan pesos.

Sea X un conjunto no vacío. Una métrica sobre X es una función real D de pares ordenados de elementos de X la cual satisface las condiciones siguientes:

D(U,V) ^{3} 0.

D(U,V) = D(V,U).

D(U,V) = 0, si y solo si, U=V.

D(U,V) + D(V,Z) ^{3} D(U,Z).

La expresión que define una distancia se dice que es una métrica si satisface las condiciones anteriores.

4.5.4.3. Función de semejanza

El concepto de semejanza (analogía, parecido, etc.) desempeña un importante papel en las ciencias poco formalizadas y constituye la base metodológica para

toda un área de desarrollo científico conocida como Reconocimiento de Patrones.

La semejanza entre dos objetos puede representarse por medio de una función de distancia, ya que çercanía" "semejanza"son conceptos similares, sinónimos. Dos objetos "se parecen más mientras más cercanos se encuentren". Existe una variedad de modos para medir el grado de parecido o semejanza entre dos objetos que puede ir desde expresiones analíticas de distancia y pseudodistancias (las que no cumplen el axioma D(X,Y)=0 Þ X=Y) hasta la descripción de un procedimiento para obtener el grado de parecido entre dichos objetos. A continuación se analiza bajo que condiciones se puede hablar de semejanza entre objetos de un universo dado y se presenta la axiomática de Voronin para determinar si una expresión para el cálculo de la distancia entre dos objetos constituye una función de semejanza.

Dados dos objetos Oi y Oj de un conjunto (universo) U respecto a un conjunto de propiedades (rasgos) R={X1,X2,...,Xn}, donde Xi toma valores en el dominio Mi, i=1,..,n (no necesariamente métrico ni necesariamente cuadrado). En el universo U podrá hablarse de semejanza entre Oi y Oj si se cumplen las condiciones siguientes:

(1) El universo es finito o numerable.

(2) Todos los objetos se describen a partir de un conjunto finito de rasgos R que resultan razonables para describirlos.

(3) Cada objeto tiene el mismo grado de estudio (lo cual excluye la posibilidad de tener rasgos no conocidos).

(4) La comparación entre los objetos se puede realizar a partir de comparaciones por rasgos.

La axiomática de Voronin da las condiciones ideales que deben cumplir las funciones de semejanza. Para él, una medida de semejanza no necesariamente tiene que ser una función continua de UxU en Â (conjunto de números reales). Cuando la medida de semejanza se expresa en forma analítica tiene que cumplir la axiomática siguiente:

V1. Condición de acotamiento. $0 \leq \lambda_R(O_i, O_j) \leq 1$

V2. Condición de simetría. $\lambda_R(O_i, O_j) = \lambda_R(O_j, O_i)$

V3. Condición de máxima semejanza. $[\forall X_p \in R X_p(O_i) = X_p(O_j)] \iff \lambda(O_i, O_j) = 1$, donde Xp(Oi) denota el valor del p-ésimo rasgo en el objeto Oi.

V4. Condición de mínima semejanza. Se considera a cada Mi con un orden total, o sea, hay un primer elemento (x_i^0) y un último elemento (x_i^{Li}).

a)$\forall Xp \in R \left[X_p(O_i) = x_i^0 \qquad \text{y} \qquad X_p(O_j) = x_j^{Lj} \right] \Longrightarrow \lambda_R(O_i, O_j) = 0$

b)$\lambda_R(O_i, O_j) = 0 \Longrightarrow \exists X_p \in R \left[X_p(O_i) = x_i^0 \qquad y \qquad X_p(O_j) = x_j^{Lj} \right]$

4.5.4.4. Matrices

Un ordenamiento rectangular de la forma:

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}$$

donde las a_{ij} son valores escalares se llama matriz. La componente a_{ij} , llamada la componente ij, ocupa la i-ésima fila y la j-ésima columna. Una matriz con M filas y N columnas se llama matriz MxN. Una matriz con una fila se llama un vector fila y con una columna un vector columna.

La transpuesta de una matriz A, representada por , es la matriz que se obtiene de A cambiando las filas por las columnas:

$$\begin{bmatrix} a_{11} & \dots & a_{1n} \\ \dots & \dots & \dots \\ a_{m1} & \dots & a_{mn} \end{bmatrix}^t = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{mn} \\ \dots & \dots & \dots & \dots \\ a_{1n} & a_{2n} & \dots & a_{nm} \end{bmatrix}$$

Una matriz con el mismo número de filas y de columnas se llama matriz cuadrada. La diagonal (o diagonal principal) de una N- matriz cuadrada A consta de los elementos A11, A22,...,Ann. En particular, la N-matriz cuadrada con unos como elementos de la diagonal y el resto de los elementos 0 se denota por In, o simplemente I, y se denomina matriz unidad o identidad.

Toda matriz cuadrada de orden N tiene $\frac{(N^2-N)}{2}$elementos por encima de la diagonal e igual cantidad por debajo.

Una matriz simétrica es una matriz cuadrada A tal que .

La simetría es respecto a la diagonal principal, o sea, que una reflexión en la diagonal principal deja a la matriz sin cambio. Una matriz simétrica de orden N no tiene sus N elementos arbitrarios, sino que . El concepto de matriz simétrica se utiliza al estudiar las características de varios modelos de redes neuronales

artificiales.

Una matriz antisimétrica es una matriz cuadrada A tal que . Luego, los elementos de la diagonal principal son iguales a 0, , y el resto cumple que .

4.5.4.5. Vectores propios

Los vectores X ^{1} 0 que satisfagan la ecuación

AX= lX

donde A es una matriz dada de orden N y l es un escalar, se denominan vectores propios o característicos o auto vectores (eigenvectors) de la matriz A.

4.6. GEOMETRÍA

4.6.1. Planos en el espacio tridimensional

Sea N un vector no nulo dado y P0 un punto dado. El conjunto de todos los puntos P, tales que el vector PP0 es perpendicular a N se dice que es un plano que pasa por el punto P0, y el vector N se dice que es normal a dicho plano.

4.6.2. Líneas en el espacio tridimensional

Una línea en el espacio tridimensional se considera básicamente como el conjunto de puntos que forman la intersección de dos planos, es decir, una línea recta en el espacio tridimensional no es la gráfica de una ecuación sino la gráfica de una proposición de la forma:

a1X + b1Y + c1Z + d1 = 0 y a2X + b2Y + c2Z + d2 = 0.

4.6.3. Hiperplanos

El conjunto de elementos (puntos) en Ân que son soluciones de una ecuación lineal con n incógnitas X1,...,Xn de la forma c1X1 + c2X2 + ... + cnXn = b, con U=(c1,...,cn) diferente de 0 en Ân , se llama hiperplano en Ân.

Como ya se vio anteriormente Holland introdujo la noción de esquemas para formalizar la noción de bloques constructores. Tal vez la forma más fácil de visualizar un espacio de búsqueda es considerando todas las cadenas y esquemas de longitud 3.

Gráficamente, podemos ver el espacio de búsqueda como un cubo donde los puntos son esquemas de orden 2 y los planos son esquemas de orden 1 (ver figura 12.1). Todo el espacio de búsqueda esta cubierto por un esquema de orden cero (***).

Este resultado puede generalizarse a espacios de L dimensiones, donde hablaríamos de hiperplanos. De tal manera, podemos pensar que un AG corta a través de diferentes hiperplanos en busca de los mejores bloques constructores.

A continuación se muestran los hiperplanos en dos y tres dimensiones respectivamente.

Otra representación para un hiperplano es la siguiente. Sea A un vector fila n-dimensional diferente de 0, y sea c un número real. El conjunto

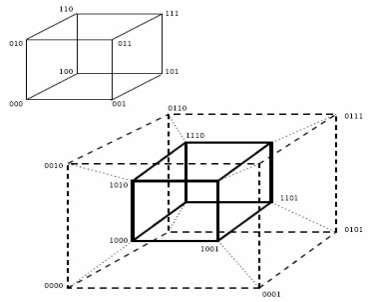


Figura 4.3: Hiperplanos de dos y tres dimensiones

H={ x Î : Ax=c}

es un hiperplano en .

4.6.4. Hipercubos

Sea el intervalo cerrado {(X1,..Xk): Ai£Xi£Bi para i=1,..,k}, donde A1,...,Ak, y B1,...,Bk son números tal que Ai<Bi para i=1,..,k. Los números positivos B1-A1,...,Bk-Ak, son las longitudes de los bordes del intervalo. Si todos los números Bi-Ai son iguales, el intervalo cerrado se denomina cubo k-dimensional.

4.7. TOPOLOGÍA

4.7.1. Espacios

Por un elemento se entiende un objeto o entidad de alguna clase. Un conjunto es una colección o agregado de tales elementos, considerados juntos o como un todo. Los términos conjunto y espacio se utilizan frecuentemente de modo contrastante. Un conjunto es una colección amorfa de elementos, sin coherencia o

forma. Cuando se impone alguna clase de estructura algebraica o geométrica sobre un conjunto, de modo que sus elementos están organizados, entonces este se convierte en un espacio.

Un espacio métrico consiste de dos elementos: un conjunto no vacío X y una métrica d sobre X. Los elementos de X se denominan puntos del espacio métrico (X,d).

Un espacio lineal real se define como sigue. Sea L un conjunto no vacío, los elementos de L se denotan por x y sea a un escalar real. El sistema algebraico L definido por las operaciones y axiomas siguientes se llama espacio lineal real.

- a(x+y) = ax + ay

- (a+ß)x = ax + ßx

- (aß)x = a(ßx)

- 1*x = x.

Un espacio lineal normado es un espacio lineal sobre el cual se define una norma, o sea, una función la cual asigna a cada elemento x en el espacio un numero real ||x|| de modo que

- ||x||=0 Û x=0

- ||x+y|| £ ||x|| + ||y||

- ||ax|| = |a| ||x||.

En términos generales, un espacio lineal normado es simplemente un espacio lineal en el cual hay disponible una distancia desde un elemento arbitrario al

origen. Un espacio lineal normado es un espacio métrico con respecto a la métrica inducida definida por d(x,y)= ||x-y||.

Sea Ân un espacio lineal. Si x=(X1,...,Xn) es un elemento arbitrario de Ân, entonces es natural definir ||x||, la distancia desde el punto x al origen o la longitud del vector x, por:

$$\|x\| = \sqrt[2]{|x_1|^2 + ... + |x_n|} = \left(\sum_{i=1}^{N} |x_i|^2 \right)^{1/2}$$

Si se considera a Ân compuesto de funciones reales f definidas sobre {1, 2, ..., n}, entonces esta definición se convierte en

$$\|f\| = \left(\sum_{i=1}^{N} |f(i)|^2 \right)^{1/2}$$

Esto es llamada la norma euclidiana sobre Ân, y el espacio lineal real Ân normado de esta manera se denomina espacio euclidiano n- dimensional.

Un espacio topológico se define de la forma siguiente. Sea X un conjunto no vacío. Una clase T de subconjuntos de X se llama una topología sobre X si satisface las siguientes condiciones:

- la unión de toda clase de conjuntos en T es un conjunto en T,

- la intersección de toda clase finita de conjuntos en T es un conjunto en T.

Un espacio topológico consiste de dos objetos: un conjunto no vacío X y una topología T sobre X. Los conjuntos en la clase T son llamados los conjuntos abiertos del espacio topológico (X,T), y los elementos de X son llamados sus puntos. Los espacios métricos son los espacios topológicos más importantes.

4.7.2. Punto interior y punto frontera

Sea X un espacio topológico y A un subconjunto de X. El interior de A, denotado por Int(A), es la unión de todos los subconjuntos abiertos de A, y un punto en el interior de A se llama punto interior de A. Está claro que A es abierto, si y solo si, A=Int(a). También, un punto en A es un punto interior de A, si y solo si, este tiene una vecindad (conjunto abierto el cual contiene el punto) la cual está contenida en A. La frontera de A, que se indica por OA, es la intersección del complemento del interior de A con el complemento del interior del complemento de A, y un punto en la frontera de A se llama punto frontera de A.$OA = (intA)^c \cap$

$(intA^c)^c$.

4.7.3. Punto fijo

Sea (E,d) un espacio métrico, aÎE y T:E®E una aplicación de E en sí mismo. Se dice que X0=a es un punto fijo si T(a)=a. El método iterativo del punto fijo se expresa de la forma siguiente. Si la secuencia X1, X2,... generada por Xn+1=g(Xn) converge a algún punto μ, y g(x) es continua, entonces μ= lim Xn+1 = lim g(Xn) = g(lim Xn) = g(μ) para n®¥, o sea, μ=g(μ), entonces μ es unto fijo de g(x).

4.7.4. Semiespacios

Sea A un vector diferente de 0 en el espacio y sea c un número real. A cada hiperplano H={X: AX=c} corresponden un semiespacio cerrado (half space)

positivo y otro negativo

$H+ = \{X: AX \geq c\}$

$H- = \{X: AX \leq c\}$

y un semiespacio abierto positivo y otro negativo

$= \{X: AX > c\}$

$= \{X: AX < c\}$

4.7.5. Regiones y Poliedros

Se llama región T a la intersección de un número finito de semiespacios cerrados $AiX \leq Di$ (i=1,...,m), cuyos hiperplanos generadores, AiX=Di, no pasan todos por el origen.

Una región es un conjunto convexo y cerrado, ya que está definida por la intersección de conjuntos convexos cerrados.

4.8. OPTIMIZACIÓN

4.8.1. Extremos

El punto de máximo o mínimo de una función se llama también punto extremo de la misma. Para el caso de funciones de un argumento, si existe un entorno bilateral del punto X0 tal que para cualquier otro punto X^1X0 de este entorno se verifica la desigualdad $f(X) > f(X0)$, el punto X0 recibe el nombre de punto mínimo de la función y=f(X) y el valor f(X0) el de mínimo de dicha función. Análogamente,

si para cualquier punto X^1X1 de un entorno determinado del punto X1 se cumple la desigualdad f(X)<f(X1), X1 recibe el nombre de punto máximo. Si X0 es un punto extremo de la función f(x), se tiene que f'(X0)=0 (punto estacionario), o no existe f'(X0) (condiciones necesarias para la existencia de un extremo).

Para el caso de funciones de dos argumentos, se dice que una función f(x,y) tiene un máximo (o mínimo) f(a,b) en el punto P(a,b), si para todos los puntos P'(x,y) diferentes de P, de un entorno suficientemente pequeño del punto P, se cumple la desigualdad f(a,b)>f(x,y) (o f(a,b)<f(x,y)). Si la función z=f(x,y) toma un extremo cuando x=X0 e y=Y0, entonces cada derivada parcial de primer orden de z se anula o no existe para estos valores de los argumentos. Análogamente se determina el extremo de una función de tres o más variables.

Se distinguen dos clases de extremos para una función f, los extremos locales y los globales. Un punto X0 es un extremo local si el entorno se toma como los puntos que se encuentran a una distancia menor o igual que un valor $\mu>0$ del punto X0. Un punto X0 es un extremo global si el entorno comprende todo el dominio de la función f.

4.8.2. Estabilidad

Algunos conceptos básicos para analizar la estabilidad de los sistemas dinámicos son:

Sistema: El sistema que se considera en este capítulo, está definido por:

$$x = f(x, t) \tag{4.14}$$

Donde x es el vector de estado (vector n-dimensional) y f(x,t) es un vector n-dimensional, cuyos elementos son funciones de x1, x2, ..., y t. Se supone que el sistema de la ecuación 4.7.1 tiene una única solución que comienza en la condición inicial dada.

Se ha de indicar la solución de la ecuación 4.7.1 como F(t;x0,t0) donde x = x0 en t=t0 y t es el tiempo observado. Así,

F(t0;x0,t0) = x0.

Estado de equilibrio:

En el sistema de la ecuación 4.7.1 a un estado x_e donde: $f(x_e, t) = 0$

se le denomina estado de equilibrio del sistema. Si el sistema es lineal, invariable en el tiempo, es decir, si f(x,t) = Ax, hay sólo un estado de equilibrio si A es no singular y hay infinitos estados de equilibrio si A es singular. Para sistemas no lineales, puede haber uno o más estados de equilibrio.

Estabilidad asintótica:

Se dice que un estado de equilibrio del sistema de la Ec 4.7.1 es asintóticamente estable, si es estable en el sentido de Liapunov y si toda solución que comienza dentro de S(d) converge sin abandonar S(e), hacia al incrementar indefinidamente t.

En la práctica, la estabilidad asintótica es más importante que solamente la

estabilidad. Igualmente, desde el momento que la estabilidad asintótica es un concepto local, establecer la estabilidad asintótica simplemente no significa que el sistema funcionará en forma correcta. Generalmente se necesita algún conocimiento del tamaño de la mayor región de estabilidad asintótica. Esa región se denomina dominio de atracción. Es la parte del espacio de estado en la cual se originan las trayectorias asintóticamente estables. En otras palabras, toda trayectoria que se origina en el dominio de atracción es asintóticamente estable.

Estabilidad asintótica global:

Si la estabilidad asintótica es válida para todos los estados (todos los puntos en el espacio de estado) del cual se originan las trayectorias, se dice que el estado de equilibrio tiene estabilidad asintótica global. Es decir se dice que el estado de equilibrio del sistema dado por la Ec. 4.7.1 tiene estabilidad asintótica global si es estable y toda solución converge hacia al incrementar definidamente t. Es evidente que una condición necesaria para que haya estabilidad asintótica global, es que haya un solo estado de equilibrio en todo el espacio de estado.

Inestabilidad:

Se dice que un estado de equilibrio es inestable si para algún número real e>0, y cualquier número real d>0, por pequeño que sea, siempre hay un estado en S(d) tal que la trayectoria que comienza en este estado abandona S(e).

4.9. POSIBILIDAD MATEMÁTICA

Posibilidad y probabilidad son dos términos muy parecidos utilizados en algunos casos de manera indiferente, pero la probabilidad es la asignación de una medida estadística cuantificable, la posibilidad no. Por eso se dice que todo en la vida es posible, pero no todo es probable.

La posibilidad ha de contar con aquello que le es entregado y con las circunstancias que limitan sus posibilidades y su libertad.

El término posibilidad se refiere a lo que satisface a las condiciones generales impuestas a un orden de realidad o que satisface las condiciones generales de la experiencia.

4.9.1. La posibilidad desde la fenomenología de la demostración matemática

El propósito es develar algunos de los rasgos distintivos del pensamiento matemático que están ocultos bajo la mecánica aparente de la demostración matemática. Se discute, usando muchos ejemplos, que la descripción de demostración matemática que usualmente se da es verdadera, pero no es realista.

Son muchos rasgos del pensamiento matemático que son dejados de lado por la noción formal de demostración, se conocen desde hace mucho tiempo, pero pocas veces se han discutido.

Lo ideal del realismo, es tomado de la fenomenología de Edmund Husserl.

Hace muchos años, Husserl dio algunas de las reglas a seguir en una descripción realista. Vale la pena recordar algunas de estas reglas.

1. Una descripción realista debe develar rasgos ocultos. Los matemáticos no predican lo que practican. Se rehúsan a aceptar formalmente lo que hacen en su trabajo diario.

2. Fenómenos laterales que normalmente se mantienen relegados deben ser tratados con su debida importancia. El vocabulario de oficio de los matemáticos incluye palabras como 'entendimiento', 'profundidad', 'tipos de demostración', 'grados de claridad' y muchos otros. Una discusión rigurosa de los roles de esos términos debería ser parte de la filosofía de la demostración matemática.

3. El realismo fenomenológico exige que no se fabriquen excusas que puedan

conducir a desechar cualquier rasgo de la matemática poniéndole el rótulo de psicológico, sociológico o subjetivo.

4. Todas los presupuestos normativos deben ser erradicados. Con excesiva frecuencia, supuestas descripciones de la demostración matemática son peticiones escondidas de lo que el autor cree que una demostración matemática debería ser. Se impone una actitud estrictamente descriptiva, a pesar de sus dificultades y peligros. Puede conducir a descubrimientos desagradables: por ejemplo, uno podría llegar a darse cuenta que no hay rasgos comunes compartidos por todas las demostraciones matemáticas. O incluso, uno podría ser llevado a admitir que las contradicciones son parte de la realidad de la matemática, a la par con la verdad.

Demostración por verificación: ¿Qué es una 'verificación'? En el sentido ordinario, una 'verificación' es un argumento que establece la verdad de una afirmación por medio de una lista de todos los casos posibles. La verificación es uno de los varios tipos de demostración matemática. Un no-matemático podría creer que una demostración por verificación es la más convincente de todas las clases de demostración. Una lista completa y explícita de todos los casos posibles parece ser irrefutable. Sin embargo, a los matemáticos no les impresiona la mera irrefutabilidad. Aunque no niegan la validez de una demostración por verificación, rara vez quedan satisfechos con tales demostraciones. La irrefutabilidad no es uno de los criterios que usará un matemático al juzgar el valor de una prueba. El valor de una demostración depende más bien de si ésta puede o no ser convertida en una técnica de demostración, esto es, de si la demostración puede ser vista como una instancia de un tipo de demostración, que se pueda usar para demostrar otros teoremas.

Las demostraciones por verificación suelen ser poco extrapolables. Un ejemplo muy sencillo de un teorema matemático para el cual puede darse una demostración por verificación, pero generalmente se evita, es el teorema de las cajas. Este teorema afirma que siempre que uno tiene n+1 fichas para colocar en n cajas, entonces siempre quedarán por lo menos dos fichas en una misma caja. Esta afirmación es tan intuitiva que parece no necesitar demostración. Observe sin embargo que la evidencia de este teorema es de una clase particular. El teorema de las cajas es

la instancia más simple de una demostración de existencia. En otras palabras, no se da ningún método para determinar cuál de las n cajas terminará con por lo menos dos fichas.

Ante un observador escéptico (o un computador), que no quiere aceptar una demostración de existencia, el argumento intuitivo debe ser reemplazado por una verificación. Una verificación del teorema de las cajas puede ser dada, pero no resultará ni simple ni esclarecedora. Consistirá en un algoritmo que escriba la lista de todas las maneras de colocar n+1 fichas en n cajas, y luego verifique que para cada posicionamiento por lo menos una caja termina con dos fichas. Cualquier algoritmo que haga eso usará inducción, y sería el hazmerreír de los matemáticos.

¿Por qué es preferible la demostración de existencia a la demostración por verificación en el caso del teorema de las cajas? Porque la demostración de existencia brilla con la luz de un principio universal, que ninguna verificación puede emular.

Una revisión somera de la historia del teorema de las cajas confirma la superioridad de la demostración de existencia. La demostración de existencia del teorema de las cajas fue el punto de partida del descubrimiento de una lista de profundos teoremas en combinatoria, ahora llamados teoremas de tipo de Ramsey. La verificación de cualquier teorema de tipo de Ramsey, que consistiría de una lista de todos los casos, es posible, pero solo en principio. Cualquier verificación de ese estilo requeriría una velocidad de cálculo más allá del alcance de los

computadores más rápidos posibles.

Las demostraciones de teoremas de tipo de Ramsey son demostraciones de existencia, que en últimas están basadas en el teorema de las cajas. Todas estas demostraciones son no-constructivas; sin embargo, proveen la evidencia incontrovertible de la posibilidad de verificación. Así, las demostraciones de tipo Ramsey son un ejemplo de una posibilidad puesta en evidencia por una demostración de existencia, aunque tal posibilidad no pueda ser convertida en acto.

Proponemos por oposición que una versión rigurosa de la noción de posibilidad sea agregada al bagaje formal de la metamatemática. Uno no puede pretender ignorar la posibilidad, alegando que las posibilidades de un resultado matemático están ocultas tras los enunciados formales. Tampoco puede uno desechar la noción de posibilidad basado en que tal noción está más allá del alcance de la lógica del presente. Las leyes de la lógica no están esculpidas en piedra, eternas e inmutables.

Una visión realista del desarrollo de la matemática muestra que las razones para que un teorema valga se encuentran tan solo después de excavar muy a fondo y enfocar las posibilidades del teorema. El descubrimiento de esas razones ocultas es la obra del matemático. Una vez se encuentran esas razones, la escogencia de enunciados formales particulares para expresarlas es secundaria.

No parece descabellado concluir que la noción de posibilidad será fundamental en la futura discusión rigurosa de la naturaleza de la matemática. Aunque parezca inquietante esta perspectiva, hay otras nociones, igualmente fundamentales y pre-

ocupantes, que reclaman un tratamiento riguroso, y que los lógicos del presente aún no han admitido en su seno. Por ejemplo, las demostraciones matemáticas vienen en tipos distintos, que aún están por clasificar. La noción de entendimiento, usada en discusión informal pero desterrada de la presentación formal, tendrá que tomar su lugar bajo el sol; aún más, la lógica deberá ser modificada para acomodar grados de entendimiento.

