UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

CENTRO UNIVERSITARIO NEZAHUALCÓYOTL

LICENCIATURA EN INGENIERÍA EN SISTEMAS INTELIGENTES

MANUAL PARA PRÁCTICAS DEL LABORATORIO DE CÓMPUTO

ASIGNATURA:

ALGORITMOS GENÉTICOS

ELABORARÓN:

DRA. DORICELA GUTIÉRREZ CRUZ
M. en C. YAROSLAF AARÓN ALBARRÁN FERNÁNDEZ
DRA. CARMEN LILIANA RODRÍGUEZ PÁEZ

CU NEZAHUALCÓYOTL

MANUAL PARA PRÁCTICAS DEL LABORATORIO DE CÓMPUTO PARA LA ASIGNATURA DE ALGORITMOS GENÉTICOS

IDENTIFICACIÓN DE LA UNIDAD DE APRENDIZAJE

<table>
<tr><td colspan="9">Espacio académico: CENTRO UNIVERSITARIO NEZAHUALCÓYOTL</td></tr>
<tr><td colspan="4">Programa educativo INGENIERÍA EN SISTEMAS INTELIGENTES</td><td colspan="5">Área de docencia: HERRAMIENTA PARA LOS SISTEMAS INTELIGENTES</td></tr>
<tr><td colspan="2">Aprobación de los HH Consejos Académico y de Gobierno</td><td colspan="2">Fecha: SEPTIEMBRE 2018</td><td colspan="5">Programa elaborado por: Doricela Gutiérrez Cruz, Yaroslaf Aarón Albarrán Fernández y Carmen Liliana Rodríguez Páez</td></tr>
<tr><td colspan="4">Nombre de la unidad de aprendizaje: ALGORITMOS GENÉTICOS</td><td colspan="5">Fecha de elaboración: agosto 2018</td></tr>
<tr><td>Clave</td><td>Horas de Teoría</td><td>Horas de Práctica</td><td>Total de horas</td><td>Créditos</td><td>Área curricular:</td><td>Carácter de la unidad de aprendizaje</td><td>Núcleo de formación</td><td>Modalidad</td></tr>
<tr><td>L41657</td><td>3</td><td>2</td><td>5</td><td>8</td><td>HERRAMIENTAS PARA LOS SISTEMAS INTELIGENTES</td><td>Obligatoria</td><td>INTEGRAL</td><td>ESCOLARIZADA CON ADMINISTRACIÓN FLEXIBLE DE LA ENSEÑANZA</td></tr>
<tr><td colspan="3">Prerrequisitos (Conocimientos previos):

PROGRAMACIÓN</td><td colspan="3">Unidad de aprendizaje antecedente:

NINGUNA</td><td colspan="3">Unidad de aprendizaje consecuente:

NINGUNA</td></tr>
<tr><td colspan="9">Programas en los que se imparte: LICENCIATURA DE INGENIERÍA EN SISTEMAS INTELIGENTES</td></tr>
</table>

EL PRESENTE MANUAL DE PRÁCTICAS HA SIDO AVALADO EN EL MES DE SEPTIEMBRE DE 2018 POR:

ÍNDICE

UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MÉXICO

DIRECTORIO

Dr. en Ed. Alfredo Barrera Baca

RECTOR

M. en S. P. María Estela Delgado Maya

SECRETARIA DE DOCENCIA

Dr. en C.I.Amb. Carlos Eduardo Barrera Díaz

SECRETARIO DE INVESTIGACIÓN Y ESTUDIOS AVANZADOS

Dr. en C.S. Luis Raúl Ortiz Ramírez

SECRETARIO DE RECTORÍA

Dr. en A. José Edgar Miranda Ortiz

SECRETARIO DE DIFUSIÓN CULTURAL

M. en Com. Jannet Socorro Valero Vilchis

SECRETARIA DE EXTENSIÓN Y VINCULACIÓN

M. en E. Javier González Martínez

SECRETARIO DE ADMINISTRACIÓN

Dr. en C.C. José Raymundo Marcial Romero

SECRETARIO DE PLANEACIÓN Y DESARROLLO INSTITUCIONAL

M. en L.A. María del Pilar Ampudia García

SECRETARIA DE COOPERACIÓN INTERNACIONAL

Dra. en C.S. y Pol. Gabriela Fuentes Reyes

ABOGADA GENERAL

Lic. en Com. Gastón Pedraza Muñoz

DIRECTOR GENERAL DE COMUNICACIÓN UNIVERSITARIA

M. en R.I. Jorge Bernaldez García

SECRETARIO TÉCNICO DE LA RECTORÍA

M. en A.P. Guadalupe Santamaría González

DIRECTORA GENERAL DE CENTROS UNIVERSITARIOS Y UAP

M. en A. Ignacio Gutiérrez Padilla

CONTRALOR UNIVERSITARIO

CENTRO UNIVERSITARIO NEZAHUALCÓYOTL

DIRECTORIO

M. en D. Juan Carlos Medina Huicochea
Encargado del despacho de C.U. Nezahualcóyotl

M. en C. José Antonio Castillo Jiménez
Subdirector Académico

Lic. en E. Ramón Vital Hernández
Subdirector Administrativo

Dra. en C. S. María Luisa Quintero Soto

Coordinadora de Investigación y Estudios Avanzados

Lic. en A. E. Víctor Manuel Durán López
Coordinador de Planeación y Desarrollo Institucional

Dr. en R.I. Rafael Alberto Duran Gómez
Coordinador de la Licenciatura en Comercio Internacional

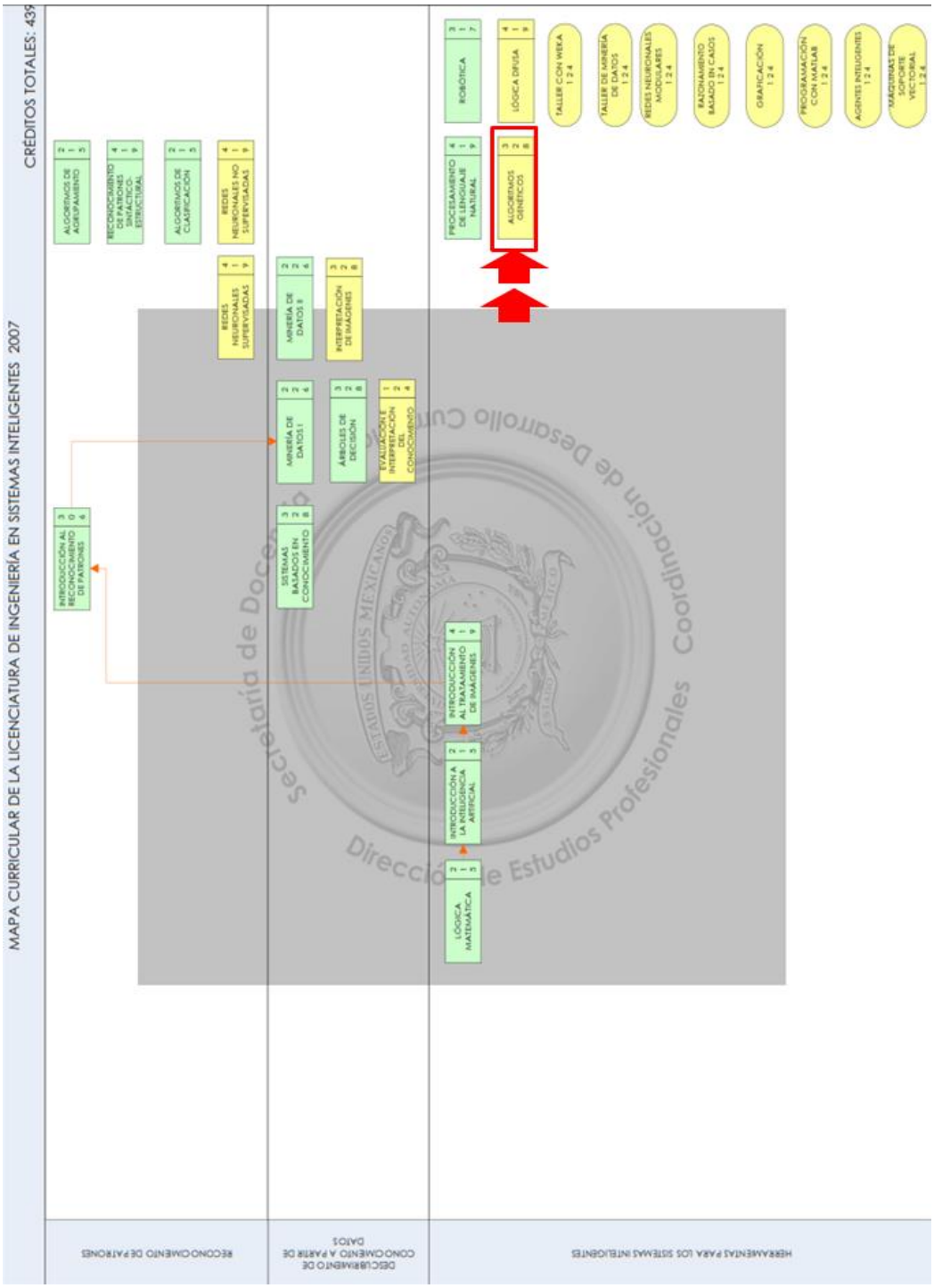
Mtra. Gabriela Kramer Bustos
Coordinadora de la Licenciatura en Educación para la Salud

Dra. Ricardo Rico Molina
Coordinador de la licenciatura en Ingeniería en Sistemas Inteligentes

D. En U. Noé Gaspar Sánchez
Coordinador de Ingeniería en Transporte

M. En C.C. Erick Nicolás Cabrera Álvarez
Coordinador de Licenciatura en Seguridad Ciudadana

Ubicación de la asignatura de Algoritmos Genéticos, dentro del programa de la Lic. en Ing. en Sistemas Inteligentes.



SECUENCIA DIDÁCTICA

PRÁCTICA 1
INTRODUCCIÓN A LOS ALGORITMOS GENÉTICOS Y BASES EVOLUTIVAS

PRÁCTICA 2
CODIFICACIÓN DE LA POBLACIÓN

PRÁCTICA 3
OPERADOR DE SELECCIÓN

PRÁCTICA 4
OPERADOR DE CRUCE DE UN PUNTO

PRÁCTICA 5
OPERADOR DE CRUCE DE DOS PUNTOS

PRÁCTICA 6
OPERADOR DE CRUCE UNIFORME

PRÁCTICA 7
OPERADOR DE MUTACIÓN

PRÁCTICA 8
ALGORITMO GENÉTICO SECUENCIAL

PRÁCTICA 9
ALGORITMO GENÉTICO ELITISTA

PRÁCTICA 10
ALGORITMO GENÉTICO SELECCIÓN POR TORNEO

PRÁCTICA 1

INTRODUCCIÓN A LOS ALGORITMOS GENÉTICOS Y BASES EVOLUTIVAS

OBJETIVO

- Introducir al alumno a los conceptos básicos de los algoritmos genéticos

INTRODUCCIÓN

Los Algoritmos Genéticos son métodos adaptativos, generalmente usados en problemas de búsqueda y optimización de parámetros, basados en la reproducción sexual y en el principio de supervivencia del más apto. Más formalmente "los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural.

Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas" La lógica es un lenguaje que permite expresar conocimiento y razonar a partir de ciertas expresiones deducir otras (deducción), sus principales características son conformadas por: sintaxis y semántica bien definidas, así como también reglas de inferencia.

La idea de evolución por si sola es un concepto abierto, es una descripción mecánica de cambio que no dice nada acerca del motor o la fuerza creadora que subyace a la transformación. La evolución biológica es el proceso histórico de transformación de unas especies en otras especies descendientes, e incluye la extinción de la gran mayoría de las especies que han existido.

LUCHA POR LA SUPERVIVENCIA.

Para Darwin "la vida comienza con la lucha por la existencia", es decir, por la supervivencia que busca todo ser vivo, ya sea de una u otra manera y del mismo modo, protegerse para no ser exterminado por un ser más fuerte. Esta es una ley natural que ha perdurado a través de los siglos y que rige toda la naturaleza, porque a fin de cuentas quienes logran sobrevivir son los más aptos o los más fuertes. Sin embargo, todo ser vivo en algún momento de su existir es exterminado por otro más fuerte que él, porque éste es un proceso natural, una cadena necesaria para que haya un equilibrio en la naturaleza. Pero las luchas más severas se dan entre los individuos de la misma especie, porque poseen rasgos muy parecidos tanto en hábitos como en complexión, y más porque entrarán en competencia por la comida o la residencia. De igual manera corren el mismo destino, tienen que escapar de los mismos depredadores e ir en búsqueda de las mismas presas, esto los llevará a una confrontación a muerte, donde sobrevivirá el más fuerte.

INSPIRACIONES TEORICAS.

EVOLUCION DE DARWIN. En 1859, Darwin publico los resultados del trabajo que había realizado durante los años precedentes en un libro titulado "On the Origin of Species by Means of Natural Selection". El éxito de este libro permite afirmar que fue en este momento cuando nació la "teoría de la evolución por medio de la selección natural".

La estructura de la teoría de la evolución por selección natural en sus escritos se apoya en tres puntos básicos: 1. Los descendientes heredan los caracteres de los progenitores de generación en generación. 2. En el proceso de la herencia ocurren variaciones espontáneas que son por azar o ciegas. Se habla de variaciones por azar o ciegas en un doble sentido. Por una parte, no se pueden determinar sus causas. Por otra parte, dichas variaciones no están orientadas a una mejor adaptación del organismo al medio, es decir, no hay ninguna orientación a priori en ellas. 3. Existe reproducción diferenciada en los individuos de una población. El motivo es doble: o bien algunos individuos poseen mayor fertilidad que otros, o bien están mejor adaptados al medio. Mejor adaptación al entorno se traducirá en una mayor supervivencia y, consiguientemente, en una mayor descendencia.

SELECCIÓN DE WEISMANN. El biólogo alemán August Weismann (1834-1914) refutó experimentalmente la hipótesis de la herencia de los caracteres adquiridos y demostró que había una suerte de permanencia de las características genéticas que van pasando inalteradas de padres a hijos. Sobre esta base, sugirió la distinción entre el "plasma germinal", que se transmite de generación en generación, y el "plasma somático", que constituye el cuerpo de los organismos. Ambos factores serían independientes, de modo que cualquier modificación que sufriera el plasma somático no sería transmitida a los descendientes. Las características somáticas adquiridas por un individuo no serían heredables dado que, como las células que forman parte del cuerpo de los organismos son diferentes de las células sexuales (los óvulos y los espermatozoides), los cambios en el cuerpo no pueden transferirse a las células germinales y, por lo tanto, no pasarían a la siguiente generación.

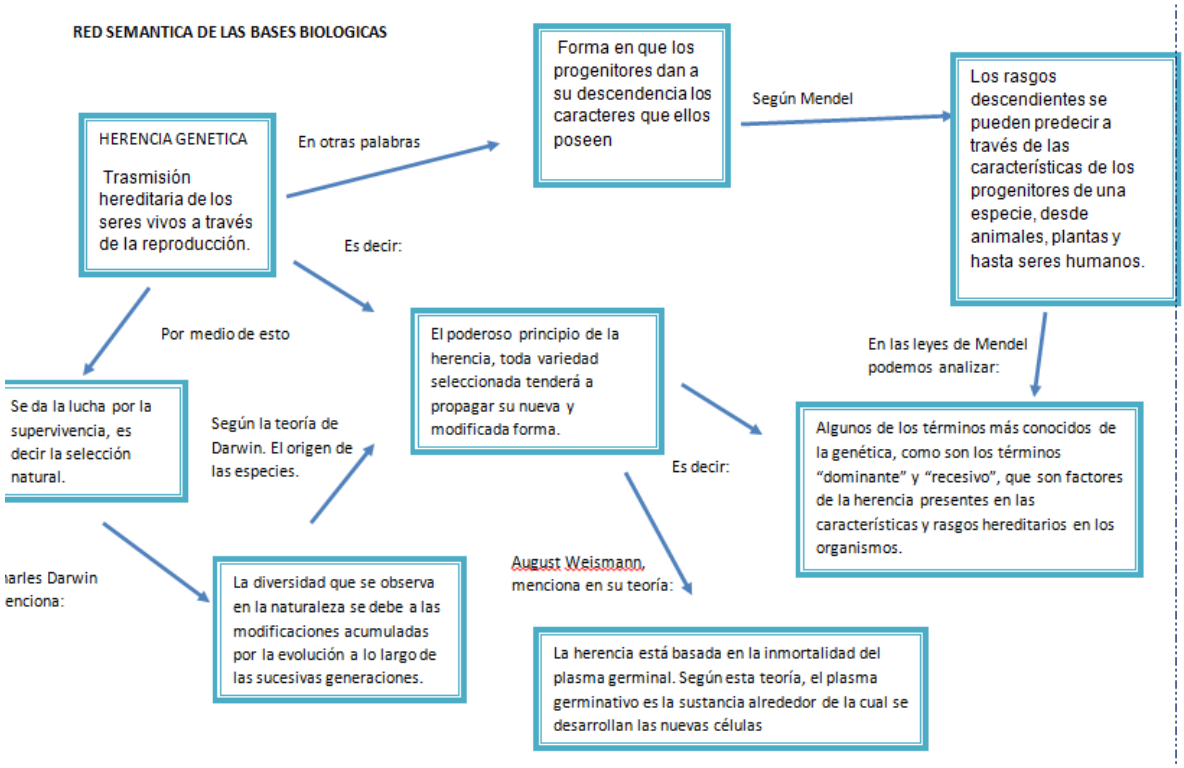
GENETICA DE MENDELL. Gregor Mendel, considerado el padre de la genética, fue un monje austriaco cuyos experimentos sobre la transmisión de los caracteres hereditarios se han convertido en el fundamento de la actual teoría de la herencia. Las leyes de Mendel explican los rasgos de los descendientes, a partir del conocimiento de las características de sus progenitores.

1. Primera Ley de Mendel dice que, si se cruzan dos padres de raza pura con diferentes rasgos, la primera generación tendrá similitudes entre sí y guardará un carácter del padre con el alelo dominante.
2. Segunda ley dice que, los factores genéticos se separan de cada uno de los padres en alelos individuales que se juntarán para procrear una descendencia con las características de la primera generación, pero en la segunda generación, se manifiestan nuevos rasgos genéticos

observados en los padres, pero unidos de manera aleatoria en la descendencia de la primera generación.
3. Tercera ley de Mendel dice que, además existen rasgos generados de forma independiente, a través de cromosomas alejados que no intervienen entre sí, y al igual que en la segunda ley, esta tercera de las leyes de Mendel se manifiesta con más claridad en la segunda generación de individuos.

DESARROLLO

El alumno diseñara una red semántica con los conceptos revisados en esta introducción. Tal como se puede apreciar en el ejemplo.



Conclusiones
Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico
Básico:

1. Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones,
ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España
2. Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.
3. Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.
4. Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

-

Bibliografía Complementaria

1. An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press
2. - Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley
3.- Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.
4.- Koza, J.R., "Genetic Programming. On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992, 819 p.

PRÁCTICA 2
CODIFICACIÓN DE LA POBLACIÓN

OBJETIVO

- El alumno conocerá la codificación binaria que representa la ristra.
- Desarrollará un programa que emule la codificación binaria y su posterior valor real.

INTRODUCCIÓN

El Algoritmo Genético Simple, también denominado Canónico, se representa en la figura 1. Como se puede ver a continuación, se necesita una codificación o representación del problema, que resulte adecuada al mismo. Además, se requiere una función de ajuste o adaptación al problema, la cual asigna un número real a cada posible solución codificada. Durante la ejecución del algoritmo, los padres deben ser seleccionados para la reproducción, a continuación, dichos padres seleccionados se cruzarán generando dos hijos, sobre cada uno de los cuales actuará un operador de mutación. El resultado de la combinación de las anteriores funciones será un conjunto de individuos (posibles soluciones al problema), los cuales en la evolución del Algoritmo Genético formarán parte de la siguiente población.

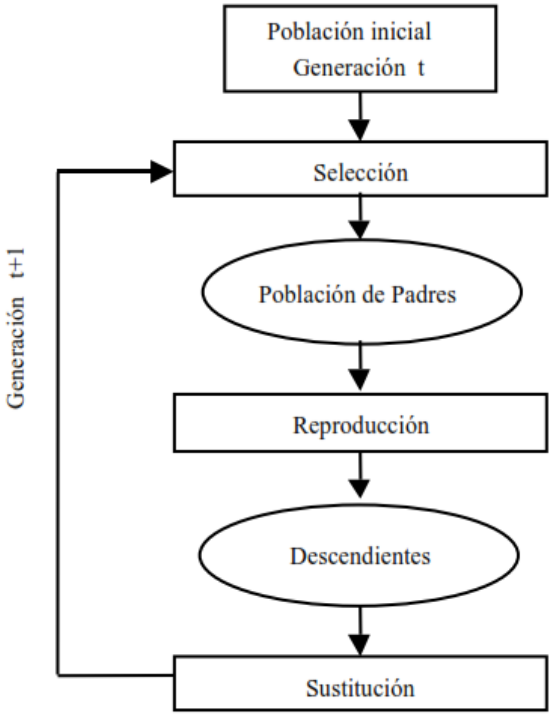


Figura 1. Estructura general del Algoritmo Genético

Codificación

Se supone que los individuos (posibles soluciones del problema), pueden representarse como un conjunto de parámetros (que denominaremos genes), los cuales agrupados forman una ristra de valores (a menudo referida como cromosoma). Si bien el alfabeto utilizado para representar los individuos no debe necesariamente estar constituido por el [0,1], buena parte de la teoría en la que se fundamentan los Algoritmos Genéticos utiliza dicho alfabeto.

En términos biológicos, el conjunto de parámetros representando un cromosoma particular se denomina fenotipo. El fenotipo contiene la información requerida para construir un organismo, el cual se refiere como genotipo. Los mismos términos se utilizan en el campo de los Algoritmos Genéticos.

La adaptación al problema de un individuo depende de la evaluación del genotipo. Esta última puede inferirse a partir del fenotipo, es decir puede ser computada a partir del cromosoma, usando la función de evaluación. La función de adaptación debe ser diseñada para cada problema de manera específica. Dado un Cromosoma particular, la función de adaptación le asigna un número real, que se supone refleja el nivel de adaptación al problema del individuo representado por el cromosoma.

Durante la fase reproductiva se seleccionan los individuos de la población para cruzarse y producir descendientes, que constituirán, una vez mutados, la siguiente generación de individuos. La selección de padres se efectúa al azar usando un procedimiento que favorezca a los individuos mejor adaptados, ya que a cada individuo se le asigna una probabilidad de ser seleccionado que es proporcional a su función de adaptación. Este procedimiento se dice que está basado en la ruleta sesgada.

Según dicho esquema, los individuos bien adaptados se escogerán probablemente varias veces por generación, mientras que los pobremente adaptados al problema, no se escogerán más que de vez en cuando.

Codificación de las soluciones

Tipos de representación

Existen distintos tipos de codificaciones de las soluciones: binarias, simbólicas, permutacional con repetición y la real. La codificación depende del problema que se quiere resolver. Según Fang (1994) las normas básicas a tener en cuenta para el diseño de una codificación óptima se pueden resumir en:

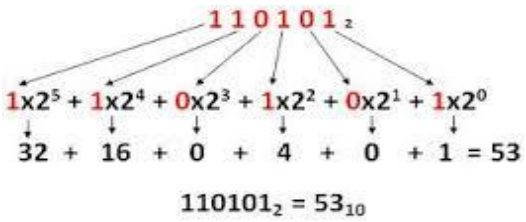
1. Cada solución del problema tendrá su correspondiente cadena codificada.
2. Para cada cadena codificada producida mediante los operadores genéticos existirá su correspondiente solución decodificada.
3. Codificación y solución se corresponderán una a una, es decir no habrá redundancia.

4. La codificación deberá permitir heredar de padres a hijos las características inherentes a los primeros.

Codificación binaria

Tradicionalmente la codificación binaria ha sido la más utilizada por la facilidad que ofrece el desarrollo de programas informáticos y porque permite la aplicación de sencillos operadores genéticos. Además, la mayor parte de la teoría subyacente en los algoritmos genéticos está desarrollada para la codificación binaria, aunque existen ampliaciones a otros tipos de representaciones

En el AG estándar, los individuos (sus estrategias) se representan mediante una cadena de bits. Entonces, un individuo genético de longitud L consiste de L símbolos 0 y 1. El conjunto completo de todos los posibles individuos genéticos distintos de longitud L será: $S \in \{0,1\}^L$ .y su cantidad $|S| \equiv N = 2^L$.