Finalmente, la noción de evidencia tendrá que obtener una posición formal que la ponga delante de la noción tradicional de verdad. Los rasgos característicos de la evidencia matemática, muchos de ellos reacios al tratamiento con lenguaje formal, tendrán que ser adecuadamente descritos.

Posibilidad en lógica difusa: El Principio de Consistencia en la lógica difusa (Consistency Principle, Zadeh, 1978) dice:

– Lo que es POSIBLE puede NO ser PROBABLE.

– Lo que es IMPROBABLE necesita ser POSIBLE.

En otras palabras, El Grado de Posibilidad o de pertenencia de cada elemento de un conjunto difuso tiene que ser Mayor o Igual a su probabilidad.

CAPÍTULO 5

INSTRUMENTACIÓN Y SENSÓRICA[1]

Discusión de la utilidad del aporte del autor de este artículo para los fundamentos básicos de los sensores: La información de este capítulo complementa la discusión técnica abordada por los autores en la solución del problema. Las aplicaciones de la electrónica , presentes actualmente en diferentes aspectos de la vida cotidiana, no serían posibles sin los sensores. Sin la capacidad que éstos ofrecen de medir las magnitudes físicas para su conocimiento o control, muchos dispositivos electrónicos no serían más que simples curiosidades de laboratorio.

La utilización de los sensores contrasta con la escasa bibliografía que se encuentra sobre ellos, en particular desde la perspectiva de la Ingeniería Electrónica.

5.1. Sistemas de Medida[2]

Se denomina sistema a la combinación de dos o más elementos, subconjuntos y partes necesarias para realizar una o varias funciones. En los sistemas de medida, es la asignación objetiva y empírica de un número a una propiedad o cualidad de un objeto o evento, de tal forma que la describa. Es decir, el resultado de la medida

[1]ESTE CAPÍTULO CONTIENE FRAGMENTOS DEL LIBRO INSTRUMENTOS ELECTRÓNICOS BÁSICOS DE RAMON PALLAS. PRIMERA EDICIÓN.

[2]www.investigacion.frc.utn.edu.ar/sensores/Tutorial/TECNO1.pdf

debe ser independiente del observador (objetiva, basada en la experimentación (empírica), y de tal forma que exista una correspondencia entre las relaciones numéricas y las relaciones entre las propiedades descritas.

5.1.1. Elementos necesarios entre la magnitud medida y el controlador

» TRANSDUCTOR

• Modifica la naturaleza de la señal que proporciona el sensor para hacerla más fácilmente medible.En general se denomina transductor a todo dispositivo que convierte una señal de una forma física a una señal correspondiente, pero de otra forma física distinta. Es por lo tanto un dispositivo que convierte un tipo de energía en otro.

Dado que hay seis tipos de señales: mecánicas, térmicas, magnéticas, eléctricas, ópticas y moleculares (químicas), cualquier dispositivo que convierta una señal de un tipo en una señal de otro tipo debería considerarse un transductor, y la señal de salida podría ser cualquier forma física "útil". En la práctica se considera por antonomasia aquellos que ofrecen una señal de salida eléctrica.

» SENSOR

• Dispositivo que está en contacto con la variable que se mide.Un sensor es un dispositivo que a partir de la energía del medio donde se mide, da una señal de salida transducible que es un función de la variable de medida.

Sensor y transductor se consideran a veces como sinónimos, pero sensor sugiere un significado más extenso: la ampliación de los sentidos para adquirir un conocimiento de cantidades físicas que por su naturaleza y tamaño no pueden ser percibidas directamente por los sentidos. Transductor, en cambio sugiere que la señal de entrada y la de salida no deben de ser homogéneas.

» ACONDICIONAMIENTO DE SEÑALES

• Los acondicionadores de señales, adaptadores o amplificadores, en sentido amplio son los elementos del sistema de medida que ofrecen, a partir de la señal de salida de un sensor electrónico, una señal apta para ser registrada o que simplemente permita un procesamiento posterior mediante un equipo o instrumento estándar. Consiste generalmente en circuitos electrónicos que ofrecen, entre otras funciones, las siguientes: amplificación, filtrado, adaptación de impedancias, y modulación y demodulación.

5.2. Características estáticas

El comportamiento del sistema de medida viene condicionado por el sensor empleado. Es por eso importante describir las características de los sensores, Sucede que en la mayoría de los sistemas de medida, la variable de interés varía tan lentamente que basta conocer las características estáticas del sensor, estas describen la actuación del instrumento en régimen permanente.

Ahora bien, las características estáticas influyen también en el comportamien-

to dinámico del sensor, es decir, el comportamiento que se presenta cuando la magnitud medida varía a lo largo del tiempo. No obstante, se suele evitar su consideración conjunta por las dificultades matemáticas que entraña, y se procede a la distinción entre las características estáticas y las características dinámicas, estudiándose por separado.

• Rango. Conjunto de valores de la variable que puede medir el instrumento. Se especifica mediante el límite inferior y el superior.

Ejemplo: rango de termorresistencia para medir temperatura: 50-150^{o}K.

• Precisión. Define la máxima desviación entre la salida real obtenida del instrumento en determinadas condiciones y el valor teórico de dicha salida que correspondería, en idénticas condiciones, según el modelo ideal especificado como patrón.

• Exactitud.Corresponde al error máximo que puede existir en una medición.

• Sensibilidad. Relación que existe entre el incremento en la señal de salida del instrumento y el incremento correspondiente en la variable medida.

• Resolución. Se mide por la mínima diferencia entre dos valores próximos que le instrumento es capaz de medir.

Ejemplo: un medidor de nivel que proporciona una variación de 10 mV por metro de altura tiene una sensibilidad de 10 mV/m.

• Histéresis. Valor máximo de la diferencia entre las medidas de un mismo valor en sentido creciente y decreciente de la variable, es la no coincidencia de

carga y descarga del instrumento. Se suele expresar en forma porcentual sobre el alcance del instrumento.

• Linealidad. Mide en qué grado la característica entrada salida del instrumento se puede aproximar a una línea recta. Se suele expresar como el error máximo que se cometería al aproximar la función por una línea recta. Esta cualidad es muy deseable ya que implica una sensibilidad similar en todo el rango de medida.

• Repetibilidad: Característica que indica, la máxima desviación entre valores de salida, optenidos al medir varas veces, un mismo valor de entrada con el mismo instrumento y el idénticas condiciones ambientales.

• Zona muerta. Es el intervalo de diferentes valores de entrada, a los que el instrumento no responde.

5.3. Características dinámicas

La presencia de inercias (masas, inductancias,...), capacidades (eléctricas, térmicas, fluhídricas, etc.) y , en general de elementos que almacenen energía, hacen que la respuesta de un sensor a señales de entrada variable sea distinta a la que se presenta cuando las señales de entrada son constantes. La descripción del comportamiento del sensor se hace en este caso mediante las denominadas características dinámicas: error dinámico y velocidad de respuesta, principalmente.

• Error dinámico: Es la diferencia entre el valor indicado y el valor exacto de la variable de medida, haciendo nulo el error estático.

- Velocidad de respuesta: Indica la rapidez con que el sistema de medida responde a los cambios en la variable de entrada.

Para describir matemáticamente el comportamiento dinámico del sensor se supone que la salida y la entrada se relacionan según una ecuación diferencial lineal de coeficientes constantes y que, por lo tanto, se tiene un sistema lineal invariante en el tiempo. En estas condiciones, la relación entre la entrada y la salida del sensor puede expresarse de manera simple, en forma de cociente empleando la transformada de Laplace de ambas señales y la función de transferencia propia del sensor. Hay que recordar no obstante que ésta última da una relación general entre la salida y la entrada, pero no entre sus valores instantáneos.

Las características dinámicas de los sensores pueden estudiarse entonces para cada señal de entrada aplicada, agrupándolos de acuerdo a la función de transferencia que los describe. Normalmente no es necesario utilizar modelos de orden superior a dos.

• Tiempo de retardo. Es el tiempo trascurrido, desde la aplicación del escalón de entrada hasta que la salida alcanza el 10 % de su valor permanente.

• Tiempo de subida. Es el tiempo transcurrido desde que la salida alcanza el 10 % de su valor permanente hasta que toma por primera vez el 90 % de dicho valor.

•Tiempo de establecimiento. Es el tiempo transcurrido desde la aplicación de un escalón en la entrada, hasta que la respuesta alcanza el régimen permanente,

con una tolerancia del 1 %.

- Constante de tiempo. Es el tiempo empleado para que la salida alcance el 63 % de su valor de régimen permanente, cuando en la entrada se aplica un escalón.

CAPÍTULO 6

LA CLONACIÓN ARTIFICIAL

6.1. ANTECEDENTES

La Universidad Autónoma de Bucaramanga en su Facultad de Ingeniería Mecatrónica, ha venido desarrollando investigaciones en tecnologías de control inteligente utilizando herramientas como lógica difusa, redes neuronales y algoritmos genéticos fortaleciendo la Línea de investigación en AUTOMATIZACIÓN Y CONTROL del Grupo de Tecnologías de la Información. Los trabajos realizados hasta el momento han consistido en el uso y aplicación de Redes Neuronales en la implementación del desarrollo de la tesis titulada "Implementación de sensores inteligentes utilizando redes neuronales aplicados en procesos de refinación del petróleo".

Continuando con esta línea de investigación se va a realizar el Diseño y Desarrollo de Medios y Sistemas de Control por Clonación Artificial, que permita proseguir el desarrollo y la investigación en Algoritmos Genéticos y en especial en la Clonación Artificial en Ingeniería donde el Dr. Muñoz, director de tesis, es pionero.

Se tiene como referente también los trabajos realizados por el Instituto de Investigación y Desarrollo de Tecnologías Aplicadas de la Universidad de Pamplona y reflejado en el artículo "Tecnologías de control inteligente: redes neuronales, algoritmos genéticos y de clonación Artificial", de la revista Colombiana de Tecnologías de Avanzada, volumen 1, Año 2003.

Etimológicamente debemos entender por clones a los individuos genéticamente iguales, siendo el clon, por ende, el doble perfecto de un ser, toda vez que posee el mismo código genético de su progenitor.

En el campo de la ingeniería la clonación son procesos tendientes a crear réplicas de dispositivos, objetos duplicados, elementos pares que cumplan con funciones equivalentes.

6.2. PLANTEAMIENTO DEL PROBLEMA Y JUSTIFICACIÓN

Los procesos de automatización industrial utilizan dispositivos de alta precisión, los cuales son de un altísimo costo, además de su dificultad para conseguirlos en el mercado tecnológico nacional, es por esto que en parte, las empresas colombianas están supeditadas en su desarrollo tecnológico a la dependencia de los países productores de alta tecnología; especialmente en el campo de los sensores para procesos industrializados, en los cuales el país tiene altos intereses y busca aumentar el margen de productividad de sus recursos, y una mejor prestación de servicios, como por ejemplo en el campo de la explotación petrolífera.

Como consecuencia de lo anterior se hace indispensable incursionar en el campo del control inteligente para encontrar soluciones óptimas a las necesidades de las empresas nacionales. Actualmente en la Empresa Nacional de Petróleos ECOPETROL refinería de Barrancabermeja Santander, se tiene la necesidad de contar con un sistema de control en tiempo real, que permita reemplazar un analizador (sensor) de viscosidad por un sensor inteligente clonado a partir del dispositivo real.

¿El diseño y desarrollo de medios y sistemas de control por clonación artificial puede convertirse en la solución a problemas en donde intervienen dispositivos de control industrial?

La respuesta al problema planteado, se busca mediante la aplicación de la metodología de clonación artificial haciendo uso de la lógica difusa y los algorritmos genéticos, tomando como referencia las características del dispositivo real.

6.3. OBJETIVOS

6.3.1. Objetivo General.

Diseñar y desarrollar Algoritmos Genéticos para Clonación Artificial de un sensor de viscosidad.

6.3.2. Objetivos Específicos.

- Desarrollar los códigos de la estructura funcional del sensor.

· Diseñar e Implementar algoritmos genéticos para manipular el mapa genético del sensor.

· Calcular los criterios de semejanza entre el sensor real o físico y el clon.

6.4. HIPÓTESIS.

Esta investigación pretende comprobar algunos resultados sobre la siguiente hipótesis:

"El sensor clonado posee un desempeño similar al sensor real, y está basado en la clonación artificial de las funciones de entrada y salida del sensor de viscosidad patrón, utilizando técnicas de inteligencia artificial"

6.5. Metodología de la clonación[1]

El vertiginoso desarrollo de los sistemas y la instrumentación inteligente y de las telecomunicaciones a lo largo de los últimos años ha impulsado el desarrollo de aplicaciones con alta interacción con el mundo externo, que funcionan en ambientes heterogéneos y con autonomía en la realización de sus acciones. Este tipo de aplicaciones es de un orden de complejidad mucho mayor que el de las aplicaciones tradicionales. El paradigma de desarrollo de tecnologías que aplican la Genética y la Clonación Artificial en Ingeniería, surge como una alternativa para la construcción de dispositivos y medios de alta precisión que permitan dar

[1]Estractado del artículo "Tecnologías de control inteligente: Redes Neuronales, Algoritmos Genéticos y de Clonación Artificial". Revista Colombiana de Tecnología de Avanzada.V1, Año 20001. Vol 3. ISSN 1692-7257. Editor IIDTA.

cumplimiento a este tipo de exigencias, combinando tecnologías existentes como son la inteligencia artificial y los sistemas distribuidos.

En la automatización de procesos industriales se usan múltiples técnicas de inteligencia artificial tales como sistemas expertos, redes neuronales, agentes inteligentes, etc., y en muchos casos se integran elementos de computación evolutiva tales como mapeo genético, entre otras, con el fin de desarrollar sistemas inteligentes para el control de los procesos. El uso e integración de esas técnicas permiten obtener componentes modulares, flexibles, reproducibles y disponibles que permiten a ingenieros de control desarrollar y adaptar herramientas y medios de automatización que satisfagan los aspectos de modularidad y flexibilidad de los sistemas de control. Uno de los tipos de componentes de interés en la automatización de procesos es los sensores industriales debido que obtienen información en línea de procesos reales y la transfieren al resto de la estructura de los sistemas.

El uso del mapeo genético permite el diseño de equipos más rápidos para el secuenciamiento, y con el desarrollo computacional se logra la creación de las bases de datos para transmitir, almacenar, analizar y clonar dicha información.

La tecnología control inteligente se basa en la clonación de sensores industriales mediante el uso de redes neuronales, la lógica difuza, los algoritmos genéticos. La clonación de sensores significa la reproducción de los códigos funcionales de dispositivos reales y la evolución de sus características a través de procedimientos de reproducción, cruce, mutación e inversión.

La redes neuronales permiten desarrollar la estructura inteligente de los sensores con el fin de mostrar el flujo de información interna del sistema de medición tomando como referencia las características de los dispositivos reales; es decir, información adicional necesaria para la interpretación de las medidas de variables. El método de activación de pesos aleatorios es usado para entrenar los sensores y realizar el aprendizaje con el fin de obtener esa información adicional y suplirla al resto de los elementos del sistema. El mapeo genético permite la generación de códigos para el proceso de clonación. Los procedimientos de mutación, cruce, reproducción e inversión son usados para la generación evolutiva del sensor clonado, el cual desarrolla características tales como mayor precisión, flexibilidad, modularidad, etc. debido a que hacen mas elemental la información.

En el momento actual los Controladores con Lógica Fuzzy transitan por una etapa del desarrollo acelerado en su fundamentación teórica y sus aplicaciones prácticas. La estructura algorítmica de los mismos, que puede llamarse tradicional, contiene entre sus particularidades el hecho de que la función de membresía de los subconjuntos difusos de entradas y salidas es del tipo "aproximadamente igual a", lo cual implica que dicha función sea simétrica en cada valor lingüístico con el máximo en el centro del intervalo correspondiente. Las ventajas de tal definición consisten en la aplicación simplificada del procedimiento de la defuzzyficación según el Método del Centro de Gravedad, y en la posibilidad del ajuste fino de la salida de mando, aprovechando los bordes difusos y solapados de valores lingüís-

ticos.

Muchos problemas de optimización del mundo real se reducen a la consideración de un solo criterio, mientras que en otros problemas es necesario evaluar varios criterios. Estos últimos son conocidos como problemas multicriterios o multiobjetivos. En el primer caso, simplemente se busca la mejor solución que pueda asumir la función objetivo predefinida. En los problemas multiobjetivos la noción de optimización no es tan obvia, esto es debido a que vamos a obtener grupos de soluciones que cumplan muy bien algunos criterios, pero no así otros. Esto lleva a definir mecanismos para lograr que todos los criterios se cumplan lo mejor posible, a pesar de que muchos entre ellos sean conflictivos.

El problema de evaluación de Funciones Multiobjetivos (FMOs) ha sido objeto de muchas investigaciones. Las técnicas de optimización convencional, tales como los métodos basados en gradiente, y unos menos convencionales, tales como recocido simulado, son difíciles de extender a casos verdaderamente multiobjetivos, ya que estos no fueron diseñados para evaluar esto. Los Algoritmos Genéticos (AGs) han sido reconocidos como apropiados para la optimización multiobjetivos por emplear individuos, los cuales permiten buscar múltiples soluciones en paralelo, según cada uno de los objetivos que se persiga.

Los Algoritmos Genéticos son algoritmos de búsqueda basados en los mecanismos de selección natural, genética y evolutiva. Es ampliamente aceptado que la

evolución de la vida comienza en un proceso que opera en cromosomas (dispositivo orgánico para codificar las estructuras de seres vivientes).

La selección natural es la conexión entre los cromosomas y el funcionamiento de sus estructuras decodificadas. El proceso de selección natural lleva a que los cromosomas de las estructuras exitosamente codificadas se reproduzcan más frecuentemente que las otras. Además la reproducción y mutación hacen que los cromosomas de los hijos sean diferentes a la de sus padres, y el proceso de recombinación puede crear cromosomas absolutamente diversos en los hijos a partir de los cromosomas de sus padres. Estas características de la evolución natural inspiraron el desarrollo de Algoritmos Genéticos. En línea general, a través de un mecanismo de codificación apropiado los Algoritmos Genéticos manipulan las cadenas de dígitos binarios (unos y ceros) llamados cromosomas, los cuales repre-sentan múltiples puntos en el espacio de búsqueda. Cada bit en la cadena se llama gen. Como en la naturaleza, los Algoritmos Genéticos solucionan el problema de encontrar los cromosomas buenos, mediante la manipulación del material oculto en los cromosomas, sin conocimiento del tipo de pro-blema que se esta resol-viendo. La única información que se da es la evaluación que cada cromosoma produce, esta evaluación se utiliza para predisponer la selección de cromosomas de modo que los cromosomas con las mejores evaluaciones tiendan a reproducirse más a menudo que aquellos con malas evaluaciones. Los Algoritmos Genéticos, usan manipulaciones simples de los cromosomas tales como codificaciones sencillas y

mecanismos de reproducción, y pueden mostrar comportamientos complicados y solucionar algunos problemas extremadamente difíciles sin el conocimiento del mundo decodificado.

Una descripción a alto nivel de los Algoritmos Genéticos es:

Dado un método de codificación de soluciones de un problema, en la forma de cromosomas y dada una función de evaluación que retorna una medida del valor de costo de cualquier cromosoma en el contexto del problema, un Algoritmo Genético consiste de los siguientes pasos:

Paso 1: Inicializar la población de cromosomas.

Paso 2: Evaluar cada cromosoma en la población.

Paso 3: Crear nuevos cromosomas por cruce entre los actuales cromosomas; aplicar mutación y recombinación.

Paso 4: Borrar los miembros de la población para permitir el ingreso de los nuevos cromosomas.

Paso 5: Evaluar los nuevos cromosomas e insertarlos en la población.

Paso 6: Si el criterio de finalización se cumple entonces parar y retornar el mejor cromosoma; de lo contrario ir al Paso 3.

6.5.1. Metodología para la clonación del sensor

La metodología utilizada para la Clonación Artificial de Sensores mediante Mapeo Genético, consiste en un conjunto de medios y procedimientos basados

en herramientas de inteligencia artificial los cuales se utilizaran en conjunto con técnicas que permitan la obtención del propósito de esta investigación tales como:

6.5.1.1. Diseño de los enfoques de optimización multiobjetivos.

Muchos problemas de optimización del mundo real se reducen a la consideración de un solo criterio, mientras que en otros problemas es necesario evaluar varios criterios. Estos últimos son conocidos como problemas multicriterios o multiobjetivos. En el primer caso, simplemente se busca la mejor solución que pueda asumir la función objetivo predefinida. En los problemas multiobjetivos la noción de optimización no es tan obvia, esto es debido a que vamos a obtener grupos de soluciones que cumplan muy bien algunos criterios, pero no así otros, por lo cual debe existir un mecanismo para lograr que todos los criterios se cumplan lo mejor posible, creándose así la necesidad de alcanzar un compromiso. Es decir, lo que se busca es poder evaluar eficientemente cada uno de los objetivos, de tal manera que la solución final sea la que mejor integre ese conjunto de objetivos (muchos de ellos conflictivos).

El problema de evaluación de funciones multiobjetivos ha sido objeto de muchas investigaciones. Las técnicas de optimización convencional, tales como los métodos basados en gradiente, y unos menos convencionales, tales como recocido simulado, son difíciles de extender a casos verdaderamente multiobjetivos, ya que estos no fueron diseñados para evaluar esto. Los AGs han sido reconocidos

como apropiados para la optimización multiobjetivos por emplear individuos, los cuales permiten buscar múltiples soluciones en paralelo, según cada uno de los objetivos que se persiga.

En este trabajo, se agrupan los enfoques multiobjetivos basados en AG's en dos grupos, los inspirados en la teoría de Pareto y los basados en la manipulación de la población como subpoblación. A continuación se presentan cada uno de los enfoques.

6.5.1.2. Enfoques basados en la manipulación de la población[2]

En este caso existen cinco enfoques, algunos de los cuales están basados en la división de la población, y otros en la evaluación de cada uno de los objetivos en la población total. A continuación se describen cada uno de ellos.

- Enfoque 1:

Se divide la población según el número de objetivos que contenga la función objetivo. Así, para un problema con n objetivos, n subpoblaciones de tamaño N/n pueden ser generadas, asumiendo un tamaño de población N. Luego, se pone a iterar un AG en cada subpoblación con una función objetivo diferente y se seleccionan los mejores individuos, esto para asegurar que cada función objetivo es evaluada. Después, se asigna prioridad (clasificación) a las funciones objetivo, esto dependiendo del problema a resolver. Finalmente, se va seleccionando cada función

[2]Estos enfoques se referencian por el Dr. Antonio Faustino Muñoz en el artículo: "Tecnologías de control inteligente: Redes Neuronales, Algoritmos Genéticos y de Clonación Artificial". Revista Colombiana de Tecnología de Avanzada.V1, Año 20001. Vol 3. ISSN 1692-7257. Editor IIDTA.

según su prioridad y se evalúa sobre cada subpoblación, esto se realiza hasta evaluar todas las funciones objetivo. Para asegurar la diversidad, se va haciendo un reemplazo parcial (p.e. X %) de los peores individuos en cada subpoblación.

- Enfoque 2:

Se divide la población según el número de objetivos que contenga la función objetivo. Así, para un problema con n objetivos , n subpoblaciones de tamaño N/n pueden ser generadas, asumiendo un tamaño de población N. Luego se pone a iterar un AG en cada subpoblación y se seleccionan los mejores individuos. Como resultado se obtienen las soluciones parciales S1,S2,S3,...,Sn. La unión de dichas soluciones permitirán conformar una nueva población global, a la cual se le aplica una función objetivo seleccionada aleatoriamente, así, hasta un cierto número de iteraciones (fijado como criterio de convergencia) para asegurar que cada función objetivo se evalúe dentro de la población global con un elevado grado de confianza. Para asegurar la diversidad, se va haciendo un reemplazo parcial (p.e. X %) de los peores individuos en cada subpoblación.

- Enfoque 3:

Se generan subpoblaciones y se evalúan como en los casos anteriores. Después, se seleccionan los mejores individuos de cada subpoblación. Esta selección es de solamente los individuos que tengan el mínimo valor de la función objetivo, que se este evaluando en esa subpoblación. El número de individuos que se seleccionan en cada subpoblación, es tomado como información para definir el coeficiente que

ponderará a cada uno de los componentes de la función multiobjetivo. Finalmente, se genera la población global con la unión de cada una de las subpoblaciones y se evalúa usando la función multiobjetivo (FMO) ponderada según los valores previamente determinados.