Decodificación del cromosoma

Para determinar el número real que la cadena binaria de caracteres representa. Es importante tener en cuenta que la cadena no representa un número real, sino que este número binario etiqueta un número dentro del intervalo inicialmente fijado.

Decodificación de un individuo, es decir, buscar el valor REAL que corresponde a la cadena binaria «c» de longitud «l»:

$$x = x_{min} + Vcad_{bin} * \frac{x_{max} - x_{min}}{2^l - 1} \text{.....(a)}$$

Ejemplo:
Se tiene una cadena binaria de longitud [5], expresada de esta manera: 01011, y se está trabajando en el intervalo [1,2], obtenga el valor real .

Solución: el valor entero de la cadena **01011** es 11, al aplicar y sustituir en [a], se tiene:

$$x = 1 + 11 * \frac{2 - 1}{2^l - 1}$$
$$x = 1.354838$$

DESARROLLO

I. Desarrollar un programa que emule la codificación binaria basándose en la figura 2

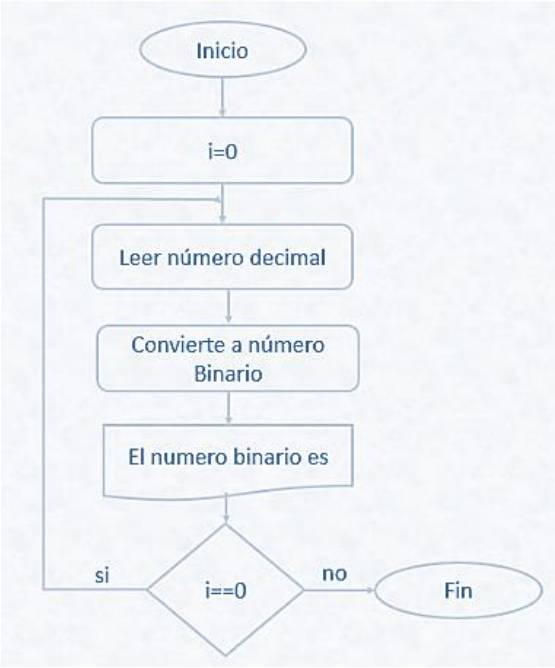


Figura 2. Diagrama de flujo para convertir de decimal a binario

Puede considerar la estructura del programa:

- Código.

```
package decimal.binario;

import java.util.Scanner;

public class DecimalBinario {

  public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int i=0;
        do {
            System.out.print("Introduzca un numero decimal: ");
            i = sc.nextInt();
            System.out.println("El numero en binario es: " +
Integer.toBinaryString(i));
        } while (i==0);
    }
}
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

II. Diseñar un programa que retome el valor binario del número decimal (programa anterior) para obtener el valor "real" que sea congruente con la fórmula:

$$x = x_{min} + Vcad_{bin} * \frac{x_{max} - x_{min}}{2^l - 1} \text{.....(a)}$$

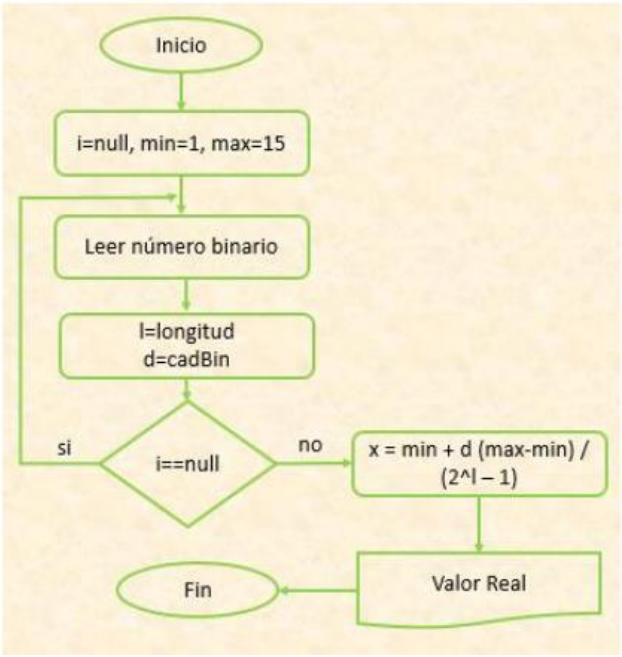


Figura 3. Diagrama de flujo para convertir el valor binario a valor real

Se sugiere considerar el código:

- Código.

```
package real;

import java.util.Scanner;

public class Real {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String i = null;
        int min = 1;
        int max = 15;
        int d, l;
        double x;
        do {
            System.out.print("Introduzca un numero binario: \n
");
            i = sc.next();
            l = i.length();
            d = Integer.parseInt(i, 2);
        } while (i == null);
        x = min + d * ((max - min) / (Math.pow(2, l) - 1));
        System.out.println("Valor real: \n " + x);
    }
}
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

III. Desarrollar un programa que retome el valor real de un individuo y sea evaluado en la función de adaptación asignada. Considere el diagrama de flujo de la figura 4.

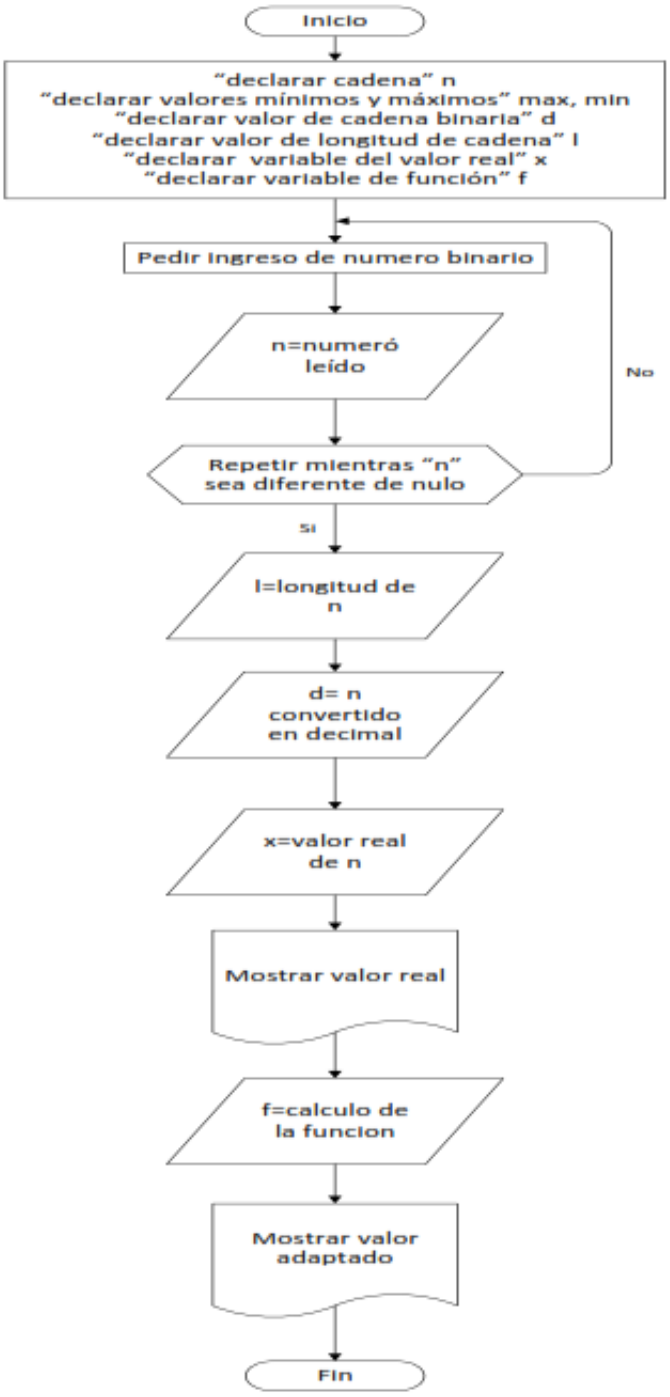


Figura 4. Diagrama de flujo para convertir el valor real a un valor adaptado

Se sugiere considerar el código:

- Código:

```
using System;//libreria

namespace Adaptado//espacio de trabajo
{
    class Program//clase primaria
    {
        static void Main(string[] args)//metodo principal
        {
            string n = null;//declaracion de variables
            int min = 1;
            int max = 15;
            int d, l;
            double x,f;
            do//estructura de control do
            {
                Console.WriteLine("Introduce un numero en binario ");//impresion en pantalla
                n = Console.ReadLine();// asignacin y convercion a entero del numero leido
            } while (n == null);//fin estructur de control y condicion que n sea igual a 0
            l = n.Length;//obtencion de el tamaño de el numero binario
            d = Convert.ToInt32(n, 2); //conversion a decimal
            x = min + d * ((max - min) / (Math.Pow(2, l) - 1));//calculo del numero real
            Console.WriteLine("Valor real: " + x);//impresion del numero real
            f = 3 * Math.Sin(x) + 2 * x + 1;//calculo del valor adaptado a la funcion
            Console.WriteLine("Valor adaptado: " + f);//impresion de valor adaptado
            Console.ReadKey();// espera a leer una tecla para que no finalice el programa
        }
    }
}
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico

Básico:

1. Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones,
2. ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España
3. Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.
4. Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.
5. Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.
6. **-Bibliografía Complementaria**
7. An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press
8. 2. - Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley
9. 3.- Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.
10. 4.- Koza, J.R., "Genetic Programming. On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992, 819 p.

PRÁCTICA 3

OPERADOR DE SELECCIÓN

Objetivo

- El alumno conocerá el funcionamiento del operador de selección
- Desarrollará un programa que emule el funcionamiento del operador de selección

Introducción

En la naturaleza, la selección sirve para determinar qué individuos sobreviven y se reproducen y cuales desaparecerán por una incorrecta adaptación al entorno. Este proceso es imitado mediante los métodos de selección. El proceso de selección es el que ha despertado un mayor interés a juzgar por la gran cantidad de estudios y métodos desarrollados.

Los métodos de selección, basándose en la calidad de las soluciones (adaptación de los individuos), seleccionan a los mejores individuos (soluciones) para que se reproduzcan, formado parte de la población de padres. Por el contrario, los peores individuos (soluciones) no serán seleccionados y desaparecerán de la búsqueda. Este proceso permite que una misma solución pase varias copias a la población de padres. Se trata, por tanto, de un muestreo sin sustitución.

Un concepto de gran importancia es la presión selectiva, que indica el grado con el que las mejores soluciones son favorecidas para formar parte de la población padre. A mayor presión selectiva, más se favorecerá a los mejores individuos y, por lo tanto, tendrán un mayor número de copias en la población padre. Esta presión dirige al algoritmo hacia poblaciones cada vez mejor adaptadas al problema.

La ratio de convergencia de un algoritmo genético está en gran parte determinado por la magnitud de la presión selectiva. Con una alta presión selectiva se obtiene una mayor ratio de convergencia, con el inconveniente de aumentar la probabilidad de convergencia prematura hacia un subóptimo. Y por el contrario, si la presión es demasiado baja, el ratio de convergencia disminuirá y el algoritmo tardará más tiempo en encontrar la solución óptima, si es que la encuentra. En el caso más extremo, donde la presión selectiva es nula, es decir, todos los individuos tienen la misma probabilidad de sobrevivir, se producirá una búsqueda aleatoria por todo el espacio de soluciones.

Los métodos existentes en la actualidad se dividen en tres, los basados en la calidad, en el orden y los que realizan la selección mediante torneos o competiciones. A continuación, y basándose en los trabajos de Goldberg (1989*)* y Michalewicz (1995), se describirán los principales métodos de selección:

Métodos proporcionales a la calidad.

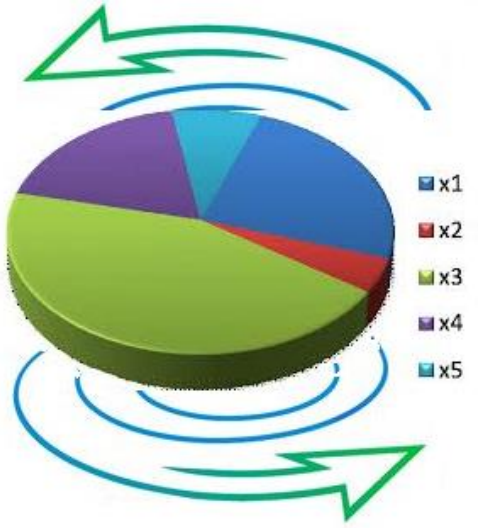
En este caso, la selección de padres se realiza teniendo en cuenta la magnitud de la calidad de las soluciones respecto de la del resto de la población. Por esto, superindividuos, es decir, los que tienen una calidad bastante superior a la del resto de la población, rápidamente coparán los puestos de la población de padres con un gran número de copias. El resultado es una pérdida de la diversidad en la población y una prematura convergencia. Dentro de esta categoría cabe destacar:

Selección proporcional o método de la ruleta, desarrollado en un principio por Holland en 1975, y ampliamente analizado por otros autores como Brindle en su Tesis Doctoral en 1981 y Goldberg (1989). La probabilidad de selección p , para el individuo i está dada por la siguiente ecuación:

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

siendo *fi* la calidad del individuo *i*, y *n* el tamaño de la población.

La representación gráfica de este sistema es una ruleta dividida en n áreas cada una de ellas proporcional a la calidad de cada individuo de la población. La selección consistirá rotar n veces la ruleta y la zona donde caiga el marcador indicará el individuo seleccionado para reproducirse.



De esta forma, los individuos con mayor calidad relativa tendrán mayores probabilidades de ser seleccionados para la población de padres, y además más de una vez. Por lo que se presenta el problema de superindividuos y, por lo tanto, de convergencia prematura.

Desarrollo

I. Desarrollar un programa emule el proceso de selección por ruleta. Considere el diagrama de flujo de la figura 4.

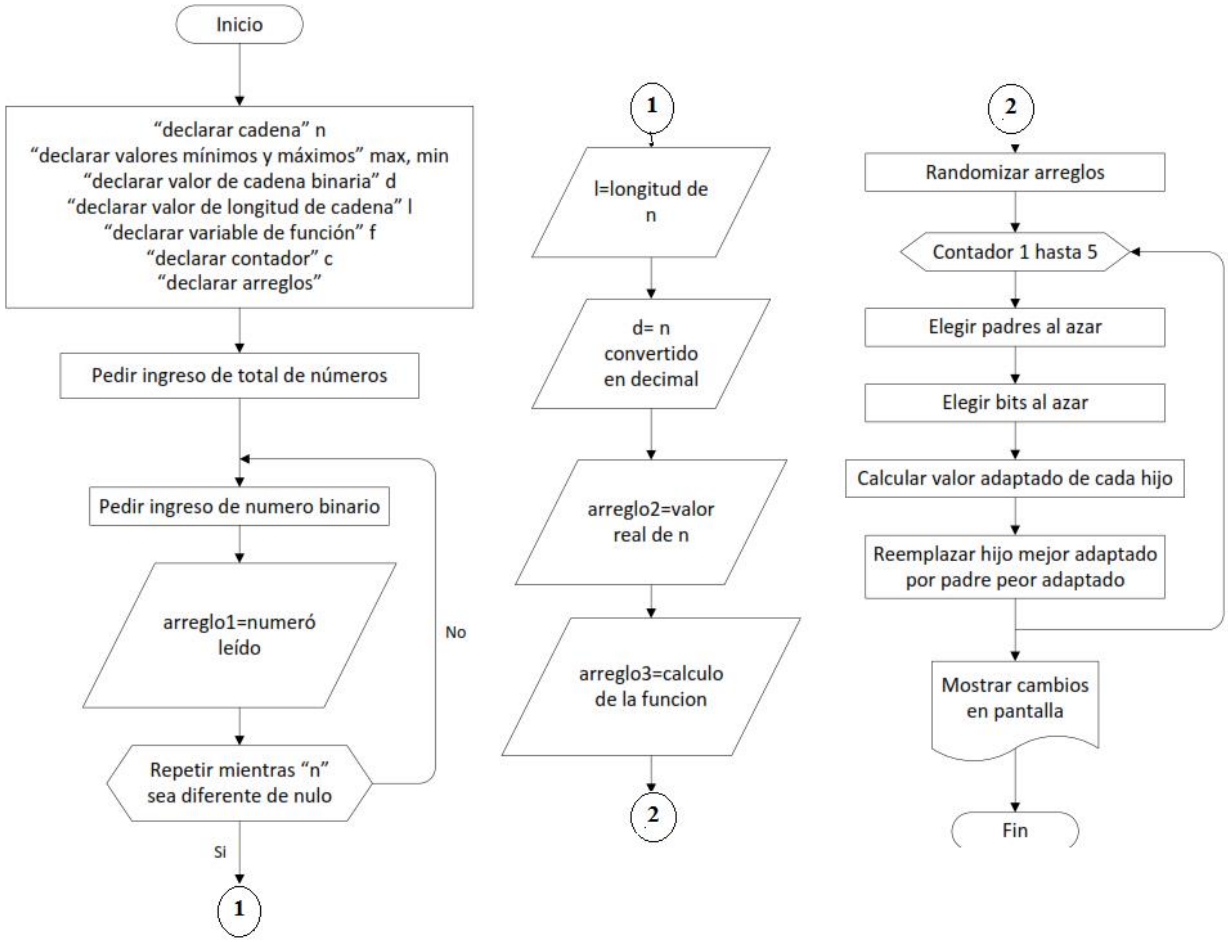


Figura 4. Diagrama de flujo del operador de selección por ruleta.

Puede considerar el programa que a continuación se presenta

Código

```
package ruleta;

import java.io.IOException;
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Scanner;
import java.util.concurrent.ThreadLocalRandom;

public class Ruleta {

    public static void main(String[] args) throws IOException {
        Scanner sc = new Scanner(System.in);
        int min = 1;
        int max = 15;
        int d, l, n, c, rand, rand2, rand3, auxd1, auxd2;
        c = 0;
        String pa1, pa12, pa2, pa22, h1, h2;
        double auxr1, auxr2, auxa1, auxa2;
        List<Integer> indicer = new ArrayList<>();
        System.out.print("Cuantos numeros se ingresaran ");
        n = sc.nextInt();
        String[] i = new String[n];
        double[] r = new double[n];
        double[] a = new double[n];
        String[] i2 = new String[n];
        double[] r2 = new double[n];
        double[] a2 = new double[n];
        do {
            System.out.print((c + 1) + ".- Introduce un numero
binario ");
            i[c] = sc.next();
            l = i[c].length();
            d = Integer.parseInt(i[c], 2);
            r[c] = min + d * ((max - min) / (Math.pow(2, l) - 1));
            a[c] = 3 * Math.sin(Math.toRadians(r[c])) + 2 * r[c] + 1;
            c += 1;
        } while (c != n);
        for (int m = 1; m <= n; m++) {
            indicer.add(m);
        }
        Collections.shuffle(indicer);
        for (int x = 0; x < n; x++) {
            i2[x] = i[indicer.get(x) - 1];


             rand = ThreadLocalRandom.current().nextInt(1, l - 1);
             do {
                 rand2 = ThreadLocalRandom.current().nextInt(0, n);
                 rand3 = ThreadLocalRandom.current().nextInt(0, n);
             } while (rand2 == rand3);
             pa1 = i2[rand2].substring(rand, l);
             pa12 = i2[rand2].substring(0, rand);
             pa2 = i2[rand3].substring(rand, l);
             pa22 = i2[rand3].substring(0, rand);
             h1 = pa22.concat(pa1);
             h2 = pa12.concat(pa2);
             auxd1 = Integer.parseInt(h1, 2);
             auxd2 = Integer.parseInt(h2, 2);
             auxr1 = min + auxd1 * ((max - min) / (Math.pow(2, l) -
 1));
```

Los resultados pueden ser muy parecidos a estos

Pantalla.

```
1.- Introduce un numero binario 01001001
2.- Introduce un numero binario 00000101
3.- Introduce un numero binario 10101010
4.- Introduce un numero binario 11011110
5.- Introduce un numero binario 11011110
6.- Introduce un numero binario 00101001
7.- Introduce un numero binario 01110100
8.- Introduce un numero binario 01011111
9.- Introduce un numero binario 00001101
10.- Introduce un numero binario 00000001
Cadena           Real             Adaptativo

01001001       5.007843137254902      11.277562603301106

00000101       1.2745098039215685      3.615747281854216

10101010       10.333333333333334      22.20479042447904

11011110       13.188235294117646      28.06092346092827

11011110       13.188235294117646      28.06092346092827

00101001       3.2509803921568627      7.672090397631533

01110100       7.368627450980392      16.122012652498988

01011111       6.215686274509804      13.756187131400694

00001101       1.7137254901960786      4.517168058759946

00000001       1.0549019607843138      3.1650353385236425
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico

Básico:

Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones, ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España

Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.

Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

Bibliografía Complementaria

An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

PRÁCTICA 4

OPERADOR DE CRUCE DE UN PUNTO

Objetivo

- Desarrollar un programa que emule el operador de cruce de un punto con su respectivo diagrama de flujo y pseudocódigo.

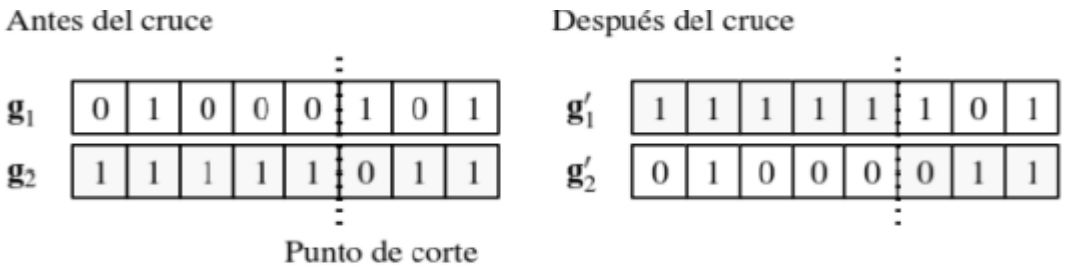
Introducción

Los Algoritmos Genéticos (AGs) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proceso genético de los organismos vivos. A lo largo de las generaciones. En los Algoritmos genéticos se trabaja con Ristras que se encuentran representadas mediante 0 y 1. Los algoritmos genéticos trabajan con distintos tipos de selección, los métodos de selección, basándose en la calidad de las soluciones (adaptación de los individuos), seleccionan a los mejores individuos (soluciones) para que se reproduzcan, formando arte de la población de padres. Este proceso perite que una misma solución pase varias copias a la población de padres. Se trata por tanto, de un muestreo sin sustitución. Existe un tipo de Selección llamado Selección de Ruleta, su probabilidad de selección p, para el individuo i está dada por la siguiente ecuación:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$ siendo *fi* la calidad del individuo *i, y n el tamaño de la población.*

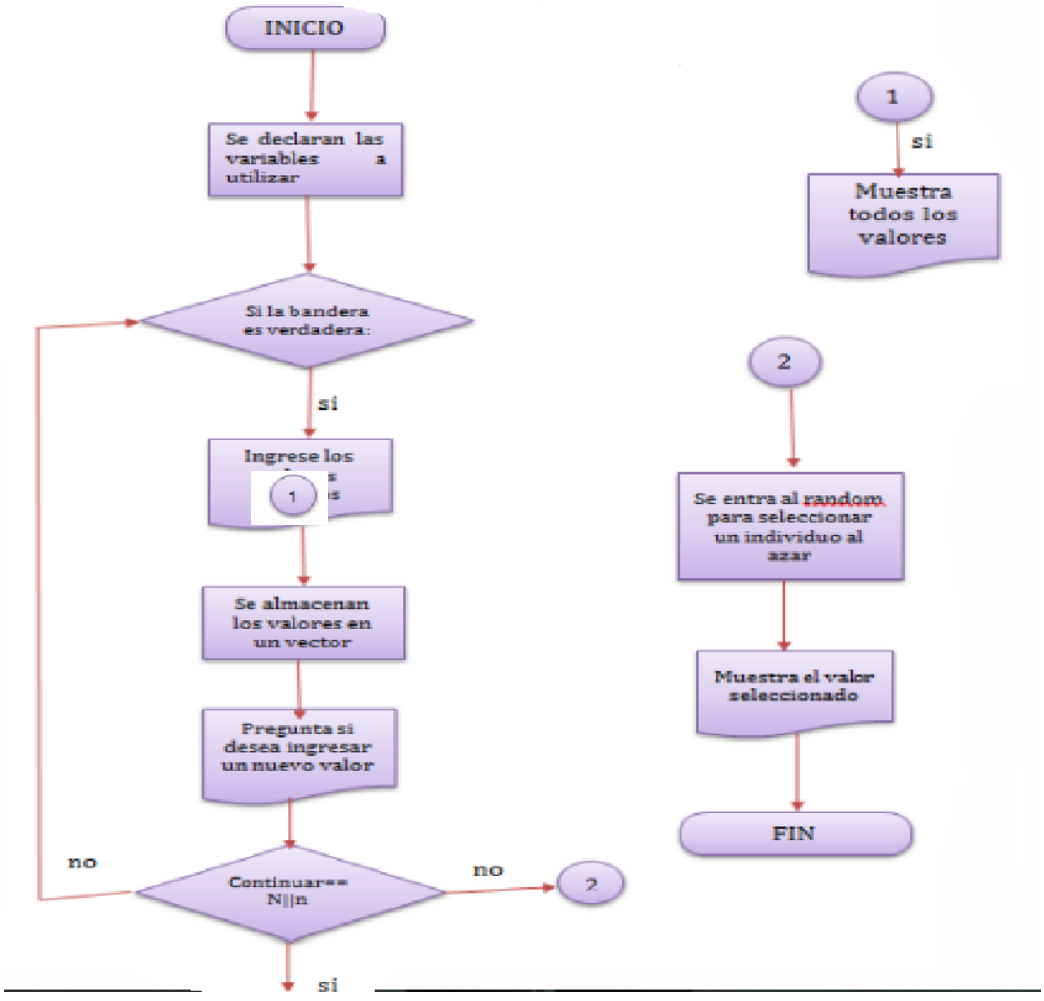
La representación de este sistema es una ruleta dividida en n áreas cada una de ellas proporcional a la calidad de cada individuo de la población. La selección consistirá en rotar n veces la ruleta y la zona donde caiga el marcador indicará el individuo seleccionado para reproducirse.