· Enfoque 4:

Se considera desde un principio toda la población (no hay subdivisión). Se selecciona aleatoriamente la función objetivo a evaluar, asegurándose que todas las funciones objetivo sean evaluadas un número mínimo de veces (parámetro a definir). La selección de la función objetivo debe realizarse de dos formas:

1. Se escoge aleatoriamente según probabilidad que va disminuyendo.

2. Se escoge cíclicamente.

· Enfoque 5:

En este enfoque se aplican los AGs dos veces, primero para optimizar los pesos o coeficientes que ponderan los elementos de la función multiobjetivo, luego para optimizar el espacio de soluciones o población presentada por el problema a resolver usando la FMO previamente definida. Para calcular los coeficientes, la población inicial de la primera fase es evaluada con cada función objetivo y se determina el número de individuos que dan solución mínina, ese será el coeficiente de ponderación de esa función objetivo en la función multiobjetivo.

6.5.1.3. Enfoques basados en la teoría de pareto.

La optimización multiobjetivo (MO) busca optimizar un conjunto de criterios (vector de costos). A diferencia de la optimización de un único objetivo, la solución a este problema no es un único punto, sino una familia de puntos conocida como el Çonjunto Optimo de Pareto". Cada punto en esa superficie es óptimo en el sentido de que no pueden realizarse mejoras en un componente del vector de costo que no conduzca a la degradación en al menos uno de los componentes restantes.

La noción de optimalidad de Pareto es sólo un primer paso hacia la solución práctica de un problema multiobjetivos, el cual, usualmente envuelve la escogencia de una única solución compromiso del conjunto no dominado de acuerdo a alguna información de preferencia. Cada elemento en el Çonjunto Optimo de Paretoçonstituye una solución no-inferior para el problema multiobjetivos. Veamos las siguientes definiciones:

Concepto de Inferioridad:

Un vector u = (u1,...,un) se dice es inferior a v = (v1,...,vn), si v es parcialmente menor que u (v p $<$ u); es decir:

$\forall$i = 1,...,n , vi $\leq$ ui y $\exists$ i = 1,...,n : Vi $<$ Uj

Concepto de Superioridad (No-Inferioridad):

Un vector u = (u1,...,un) se dice es superior a v = (v1,....,vn) si v es inferior a u.

Para obtener soluciones no-inferiores se pueden usar los siguientes métodos estadísticos: Enfoque de Sumas Ponderadas, Método de Restricción y programación de objetivos. Otra manera puede ser usando AGs. Así, manteniendo una población de soluciones, los AGs pueden buscar muchas soluciones no-inferiores en paralelo. Esta característica hace a los AGs muy atractivos para resolver problemas de optimización multiobjetivos usando la teoría de Pareto.

En este caso, el enfoque permite definir el AG en dos fases. En la primera fase se clasifican los individuos en dominados o no dominados (No-inferiores), para cada una de las funciones objetivo que conforman la función multiobjetivo. En la segunda fase, se utilizan los individuos no dominados como los padres para la fase de reprod^{ooo}cción.

6.5.2. Proceso de Clonación del sensor

De acuerdo a cada enfoque se conciben cinco etapas que componen el proceso de clonación de sensores artificiales. Las etapas son:

Etapa 1: En esta etapa se seleccionan los dispositivos a clonar. Se divide la población según el número de objetivos dados en unidades operativas funcionales; el conjunto de unidades operativas es llamada función objetivo. Por ejemplo, para un numero de N dispositivos que constituyen la población y un número de n unidades operativas, se divide la población en subpoblaciones acorde con las unidades, cuyo tamaño es N/n. Luego, se itera con un algoritmo genético ca-

da subpoblación con una función objetivo diferente con el fin de seleccionar los mejores individuos; esto es, para asegurar que cada función objetivo sea evaluada. Después, se asigna prioridad (clasificación jerárquica) a las funciones objetivo dependiendo del problema a resolver. Finalmente, se selecciona cada función según su prioridad y se evalúa sobre cada subpoblación. Esto se realiza hasta evaluar todas las funciones objetivo. Para asegurar la diversidad, se reemplazan los peores individuos en cada subpoblación.

Etapa 2: En esta etapa se obtienen las soluciones parciales S1,S2,S3,...,Sn por cada unidad operativa. La unión de dichas soluciones permitirá conformar una nueva población global, a quien se aplica una función objetivo seleccionada aleatoriamente. Esto es un proceso repetitivo hasta un cierto número de iteraciones (fijado como criterio de convergencia) para asegurar que cada función objetivo se evalúe dentro de la población global con un elevado grado de confianza.

Etapa 3: En esta etapa, en cada subpoblación se seleccionan los individuos que tengan el mínimo valor de la función objetivo que se evalúa. El número de individuos que se seleccionan (por cada subpoblación) es tomado como información para definir el coeficiente que ponderará a cada uno de los componentes de la función multiobjetivo (el conjunto de unidades operativas diferentes). Finalmente, se genera la población global como la unión de cada una de las subpoblaciones y se evalúa usando la función multiobjetivo ponderada según los valores previamente determinados.

Etapa 4: En esta etapa se selecciona la función objetivo a evaluar, de entre las unidades operativas que componen la función multiobjetivo. Debe asegurarse que todas las funciones sean evaluadas un definido mínimo número de veces.

Etapa 5: En esta etapa se realiza un proceso de optimización de los pesos y espacios propios de las soluciones parciales obtenidas en la etapa 3 usando la función multiobjetivo resultante en la etapa 4. Luego, se determina el número de individuos que dan una solución mínima; es decir, que satisfacen el coeficiente de ponderación de las funciones objetivos con respecto a la función multiobjetivo. Esto representa el dispositivo clonado.

CAPÍTULO 7

SOLUCIÓN PROPUESTA

7.1. LA RED NEURONAL[1]

Se desarrolló un sensor virtual para predecir el índice de viscosidad, utilizando las variables de temperatura, presión, nivel y flujo, tomadas en diferentes partes del proceso de extracción de aceites lubricantes con Fenol, utilizando una red neuronal entrenada por el método de Activación de Pesos Aleatorios, técnica no iterativa mucho más rápida que otros métodos tradicionales. Esta investigación fué desarrollada por los ingenieros Harry José Paba Argote y Eudilson Nuñes Cossio, miembros del grupo de invetigación en clonación artificial; este trabajo da continuidad al aporte que hacen los autores de la investigación desarrollada con algoritmos genéticos.

La herramienta de software elaborada puede ser utilizada en el entrenamiento de cualquier sistema de entradas-salida, aplicando redes neuronales. Ayuda en la selección de variables, pruebas de linealidad, tratamiento de señal para filtrar ruido y eliminar datos falsos, entrenamiento-validación y pruebas por simulación fuera de línea y en línea con el proceso real. La medición del Error RMS y el Máximo Error encontrado, son usados como el parámetro de comparación que

[1]Tomado de la tesis de Maestría en Ciencias Computacionales de Pava, H. y Nuñez, E.

permite evaluar el desempeño del modelo obtenido.

El diseño del experimento en este trrabajo se realiza con los datos utilizados por los ingenieros Harry Pava y Eudilson Nuñez en el desarrollo con redes neuronales, a partir de las mismas bases de datos que contienen las lecturas de temperatura (t), presión (p), nivel (n) y flujo (f), se obtuvo la ecuación que modela la superficie de respuesta, dada en función de f(t,p,n,f), hallada después de aplicar los operadores genéticos mediante programación genética.

7.1.1. La red Neuronal con Activación de Pesos Aleatorios (Rawn)

La función de aproximación general puede ser obtenida por redes neuronales retroalimentadas consistentes de sólo una capa oculta y neuronas no lineales. La innovación está en que el entrenamiento de los pesos entre las entradas y la capa oculta no se requiere. Tomando estos pesos de activación como aleatorios, el problema resultante es lineal en parámetros y puede ser fácilmente resuelto por métodos ordinarios estándar de mínimos cuadrados. Aquí se demuestra que utilizando tales redes con pesos de activación aleatoria (RAWNs), se puede obtener un excelente mapeo, siempre y cuando los pesos de activación sean escogidos de tal forma de que la matriz de regresión no sea singular. Se pueden alcanzar mejoras más adelante por regularización y con la selección adecuada de la señal de excitación usada para el entrenamiento. Se halla que: (i) se obtienen errores mucho más pequeños comparado con las redes de backpropagation utilizando los

mismos grados de libertad, (ii) el mapeo depende sólo un poco de los valores reales de los pesos ocultos, a condición de que la red tenga suficientes neuronas y (iii) debido a que no se necesitan iteraciones, la velocidad de computación es incomparablemente más rápida que el backpropagation. Estas propiedades hacen que la red RAWN sea particularmente adecuada para aplicaciones de control.

Las redes neuronales estáticas pueden ser vistas como un mapeo no lineal conveniente del espacio de las variables independientes al espacio de las variables dependientes. Un tipo de red neuronal de alimentación hacia adelante particularmente útil, consiste de una capa oculta con función de activación sigmoidal, no lineal y una capa de salida con una solución de activación lineal.

Ha sido demostrado teóricamente por [Cybenko] que este tipo de red puede aproximar cualquier función continua a cualquier precisión siempre que existan suficientes neuronas. Los elementos de los pesos en las matrices pueden ser vistos como parámetros que deben ser encontrados para obtener un acercamiento aceptable a los datos disponibles. Este proceso es llamado "entrenamiento". El procedimiento usual para entrenar redes neuronales en adelanto, es utilizando backpropagation. A pesar de las mejoras tales como la aceleración, rata de aprendizaje adaptativo [Drago], y redes auxiliares [Fortuna], el backpropagation es lento y tiene propiedades de convergencia pobres.

Recientemente, un número de documentos ha enfocado el entrenamiento de redes neuronales continuas de alimentación hacia delante, como un problema de esti-

mación de parámetros [Singhal, Scalero]. Se han reportado mejoras significantes en velocidad y convergencia. También, están disponibles esquemas recursivos [Chen, Billings]. Cualquiera que sea el procedimiento, se necesita gran cantidad de datos fuera de línea para obtener errores residuales aceptables. Por ello, a pesar del progreso significativo, el entrenamiento de las redes neuronales sigue siendo el cuello de botella en las aplicaciones de control en tiempo real.

Una red neuronal del tipo descrito con entradas, neuronas en la capa oculta y salidas, tiene pesos. De este modo, para cualquier utilidad práctica, el número de parámetros llega a ser muy grande. Puesto que, además, la red es no lineal, la hiper-superficie del criterio de error como función de los parámetros puede tener una forma muy irregular y usualmente tiene muchos mínimos locales. A pesar del hecho de obtener un buen mapeo con cualquiera de las funciones de arriba, no hay garantía de encontrar el mínimo global. De hecho, es bastante probable que el mínimo obtenido sea un mínimo local; y así la red será usada solamente con mapeo en el rango de calibración, asegurándose que posea buenas propiedades de Interpolación local. La forma clásica para probar el desempeño es revisando la salida contra el segundo conjunto de datos independientes.

La novedad básicamente obtiene muy buenos mapeos tomando tan sólo los pesos aleatoriamente en la capa oculta. En tal caso, la estimación de los pesos en la capa de salida, se vuelve lineal en los parámetros y por lo tanto puede ser fácilmente resuelta por métodos ordinarios estándar de mínimos cuadrados. Una

red construida en esta forma puede ser llamada red con pesos de activación aleatorios (RAWN). Se demuestra que con algunas precauciones, las redes RAW pueden entrenarse por el método de mínimos cuadrados con excelente desempeño sin la necesidad de iteraciones. Esta propiedad es particularmente útil para aplicaciones de control.

7.1.1.1. Procedimiento de Entrenamiento

Dada una red neuronal de alimentación hacia adelante con solo una capa oculta, los elementos de las matrices de pesos son parámetros que deben ser encontrados con el fin de obtener un ajuste aceptable de los datos disponibles, este proceso es llamado entrenamiento. En los casos de la redes neuronales de alimentación hacia delante, el entrenamiento no es nada más que encontrar el mínimo local o global de las funciones objetivo.

Los métodos de entrenamiento más comúnmente usados, tales como el método de propagación hacia atrás o Levenberg–Marquardt, son métodos basados en gradientes, así estos métodos tratan de encontrar el mínimo descendiendo en la dirección del gradiente negativo [Hassoun]. Los métodos de entrenamiento basados en gradientes tienen dos desventajas principales, primero, la rutina de búsqueda puede quedarse estancada en el mínimo local y segundo, el entrenamiento basado en gradiente es un procedimiento iterativo posiblemente resultando en tiempos de entrenamientos largos antes de que se encuentre el mínimo. En esta sección

el esquema RAWN alternativo se desarrolla con el único objetivo de acelerar el proceso de entrenamiento, sin embargo, ambas aproximaciones pueden aun resultar en soluciones sub-óptimas. La aproximación al entrenamiento de la redes neuronales de pesos por activación reguladas se basa en la división del problema en dos subproblemas mutuamente independientes que pueden ser resueltos óptimamente [Braake 1996]. Sin embargo, no implica que el problema sea resuelto total y óptimamente aunque la aplicación del método propuesto en varios procesos reales haya mostrado que los resultados satisfactorios pueden ser obtenidos, el primer subproblema es la estimación de los pesos de activación , el segundo subproblema es la estimación de los pesos de salida . Como se mostrará de esta manera los subproblemas resultantes tienen parámetros lineales y la solución pueden ser encontrada por algoritmos no iterativos rápidos. Esquemáticamente el procedimiento puede ser dividido en los 2 siguientes pasos:

El cálculo de los pesos de activación.

El cálculo de los pesos de la salida.

Cada paso puede ser dividido en diferentes acciones y cálculos , en la figura se muestra un resumen de la aproximación total del RAWN. La primera parte del algoritmo tiene que ver con la estimación de los pesos de activación, esto se basa en la construcción de modelos lineales locales, pero para construir estos modelos lineales, el conjunto de datos primero necesita ser particionado en diferentes subconjuntos, los cuales serán utilizados para la estimación de un modelo lineal local

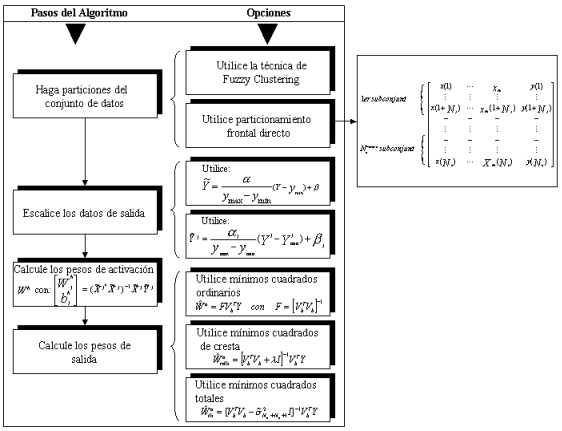


Figura 7.1: Resumen del algoritmo RAWN para entrenar un red neuronal, alimentada hacia adelante con una capa oculta

particular, para evitar la saturación de la función de activación, estos subconjuntos necesitan ser escalizados, este es el segundo paso del algoritmo, el tercer paso es, con los parámetros de estos modelos locales lineales que deben ser calculados por técnicas cuadráticas lineales. Los parámetros resultantes de los modelos lineales locales se utilizan como los pesos de activación de la red neuronal. Ahora que los pesos han sido calculados, el último paso del proceso de entrenamiento RAWN es la estimación de los pesos de salida por técnicas cuadráticas lineales estándar. Debe resaltarse en este punto que el algoritmo total no necesita iteraciones en principio a través del conjunto de datos. Cada uno de estos pasos mencionados de construcción deben ser ejecutados solo una vez.

7.1.2. Pruebas

7.1.2.1. Descripción de las pruebas efectuadas

Las pruebas propuestas buscan encontrar los parámetros óptimos para la red neuronal que garanticen el mejor desempeño del sensor virtual a implementar en la Unidad de Fenol, dadas las condiciones operacionales a que está sometido el proceso, principalmente por los cambios de carga (Liviano, Medio y Pesado).

Para ello se debe recopilar suficiente información que abarque en lo posible la mayor variación en las entradas y la salida, si las condiciones operacionales y de control del proceso lo permiten sin poner en riesgo el proceso y la calidad del producto a entregar (rafinato).

Esto puede tomar alrededor de 6 meses, recopilando archivos con tendencias históricas de las diferentes variables a utilizar en la que se incluyan, arrancadas y paradas de la planta, emergencias, condiciones normales de operación, puntos de máximo rendimiento y lógicamente los cambios de carga intrínsecos a esta operación en particular.

Posteriormente se deben reconciliar los datos capturados. Este proceso consiste en validar la consistencia de los datos que entrega la instrumentación, descartar e inclusive corregirlos para que la red neuronal aprenda adecuadamente de dicha información.

De esta forma se preparó un archivo de datos históricos con buena cantidad de información y buena calidad de la misma (case-27-sep-2000.mat) con 8640 datos,

suficientes para entrenar y validar el modelo, que incluye un cambio de carga con su correspondiente variación en la salida (índice de refracción).

Para centrar más el problema en la solución de los aspectos que matemáticamente no están establecidos, se toman como constantes los parámetros definidos a continuación, de esta forma la complejidad del problema se reduce y permite hallar parámetros óptimos para el desempeño de la red neuronal para el propósito establecido.

Función de Activación: Sigmoide. Según las conclusiones de [Braake] y [Otero] en sus publicaciones (comprobadas de igual forma en nuestros experimentos).

Método de Entrenamiento: RAWN. Para obtener rápidamente los parámetros de la arquitectura la red neuronal, permitiendo el análisis de muchos casos en corto tiempo y decidir el mejor comportamiento variando los parámetros restantes. Posteriormente se re-entrena con un método completo, con los parámetros óptimos encontrados por RAWN, lográndose resultados superiores.

Así, los demás parámetros, tales como, el número de neuronas en la capa oculta, el número de muestras requeridas para el entrenamiento y el número de pasos hacia atrás, se convierten en el objetivo para determinar la red ideal.

Por cada entrenamiento se evalúa el RMSE (Error RMS) complementado con el MAXE (Máximo Error: con los datos de validación), para determinar cuál de los experimentos tiene el mejor desempeño y tomar así sus parámetros como óptimos.

El método descrito puede ser ampliado y mejorado en el futuro dado que la

aplicación desarrollada permite experimentar libremente sobre nuevas teorías para obtener mejores resultados. El aporte computacional radica en la reducción del tiempo de ejecución necesario para el entrenamiento que requieren los algoritmos tradicionales en la obtención de los parámetros de la red neuronal (pesos y ajustes).

7.1.3. Resultados por modelos propuestos

7.1.3.1. Modelo por Regresión

Tomando como entradas las variables seleccionadas y entrenando por RAWN se obtuvieron los resultados que se muestran en la siguiente tabla:

NUM	$RMSE$	$MAXE$
1	0,0017439	0,0033676
3	0,003600	0,0031367
4	0,0021229	0,0038839
5	0,0015836	0,0035062
6	0,0016774	0,007212
7	0,0019241	0,0061608
8	0,0017042	0,008346
10	0,0013473	0,0058143
20	0,0018437	0,011341
30	0,0020754	0,031088
40	0,0058415	0,12686

NUM= Número de neuronas.

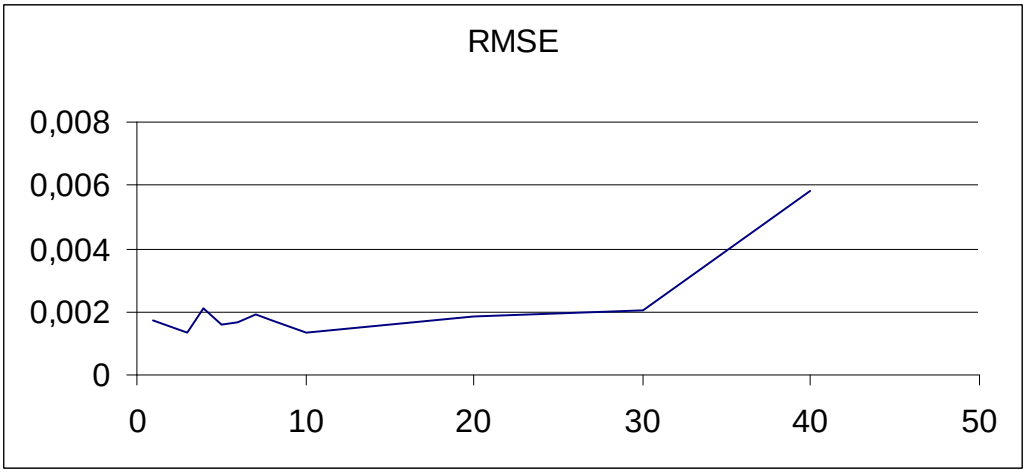


Figura 7.2: RMSE por Número de Neuronas con Regresión.

Resultados de los Experimentos con el Modelo por Regresión

De los resultaods anteriores se encuentra el caso óptimo en donde RMSE y MAXE tienen los picos más bajos simultáneamente es con 10 neuronas para la capa oculta.

7.1.3.2. Modelo por Correlación y Dispersión

Tomando como entradas las variables seleccionadas y entrenando por RAWN se obtuvieron los resultados que se muestran en la siguiente tabla:

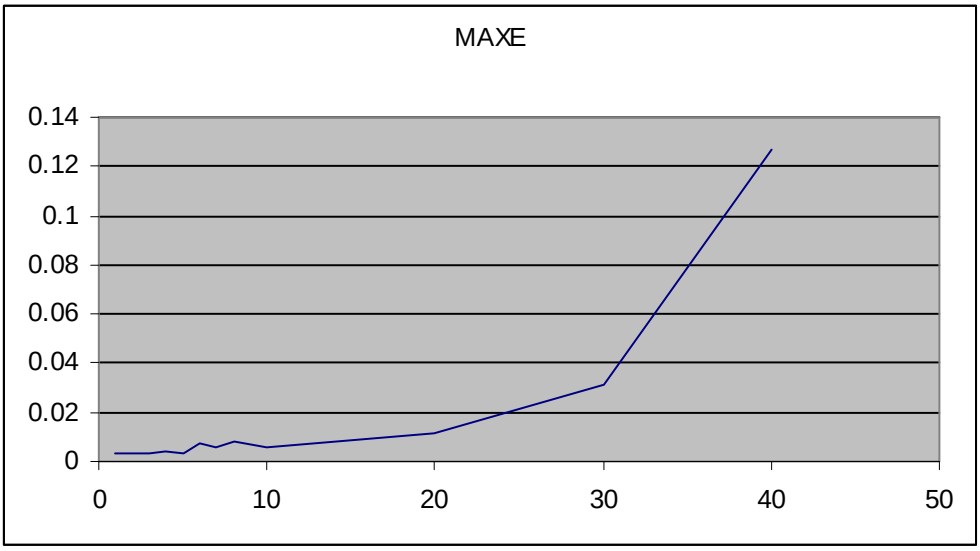


Figura 7.3: MAXE por Número de Neuronas con Regresión.

NUM	$RMSE$	$MAXE$
3	0,0010291	0,0027133
5	0,0011054	0,0024877
10	0,003029	0,012585
12	0,00096497	0,012741
15	0,0036674	0,0073696
20	0,00072544	0,0071936

Resultados de los Experimentos con el Modelo por Correlación y Dispersión.

De las figuras anteriores se encuentra el caso óptimo en donde RMSE y MAXE tienen los picos más bajos simultáneamente es con 5 neuronas para la capa oculta.

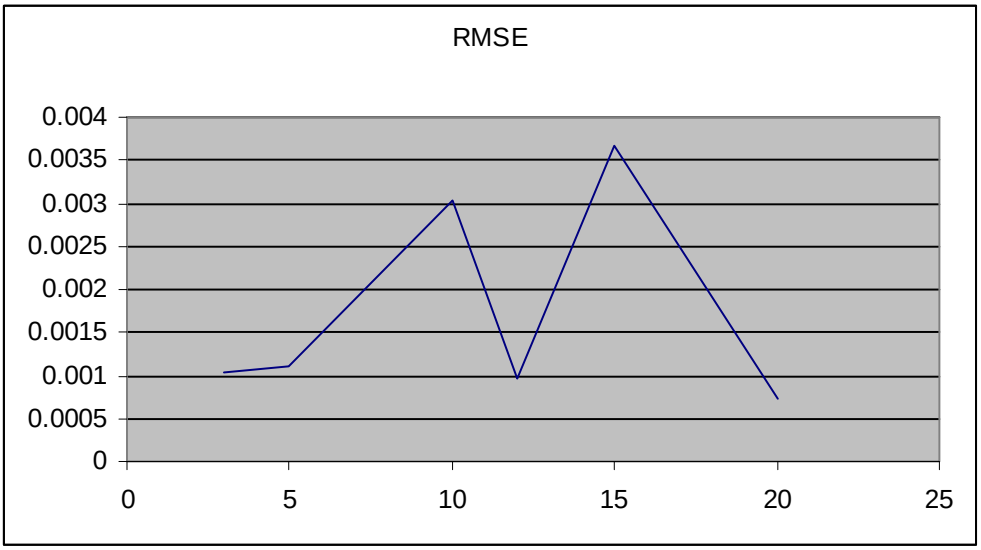


Figura 7.4: RMSE por Número de Neuronas con Correlación y Dispersión.