Existen diferentes tipos de Operadores Genéticos, uno de ellos es el CRUCE DE UN PUNTO, que juega el papel más importante dentro de los operadores genéticos, ya que se permite el intercambio de características de una generación a la siguiente. Se desarrolla probabilísticamente hablando, se le asigna na probabilidad (p) para que se pueda ejecutar. Su valor depende del tipo de problema, de la representación utilizada, y de muchos otros factores. Este cruce es operado a partir de una pareja de genotipos seleccionados g1 y g2 y seleccionado un punto de corte aleatorio, se generan dos descendientes h1 y h2 intercambiando parte de su información.



DESARROLLO

- Se desarrollará un programa para calcular el cruce de un punto y que calcule su mutación correspondiente.



INICIO.

- Se declaran las variables.
- MIENTRAS que la bandera sea verdadera:
- Se solicitan los valores binarios.
- Se almacenan en un vector
- Pregunta si desea ingresar mas valores:
- SI no termina los ciclos anteriores
- Comienza a imprimir los valores binarios almacenados en el vector
- SI NO regresa al ciclo while para agregar un valor nuevo y se repite.
- Se seleccionan dos individuos al azar para ser los padres.
- Se calcula el punto de corte de manera aleatoria.
- Se hace el Cruce de un Punto.
- Se muestra cuales fueron los hijos obtenidos.
- Se calcula: Decimal->Real->Funcion Adaptativa de ambos hijos.
- Se almacena el valor de la funcion adaptativa de cada hijo.
- Se comparan los valores obtenidos contra los de los padres.
- Se remplaza el padre por el hijo más fuerte.
- Se repite varias veces hasta que se cumple el umbral.
- Se ingresan los individuos a un random para elegir el individuo a mutar.
- Se calcula la mutación.
- Se imprime el resultado de la mutación y los cruces. FIN.

```
27 -    end
28      %_______
29      % extrae aleatoriamente del vector N los numeros binario
30 -    b=round((i-1)*rand(1,2)+1); extraccion=N(b);%numeros de extraccion ose
31      %los numeros binarios extraidos aleatroriamente se tranforman en binar
32 -    A=dec2bin(extraccion)
33      %
34 -    P=A(1,:)
35 -    str_P = num2str(P)
36 -    O=A(2,:)
37 -    str_O= num2str(O)
38 -    puntoC=input('\n Ingrese el punto de corte para esta iteracion:\n');
39 -    numf=longitud-puntoC;
40 -      for k = 1:numf
41 -      w1=str_P(k)~=str_O(k)
42 -       str_P(k)= str_O(k)
43 -        w3=str_O(k)~=str_P(k)
44 -       str_O(k)= str_P(k)
45 -      end
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente. Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico

Básico:

5. Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones,
   ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España
6. Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.
7. Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.
8. Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

-

Bibliografía Complementaria

2. An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

2. - Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

3.- Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

4.- Koza, J.R., "Genetic Programming. On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992, 819 p.

PRÁCTICA 5
OPERADOR DE CRUCE DE DOS PUNTOS

Objetivo

- Desarrollar un programa que realice el cruce de dos puntos con las ristras seleccionadas.

Introducción

Los Algoritmos Genéticos (AG) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización. Están basados en el proc4eso genético de los organismos vivos. A lo largo de las generaciones.

En los Algoritmos genéticos se trabaja con Ristras que se encuentran representadas mediante 0 y 1.

Los algoritmos genéticos trabajan con distintos tipos de selección, los métodos de selección, basándose en la calidad de las soluciones (adaptación de los individuos), seleccionan a los mejores individuos (soluciones) para que se reproduzcan, formando arte de la población de padres. Este proceso perite que una misma solución pase varias copias a la población de padres. Se trata, por tanto, de un muestreo sin sustitución.

Existe un tipo de Selección llamado Selección de Ruleta, su probabilidad de selección p, para el individuo i está dada por la siguiente ecuación:

$$p_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$ siendo *fi* la calidad del individuo *i, y n el tamaño de la población.*

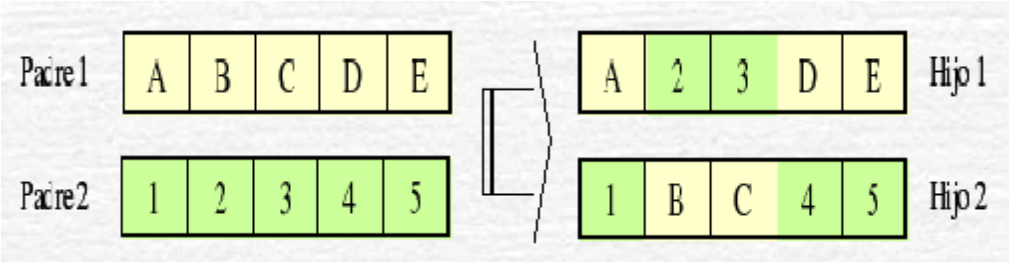
La representación de este sistema es una ruleta dividida en n áreas cada una de ellas proporcional a la calidad de cada individuo de la población. La selección consistirá en rotar n veces la ruleta y la zona donde caiga el marcador indicará el individuo seleccionado para reproducirse.

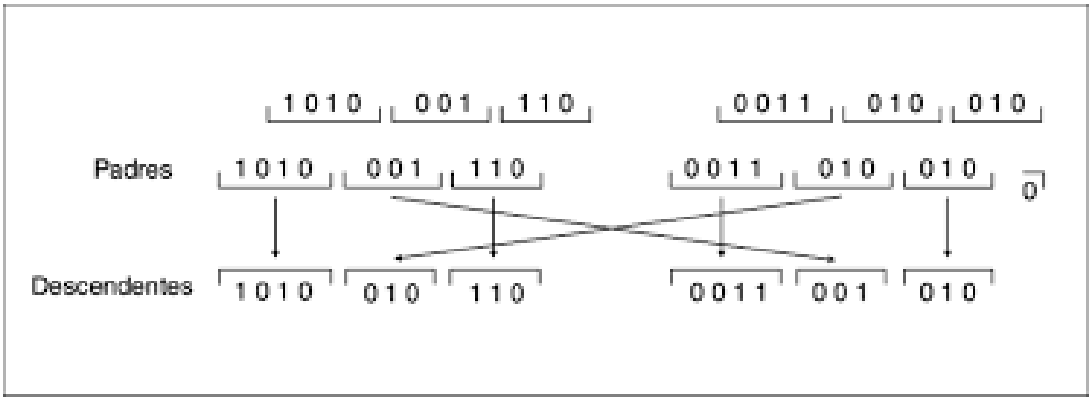
Existen diferentes tipos de Operadores Genéticos, uno de ellos es el CRUCE DE DOS PUNTOS.

CRUCE DE DOS PUNTOS

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

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

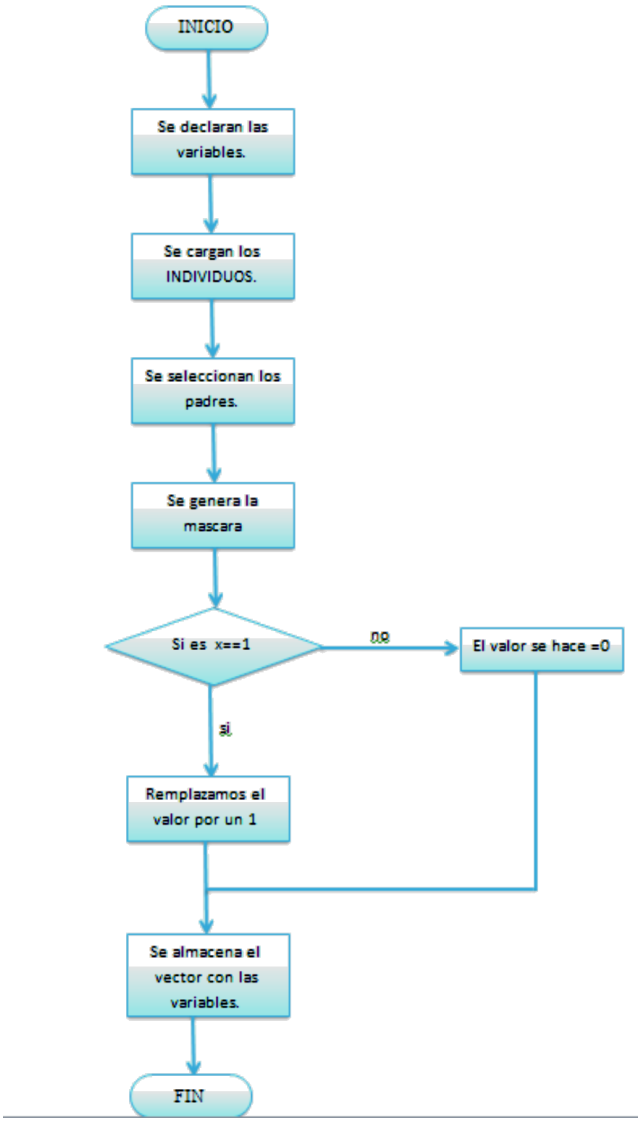




DESARROLLO

Considere la información del Cruce de dos Puntos para realizar el programa

Diagrama de flujo



Código:

```
//cruces
for(int i=0;i<cadenas.length;i++){

    c1=cadenas[i].substring(0,3);
    p1.add(c1);
    c2=cadenas[i].substring(3,7);
    p2.add(c2);

}
```

Funcionando:

```
        Todos los sujetos
01010110
11100011
00100101
10100100
11110001
00110100
10111111
00101001
Mascara = 00110101
Padre uno 11100011
Hijo uno 00100001
Padre dos 00101001
Hijo dos 00100001
Padre dos 00101001 es sustituido por hijo dos 00100001
        Todos los sujetos
01010110
11100011
00100101
10100100
11110001
00110100
10111111
00100001
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico

Básico:

Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones, ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España

Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.

Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

Bibliografía Complementaria

An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

PRÁCTICA 6

OPERADOR DE CRUCE UNIFORME

Objetivo

- El alumno conocerá el funcionamiento del operador de cruce uniforme
- Desarrollará un programa que emule el funcionamiento de este operador

Introducción

El operador cruce es el que juega el papel más importante dentro de los operadores genéticos, ya que permite el intercambio de características de una generación a la siguiente; es decir, el que hace evolucionar a las especies. Su realización es probabilística, se le asigna una probabilidad (*p*) para que se pueda ejecutar. Su valor depende del tipo de problema, de la representación utilizada, y de muchos factores más. Inicialmente el valor dado fue del 60% (Goldberg, 1989), pero han surgido muchos estudios de cómo adaptar estas probabilidades al estado del proceso genético (Michalewicz, 1995).

Es importante considerar que el operador cruce realiza dos trabajos, los denominados mecanismo e idea (Jones, 1995). La *idea* del cruce se basa en el hecho de que cada uno de los padres seleccionados tiene una probabilidad mayor que la media de la generación actual, de ofrecer un mejor material genético. La razón principal de mantener un conjunto de soluciones y utilizar el cruce es formar cada vez mejores individuos y combinarlos a su vez para conseguir otros nuevos aún mejores: cumplir el teorema de los bloques constitutivos. Si la idea del cruce así definida no se desarrolla en el algoritmo elegido, entonces no merece la pena gastar recursos en mantener una población de soluciones.