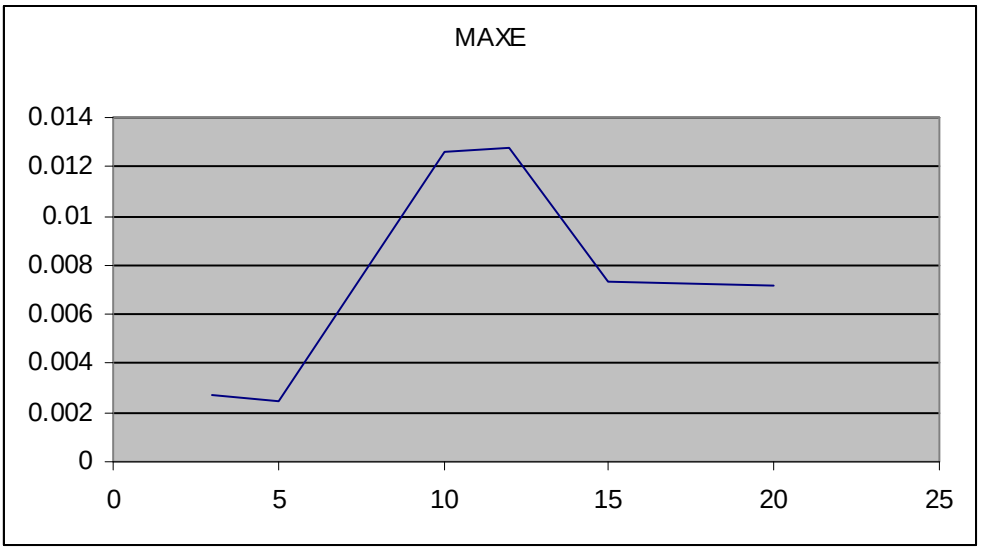


Figura 7.5: MAXE por Número de Neuronas con Correlación y Dispersión.

7.1.3.3. Modelo por el Experto

Tomando como entradas las variables seleccionadas y disponibles en la tabla 1.5 y entrenando por RAWN se obtuvieron los resultados que se muestran en la siguiente tabla:

NUM	$RMSE$	$MAXE$
5	0,0019238	0,003634
10	0,0017794	0,0037834
12	0,0016122	0,0035634
15	0,0021772	0,0071826
20	0,0035932	0,020569

Resultado de los experimentos con el modelo por el experto.

De las figuras anteriores se encuentra el caso óptimo en donde RMSE y MAXE tienen los picos más bajos simultáneamente es con 12 neuronas para la capa oculta.

7.1.3.4. Optimización Final de Parámetros de la Red.

El mejor modelo fue entregado por la red alimentada con datos de entrada seleccionados por correlación y dispersión. Sin embargo el modelo que mayor cantidad de neuronas en la capa oculta utiliza es el propuesto por el Experto, lo cual daría una mayor robustez con los otros casos de prueba.

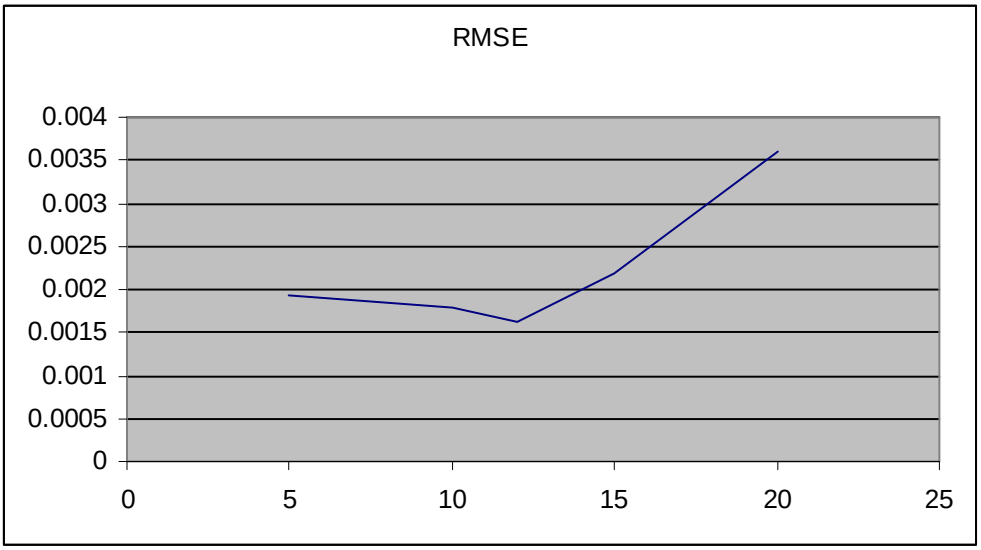


Figura 7.6: RMSE por Número de Neuronas por el Modelo del Experto.

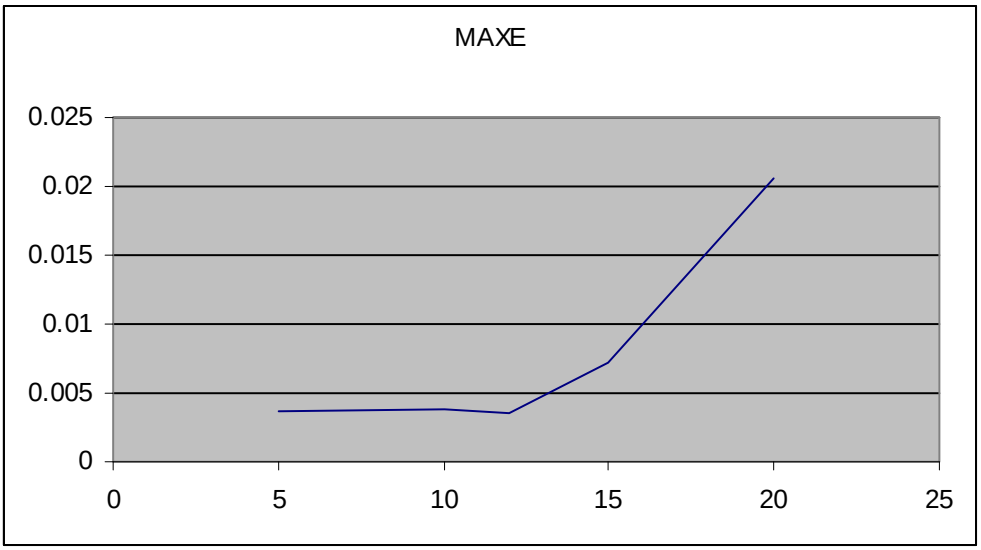


Figura 7.7: MAXE por Número de Neuronas por el modelo del Experto.

$Modelo$	$\#Neuronas$	$RMSE$	$MAXE$
$REGRESIÓN$	10	0,0013473	0,0058143
$CORRELACION - DISPERSION$	5	0,0011054	0,0024877
$EXPERTO$	12	0,0016122	0,0035634

Comparación de los Diferentes Modelos.

7.1.4. Analisis de resultados.

La técnica utilizada para analizar el comportamiento de los diferentes casos es novedosa y sencilla. Combinada con el entrenamiento por RAWN permite poblar en pocas horas un buen número de pruebas para obtener muy buenos resultados para el sensor virtual.

Dentro de la información consignada en las tablas y figuras de este mismo capítulo se aprecia la evolución de estas pruebas y nos lleva a seleccionar el caso del modelo que utiliza entradas de variables obtenidas por correlación y dispersión, para el cual se muestra el comportamiento final del sensor virtual contra el real en la figura.

El número de muestras necesarias para obtener buenos resultados no es función de la cantidad como si lo es de la calidad de los datos tanto de entrada como de salida y para ello además se deben escoger rangos de entrenamiento donde se cubra la mayor variabilidad siempre y cuando esta información sea verdadera para el proceso.

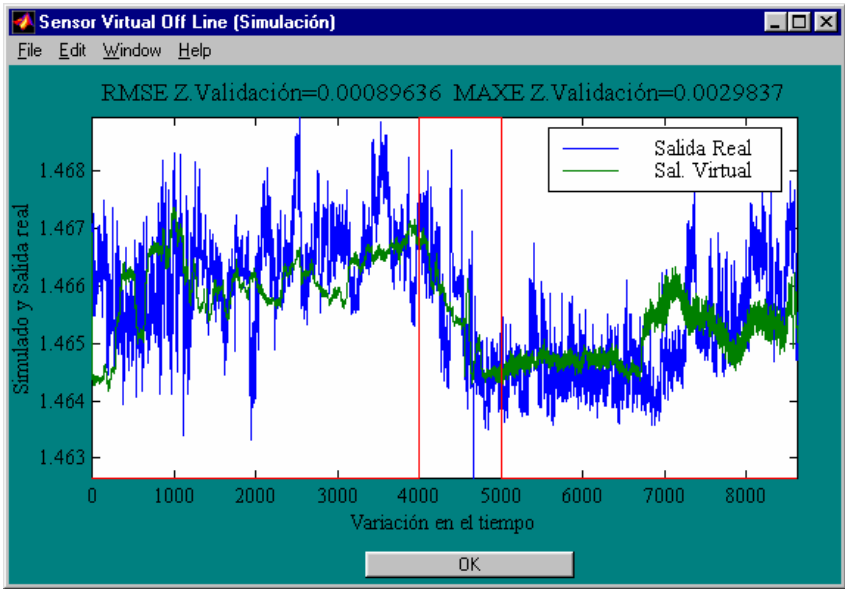


Figura 7.8: Comportamiento del Sensor Virtual Obtenido (Zona de Entrenamiento con 1000 muestras y Validación de las restantes)

7.1.5. Conclusiones y recomendaciones del trabajo con redes neuronales

7.1.5.1. CONCLUSIONES

Las redes neuronales artificiales son capaces de manejar problemas complejos y no lineales, pueden procesar información muy rápidamente y reducen el esfuerzo computacional requerido en el desarrollo de modelos computacionales intensivos, encontrando formas funcionales para modelos empíricos como el de nuestro caso con el sensor virtual.

En una red neuronal artificial solo se necesitan datos de entrada y salida

para que la red reconozca un patrón envuelto en el mapeo de las variables de entrada a la respuesta de la salida. Es verdad que las redes neuronales han sido descritas como una "caja negra" para solucionar problemas, pero la habilidad de la red neuronal para dar valores rápidos y precisos para solucionar problemas de ingeniería, hace de ellas una herramienta muy útil. Su habilidad para ejecutar fácilmente el problema inverso de intercambiar los vectores de entrada y la salida de la red, también se constituye en otra ventaja para el análisis y diagnóstico de un sistema dado.

Se puede obtener un excelente mapeo con redes neuronales en adelanto con tan sólo una capa de neuronas no lineales tomando los pesos de activación aleatoriamente, seguido por un entrenamiento de los pesos de salida por mínimos cuadrados ordinarios. Ejemplos estáticos y dinámicos muestran la factibilidad de esta aproximación. Como en cualquier identificación no lineal, se debe tener cuidado para asegurarse de que la entrada de excitación usada para identificación, esté en el mismo rango de frecuencia y amplitud como en la aplicación. Posteriores mejoras se pueden obtener por regularización de los pesos de activación.

Habiendo observado las extraordinarias propiedades de la red RAW, podemos decir que la selección aleatoria de pesos puede quizás suponer la necesidad de un gran número de neuronas que son estrictamente requeridas cuando se ejecuta un entrenamiento completo.

A pesar de esto, el número de grados de libertad con una RAWN permanece

más pequeño que con redes completas entrenadas por backpropagation.

La escogencia aleatoria de los pesos de activación no impide la solución de problemas lineales no separables. No se debe olvidar que el número de parámetros libres en una red en adelanto de dos neuronas entrenada por backpropagation es cuatro, si no se utilizan entradas de ajuste (bias), de lo contrario serían seis, mientras que una red neuronal RAWN sólo tiene tres grados de libertad.

Una vez los pesos de activación se escojan, no se necesitan más iteraciones y siempre se encuentra un mínimo a diferencia del backpropagation que sufre de propiedades de convergencia serias. Un buen ejemplo de esto es el problema clásico de la .ºr exclusiva"(XOR), donde la salida es uno si cualquiera de las dos entradas es uno, pero no ambas. Este problema es no lineal separable. Si una red neuronal es entrenada con backpropagation, se necesitan dos neuronas para un mapeo apropiado de la XOR. Con una red neuronal RAWN se necesitan tres neuronas, pero este caso se obtiene un mejor mapeo.

Los resultados presentados aquí sugieren que la regularización puede aún incrementar el desempeño de RAWN. Ciertamente vale la pena investigar por esquemas, como el sugerido por Drao y Ridella. Resultados preliminares con una modificación de la regularizacion de entrada inteligente presentada aquí, sugieren que en algunos casos se pueden alcanzar mejoras adicionales. En el siguiente gráfico se puede observar la representación gráfica utilizada en la literatura para una red neuronal.

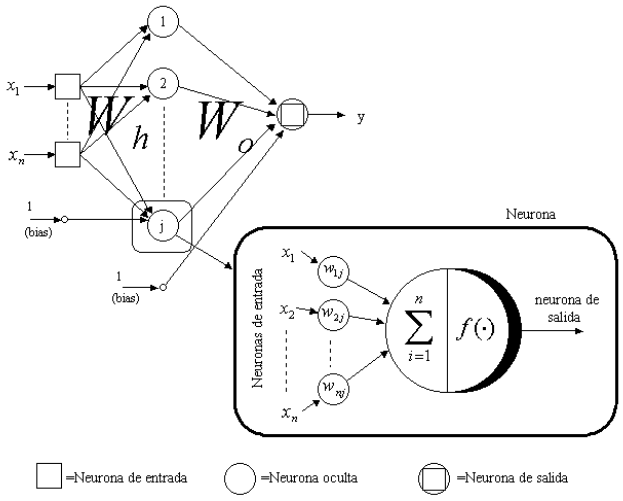


Figura 7.9: Ejemplo de una red neuronal alimentada hacia adelante. Los detalles muestran la también llamada "neurona de McGulloch-Pitts"

7.2. PRUEBA EXPERIMENTAL Y ESTRUCTURA FUNCIONAL DEL SENSOR

Para lograr los objetivos de la investigación se ha utilizado el método científico, desarrollando los siguientes pasos:

· Análisis de la situación actual: En la experiencia de los expertos que manejan el proceso de refinamiento del fenol, se ha llegado a la conclusión de que el índice de viscosidad del fenol depende de diferentes variables, de las cuales, las más relevantes han sido la presión, la temperatura, el nivel y el flujo del líquido en su proceso de refinado; es por esto que se plantea el desarrollo de esta investigación a partir de las teorías de Lógica Difuza , que permite hacer razonamientos

aproximados a los que hace el ser humano, y de Algoritmos Genéticos, que lleva implicita la evolución natural darwiniana; técnicas de inteligencia artificial, que pueden ser aplicados para la obtención del sensor clonado, cuya base fundamental es el comportamiento de las variables dadas en el entorno determinado en el proceso, junto con la metodología de clonación para obtener una replica del sensor real.

· Planteamiento de la hipótesis: "El sensor clonado ofrece una eficiencia similar a la del sensor físico en la obtención del índice de viscosidad en la refinería de ECOPETROL".

· Selección del método para probar la hipótesis: El método seleccionado para probar la hipótesis hace parte de lo presentado en el capítulo de la Metodología de Clonación.

· Recolección de datos: Las trazas de los datos son suministrados por un trabajo desarrollado por E.Nuñes y H.Paba en su tesis de maestría.[2]

· Diseño del experimento: Se presenta en el apartado de Prueba Experimental.

· Conclusiones sobre la hipótesis: Después de realizadas las pruebas experimentales con los resultados del sensor clonado contra el sensor original se deben concluir los resultados de la investigación.

[2]Paba Argote Harry J. y Nuñez Cossio Eudilson. Proyecto de grado para otar el título de Maestria en Ciencias Computacionales."Implementación de sensores inteligentes utilizando redes neuronales aplicados en procesos de refinación del petróleo". Universidad Autónoma de Bucaramanga. Colombia. 2000.

Este capítulo describe las pruebas que se realizaron para comprobar

la hipótesis, se establece el diseño experimental basado en las superficies de respuesta, des-pués de un análisis posibilístico de los datos, y se aplica la metodología de clonación propuesta para el sensor de viscosidad de fenol, permitiendo encontrar el sensor clonado y entregar las conclusiones de la investigación en el siguiente capítulo.

7.2.1. Diseño Experimental

A partir de la unidades experimentales o variables determinadas para la investigación (Temperatura, Nivel, Flujo y Presión) se determinó aplicar el tratamiento a los datos obtenidos en la planta industrial de ECOPETROL, como se explica en el apartado de la metodología de la experimentación.

7.2.1.1. Clusters

En el desarollo de la metodología se obtuvieron los cluster de entrada y de salida al sensor clonado. El cluster de entrada fué conformado por la superficie de respuesta dada por los sensores de temperatura, presión, nivel y flujo y una caracterización de la viscosidad, generada de manera aleatoria y el cluster de salida inicial se constituyó a partir de la superficie de respuesta dada por la información fuzificada de los sensores, con la información real del sensor de viscosidad.

Se dispone de la información de entrada y de salida ya filtrada y lista para la realización de los clusters, se debe entonces seleccionar el número de clusters

para cada variable, esta tarea, es sin lugar a dudas una de las más complejas del proceso, un gran número de clusters puede exigir una mayor cuantía de ciclos de procesamiento de señales y una cantidad mínima de ciclos, evitará que el sistema encuentre una solución con la precisión necesaria para ser considerado un sistema experto.

La determinación del volumen de la muestra se estableció a partir de las superficies de respuesta[3], entregadas por la integración de los cluster fuzzy por medio del surface obtenido con el toolbox de Fuzzy Logic de Matlab.

Los AG utilizan la teoría de probabilidades en las operaciones de cruzamiento y mutación; sin embargo la teoría de posibilidades surgió como resultado del desarrollo de la lógica difusa (Zadeh Lofti, 1998), que tiene diferencias esenciales con la probabilidad; la lógica difuza establece la diferencia entre la incertidumbre probabilística y la imprecisión posibilística, así, al aplicar los conjuntos borrosos (difusos), estos interpretan planteamientos imprecisos de los operadores genéticos al reproducirse las generaciones de poblaciones; cuyos individuos se combinan con pertenencias gradadas (grado de pertenencia-membership functions), de modo que cada evento en la clonación tiene un grado de pertenencia y no de probabilidad.

A continuación se muestran los cluster explicados anteriormente, donde:

Variable: Variable ALTA

$\overline{Variable}$: Variable BAJA

$\overleftrightarrow{Variable}$: Variable MEDIA

[3]Kuehl, R. O. Diseño de Experimentos. Ed. Thomson Learning. Ed. 2. 2001. pp. 423 a 462.

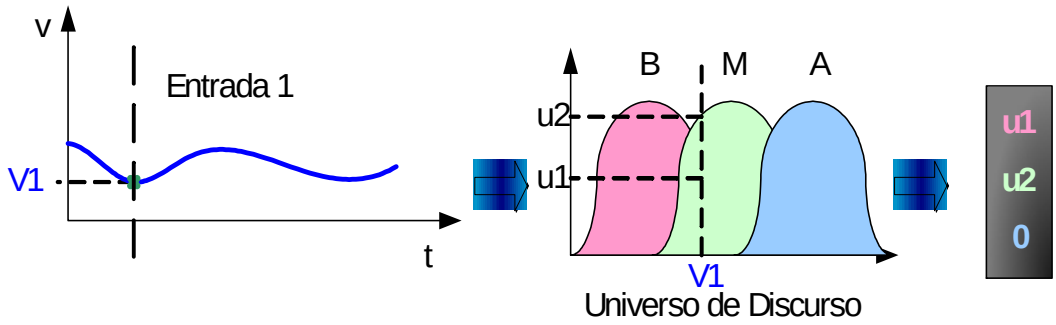
Cluster de Salida:

$\langle VI \rangle$	TF	$\overline{T}F$	$\overline{TF}$	$T\overline{F}$	$\overleftrightarrow{T}F$	$\overleftrightarrow{T}\overleftrightarrow{F}$	$T\overleftrightarrow{F}$	$\overline{T}\overleftrightarrow{F}$	$\overleftrightarrow{T}\overline{F}$
PN	V	$\overline{V}$	V	V	V	V	V	V	V
$\overline{P}N$	V	V	V	V	V	V	V	V	V
$\overline{PN}$	V	V	$\overleftrightarrow{V}$	V	V	V	V	V	V
$P\overline{N}$	V	V	V	V	V	V	V	V	V
PN	V	V	V	V	V	V	V	V	V
$\overleftrightarrow{P}\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$P\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$\overline{P}\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$\overleftrightarrow{P}\overline{N}$	V	V	V	V	V	V	V	V	V

$\langle VI \rangle$	TF	$\overline{T}F$	$\overline{TF}$	$T\overline{F}$	$\overleftrightarrow{T}F$	$\overleftrightarrow{T}\overleftrightarrow{F}$	$T\overleftrightarrow{F}$	$\overline{T}\overleftrightarrow{F}$	$\overleftrightarrow{T}\overline{F}$
PN	V	$\overline{V}$	V	V	V	V	V	V	V
$\overline{P}N$	V	V	V	V	V	V	V	V	V
$\overline{PN}$	V	V	$\overleftrightarrow{V}$	V	V	V	V	V	V
$P\overline{N}$	V	V	V	V	V	V	V	V	V
PN	V	V	V	V	V	V	V	V	V
$\overleftrightarrow{P}\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$P\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$\overline{P}\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$\overleftrightarrow{P}\overline{N}$	V	V	V	V	V	V	V	V	V

Cluster de entrada:

$\langle VI \rangle$	TF	$\overline{T}F$	$\overline{TF}$	$T\overline{F}$	$\overleftrightarrow{T}F$	$\overleftrightarrow{T}\overleftrightarrow{F}$	$T\overleftrightarrow{F}$	$\overline{T}\overleftrightarrow{F}$	$\overleftrightarrow{T}\overline{F}$
PN	V	$\overline{V}$	V	V	V	V	V	V	V
$\overline{P}N$	V	V	V	V	V	V	V	V	V
$\overline{PN}$	V	V	$\overleftrightarrow{V}$	V	V	V	V	V	V
$P\overline{N}$	V	V	V	V	V	V	V	V	V
PN	V	V	V	V	V	V	V	V	V
$\overleftrightarrow{P}\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$P\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$\overline{P}\overleftrightarrow{N}$	V	V	V	V	V	V	V	V	V
$\overleftrightarrow{P}\overline{N}$	V	V	V	V	V	V	V	V	V



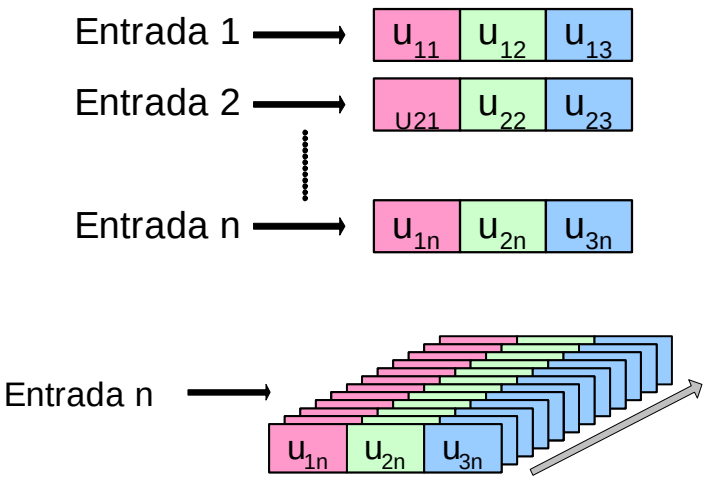
7.2.1.2. Creación de clusters.

El primer paso en la etapa de clonación consiste en crear los clusters para los valores de las entradas y salidas, esto contribuirá con la concepción de una metodología que pueda trabajar con problemas multiobjetivo.

Para lograr este enfoque, es necesario, aplicar el "fuzzy c-mean" para encontrar los respectivos clusters de cada señal, estos clusters tienen una representación en conjuntos difusos, por lo que un valor V1 se puede representar en n valores de pertenencia, donde n es el número de clusters de la variable en mención. Este procedimiento se aprecia en la siguiente gráfica.

Este procedimiento repite las entradas del sistema y para todas las salidas, pasando de una representación por valor a una representación por grado de pertenencia en los clústeres, tal y como se aprecia en la siguiente gráfica.

Para finalizar esta etapa, se convierte las señales de conjuntos de representaciones en clusters, lo que permitirá finalmente convertir todo el contenido de la señal en clusters, tal y como se aprecia en la siguiente gráfica.

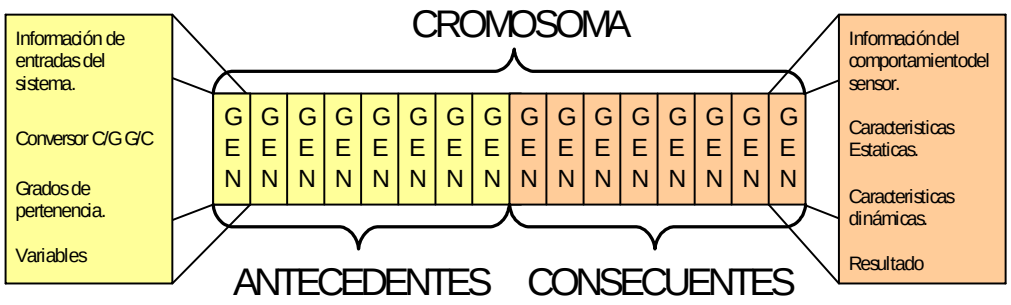


7.2.1.3. Cromosomas

Para esta creación de cromosoma, fue implementada la visión dada por la referencia [Delgado98], en la cual se utiliza una división del cromosoma en antecedentes y consecuentes, los antecedentes corresponden a las entradas del sistema, es decir, todas la diferentes variables que influyen en la inferencia de la o de las variables de salida, en esta sección también se pueden encontrar la información codificada de los clusters, grados de pertenencia, tipos de conjuntos difusos, entre otros.

Los consecuentes del cromosoma contienen información que ha sido obtenida del los antecedentes, estas pueden ser, características estáticas y dinámicas [Muñoz97], valor de salida propuesto, error estático y dinámico, por mencionar algunos.

La selección de los antecedentes y los consecuentes es libre y constituye una de las tareas más importantes para el desarrollo del proceso de clonación, esto es debido a que es en este punto del proceso, en donde se selecciona la información

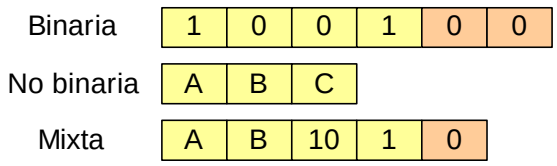


relevante para el proceso de clonación, la cantidad y variedad de los denominados "genes" del cromosoma será por lo tanto una de las decisiones particulares en cada proceso, esta representación de antecedentes y consecuentes (AC) se aprecia en la siguinete gráfica.

Codificación del cromosoma. La codificación del cromosoma es uno de los pasos de mayor relevancia, esto es debido a la gran variedad de representaciones para los valores y estados de un sistema.

La codificación binaria es la más utilizada para realizar operaciones, esto debido a la gran facilidad de realizar operaciones genéticas (cruce, mutación), exige generalmente que el tamaño de los cromosomas crezcan cuando buscan la representación de varios valores con un alto grado de precisión.

La codificación no binaria permite una lectura mas sencilla de los valores de cada "gen", por lo que el tamaño del cromosoma se reduce considerablemente, exige un poco mas de atención a la hora de aplicar operadores genéticos en vista de no generar individuos incorrectos, en esta codificación se pueden encontrar



valores enteros, caracteres (ideales para la representación fuzzy).

Codificación mixta: Incluye las dos codificaciones anteriormente mencionadas, y busca la flexibilidad de la codificación binara en la implementación de los operadores genéticos y las bondades de la codificación no binaria en la disminución del tamaño del cromosoma, una representación de todas las posibles codificaciones se aprecian en la siguinete gráfica.

7.2.1.4. Genoma

El genoma conformado en su estructura por varios cromosomas y los cromosomas a su vez formados por genes, los cuales contienen la información característica de cada uno de los individuos mediante alelos[4]. En este caso el genoma se puede interpretar geométricamente, ubicando los centros de las superficies de respuesta individuales de las variables en un cubo, cuyos vertices representan los valores de las variables, sus aristas indican la presencia de dos variables y las superficies contienen la información codificada de manera posibilística.

Sea una representación de un árbol binario ubicado en los vértices de un hipercubo (Ver figura 7.1.1), se definen las hojas del árbol como los genes de presión, temperatura, flujo y nivel; además los nodos definen una función lineal tipo sugeno

[4]Delgado A. Inteligencia Artificial y Minirobots. Ediciones Ecoe. Ed. 2. 1998. pp.147 a 157.

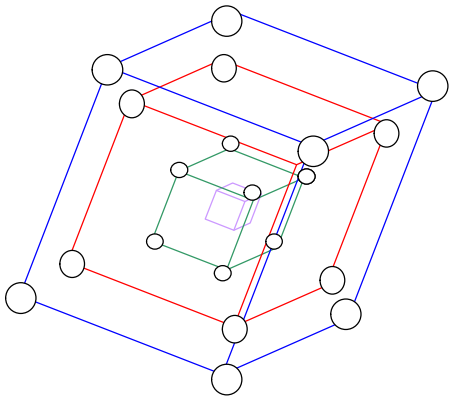


Figura 7.10: Estructura del genoma del sensor

f (P,T,N,F) que relaciona las hojas del árbol, para obtener el valor inicial de la viscosidad. Inicialmente los cromosomas tienen una profundidad homogénea y luego varía de acuerdo al algoritmo implementado, es decir dependiendo del punto de cruce escogido .

SELECCIÓN: El proceso de selección se realiza tomando como referente el sensor real de manera que se asigna el fitness de los cromosomas o árboles, teniendo en cuenta su relación funcional (Sugeno) y la salida del sensor real.

Se realiza primero la evaluación de las funciones que conforman la población punto a punto, igualmente recorriendo esta vez la totalidad del árbol y calculando su valor, todo esto en preorden.

CRUZAMIENTO: Contribuye con la variedad evolutiva de una población, en cada generación, los individuos que son más aptos y que tienen la posibilidad de permanecer exitosos en el ambiente cruzan parte de su contenido genético

con algún otro individuo de la población, formado un nuevo individuo, éstos dos cromosomas son padres de los dos nuevos individuos formados al cruzarse entre sí. Los pasos a seguir para realizar el cruce son los siguientes:

1. Seleccionar árboles o cromosomas a cruzar.

2. Seleccionar puntos o nodos de cruce en cada cromosoma.

3. Buscar nodos.

4. Calcular número respectivo de descendientes de los dos nodos y actualizar el valor en la raíz.

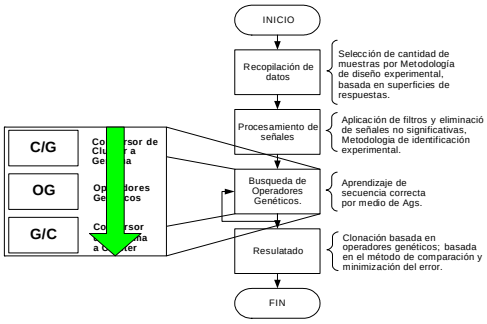
5. Realizar el cruce.

MUTACIÓN: Ésta se aplica utilizando una función determinada a partir del análisis de la superficie de respuesta:

(1-(t+T/2)/T)^1/4;

Éste proceso que se lleva hasta la mitad de las generaciones, tiempo en el cual se supone, desde la heurística, que ya se ha hecho toda una exploración en el espacio de búsqueda de la soluciones, en consecuencia después causaría perturbaciones cuando se está buscando la convergencia, lo que aumentaría la complejidad computacional y demoraría en encontrar la solución, distorsionando la información.

CRITERIO DEL NÚMERO DE GENERACIONES. El criterio del número de generaciones se determina de manera heurística y su finalidad es encontrar un número de generaciones que permitan hallar la convergencia, en este caso, la convergencia se ha encontrado máximo con 1000 generaciones, más de éste



número aumenta la complejidad en tiempo del algoritmo, según se explica en el manual de usuario, se puede experimentar con diferente catidad de generaciones y seleccionando la mayor profundidad de los árboles n-arios.

El siguiente algoritmo muestra claramente lo explicado anteriormente.

Finalmente, el resultado obtenido con esta metodología, son funciones de salida (para problemas multiobjetivo) que contienen la información solicitada por el diseñador.

7.2.2. Metodología de la experimentación

Despues de realizar el estudio detallado del estado del arte, la investigación se concentra en primera instancia en el análisis de los datos utilizados por Pava. H y Nuñes E., ingenieros de ECOPETROL quienes realizaron el análisis y toma de los mismos.

Para la realización de los experimentos se utilizaron las trazas de datos, las cuales corresponden a los siguientes dispositivos: 17 sensores de temperatura, 6 sensores de flujo, 4 sensores de presión, y 2 de Nivel.

Se realizó una discriminación de los dispositivos de acuerdo a la amplitud de su oscilación, éste criterio se utilizó en razón a que los sensores de menor oscilación son subconjuntos del sensor de mayor oscilación para cada variable.

A continuación se determinó el comportamiento dinámico de los sensores, haciendo uso de la función de transferencia. Esto con el fin de determinar la función de pertenencia para los conjuntos borrosos de las variables linguisticas de temperatura, presión, nivel y flujo, como sus valores linguísticos (Ejm. temperatura baja, tempe-ratura media o temperatura alta).[5]

Las funciones de pertenencia que indican la posibilidad con que los elementos del dominio de discurso pertenecen a los respectivos subconjuntos difusos, permitieron construir los clusters con la información propia de los sensores, a diferencia de la obtención de los clusters tradicionales que se hacen con las reglas difusas elaboradas a partir del conocimiento del experto.

Parte de la investigación se encuentra reflejada en los siguientes archivos que se anexan al presente documento, en medio magnético con su explicación respectiva. La siguiente información sienta las bases para la clonación artificial de un sensor de viscosidad de fenol, haciendo uso de la técnica posibilística que se encuentra implícita en la teoría de la lógica difusa de Zadeth, de la metodología general de clonación propuesta por el director del trabajo de grado y la metodología utilizada por los autores del trabajo, abriendo así, un nuevo campo para la investigación en

[5]Delgado, A. Inteligencia Artificial y Minirobots.Ediciones ECOE. Segunda Edición. 1998. pp 169 a 190.

esta área.

GráficasMAtlab.doc: Primer gráficación de los datos, con la cual se buscó una relación entre viscosidad y temperatura. Se señala un cambio identificado en los datos del 22 al 29 de Julio de 2000. La muestra utilizada fué de 30560 datos.

GráficasViscosidad.doc: Muestra la búsqueda de cambios bruscos en la viscosidad con el fin de obtener la respuesta dinámica del sensor, el mes de Junio se omitió por la escacés de la información, además se aplicó un filtro pasa bajas. Las referencias de las gráficas pueden leerse mes/dia inicial del muestreo/dia final del muestreo.

Graficas de la data.ppt: Gráficos de todos los sensores , buscando cuál de ellos presentaba mayor variación, para tomarlo como referencia, muestra el enálisis estadístico que se realizó.

MemberShipViscisidad.ppt: Aplicación del algoritmo C-mean en los datos de viscisidad, así se demostró que en diferentes meses de muestreo las gráficas son distintas, pero si se realiza una muestra aleatoria de todos los datos la gráfica es sismilar a la que arroja el C-maen en toda la población, este criterio permitió aplicar el mismo principio para el análisis de los demás sensores.

MemberShipFlujo: Agrupamiento con 3,5 y 7 centros realizado con C-mean.

MemberShip IN/OUT: Agrupamiento con C-means con tres centros realizado con los sensores seleccionados.

Tabla.xls: Datos obtenidos a partir de la aplicación del cluster con la técnica

de agrupamiento C-means.

Proceso.ppt: Muestra el proceso que se siguió para obtener las funciones de transferencia de todos los sensores, el punto de trabajo se establece hallando la moda de la data, se buscó un sitio en la data donde su comportamiento obedeciera a funciones de primer o segundo orden y se determina una banda de tolerancia del 10 %, los datos utilizados fueron 187.842.de muestras tomadas en Junio, Julio, Septiembre, Octubre, Noviembre y Diciembre. los datos de agosto no permitieron un análisis adecuado.

Mf-reales.ppt: Cluster construidos con las funciones de pertenencia obtenidas a partir de las funciones de transferencia de los sensores, se tomaron datos que estuvieran en la banda de tolerancia, los demás no fueron utilizados, se trabajó con el OR Fuzzy.

BWtol.xls: Datos que se encontraron dentro de la banda de tolerancia, se obtuvieron 187.843 datos.

Tabla2.xls: Datos de BWtol.xls y construcción del cluster.

bwtololat.m: Archivo de Matlab que carga 187.000 datos, evalua que los datos de los cinco sensores se encuentren paralelamente en la banda de tolerancia para obtener un archivo de salida con éstos datos. Obtiene el cluster cálculando la pertenencia del dato y el tipo de conjunto al que pertenece. Entrega el resultado en una matriz de 15 columnas.

Clus.m: Construye el cluster utilizando C-means. La data debe estar en una

columna.

Muestreo.m: Hace un muestreo aleatorio sobre la data.

Verif.m: Verifica si hay incoherencias en las reglas.

Vardat.m: Contiene la matriz de los cinco sensores seleccionados, con 187.842 datos de Flujo, Nivel, Presión, Temperatura y Viscosidad.

Luego del análisis de datos se determina con cual cluster se va a trabajar y se aplicala metodología de clonación del sensor obteniendo como resultado el programa de computador viscosidad.exe, que se desarrolló en el lenguaje C#,Se anexan el diagrama de clases específico (Diagama de clases específico.png), el diagrama de clase general (Diagrama de clase general.png) y el diagrama de flujo (diagrama de Bloques.ppt).

7.3. DIAGRAMA DE FLUJO PARA LA ELABORACIÓN DEL ALGORITMO DEL SENSOR CLONADO

El diagrama de flujo muestra la secuencia en que se realizan los procedimientos necesarios para implementar el algoritmo genético mediante procedimientos que son sencillos; y lo mas sorprendente es que efectivamente, el método evolutivo conduce a soluciones idóneas para el problema en cuestión. La sencillez es entendida, de como a partir de una población inicial y a la continua iteración de los operadores genéticos sobre poblaciones a partir de esta, llevan hasta encontrar la solución del problema, como puede observarse en el diagrama de flujo.

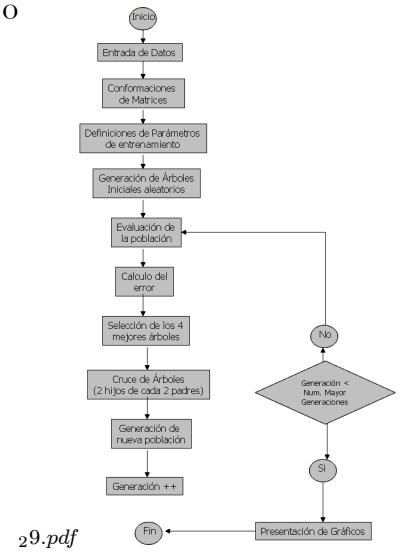
El potencial de aplicación de estos algoritmos es inmenso, y aparece como un campo de investigación interesante en muchas áreas del conocimiento. El esfuerzo principal para poder utilizarlos en problemas específicos lo constituye la búsqueda de la representación de la solución mediante cromosomas, y la definición adecuada de los operadores de selección, cruce y mutación, y de la función de aptitud. Luego con ayuda del computador se observa como "misteriosamente", el algoritmo genético evoluciona hasta encontrar la solución. En lo que respecta a las ciencias computacionales, la investigación está abierta para la búsqueda de nuevos operadores, y la asimilación de nuevos conceptos estudiados en la teoría genética y evolutiva, con el ánimo de enriquecer este nuevo enfoque de solución de problemas no solo del área de la ingeniería sino de muchas disciplinas.

El código fuente documentado en detalle se encuentra anexo en el medio magnético adjunto a este trabajo.

7.4. CRITERIOS DE SEMEJANZA CON EL SENSOR REAL

El criterio de semejanza entre los sensores real y clonado, se establece cuando se obtiene, por medio del programa la convergencia del algoritmo, como consecuancia de la aplicación del fitness a los individuos en cada población, convergencia que representa la solución del problema; comparados los valores de esta solución, con los valores del indice de viscosidad del sensor real, se obtiene una diferencia mediante un medición adimencional o dada en porcentaje, que se conviente en

de flujo



$_{2}9.pdf$

Figura 7.11: Diagrama de flujo del programa del sensor clonado

la aproximación de los resultados de los sensores. Gráficamente el programa establece la diferencia de los valores medidos de viscosidad y los obtenidos con el sensor clonado, como puede observarse en la gráfica obtenida mediante la manipulación de la interfaz de usuario.El manejo de la interfaz se presenta en la sección del manual de usuario del programa.

7.5. SITEMAS DE CONTROL

En el ambiente genético hay un control conocido como Selección Natural de Darwin, que se puede resumir así:

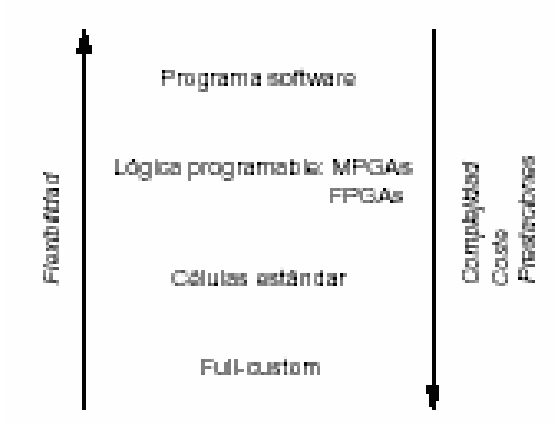
"La información genética no válida, ya sea por cambios en los ambientes o por mutaciones, no se trasmite a las generaciones siguientes evitando fallas en las generaciones posteriores". Este criterio es coherente con el software desarrollado

ya que se basa en la aptitud de las estructuras de datos: árboles, los árboles menos aptos no se pueden reproducir, no pueden transmitir información genética a las generaciones siguientes.

Los sistemas de control electrónico que se utilizan en la investigación, se basan en la utilización de los arreglos lógicos programables (File Programming Gate Array, FPGA), a continuación se entrega la información básica necesaria para la comprensión de la proyección de la investigación hacia el uso de hardware evolutivo.

7.5.1. Introducción a los FPGA

Cuando se aborda el diseño de un sistema electrónico y surge la necesidad de implementar una parte con hardware dedicado, son varias las posibilidades que hay. En la siguiente figura se han representado las principales aproximaciones ordenándolas en función de los parámetros costo, comunicaciónes, prestaciones y complejidad. Como se puede ver, las mejores prestaciones las proporciona un diseño full-custom, consiguiéndose a perjuicio de elevados costos y enorme complejidad de diseño. En el otro extremo del abanico de posibilidades se encuentra la implementación software, que es muy barata y exhibe, pero que en determinados casos no es valida para alcanzar un nivel de prestaciones relativamente alto.



digitales

Entre estas dos opciones se puede elegir la fabricación de un circuito electrónico realizado mediante diseño SEMIcustom, utilizando células estándar, o recurrir a un circuito ya fabricado que se pueda programar"insitu, como son las Fugas. De estas dos opciones la primera proporciona mejores prestaciones, aunque es muy cara y exige un ciclo de diseño relativamente largo. Por otro lado, los dispositivos lógicos programables constituyen una buena oferta para realizar diseños electrónicos digitales con un buen compromiso costo-prestaciones. Y lo que es mejor, permiten obtener una implementación en un tiempo de diseño asombrosamente corto .Otro aspecto que se debe tener en cuenta para decidirse por este tipo de implementación es que el costo de realización es muy bajo, por lo que suele ser una buena opción para la realización de prototipos. En estas notas vamos a describir de forma muy somera en que consisten estos dispositivos, particularizando para una familia del fabricante Xilinx. La información contenida en ellas se basa en gran medida en las siguientes fuentes: sobre arquitectura de FPGAs y sobre

diseño de circuitos.

7.5.2. Implementación del microcontrolador en el diseño evolutivo

Para este diseño se implementara el microcontrolador que es un dispositivo más accesible y tiene las prestaciones necesarias para el diseño.

Para tal fin se tomaran las entradas de el micro como sigue en el siguiente apartado:

Puerto B : Este puerto servirá como salida de mutación para crear las mutaciones necesarias para encontrar la solución de la señal a concatenar, sus salidas soportaran dos cromosomas de cuatro bits del algoritmo genético.

Puerto C : Puerto de comparación con la función fitness .

Puerto D : Estará comparando la señal de salida la solución que se esta buscando.

Puerto A : señal de entrada de los bits a guardar en la eeprom una vez se encuentra la solución.

El algoritmo genético constara de varios operadores mutación, cruce, inversión

A continuación se dará el diagrama de flujo del algoritmo genético.

7.5.3. Estructura general de los FPGAs

El proceso de diseño de un circuito digital utilizando una matriz lógica programable puede descomponerse en dos etapas básicas:

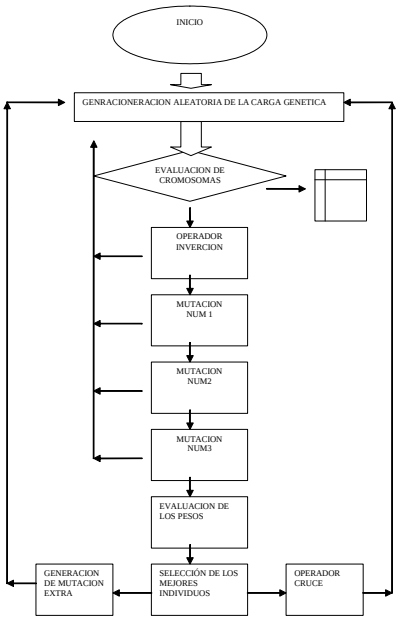


Figura 7.12: Diagrama de flujo del algoritmo evolutivo

1. Dividir el circuito en bloques básicos, asignándoles a los bloques configuraciones del dispositivo.

2. Conectar los bloques de lógica mediante los conmutadores necesarios.

Los elementos básicos constituyentes de un FPGA como las de Xilinx se pueden ver en la figura anterior y son las siguientes:

1. Bloques lógicos, cuya estructura y contenido se denomina arquitectura. Hay muchos tipos de arquitecturas, que varían principalmente en complejidad (Desde una simple puerta hasta módulos más complejos o estructuras tipo PLD). Suelen incluir biestables para facilitar la implementación de circuitos secuenciales. Otros módulos de importancia son los bloques de Entrada/Salida.

2. Recursos de interconexión, cuya estructura y contenido se denomina arqui-

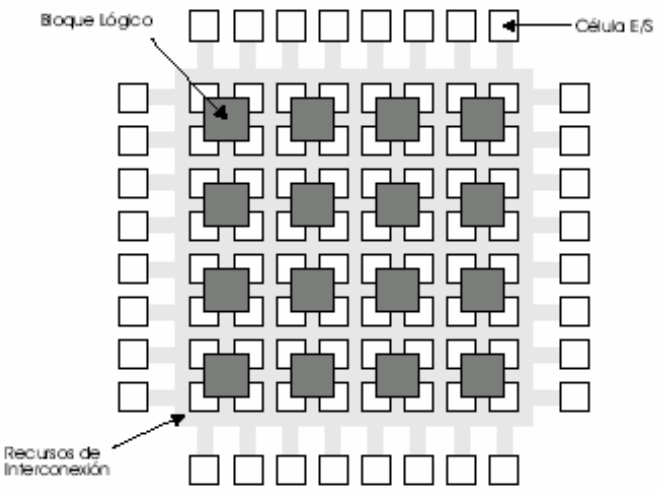


Figura 2: Estructura general de una FPGA (en concreto de XILINX)

tectura de rutado.

3. Memoria RAM, que se carga durante el RESET para configurar bloques y conectarlos.

Entre las numerosas ventajas que proporciona el uso de FPGAs se destacan dos principalmente:

El bajo costo de prototipado y el corto tiempo de producción, no todo son ventajas, entre los inconvenientes de su utilización están su baja velocidad de operación y baja densidad lógica (poca lógica implementa en un solo chip). Su baja velocidad se debe a los retardos introducidos por los conmutadores y las largas pistas de conexión.

Por supuesto, no todas las FPGA son iguales. Dependiendo del fabricante podemos encontrar diferentes soluciones. Los FPGAs que existen en la actualidad

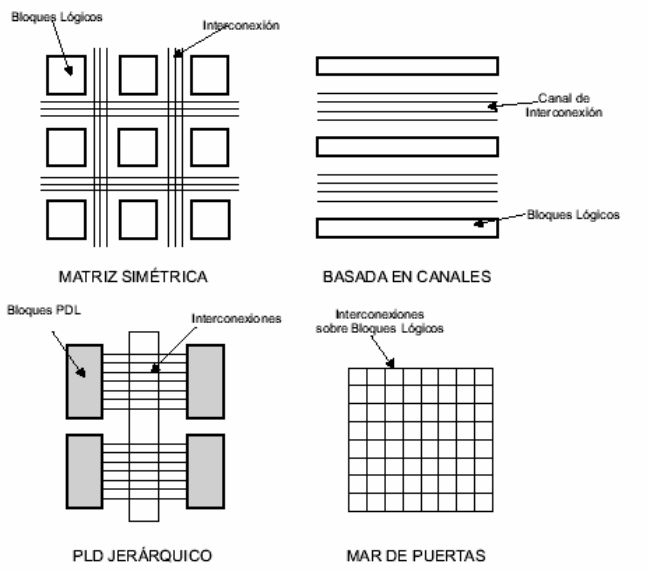


Figura 3: Tipos de FPGAs

en el mercado se pueden clasificar como pertenecientes a cuatro grandes familias, dependiendo de la estructura que adoptan los bloques lógicos que tengan definidos. Las cuatro estructuras se pueden ver en la siguiente figura, sin que aparezcan en la misma los bloques de entrada/salida.