El *mecanismo* del cruce es el proceso por el que se lleva a cabo la idea. Es decir, la forma del intercambio, la aplicación literal de los métodos para la implementación de cada operador cruce que serán explicados a continuación.

Es importante separar estos dos conceptos del cruce ya que, si un algoritmo realiza sólo el mecanismo, el resultado final es tal que no importa entonces que los padres seleccionados sean los mejores, al no aprovecharse sus buenas características. Se puede hablar de macromutación de los padres: cambio de la mayor parte del material genético de cada uno de los padres por otro que, en principio, no les aporta beneficios, tal y como pasa en las mutaciones.

Existe otro método menos estudiado de producir hijos, pero en cuyo proceso no están implicados directamente dos padres. Fue desarrollado inicialmente por Syswerda en 1993 y modificado por Eshelman y Schaffer (1993). Consiste en la creación de los hijos teniendo en

cuenta toda la información que proporciona el total de individuos de la generación actual. Para crear los nuevos individuos, se observa la frecuencia de los valores existentes en cada posición, y por medio de dicha frecuencia se van construyendo los hijos de forma estocástica.

Dentro de la gran cantidad de operadores cruce existentes, en particular se hablará del cruce uniforme.

Operador cruce uniforme. Es la generalización de los operadores anteriores. Cada gen tiene una probabilidad del 50% de ser intercambiado por el correspondiente del otro padre, utilizándose para ello una máscara (Figura 5). Fue desarrollado por primera vez por Ackeley en 1987, y posteriormente Syswerda (1989) realizó un estudio comparativo entre este operador y los operadores 1-punto y 2puntos. La conclusión Fue que el cruce uniforme funciona mejor que los otros dos en estudio.

El cruce uniforme es una técnica completamente diferente de las vistas hasta el momento. Cada gen de la descendencia tiene las mismas probabilidades de pertenecer a uno u otro padre. Aunque se puede implementar de muy diversas formas, la técnica implica la generación de una máscara de cruce con valores binarios. Si en una de las posiciones de la máscara hay un 1, el gen situado en esa posición en uno de los descendientes se copia del primer padre. Si por el contrario hay un 0 el gen se copia del segundo padre. Para producir el segundo descendiente se intercambian los papeles de los padres, o bien se intercambia la interpretación de los unos y los ceros de la máscara de cruce.

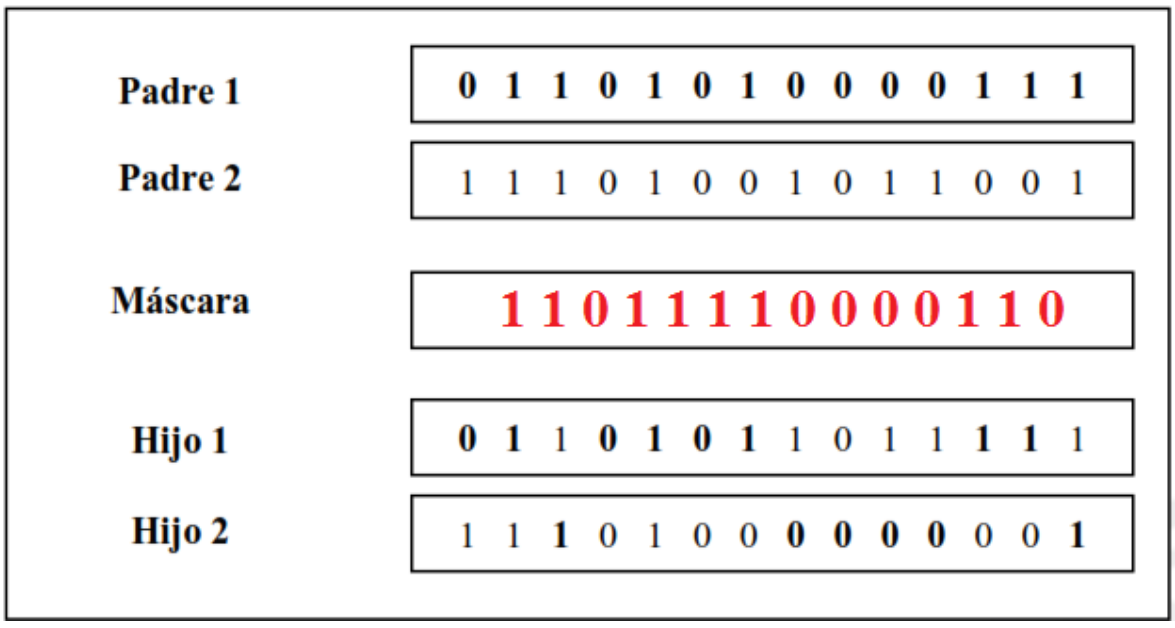
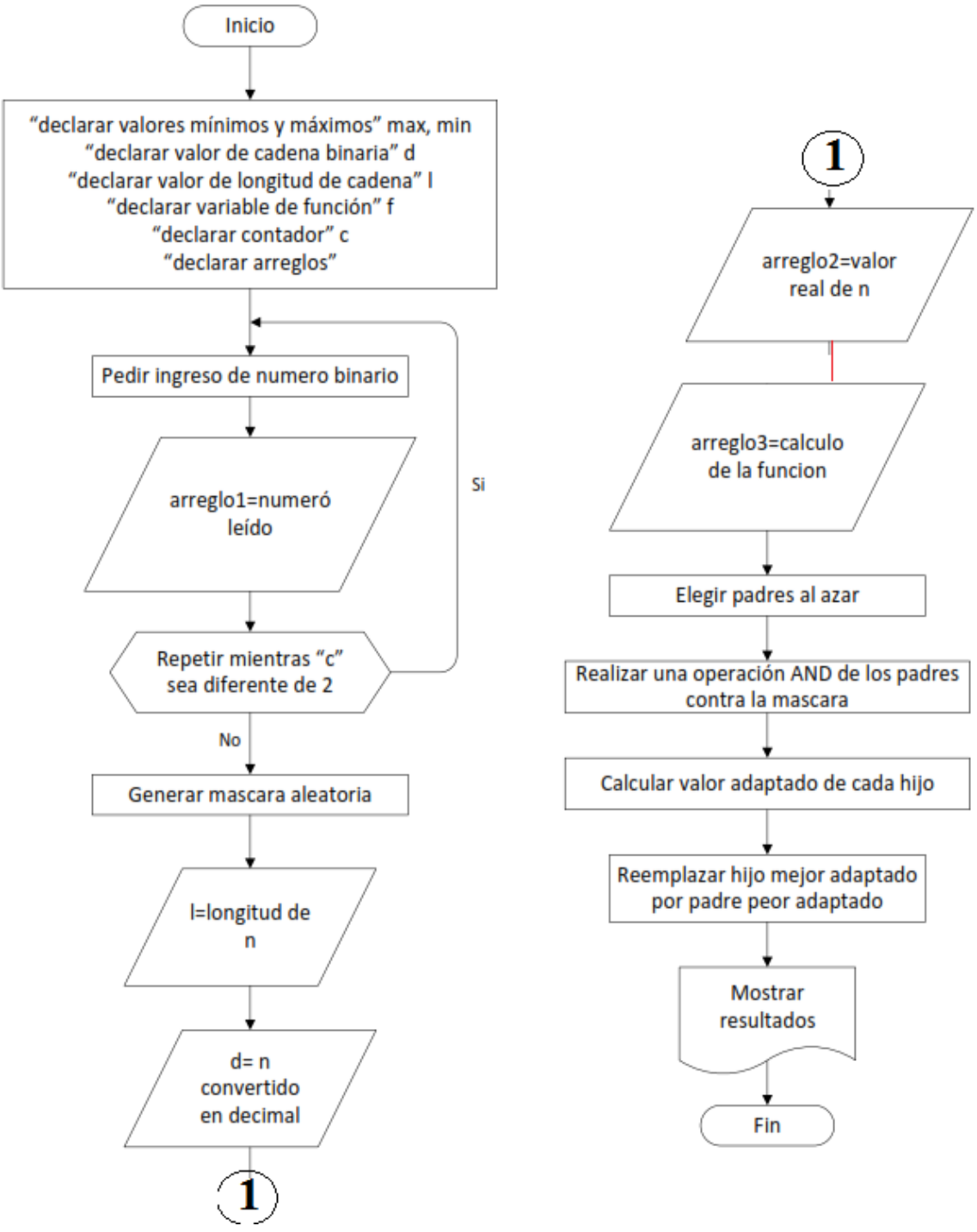


Figura 5. Operador de Cruce Uniforme

Desarrollo

II. Desarrollar un programa emule el proceso del operador de cruce uniforme. Considere el diagrama de flujo de la figura 6.