1. Matriz simétrica, como son las de XILINX

2. Basada en canales, ACTEL

3. Mar de puertas, ORCA

4. PLD jerárquica, ALTERA o CPLDs de XILINX.

En concreto, para explicar el funcionamiento y la estructura básica de este tipo de dispositivos programables solo se consideraran las familias de XILINX.

7.5.3.1. Arquitectura de las FPGA de Xilinx

TECNOLOGÍA DE PROGRAMACIÓN

Antes de continuar con conocimientos mas avanzados acerca de FPGAs (de XILINX en concreto), hay que aclarar como se realiza el proceso de programación (ie., las conexiones necesarias entre bloques y pistas). En primer lugar, si se piensa que el número de dispositivos de conexión que hay en una FPGA es muy grande (típicamente superior a 100.000), es necesario que cumplan las siguientes propiedades:

Ser lo mas pequeños posible.

Tener la resistencia ON lo mas baja posible, mientras la OFF ha de ser lo mas alta posible (para que funcione como conmutador).

Se deben poder incorporar al proceso de fabricación de la FPGA.

El proceso de programación no es único, sino que se puede realizar mediante diferentes tecnologías, como son células RAM estáticas, EPROM y EEPROM. En el caso de las FPGAs de XILINX los elementos de programación se basan en células de memoria RAM que controlan transistores de paso, puertas de transmisión o multiplexores. En la figura se puede ver esquemáticamente como son. Dependiendo del tipo de conexión requerida se elegirá un modelo u otro.

Es importante destacar que si se utilizan células SRAM, la configuración de la FPGA será valida únicamente mientras este conectada la alimentación, pues es memoria volátil. En los sistemas finales esta claro que hace falta algún mecanis-

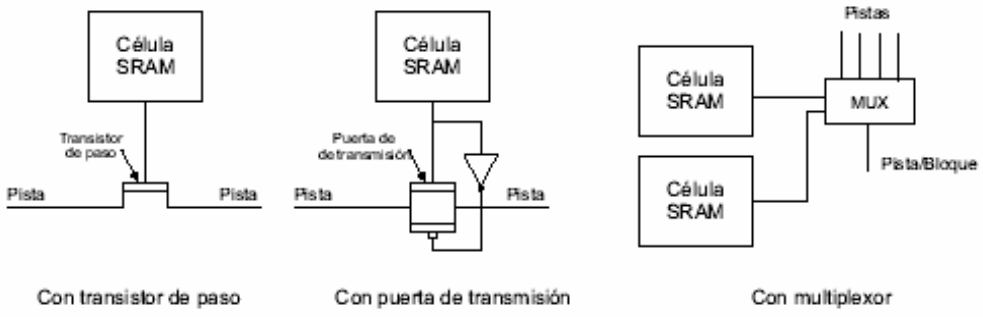


Figura 4: Tipos de conectores utilizados por XILINX

mo de almacenamiento no volátil que cargue las células de RAM. Esto se puede conseguir mediante EPROMs o disco. Este elemento de programación es relativamente grande (necesita por lo menos 5 transistores), pero se puede implementar en el proceso normal de fabricación del circuito (es CMOS).

Además, permite reconfigurar la FPGA de una forma muy rápida.

7.5.4. Descripción de las principales familias

Hay múltiples familias lógicas dentro de XILINX. Las primeras que surgieron son: XC2000 (descatalogada en el año 1999), XC3000 y XC4000, correspondientes respectivamente a la primera, segunda y tercera generación de dispositivos, que se distinguen por el tipo de bloque lógico configurable (CLB) que contienen. En la actualidad existen también las familias de FPGA SpartanII, SpartanIII, Virtex, VirtexII y VirtexPro. El siguinete cuadro, muestra la cantidad de CLBs que puede haber en cada FPGA de las familias base y ese mismo valor expresado en puertas equivalentes.

SERIE	Tipo CLB	Nº de CLBs	Puertas Equivalentes
XC2000	1 LUT, 1 FF	64-100	1.200-1.800
XC3000	1 LUT, 2 FF	64-484	1.500-7.500
XC4000XL	3 LUT, 2 FF	64-3.136	1.600-180.000

Cuadro 1: Familias del fabricante XILINX

El bloque lógico ha de ser capaz de proporcionar una función lógica en general y reprogramable. La mejor forma de realizar esto es mediante una tabla de valores preasignados o tablas de look-up. Básicamente, una tabla de look-up (LUTs en adelante) es una memoria, con un circuito de control que se encarga de cargar los datos. Cuando se direccionan las entradas de la función boleana, la memoria devuelve un dato, lo que se puede hacer corresponder con la salida requerida. Falta añadir los componentes necesarios para desempeñar funciones no implementables con una memoria, tales como una batería de registros, multiplexores, buffers etc. Estos componentes están en posiciones fijas del dispositivo.

La ventaja de la utilización de este tipo de tablas es su gran flexibilidad, una LUT de k entradas puede implementar cualquier función booleana de k variables, y hay 22k funciones posibles. El inconveniente es obvio: ocupan mucho espacio y no son muy aprovechables.

Los bloques lógicos configurables de la familia XC2000 se componen de una tabla de lookup con cuatro entradas y un biestable, con lo que puede generar cualquier función de hasta 4 variables o dos funciones de 3 variables. El de la familia XC3000 es mas complejo: permite implementar una función de 5 variables

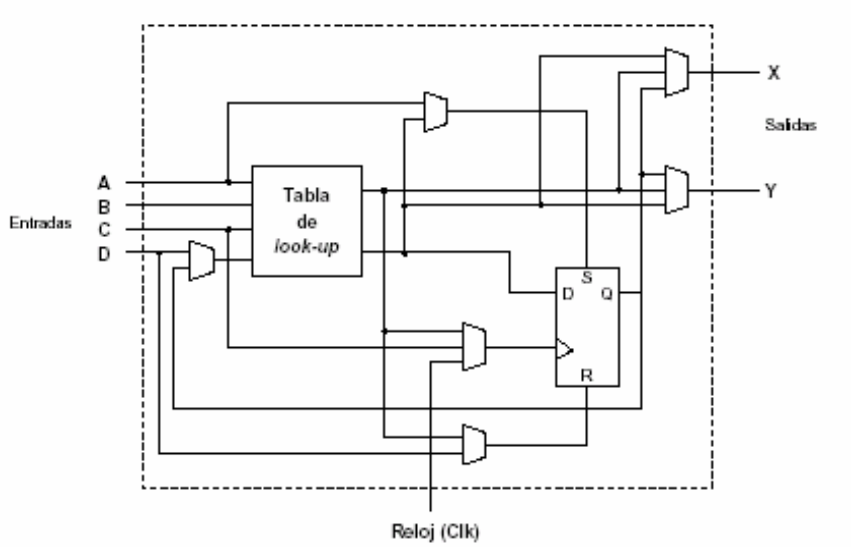


Figura 5: Arquitectura del CLB de la XC2000

o dos funciones de 4 variables (limitadas a 5 diferentes entradas). Además contiene dos biestables y cierta lógica. La familia XC4000 es más sofisticada. Tiene tres tablas de look-up dispuestas en dos niveles, llegando a poder implementar funciones de hasta 9 variables.

En general, los recursos de interconexión son de tres tipos:

Conexiones directas, permiten la conexión de las salidas del CLB con sus vecinos mas directos (N, S, E y O).

Interconexiones de propósito general, para distancias superiores a un CLB (mas allá del vecino). Son pistas horizontales y verticales del tamaño de un CLB, pero que se pueden empalmar para crear pistas mas largas.

Líneas de largo recorrido, suelen cubrir lo ancho o largo de la pastilla. Permiten conexiones con un retardo mucho menor que uniendo las anteriores. El camino

critico de un circuito es el recorrido que, desde una entrada hasta una salida, presenta un retardo máximo.

7.5.5. Arquitectura de la familia XC2000

Aunque hoy en día no se encuentran disponibles las FPGAs de esta familia, dado que contienen la arquitectura más sencilla, se utilizarán como base para comprender el funcionamiento de este tipo de dispositivos.

En la figura se puede ver como es el bloque configurable básico de las XC2000. Contiene como elementos principales una tabla de look-up de 4 entradas y un biestable D. La tabla de look-up puede reproducir cualquier función de cuatro variables o dos funciones de tres variables.

De las dos salidas del CLB una se puede registrar, o se pueden dejar las dos combinacionales.

Adicionalmente, en el bloque hay 6 multiplexores que permitirán seleccionar las conexiones que se desea hacer. Por ello, en sus terminales de selección necesitaran un elemento de memoria con el valor deseado. Nótese que la salida del biestable se puede llevar de vuelta a una de las entradas de la LUT, siempre y cuando se configuren adecuadamente los selectores oportunos. Esto es muy útil, pues permite implementar estructuras realimentadas como son contadores o máquinas de estados.

Por otro lado, la arquitectura de rutado de la familia XC2000 utiliza tres tipos

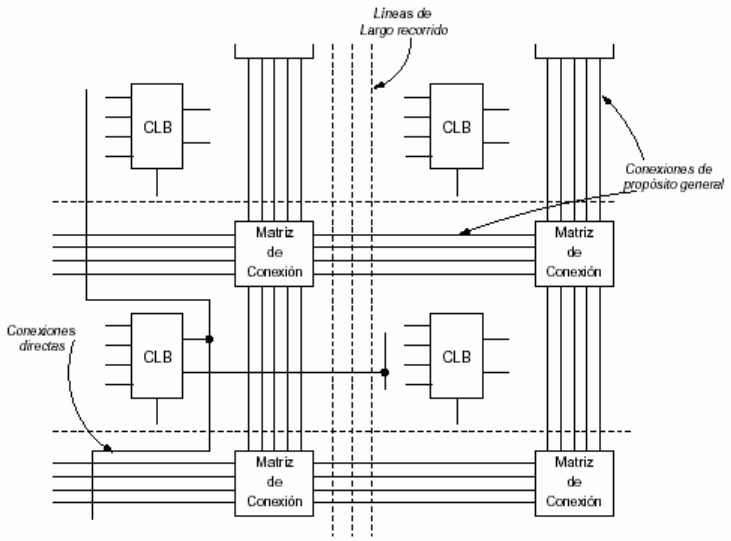


Figura 6: Recursos de interconexión en la familia XC2000

de recursos de interconexión: conexiones directas, conexiones de propósito general y líneas de largo recorrido.

Las conexiones directas proporcionan enlace desde la salida de un CLB hasta sus vecinos superior, inferior y a la derecha. Si hay que conectar una red a un bloque mas lejanodeben utilizarse las conexiones de propósito general, que son segmentos de pista dispuestas horizontal y verticalmente a lo largo de todo el FPGA. En particular, en esta familia hay cuatro segmentos horizontales y cinco verticales por canal. Su longitud esta limitada siempre a la distancia fija entre 2 CLBs, por lo que para realizar conexiones mas largas hay que utilizar las matrices de interconexión.

Es importante observar que la utilización de estos recursos repercutirá negativamente en las prestaciones del diseño, pues los conectores de la matriz introducen

forzosamente un retardo. Las líneas de largo recorrido se utilizan para conexiones que han de llegar a varios CLBs con bajo skew.

7.5.6. Diseño e implementación de la célula madre electrónica

Desde finales del siglo XX, se han producido avances notables en la investigación sobre células stem, llamadas también células troncales (entre ellas se encuentran denominaciones como células madre, células primordiales o células progenitoras), constituyen una enorme esperanza para el tratamiento de enfermedades que afectan a millones de personas en el planeta, algunas de las cuales aún son incurables.

El diseño de circuitos digitales, entre los paradigmas ya hechos se han propuesto los diseños de compuerta AND y OR y sus correspondientes inversores, NAND y NOR, con estos operadores básicos se puede diseñar cualquier clase de los circuitos lógicos existentes, centrando la atención en las compuertas NAND y NOR, la característica mas importante de estos operadores es que uno o cualquiera de los dos es el resultado de negar o invertir las entradas de señal del otro es por esto que el diseño del circuito evolutivo se enfocará en la implementación de estas dos compuertas. Sustentando lo anterior en el hecho de que en los laboratorios que se realizan en diseño de circuitos digitales los resultados son los esperados con respecto a los que implementan compuertas AND, OR y negadores.

Este enfoque de solución da un resultado que mas adelante discutiremos.

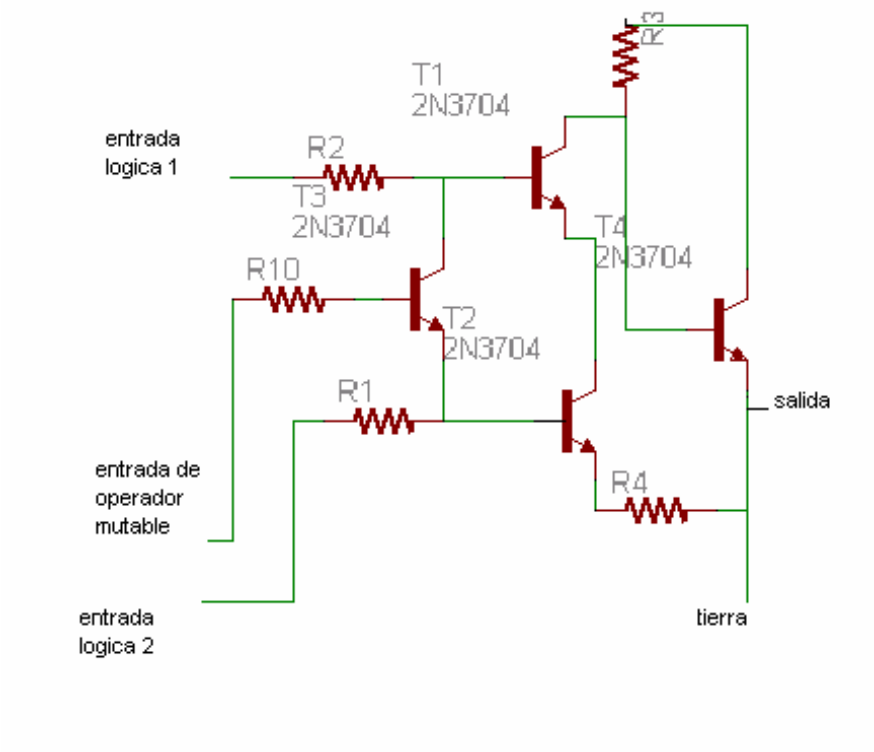


Figura 7.13: Circuito operador evolutivo NAND NOR

Citando la teoría vista sobre células madres, para este caso se implementa el diseño enfocado a OPERADORES LÓGICOS MUTABLES, que será el tema a tratar a continuación.

7.5.6.1. Operador mutable NAND / NOR

Para este punto el diseño es un circuito que tiene las características de una compuerta NAND ante una señal de control y una compuerta NOR ante la señal inversa de control. Se propone el siguiente diseño y la simbología del circuito:

Como ya se ha dicho trabajaremos con el transistor 2n2222 de muy fácil acceso.

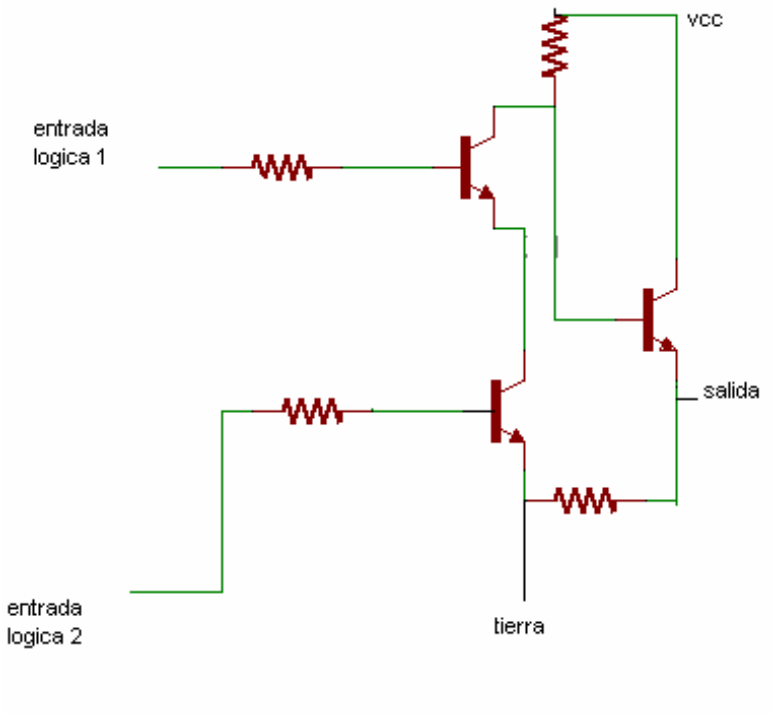


Figura 7.14: Circuito operador evolutivo NAND

El circuito consta básicamente de tres transistores; dos de ellos son la señal de entrada binaria y el transistor del centro funciona como conmutador entre las bases de los transistores de los lados T1 y T2.

Pero para poder analizar mas detalladamente el funcionamiento de el circuito se separa T3 . El circuito resultante se muestra en la siguinete figura. Este circuito funciona como una compuerta NAND; dado que los transistores se encuentran trabajando en zona de saturación, según este concepto el transistor esta trabajando en dos puntos de la recta de carga: como un interruptor cerrado o como un interruptor abierto.

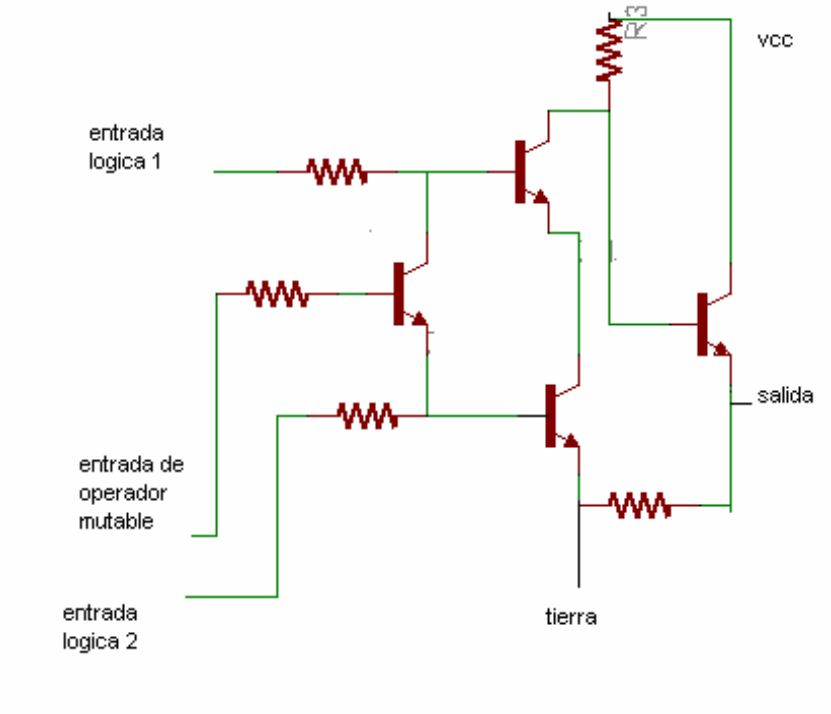


Figura 7.15: Circuito Operador lógico nor.

Siguiendo con explicación del diseño tomaremos la otra parte en la que el transistor de mutación genera un nuevo circuito y cuyo comportamiento se espera sea el de una compuerta nor.

En la figura podemos observar que el transistor de mutación conecta la dos bases es decir ante un uno en la entrada comunica las dos bases y con una señal de un 1 lógico tendremos la misma señal en el otro transistor.

Se puede recurrir a las tablas de valores lógicos y verificar que:

Tabla de valor lógico nor.

Entrada lógica 1	Entrada lógica 2	salida

0	0	1
0	1	0
1	0	0
1	1	0

Gracias a esta tabla se puede ver que el comportamiento del circuito es el de una compuerta nor y que una vez hay un uno lógico en la entrada del transistor el circuito se comporta como un circuito nor.

Finalmente para analizar todo el circuito se estudiara todo el comportamiento lógico a través de la tabla de verdad siguiente.

Señal de mutación	Entrada logica1	Entrada logica2	Salida
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Tabla de verdad para la compuerta mutable NAND – NOR

A continuación proponemos la simbologia

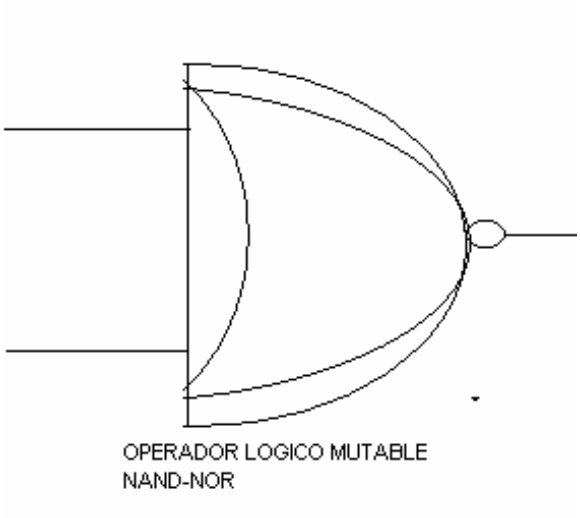


Figura 7.16: Símbolo operador logico mutable NAND NOR.

7.5.7. OPERADOR MUTABLE NOR / OR

A continuación se presenta el diseño y la simbología del circuito.

Como en el apartado anterior se separa el circuito en sus dos componentes de la operador lógica or y la nor. En la figura podemos observar el circuito para el operador lógico OR. En la entrada observamos que cualquier valor lógico de 1 en una de las entradas producirá un voltaje en el emisor de el transistor pero en la salida dependerá de un cero lógico en la entrada de la base de el transistor de mutación, que en este caso es un 2n3906 que al igual que el 2n2222 es muy comercial y que cuyas prestaciones al circuito resultaron ser muy buenas.

El análisis de lo que ocurre con el circuito es muy sencillo; ante una entrada de un cero lógico en los terminales de la base de el transistor, el transistor se polarizara y dejara pasar la señal presente en el emisor. Se puede mostrar mediante una tabla

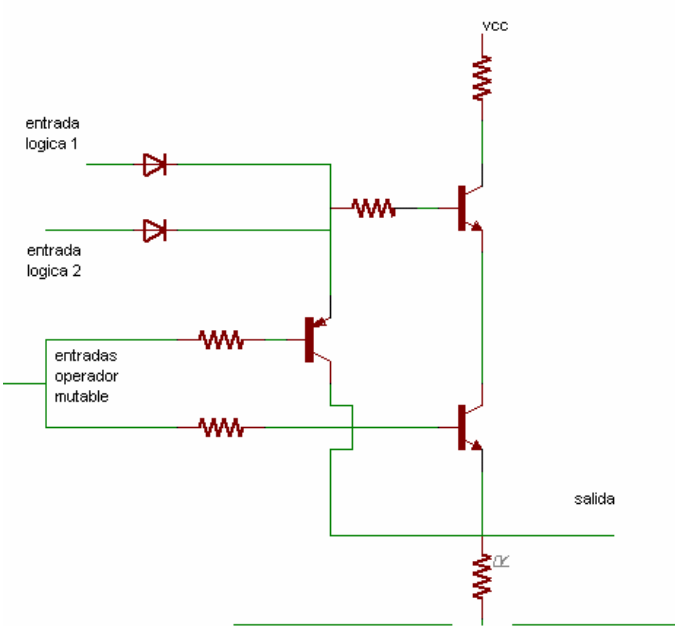


Figura 7.17: Circuito operador mutable NOR / OR

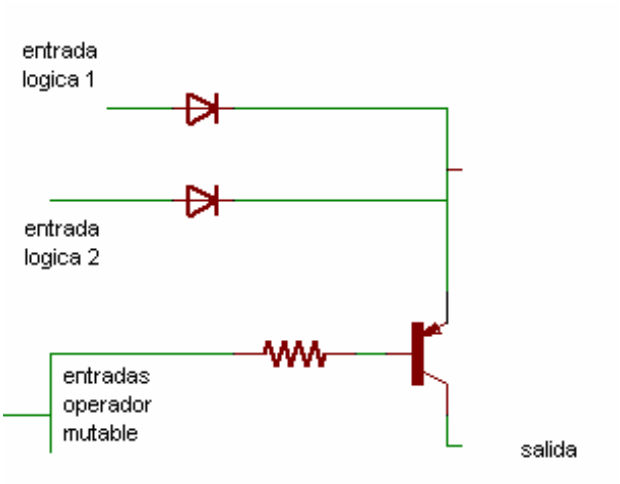


Figura 7.18: Operador lógico OR

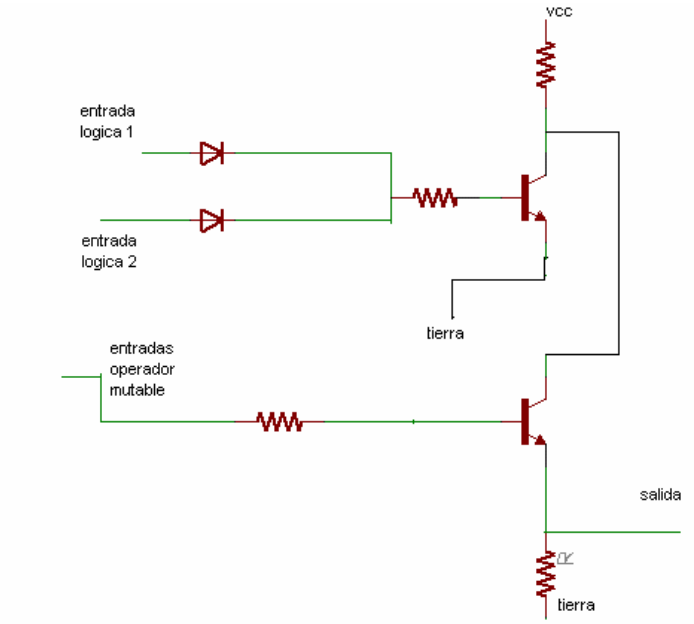


Figura 7.19: Operador lógico nor.

un operador or.