Puede considerar el programa que a continuación se presenta

Código.

```
package cruceu;

import java.util.Scanner;
import java.util.concurrent.ThreadLocalRandom;

public class CruceU {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int c = 0;
        int min = 1;
        int max = 15;
        String[] i = new String[2];
        double[] r = new double[2];
        double[] a = new double[2];
        int l, d, auxd1, auxd2, mascara, rand3;
        String pa1, pa2, mas = null;
        double auxr1, auxr, auxa1, auxa2;
        StringBuilder masc;
        masc = new StringBuilder();
        StringBuilder h1 = new StringBuilder();
        StringBuilder h2 = new StringBuilder();
        do {
            System.out.print((c + 1) + ".- Introduce un numero binario ");
            i[c] = sc.next();
            l = i[c].length();
            d = Integer.parseInt(i[c], 2);
            r[c] = min + d * ((max - min) / (Math.pow(2, l) - 1));
            a[c] = 3 * Math.sin(Math.toRadians(r[c])) + 2 * r[c] + 1;
            c++;
        } while (c != 2);
        c = 0;
        while (c < l) {
            char car = '1';
            masc.append(car);
            c++;
        }
        mascara = Integer.parseInt(masc.toString(), 2);
        do {
            rand3 = ThreadLocalRandom.current().nextInt(1, mascara);
            if (Integer.toBinaryString(rand3).length() < l) {
                c = l;
```

```
}
mascara = Integer.parseInt(masc.toString(), 2);
do {
    rand3 = ThreadLocalRandom.current().nextInt(1, mascara);
    if (Integer.toBinaryString(rand3).length() < l) {
        c = l;
        masc.delete(0, masc.length());
        while (c > Integer.toBinaryString(rand3).length()) {
            char car = '0';
            masc.append(car);
            c--;
        }
        masc.append(Integer.toBinaryString(rand3));
        mas = masc.toString();
    }

  } while (mas == null);

    pa1 = i[0];
    pa2 = i[1];
    System.out.println("Mascara " + mas);
    for (int x = 0; x < l; x++) {
        if (pa1.substring(x, x + 1).equals("1")) {
            if (mas.substring(x, x + 1).equals("1")) {
                h1.append('1');
            } else {
                h1.append('0');
            }
        } else {
            h1.append('0');
        }
        if (pa2.substring(x, x + 1).equals("1")) {
            if (mas.substring(x, x + 1).equals("1")) {
                h2.append('1');
            } else {
                h2.append('0');
            }
        } else {
            h2.append('0');
        }

      if (pa2.substring(x, x + 1).equals("1")) {
          if (mas.substring(x, x + 1).equals("1")) {
              h2.append('1');
          } else {
              h2.append('0');
          }
      } else {
          h2.append('0');
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico

Básico:

Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones, ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España

Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.

Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

Bibliografía Complementaria

An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

PRÁCTICA 7

OPERADOR DE MUTACIÓN

Objetivo

- El alumno conocerá el funcionamiento del operador de mutación
- Desarrollará un programa que emule el funcionamiento de este operador

Introducción

Operadores Mutación.

Mientras el operador cruce combina aquellas buenas características de los individuos llevando a cabo la explotación de las áreas beneficiosas detectadas por la selección, el operador mutación incluye la diversidad en la búsqueda facilitando la exploración de áreas aún no tratadas. Por lo tanto, se puede decir que la principal característica de este operador es permitir que todos los individuos del espacio de búsqueda tengan una probabilidad de ser explorados mayor de cero.

Además, su funcionamiento permite recuperar características de los individuos que por medio de la selección y el cruce podrían haber perdido, cayendo en el olvido y que, aún siendo buenas características, sería imposible recuperar sin la acción del operador mutación.

Mutación estándar. Es el operador por excelencia utilizado en la codificación binaria. La probabilidad de aplicación (*pm*) muy baja, para no incluir demasiada diversidad en la búsqueda y perder de esta forma la dirección de la misma. Según los estudios realizados por Holland, la probabilidad aceptable sería la comprendida entre 0.1 y 1%. Esta probabilidad se aplicará bit a bit, a diferencia de los operadores cruce que se refieren a cada pareja o incluso a otros operadores mutación, generalmente los aplicados sobre codificaciones no binarias, donde se aplicará sobre cada cadena o individuo. El proceso es sencillo y consiste en mutar el bit correspondiente según la probabilidad de mutación asignada (Figura 7).

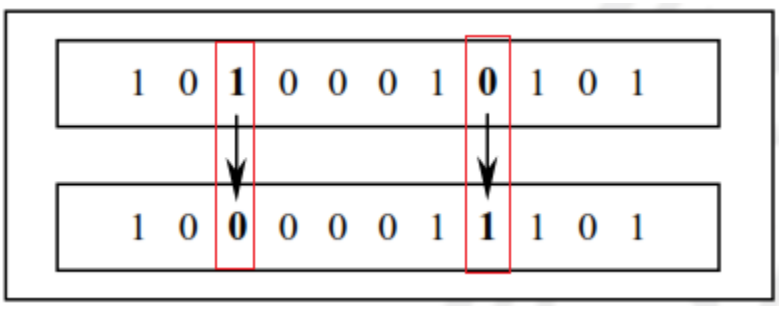
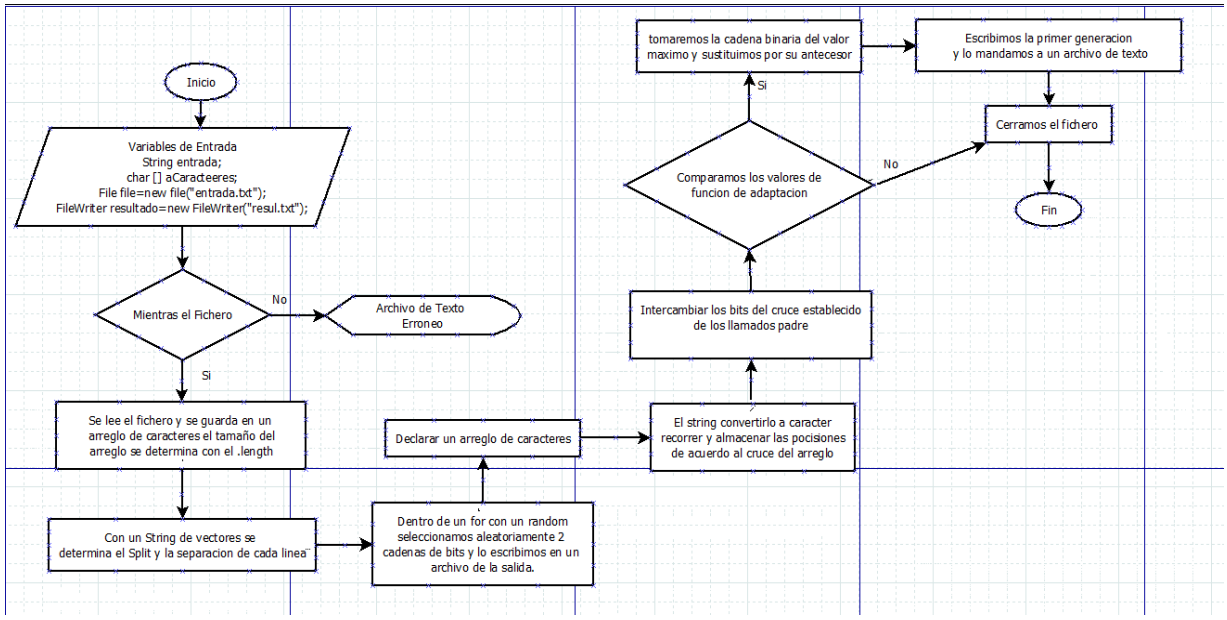


Figura 7. Operador mutación estándar

DESARROLLO

Se desarrollará un programa para calcular el cruce de un punto y que calcule su mutación correspondiente.

Diagrama de flujo:



Código:

```
private void jButton5ActionPerformed(java.awt.event.ActionEvent evt) {
    String array1[]= new String[4], array2[]= new String[4],
           padres[]= new String[2];
    String array[]= new String[4];
    int N= lista.size(), padre1=0, padre2=0, decimal1, decimal2;
    int corte= (int)(Math.random()*Tmax + 1);

    while(padre1 == padre2){
        padre1= (int)(Math.random()*N + 1);
        padre2= (int)(Math.random()*N + 1);
    }

    String cad1= lista.get(padre1-1), cad2= lista.get(padre2-1);

    padres= alg.cruce(cad1, cad2, corte);

    data(padre1, padre2, corte);

    cad1= padres[0];
    cad2= padres[1];

    array1[0]= cad1;
    array1[1]= ConversionDec(cad1)+"";
    array1[2]= alg.Real(Double.parseDouble(array1[1]), min, max, Tmax)+"";
    array1[3]= alg.Funcion(Double.parseDouble(array1[2]))+"";

    array2[0]= cad2;
    array2[1]= ConversionDec(cad2)+"";
    array2[2]= alg.Real(Double.parseDouble(array2[1]), min, max, Tmax)+"";
```

```
array2[0]= cad2;
array2[1]= ConversionDec(cad2)+"";
array2[2]= alg.Real(Double.parseDouble(array2[1]), min, max, Tmax)+"";
array2[3]= alg.Funcion(Double.parseDouble(array2[2]))+"";

modelo2.addRow(array1);
modelo2.addRow(array2);

if(Double.parseDouble(array1[3])> Double.parseDouble(array2[3]))  lista.set(padre1-1, cad1);
else    lista.set(padre2-1, cad2);

String espacio[]= {"****","****","****","****"};
modelo.addRow(espacio);
modelo2.addRow(espacio);
for(String cad: lista){
    array[0]= cad;
    array[1]= ConversionDec(cad)+"";
    array[2]= alg.Real(Double.parseDouble(array[1]), min, max, Tmax)+"";
    array[3]= alg.Funcion(Double.parseDouble(array[2]))+"";
    modelo.addRow(array);
}
```

Ejecución:

Iteracion	18	***	***
0101010110	342	6.686217008...	93.17062426...
0101010110	342	6.686217008...	93.17062426...
Iteracion	19	***	***
0101010101	341	6.666666666...	92.67099199...
0101010110	342	6.686217008...	93.17062426...
Iteracion	20	***	***
0101010110	342	6.686217008...	93.17062426...
0101010101	341	6.666666666...	92.67099199...

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico

Básico:

Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones, ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España

Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.

Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

Bibliografía Complementaria

An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

PRÁCTICA 8

ALGORITMO GENÉTICO SECUENCIAL

OBJETIVO

Que el alumno genere un programa que agrupe el proceso del algoritmo genético secuencial

INTRODUCCIÓN

Los *algoritmos genéticos* han sido muy importantes dentro de las técnicas evolutivas. tradicionalmente han utilizado codificaciones independientes al problema, como son las cadenas binarias, sin embargo, muchas de las aplicaciones recientes se han centrado en otras representaciones tales como grafos, listas ordenadas, números reales, etc. Para la formación de los descendientes la reproducción es sexual, es decir, hay intercambio del material genético de los padres mediante los operadores genéticos, cruce y mutación. En este caso, el operador mutación se considera un operador de segundo orden frente al operador cruce, como principal diferencia con las técnicas evolutivas anteriores.
Se puede decir que los algoritmos genéticos son métodos metaheurísticos poco convencionales.
Esto es debido a que trabajan con un conjunto de soluciones al mismo tiempo, imitando los procesos que se llevan a cabo en la evolución natural de las especies. Esta es la principal diferencia con otras metaheurísticas tales como la búsqueda tabu, el recocido simulado, etc. donde una única solución es modificada hasta alcanzar un cierto criterio de parada.

En los últimos años se están desarrollando las técnicas de *programación genética*. En este caso la representación es más compleja el ser estructuras jerarquizadas y de tamaño variable (Koza, 1993). La población está formada por cientos o miles de dichas estructuras que son unidos por operadores apropiados pretendiéndose encontrar aquella combinación mejor adaptada a un determinado problema.
Los algoritmos genéticos es el campo de los algoritmos evolutivos que ha tenido un mayor desarrollo. Prueba de ello es el gran número de publicaciones existentes en revistas de gran prestigio como, por ejemplo, *Management Science*, *Operational Research*, etc., y por la creación de grandes conferencias internacionales bianuales: *International Conference on Genetic Algorithms*, *Parallel Problem Solving from Nature*, *Foundations of Genetic Algorithms*.

El pseudo código de un algoritmo genético es el siguientes:

1. Elegir la población inicial de individuos
2. Evaluar la función objetivo (fitness) de cada individuo de la población
3. Repetir en esta población hasta terminar (límite de tiempo, alcanzar un valor en la función objetivo, etc.)
1. Seleccionar los mejores individuos para la reproducción (padres).

2. Crear nuevos individuos (hijos) a través de los operadores cruce y mutación.
3. Evaluar la función objetivo en los hijos.
4. Reemplazar los padres por los hijos

Se muestra el diagrama de flujo simplificado con las partes más importantes de un algoritmo genético clásico: selección, reproducción y sustitución. Aparte de estos procesos, que pertenecen al núcleo principal y exclusivo de los algoritmos genéticos, existen otros, de igual importancia, que sirven de apoyo a los anteriores: descripción del problema a resolver, codificación de sus soluciones e inicialización. Selección y codificación son los puntos críticos de un algoritmo genético.

DESARROLLO

Realizar un programa que emule el proceso de un algoritmo genético secuencial, considerando el diagrama de flujo que sigue:

Seudocódigo.

INICIO.

- Ingresa el usuario el valor
- Se guarda en la variable llamada “binario”.
- Se declaran vectores “ristra” y “ristraAux”.
- MIENTRAS el valor stream >> aux + Se llena el vector con cada componente del #binario.
- HASTA que se termine de recorrer el vector se realizara: +Se invierte el valor binario para su manipulación correcta.
- SI el valor individual de la ristra
- -1 es igual a uno: + La variable decimal será igual a decimal más núm.
- SI NO : + num es igual a num *2.
- Se almacena el valor decimal.
- Lee el valor decimal, el mínimo el máximo y el rango.
- Se guarda en la variable llamada “x-min”, “x-max”.
- Se hace la conversión usando la formula obtenida en clase.
- Se almacena el valor obtenido.
- Se lee el valor obtenido.
- Se aplica la formula vista en clase.
- Se imprimen os 3 resultados obtenidos.
- FIN.

Programa funcionando.

```
#include <iostream>
#include <sstream>
#include <vector>
#include <math.h>

using namespace std;

class AlgoritmosGeneticos{

    //Declararemos las variables a utilizar
    int decimal;
    string binario;
    double real;
    double funcionX;
    int longitud;
    stringstream stream;


    public:

    //inicializaremos la variables a utilizar
    AlgoritmosGeneticos(){
    binario = "0101011";
    decimal=0;
    real = 0.0;
    funcionX = 0.0;
    longitud = 0;
    stream.str(binario);
    }
```

```
//En esta seccion se realizara la combercion de un numero binario a un numero decimal
void Decimal(){

char aux;
int num = 1;
vector<char> ristra;
vector<char> ristraAux;

   //utilizando una variable de tipo stream donde se pasaran cada unos de los valores
      //que contenga el string a un tipo de dato char

    while(stream >> aux)
        ristraAux.push_back(aux);

    for(int i=1;i<=ristraAux.size();i++){

        ristra.push_back(ristraAux.at(ristraAux.size()-i));
        num = num;

        if(ristra.at(i-1) == '1')
            decimal = decimal+num;

        num = num*2;

    }
cout<<"Binario "<<binario<<endl<<"Decimal "<<decimal<<endl;
longitud = ristra.size();

}

void Real(){

double Xmin=0.0,Xmax=20.0;

real = (Xmin + (double)decimal) * ((Xmax-Xmin)/(double)(pow(2,longitud) - 1));

cout<<"Real[x] "<<real<<endl;

}

void FuncionAdaptacion(){

funcionX = 2.0*((double)pow(real,2))+(3.0*(double)(cos(real)))+1;

cout<<"f(x) = "<<funcionX<<endl;

}
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

_____

_____

_____

_____

_____

Acervo bibliográfico

Básico:

9. Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones,
   ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España
10. Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.
11. Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.
12. Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

-

Bibliografía Complementaria

3. An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

2. - Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

3.- Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

4.- Koza, J.R., "Genetic Programming. On the Programming of Computers by Means of Natural Selection", The MIT Press, 1992, 819 p.