Entrada lógica 1	Entrada lógica 2	salida
0	0	0
0	1	1
1	0	1
1	1	1

Tabla de verdad para el operador lógico or.

Es claro que la señal de mutación de el transistor es un cero lógico por ese motivo nuevamente para simplificar y por comodidad se omite.

Ahora estudiaremos la otra parte que es el operador lógico nor como sigue y viendo la figura:

Analizando todo el circuito mientras hay un cero lógico en la entrada de control de mutación estar funcionando como un operador lógico or y cuando haya un uno se estar comportando como un operador lógico nor, a continuación se da la tabla de verdad para el operador mutable NOR- OR.

Señal de mutación	Entrada lógica 1	Entrada lógica 2	salida
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

Tabla de verdad de valores lógicos para el operador lógico NOR-OR

Y propone la simbología del operador:

7.5.7.1. Implementación de la célula madre

Una vez presentados los operadores lógicos a implementar y sabiendo ya el resultado de dichas mutaciones subsiste una pregunta.¿Cuánta señal debe conocer un operador lógico para que involucre los cambios necesarios a la salida?

Esta pregunta es importante por que enfoca el problema del arreglo lógico y

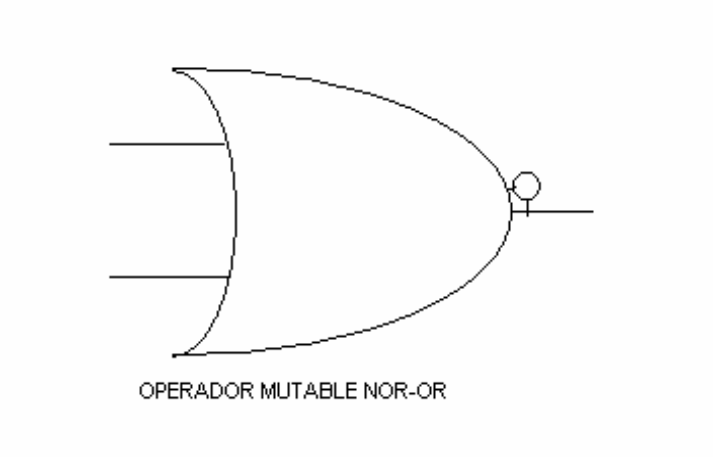


Figura 7.20: Símbolo operador lógico mutable NOR OR

es que si en la señal es necesario conocer toda trama de bits o solamente se deben conocer uno bits de información.

La solución del problema es el cambio en una cadena de bits, a no ser que la información sea completamente arbitraria y eso no ocurre; los cambios de los bits se hacen armónicamente y para ello veremos el conjunto de posibilidades de una entrada de cuatro bits como sigue.

lsb		msb
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1

1 1 0

1 1 1

Podemos observar que el cambio de los bits como es lógico se genera en los dos bits inmediatamente seguidos de el bit que nos interesa, entonces por ejemplo,para el numero 4 que es el binario 1-0-0 tenemos que el cinco es 1-0-1 esto da una pista de la solución a implementar y es que el cambio se da en los bits seguidos de el BIT que nos interesa. Este razonamiento va a ser importante a la hora de designar las entradas de la célula madre electrónica

A continuación veremos como se implementara dicho circuito y que ante cualquier entrada arbitraria debe proporcionar una salida al concatenar el problema que nos involucra.

En este punto los dos operadores mutables se deben interconectar de modo que cualquier entrada debe poder proporcionar cualquier salida, para ello veremos el siguiente arreglo lógico ya habiendo visto la simbología:

Seguidamente veremos los resultados del análisis anterior y ante cualquier entrada se podrá obtener un uno o un cero según la conveniencia. Para esto utilizaremos la tabla de verdad que dará los resultados:

La entrada 0 es el msb y la entrada 2 es el lsb este análisis se hará para tres bits, los necesarios para este diseño, las entradas ent1 y ent 0 van al operador lógico nor _ or y la entrada al operador lógico nad-nor.

Ent2 Ent1 Ent0 Bit control

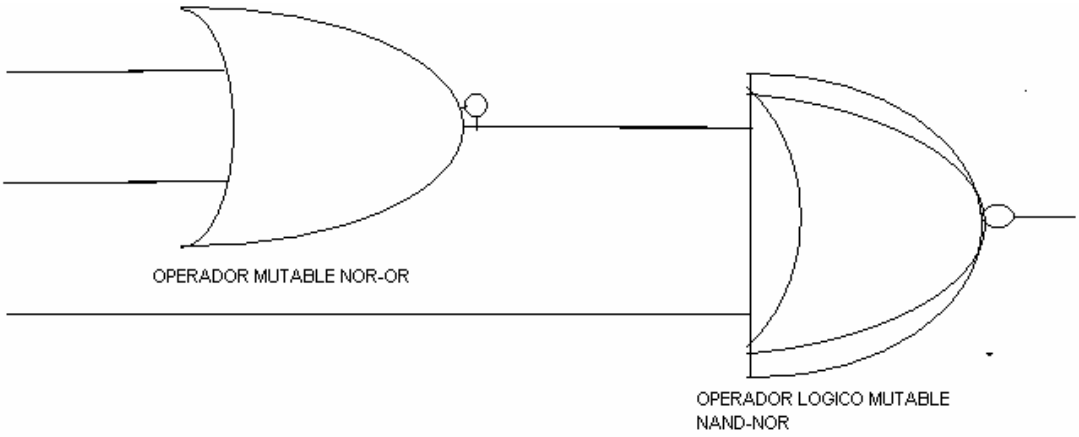


Figura 7.21: Arreglo lógico mutable , célula madre electrónica.

or_nor Op mut n-or Salida

1er

operador Bit de contro

Nand

nor Op mut

usad Salida

encontrada

Salida esperada

0	0	0	1	nor	1	1	nor	0	0
0	0	1	0	0r	0	0	nand	1	1
0	1	0	0	or	1	0	nor	0	0
0	1	1	0	or	1	1	nor	0	0
1	0	0	0	or	0	0	nand	1	1

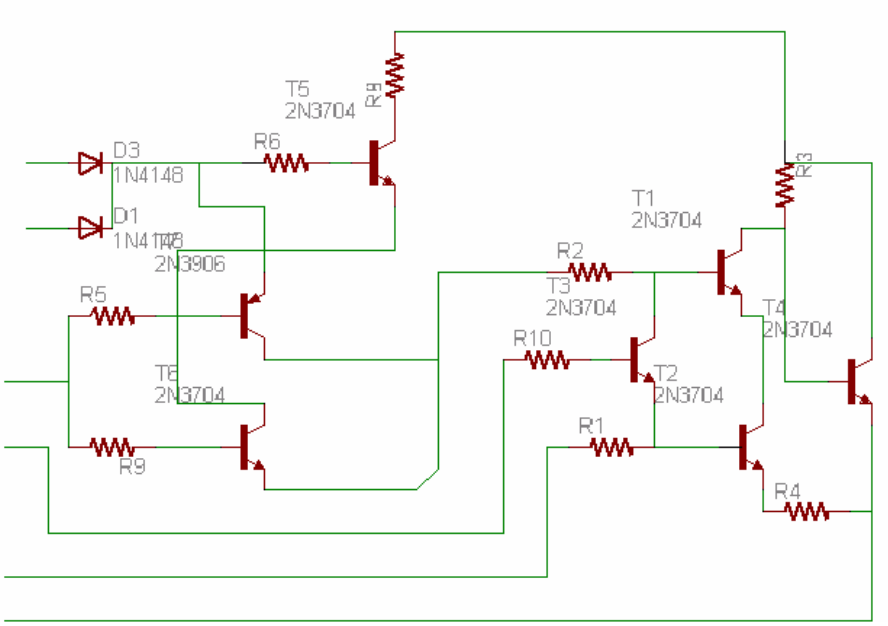


Figura 7.22: Circuito de la célula madre electrónica.

1	0	1	0	or	1	1	nor	0	0
1	1	0	1	nor	0	0	nand	1	1
1	1	1	1	nor	0	1	nor	0	0

En la tabla anterior se observa la salida esperada y la encontrada, el operador lógico implementado en cada operación y su bit de mutación y las entradas arbitrarias, este ejemplo solo se hizo con la mitad de las posibles salidas por lo que aún en cada ejemplo falta la solución inversa con la misma entrada.

Seguidamente veremos el esquema electrónico del anterior arreglo lógico que es finalmente el diseño de la célula madre electrónica:

Este esquema es el que finalmente dio solución a todas las posibilidades de mutación, no importando la clase de entrada.

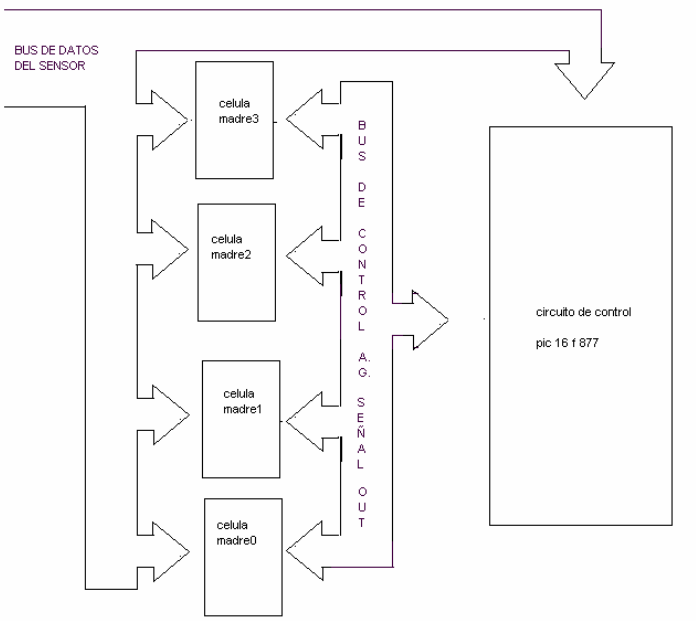


Figura 7.23: Bloque lógico mutable junto con el microcontrolador que lo controla

El siguiente paso una vez encontrado el arreglo lógico mutable es crear un bloque lógico en el F.P.G.A.

Tenemos que hallar un arreglo lógico más grande para varias entradas, en este trabajo se considerara un arreglo para cuatro bits por cada bloque lógico.

7.5.7.2. Implementación del bloque lógico mutable

En este punto se propone el uso de un arreglo lógico de cuatro células madres electrónicas cuyas entradas arbitrarias deben encontrar la solución al circuito.

Para el control de la células madres se implementara el uso de el pic 16f877 de microchip, su cantidad de puertos disponibles justifican su uso.

El siguiente diagrama en bloques es la idea de la solución.

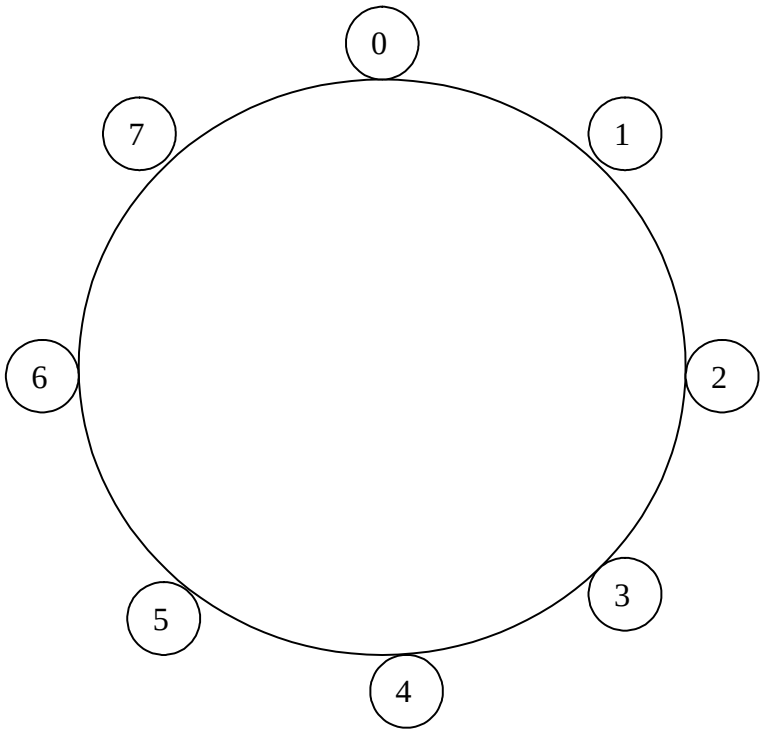


Figura 7.24: Arreglo de bits a implementan en las entradas de B.L.M.

Este sesillo esquema da una idea de cómo el microcontrolador controla el bloque lógico. Retomado lo dicho anteriormente sobre cuantas entradas utilizar para encontrar una salida a concatenar, el número de cambios de bits por trama es de cuatro bits.

Se propone una solución en la que los bits inmediatamente seguidos intervienen en la solución, utilizando un esquema de anillos para utilizarlos como entradas de el B.L.M. en las que se usaran tramas de tres bits.

En el siguiente diagrama se explica lo dicho en el párrafo anterior:

Cada uno de los números en añillo representan los bits en la entrada entonces

seguidamente se hará una tabla de bits teniendo en cuenta la información de el anillo dado para un número de cuatro bits de información que es el caso que se va a implementar:

	Entrada 2	Entrada 1	Entrada 0
Bit cero c madre 1	Bit 3	Bit 0	Bit 1
Bit uno c madre2	Bit 0	Bit 1	Bit 2
Bit dos c madre3	Bit 1	Bit 2	Bit 3
Bit tres c madre4	Bit 2	Bit 3	Bit 0

Este mismo caso nos ocupara para la trama de bits desde el bit4 hasta el bit7.

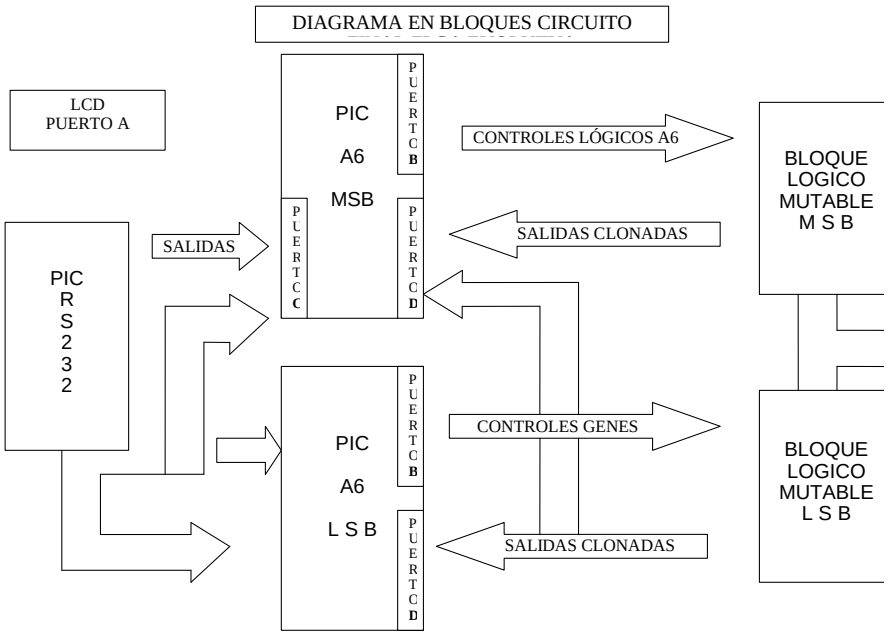
7.5.8. Implementación del FPGA evolutivo

Ahora que ya se tiene el algoritmo evolutivo, el diseño de los B.L.M, y la configuración de las entradas en la compuertas lógicas será el turno de implementar el diseño final de la fuga evolutiva . En este diseño tenemos :

Microprocesador de comunicación : se trata de un micro 16f877 que comunica el circuito serial con la cpu a través de el puerto serial rs 232 o con el sensor directamente a través de uno de los puertos c mas específicamente Rc3, comunica a los otros micros con la cpu.

Módulo lcd comunica el dispositivo con el usuario brindando información del estado de la clonación.

Micro de control del circuito evolutivo genera las posibles soluciones siendo



un total de DOCIENTAS CINCUENTA Y SEIS soluciones por entrada, para este caso se implementara dos bloque logicos mutables. En total se implementaran un total de 4096 posibles mutaciones.

En el siguiente diagrama en bloques se da el diseño implementado.

La explicación del funcionamiento del circuito evolutivo es el siguiente.

7.5.8.1. Bloque Lógico Mutable 1 (B.L.M.)

Es el circuito que contiene las célula madres cuyas salidas estan entre el bit0 y el bit3, se realiza la comunicación con el pic número uno el cual genera todas las mutaciones necesarias para encontrar la solución.

El diagrama en bloque podemos tres microcontroladores, dos de ellos son los controles de las células madres electrónicas, implementadas a través del algorit-

mo genético programado en cada pic de control, generando en cada momento cromosomas de 8 bitsl.

Estos estarán buscando la solución para concatenar, una vez esto ocurra en el micro del LSB (numero 2) envia una señal al pic de MSB (numero 1) con la solución encontrada; una vez el pic numero 1 tenga la solución completa, se comunica con el pic de comunicación y le comunica que la "replica esta hecha".

El modulo LCD constantemente esta comunicando al usuario el estado del proceso de clonación.

La corriente de colector que se implemento en los transistores de saturación aseguran un retardo de 50nseg, tiempo suficiente para asegurar que no hayan desajustes en la concatenación hecha por el micro que esta trabajando a 4Mhz.

Finalmente el ciclo de instrucción en el pic es de 1 micro segundo, velocidad del proceso suficientemente necesaria.

A continuación se estudiara la forma en que los dispositivos activos y pasivos interviene en el diseño.

Tomando las curvas características del 2n2222 ; y enfocándonos en el esquema que acontinuacion vemos.

Retomando un poco lo que es diseño que es un sistema digital:

1. la impedancia de entrada debe ser alta.
2. admitancia de salida parámetro que tiende a cero.
3. consumo de corriente lo mas bajo posible, para evitar recalentamiento

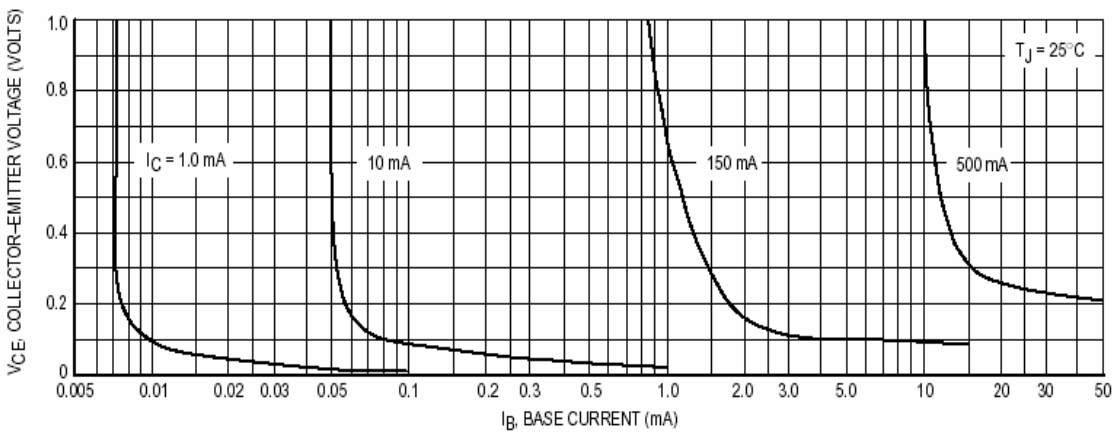


Figure 4. Collector Saturation Region

Figura 7.25: Curvas de saturación para el 2n2222.

que pueden degenerar losa componentes de el circuito.

4. la rapidez de respuesta debe ser otro parámetro a tener en cuenta.

5. debe ser sencillo ala hora de implantarse.

Ya habiendo dado unos parámetros de diseño podemos empezar el análisis.

Para este diseño se ha escogido la corriente se saturación lo mas pequeña posible dentro del rango que el dispositivo otorga en sus hojas características. Por este hecho se tomara como referencia la una corriente igual a 1mA que es una de la curvas que podemos observar.

La recta de carga para el circuito en este caso seria la siguiente.

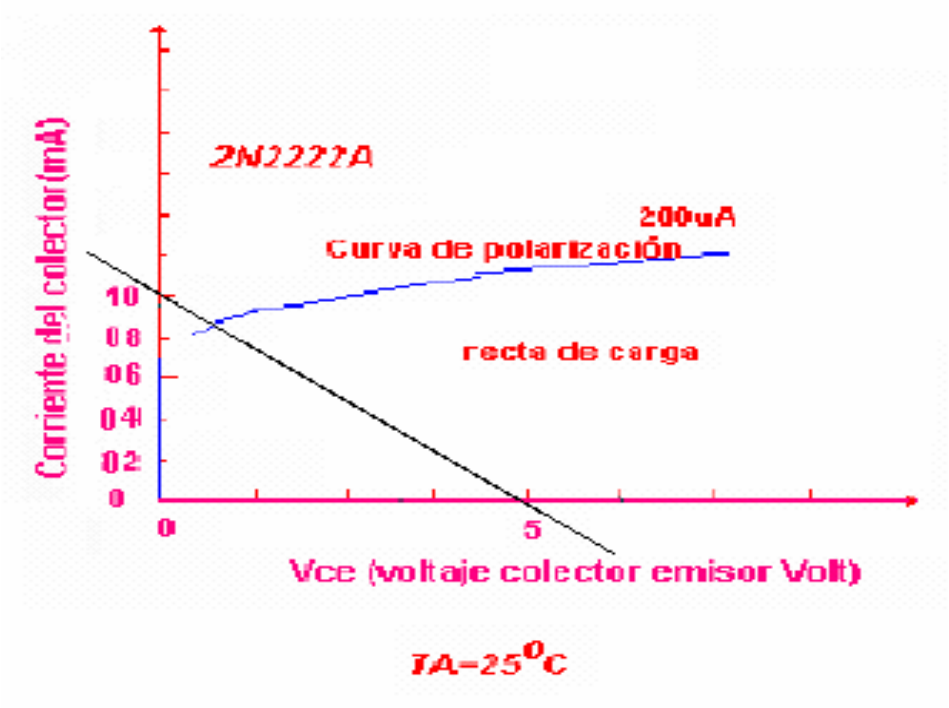


Figura 7.26: Recta de carga para el transistor en saturación.

CAPÍTULO 8

CONCLUSIONES Y RECOMENDACIONES

El índice de viscosidad es una variable de difícil medición, se puede obtener su medición mediante variables que intervienen en el proceso generalmente no lineal.

Al identificar la no linealidad del proceso se puede establecer una aproximación de su comportamiento real, con base en la metodología de clonación que se adapta a su dinámica.

Para una acertada selección de variables no es sufciciente los modelos teóricos, fué necesario apoyarse en los procesos y realizar pruebas para determinar el comportamiento del patrón a replicar.

La programación genética es una herramienta computacional que permite diseñar el modelo, realizar el entrenamiento, y la validación a través del modelo, comparando las salidas del programa con los valores dados por el sensor real.

La metodología de clonación es aplicable a modelos que representan sistemas no lineales, como el problema solucionado en esta investigación.

Se logró replicar el sensor por mapeo genético evolutivo con base en algoritmos y programación genética.

Haciendo uso de las técnicas de clonación artificial y programación genética se codificó la estructura funcional del sensor, comprobándose que es posible construir

sistemas capaces de solucionar problemas con gran calidad.

Se pueden realizar acercamientos al comportamiento dinámico de los sensores mediante el análisis de datos de salida de los mismos.

El diseño experimental es un proceso fundamental para el desarrollo de la estructura funcional del sensor y de la clonación artificial, por lo tanto , se recomienda profundizar en este tema para futuras aplicaciones.

La siguiente etapa de la investigación es desarrollar metodologías de diseño para clonación en circuitos integrados, cuyo desarrollo se puede soportar utilizando FPGA.

CAPÍTULO 9

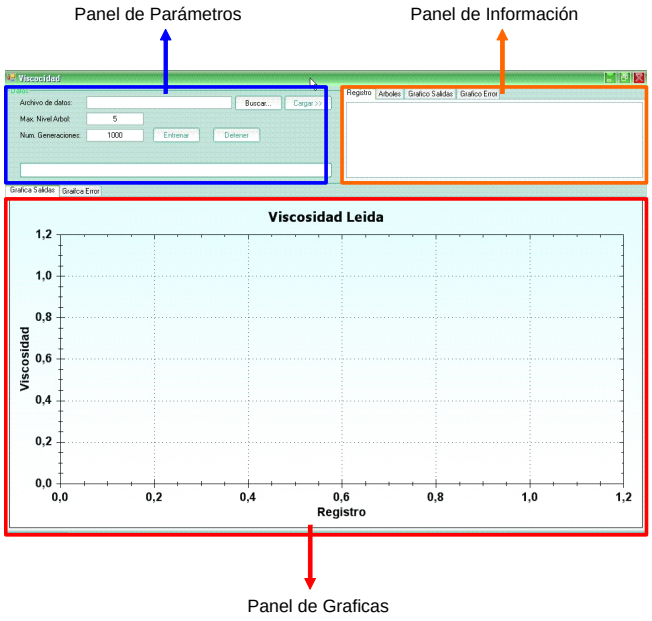
MANUAL DEL USUARIO

Al iniciar la aplicación se presenta un formulario que esta dividido en tres paneles:

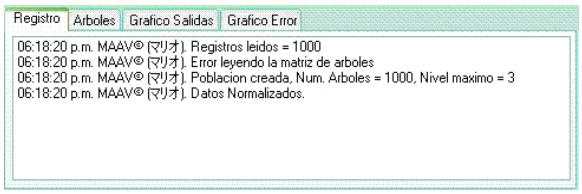
Para comenzar a usar la aplicación, lo primero que se debe hacer es establecer los parámetros iniciales del entrenamiento, entre los cuales esta, la ruta del archivo de Excel que contiene los datos históricos de las variables que se usan en el modelo para calcular la viscosidad (Temperatura, Flujo, Nivel y Presión). Esta ruta debe ser introducida en el cuadro de texto que contiene la etiqueta "Archivo de Datos", otra forma de introducir la ruta del archivo es presionando el botón Buscar, el cual presenta un dialogo de búsqueda de archivos, allí se puede buscar y seleccionar el archivo que contiene los datos, luego al dar en clic en aceptar, la ruta queda registrada en el cuadro de texto correspondiente:

De esta forma, se activa el botón para poder cargar los datos del archivo a la memoria de la aplicación. Otro de los parámetros iniciales de la aplicación es el máximo nivel de los árboles al cual se quiere restringir la creación de los mismos (la profundidad), el número de generaciones que va a iterar la aplicación para poder ejecutar el algoritmo genético.

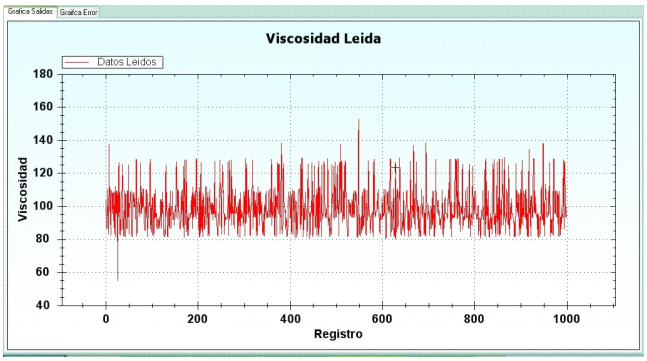
Una vez cargado el archivo de datos, el panel de información, en la pestaña


Panel de Parámetros
Panel de Información
Viscosidad Leida
Viscosidad
Registro
0,0
0,2
0,4
0,6
0,8
1,0
1,2
Panel de Graficas


Datos
Archivo de datos:
C:¥Viscosidad Final¥muestra2.xls
Buscar...
Cargar >>
Max. Nivel Arbol:
5
Num. Generaciones:
1000
Entrenar
Detener



Mensajes de la aplicación



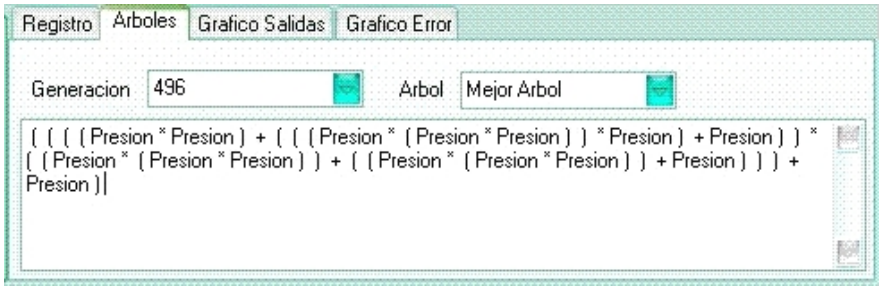
Grafica de los datos leídos

"Registro" muestra los mensajes correspondientes a los eventos realizados, informa el número de datos leídos, la población creada e informa en general el resultado de los procesos. También, una vez cargado el archivo, se presenta una grafica de los datos leídos en el panel de graficas:

Cuando se han definido todos los parámetros y se ha cargado el archivo de datos, se activa el botón de Entrenar, el cual permite iniciar el proceso de iteración del algoritmo. El progreso del algoritmo es informado mediante la barra de progreso situada en la parte inferior del panel de datos, a medida que avanza el proceso, la barra indicadora avanza en la misma medida para de esta forma indicar que el entrenamiento ha finalizado cuando la barra se halla llenado, también aparece un dialogo informando la terminación del proceso:

Cuando el entrenamiento finaliza, se activan el resto de pestañas en el panel

Indicador de Progreso

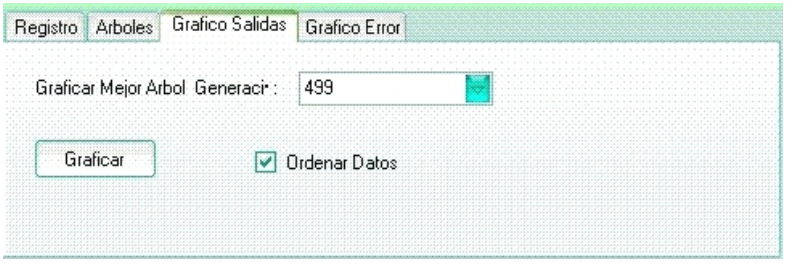


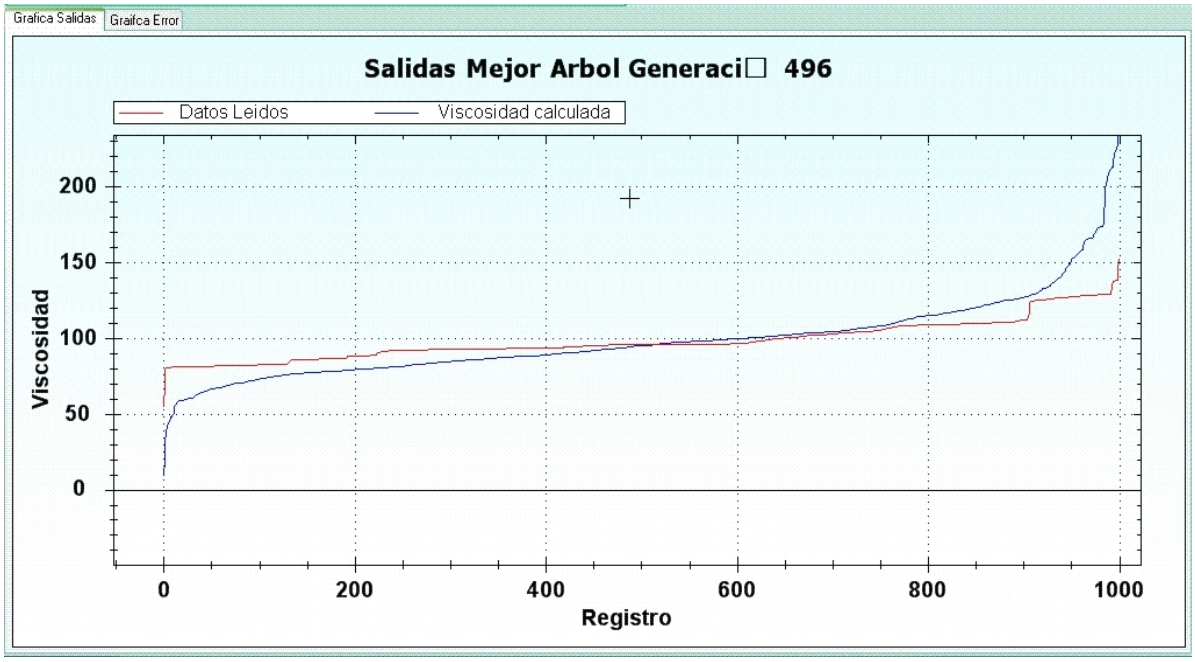
d e información, desde allí, se puede observar como fue todo el proceso de entrenamiento, cuales fueron los árboles generados en cada una de las iteraciones y también cual fue el mejor de ellos:

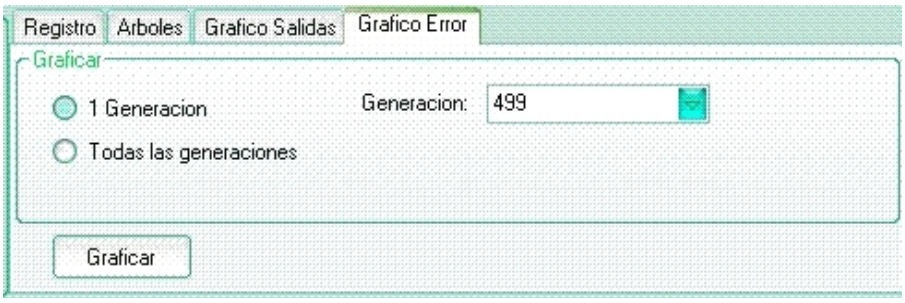
También se permite en la pestaña "Grafico Salidas", graficar cada uno de los mejores árboles en cada generación con la opción de graficarlos en el orden en que se leyeron los datos del archivo o ordenar esos datos en función del eje y, el de la viscosidad:

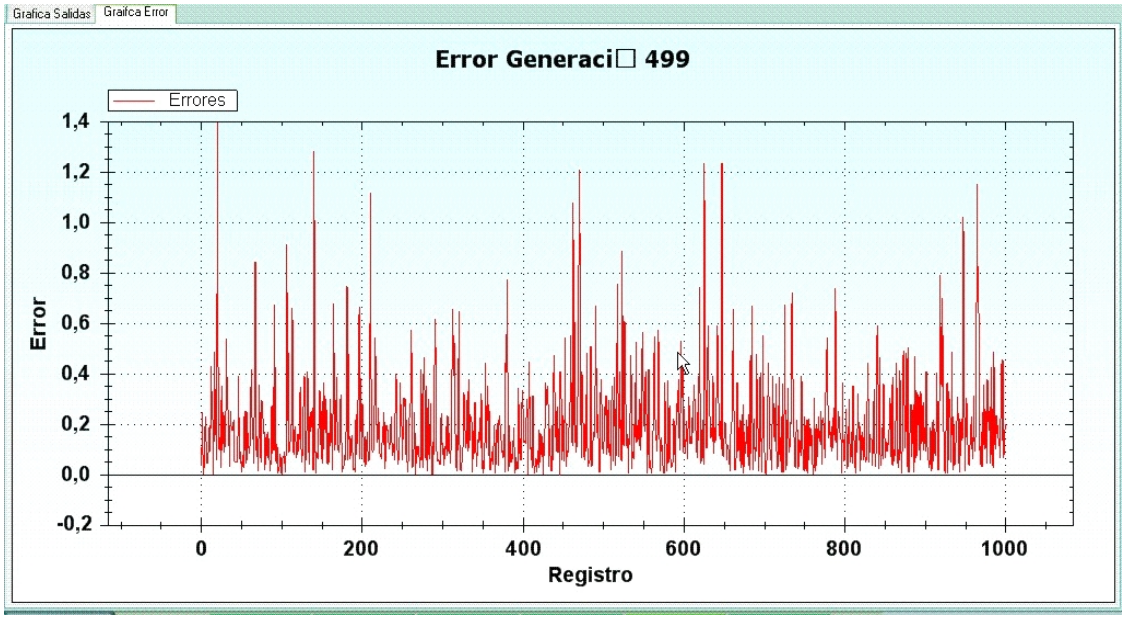
El árbol graficado se compara contra la curva de los datos de viscosidad leídos del archivo de datos, los valores del árbol de la generación se grafican en color azul y los leídos del archivo se grafican en color rojo:

En la ultima pestaña del panel de información se puede escoger el error gen-


Registro
Arboles
Grafico Salidas
Grafico Error
Graficar Mejor Arbol Generaci :
499
Graficar
Ordenar Datos

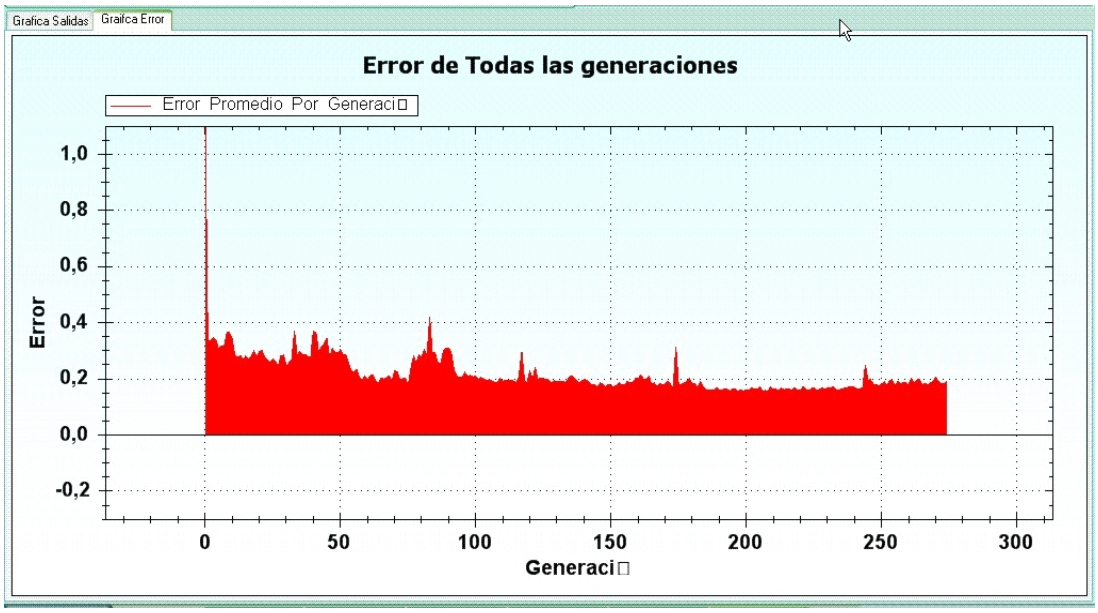

Grafica Salidas
Graifca Error
Salidas Mejor Arbol Generaci 496
Datos Leidos
Viscosidad calculada
Viscosidad
200
150
100
50
0
0
200
400
600
800
1000
Registro





erado en cada una de las generaciones y graficarlo o simplemente graficar un error general de todo el entrenamiento, este error es el error global calculado entre las salidas de los mejores árboles y los datos de viscosidad leídos del archivo de datos:

De esta forma, al finalizar el algoritmo se ha obtenido una solución aceptable para el calculo de la viscosidad, y si no fuera si, el entrenamiento puede ser iniciado nuevamente desde el punto en el que se termino el entrenamiento anterior, para así poder partir de esa solución encontrada hasta ese momento.


Grafica Salidas
Graifca Error
Error de Todas las generaciones
Error Promedio Por Generaci
1,0
0,8
0,6
0,4
0,2
0,0
-0,2
Error
0
50
100
150
200
250
300
Generaci

CAPÍTULO 10

REFERENCIAS Y BIBLIOGRAFÍA

[MUÑOZ 98] MUÑOZ, A.F., Aplicación de los algoritmos genéticos en la identificación y control de bioprocesos por clonación artificial. IEEE Transactions on Systems, Man, and Cybernetic V 19 No. 2 58-76, 1998

[MUÑOZ 98] MUÑOZ, A.F., Tecnología de clonación artificial on-line de sensores y controladores. Oficina Internacional de Invenciones, Patentes y Marcas, República de Cuba. Registros No. 7-789735, 2000

[MUÑOZ 98] MUÑOZ, A.F., Equipo de control genético de la composición en medios continuos on-line. Oficina Internacional de Invenciones, Patentes y Marcas, República de Cuba. Registros No. 7-789734, 2001.

[ADAM 94] ADAMI, C., Learning and complexity in genetic autoadaptive systems. California Institute of Technology, 1994.

[ADEL 95] ADELI, H., Machine Learning: Neural Networks, Genetic Algorithms, and Fuzzy Systems. John Wiley and Sons, Inc, 1995.

[AGUI 99] AGUILAR José y Pablo Miranda, Resolution of the Left Ventricle 3D Reconstruction Problem using Approaches based on Genetic Algorithms for MultiObjectives Problems. En: Proceeding of the 1999 Conference on Evolutionary Computation, pags. 913-920, Washington, USA, 1999.

[AIZA 97] AIZAWA, A., In Foundations of Genetic Algorithms. Morgan Kaufmann, 1997.

[ANDR 94] ANDRE, D., Evolution of mapmaking: Learning, planning, and memory using genetic programming. In Proceedings of the First IEEE Conference on Evolutionary Computation, Volume 1. Piscataway, NJ: IEEE Service Center, 1994.

[BOND 88] BOND A.H., Gasser L. (eds.) Readings in Distributed Artificial Intelligence. Morgan Kaufmann. 1988.

[BRAD 97] BRADSHAW, J. (ed.). Software Agents. AAAI Press/ The MIT Press. 1997.

[CHAK 91] CHAKRABORTY, U. K., & Dastidar, D. G., Artificial genetic search in the nqueens problem. Proceedings of the International AMSE Conference on Signals, Data & Systems, 1991.

[CHAK 93] CHAKRABORTY, U. K., & Dastidar, D. G., Using reliability analysis to estimate the number of generations to convergence in genetic algorithms. Information Processing Letters, 1993.

[CHAK 95] CHAK, C. K., & Feng, G., Accelerated genetic algorithms: Combined with local search techniques for fast and accurate global search. In 1995 IEEE International Conference on Evolutionary Computation, Volume 1. IEEE Service Center, 1995.

[DAVI 91] DAVIS, L., Handbook of Genetic Algorithms. New York: Van Nos-

trand Reinhold, 1991.

[DORI 93] DORIGO, M., & Maniezzo, V., Parallel Genetic Algorithms: Theory and Applications. Amsterdam, IOS Press, 1993.

[DORS 94] DORSEY, R. E., Johnson, J. D., & Mayer, W. J. The genetic adaptive neural network training (GANNT) algorithm for generic feedforward artificial neural systems (Technical Report). University, MS: The University of Mississippi, 1994.

[FOX 91] FOX, B. R., & McMahon, M. B., Foundations of Genetic Algorithms. Morgan Kaufmann, 1991.

[FURU 97] FURUHASHI, T., Matsushita, S., & Tsutsui, H. (1997). Evolutionary fuzzy modeling using fuzzy neural networks and genetic algorithm. In Proceedings of 1997 IEEE International Conference on Evolutionary Computation, IEEE, 1997.

[GOLD 89] GOLDBERG, D.E., Genetic Algorithms in Search, Optimization & Machine Learning. Reading: Addison-Wesley, 1989.

[GOLD 97] GOLDBERG, D. E., Zakrzewski, K., Chang, C., Gallego, P., Sutton, B., Miller, B. L., & Cant'u- Paz, E. (1997). Genetic algorithms: A bibliography (IlliGAL Report No. 97002). Urbana: University of Illinois, Illinois Genetic Algorithms Laboratory.

[HEIT 00] HEITKOETTER, J. and D. Beasley, The Hitch-Hiker's Guide to Evolutionary Computation: A List of Frequently Asked Questions (FAQ).

USENET: comp.ai.genetic. Disponible en http://www.cs.bham.ac.uk/Mirrors/ftp.de.uu.net/EC/clife/w
2000.

[HOFF 91] HOFFMEISTER, F., & Back, T., Genetic algorithms and evolution strategies—Similarities and differences. SpringerVerlag, 1991.

[HOLL 87] HOLLAND, J. H. (1987). Genetic algorithms and classifier systems: Foundations and future directions. Proceedings of the Second International Conference on Genetic Algorithms. Erlbaum Associates, 1987

[HOLL 92] HOLLAND, J.H., Adaptation in Natural and Artificial Systems. Second edition. Cambridge: MIT Press, 1992

[KARR 92] KARR, C. L.. Artificial Intelligence in RealTime Control. Pergamon Press. 1992.

[MITC 02] MITCHELL, M. An Introduction To Genetic Algorithms. Eight edition. Cambridge: MIT Press, 2002.

[PARK 95] PARK, Y. J., Cho, H. S., & Cha, D. H., Genetic algorithm based optimization of fuzzy logic controller using characteristic parameters. In 1995 IEEE International Conference on Evolutionary Computation, IEEE Service Center, 1995.

[ROTA] Gian-Carlo Rota – Pensamientos Indiscretos – traducción: A. Martín – A. Villaveces

---

REFERENCIAS

[1] A.F. Muñoz M, Çloning process for genetic algoritms:part I, Fundamentals", University of Havana, 15 (2), 1998, pp.58-69.

[2] A.F. Muñoz M, Çloning process for genetic algoritms:part II, Research Topics", University of Havana, 15 (4), 1998, pp.170-181.

[3] Downsland, Kathryn, "Simulated Annealing", en Modern Heuristic Techniques for Combinatorial Optimization Problems, editado por Colin R. Reeves, John Wiley & Sons, 1993, pp. 20-69.

[4] Glover, F. y M. Laguna, Tabu Search, Kluwer Academic Publishing, 1997.

[5] Glover, F., "Future Paths for Integer Programming and Links to Artificial Intelligence", Computers and Operations Research, No. 5, 1986, pp. 553- 549.

[6] Goldberg, D., Genetic Algorithms in Search, Optimization and Machine Learning, Addison Wesley, 1989.

[7] Holland, J. H., Adaptation in Natural and Artificial Systems, 2a ed., MIT Press, 1992.

[8] Kirkpatrick, S., C. Gelatt y M. Vecchi, "Optimiztion by Simulated Annealing", Science, No. 220, 1983, pp. 671-679.

[9] Marsden, J., y A. Tromba, Cálculo Vectorial, Fondo Educativo Interamericano, 1981.

[10] Metropolis, N., A. Rosenbluth, A. Teller y E. Teller, "Equations of State Calculations by Fast Computing Machines", The Journal of Chemical Physics, Vol. 21, No. 6, 1995, pp. 1087-1092.

[11] Reeves, C. (editor), Modern Heuristic Techniques for Combinatorial Problems, John Wiley & Sons, 1993.

[12] Shoup, T., y Farrokh Mistree, Optimization Methods with Applications for Personal Computers, Prentice Hall, 1987.

[13] Smith-Keary, P., Genetic, Structure and Function, Macmillan Press, 1979.

[14] Winston, W. L., Introduction to Mathematical Programming, Applications and Algorithms, 2a ed., Duxbury Press, 1995.

[15] Zadeth, L., Fuzzy Logic and Approximate Reasoning, Synthese 30, 1975, pp. 407-428.