PRÁCTICA 9

ALGORITMO GENÉTICO SELECCIÓN ELITISTA

OBJETIVO

- Generar el programa del algoritmo elitista con lo visto en clase y subir la evidencia que corresponda

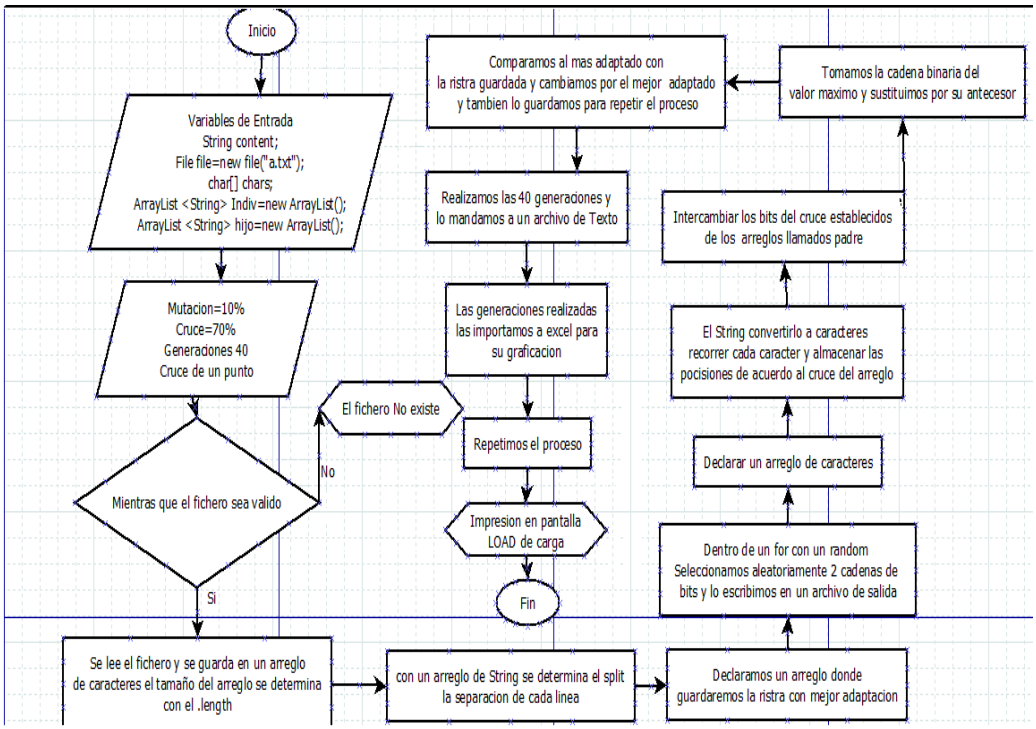
INTRODUCCIÓN

El elitismo es un caso particular del operador de copia consistente en copiar siempre al mejor, o en su caso mejores, individuos de una generación en la generación siguiente. De esta manera se garantiza que el proceso de búsqueda nunca dará un paso atrás en cuanto a la calidad de la mejor solución obtenida, sino que un cambio en ésta siempre implicará una mejora.

Una variación de este proceso consiste en copiar a los mejor o mejores individuos de una generación en la siguiente, únicamente cuando tras el paso de una generación no se haya mejorado con los operadores de cruce o mutación la mejor solución de la generación actual.

DESARROLLO

Considerar el siguiente diagrama de flujo para la implementación y ejecución del algoritmo genético elitista.



Código:

```
Configuración Algoritmo Genético
% ********************************************************
options = gaoptimset;
% Población Genética
options = gaoptimset(options,'PopulationSize' , 25);
options = gaoptimset(options,'CreationFcn', ...
@PermCreation);
% Criterios de parada
options = gaoptimset(options, 'Generations', 100);
options = gaoptimset(options, 'FitnessLimit', 0);
options = gaoptimset(options,'TolFun', 1e-12);
options = gaoptimset(options,'StallTimeLimit', 100);
options = gaoptimset(options,'StallGenLimit', 100);
%
Operadores Genéticos
% ********************************************************
% Elitismo
options = gaoptimset(options, 'EliteCount', 1);
% Ajuste del operador de selección
options = gaoptimset(options, 'SelectionFcn',...
@selectionroulette);
% Ajuste del algoritmo de cruce:
options = gaoptimset(options, 'CrossoverFcn', ...
@PermCrossover);
options = gaoptimset(options, 'CrossoverFraction', 0.8);
% Ajuste del algoritmo de mutación:
options = gaoptimset(options, 'MutationFcn', ...
@PermMutation);
algoritmos
geneticos

% Configuración de Salida
% ********************************************************
options = gaoptimset(options, 'Display', 'diagnose');
options = gaoptimset(options, 'PlotInterval', 1);
options =
gaoptimset(options,'PlotFcns', ...
[@gaplotbestindiv, @gaplotbestf]);
% Ejecución algoritmo
[x, fval, reason, output, population, scores]= ...
```

```
ga(@fitness,numeroReinas,options);
disp('');
disp('Mejor individuo: ');
disp(x);
disp('Número de colisiones: ');
disp(fval);
```

Ejecución:

File Edit Format View Help

```
Generacion1
0     0     37    individuo-->  1111111  Real-->  127  decimal-->  -0,999876
1     1     33    individuo-->  1110001  Real-->  113  decimal-->  -0,999908
2     2     36    individuo-->  1110111  Real-->  119  decimal-->  -0,999901
3     3     17    individuo-->  1001111  Real-->  79   decimal-->  -0,999890
4     4     29    individuo-->  1111100  Real-->  124  decimal-->  -0,999841
5     5     25    individuo-->  1110001  Real-->  113  decimal-->  -0,999881
6     6     23    individuo-->  1000011  Real-->  67   decimal-->  -0,999907
7     7     1     individuo-->  1000000  Real-->  64   decimal-->  -0,999858
8     8     24    individuo-->  1011100  Real-->  92   decimal-->  -0,999861
9     9     8     individuo-->  1100001  Real-->  97   decimal-->  -0,999866
10    10    9     individuo-->  1011110  Real-->  94   decimal-->  -0,999868
11    11    11    individuo-->  1000000  Real-->  64   decimal-->  -0,999908
12    12    39    individuo-->  1111101  Real-->  125  decimal-->  -0,999847
13    13    36    individuo-->  1110111  Real-->  119  decimal-->  -0,999888
14    14    48    individuo-->  1101110  Real-->  110  decimal-->  -0,999904
15    15    12    individuo-->  1101011  Real-->  107  decimal-->  -0,999831
16    16    31    individuo-->  1010101  Real-->  85   decimal-->  -0,999901
17    17    37    individuo-->  1111111  Real-->  127  decimal-->  -0,999887
18    18    26    individuo-->  1110101  Real-->  117  decimal-->  -0,999908
19    19    29    individuo-->  1111100  Real-->  124  decimal-->  -0,999884
20    20    18    individuo-->  1000000  Real-->  64   decimal-->  -0,999823
21    21    7     individuo-->  1100011  Real-->  99   decimal-->  -0,999904
22    22    20    individuo-->  1111100  Real-->  124  decimal-->  -0,999878
23    23    31    individuo-->  1010101  Real-->  85   decimal-->  -0,999904
24    24    13    individuo-->  1001110  Real-->  78   decimal-->  -0,999868
25    25    49    individuo-->  1011111  Real-->  95   decimal-->  -0,999838
26    26    3     individuo-->  1001101  Real-->  77   decimal-->  -0,999833
27    27    18    individuo-->  1000000  Real-->  64   decimal-->  -0,999864
28    28    6     individuo-->  1000001  Real-->  65   decimal-->  -0,999864
29    29    40    individuo-->  1111011  Real-->  123  decimal-->  -0,999823
30    30    24    individuo-->  1011100  Real-->  92   decimal-->  -0,999878
31    31    1     individuo-->  1000000  Real-->  64   decimal-->  -0,999878
32    32    20    individuo-->  1111100  Real-->  124  decimal-->  -0,999838
33    33    42    individuo-->  1111111  Real-->  127  decimal-->  -0,999838
34    34    29    individuo-->  1111100  Real-->  124  decimal-->  -0,999824
```

Algoritmo terminado

File Edit Format View Help

```
5     25    44    individuo-->  1110000  Real-->  112  decimal-->  0,379395  Adaptado-->  0,126651
6     26    45    individuo-->  1111100  Real-->  124  decimal-->  0,428223  Adaptado-->  0,138774
7     27    18    individuo-->  1000000  Real-->  64   decimal-->  0,159668  Adaptado-->  0,062434
8     28    9     individuo-->  1011110  Real-->  94   decimal-->  0,159668  Adaptado-->  0,062434
9     29    17    individuo-->  1001111  Real-->  79   decimal-->  0,513672  Adaptado-->  0,158441
0     30    36    individuo-->  1110111  Real-->  119  decimal-->  0,037598  Adaptado-->  0,016850
1     31    50    individuo-->  1011111  Real-->  95   decimal-->  0,037598  Adaptado-->  0,016850
2     32    38    individuo-->  1011110  Real-->  94   decimal-->  0,379395  Adaptado-->  0,126651
3     33    45    individuo-->  1111100  Real-->  124  decimal-->  0,379395  Adaptado-->  0,126651
4     34    27    individuo-->  1011111  Real-->  95   decimal-->  0,501465  Adaptado-->  0,155747
5     35    8     individuo-->  1100001  Real-->  97   decimal-->  0,452637  Adaptado-->  0,144588
6     36    40    individuo-->  1111011  Real-->  123  decimal-->  0,452637  Adaptado-->  0,144588
7     37    1     individuo-->  1000000  Real-->  64   decimal-->  0,550293  Adaptado-->  0,166303
8     38    7     individuo-->  1100011  Real-->  99   decimal-->  0,147461  Adaptado-->  0,058290
9     39    34    individuo-->  1111011  Real-->  123  decimal-->  0,525879  Adaptado-->  0,161099
0     40    9     individuo-->  1011110  Real-->  94   decimal-->  0,501465  Adaptado-->  0,155747
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente.

Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

Acervo bibliográfico

Básico:

Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones, ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España

Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.

Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

Bibliografía Complementaria

An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.

PRÁCTICA 10

ALGORITMO GENÉTICO SELECCIÓN POR TORNEO

OBJETIVO

- Generar el programa que emule el proceso de un algoritmo genético con selección por torneo.

INTRODUCCIÓN

La función de selección es la encargada de elegir a los individuos que harán de progenitores para la nueva generación. Existen diferentes formas de tomar esta decisión. La más evidente sería la de escoger a los individuos más adaptados, pero esto degeneraría en problemas de *diversidad genética,* donde rápidamente aparecerían *súper individuos* que provocarían uno de los grandes problemas que presentan los algoritmos genéticos: *la convergencia prematura*. Estos tres términos son básicos y aparecerán en multitud de ocasiones de ahora en adelante, así que es necesario hacer un alto en el camino y ofrecer una pequeña definición:

- ***Diversidad genética:*** es quizás el más intuitivo de los tres términos. Hace referencia a la distribución de las soluciones en el espacio de búsqueda. Uno de los problemas que presentan los AG's es el de cómo mantener una diversidad genética aceptable evitando que las soluciones se agolpen alrededor de óptimos locales impidiendo que se explore todo el espacio de búsqueda.

- ***Súper individuos:*** son óptimos locales. En esencia son buenas soluciones al problema (en ocasiones las mejores soluciones) pero que, si no se tratan correctamente, "absorben" a la población hacia ellos de forma que el resto del espacio de búsqueda queda sin explorar.

- ***Convergencia prematura****:* se produce cuando los súper individuos colapsan la población; es decir, los súper individuos (como mejores soluciones que son) se eligen una y otra vez como progenitores de forma que asfixian a los individuos menos dotados haciéndolos desaparecer. Al final el resultado es una población formada en su mayoría por individuos muy "similares" al súper individuo y por tanto muy próximos a él. De esta forma el AG no tiene ninguna posibilidad de explorar nuevos territorios de manera que la población *converge* hacia el *súper individuo* (óptimo local).

Una solución tan trivial como escoger siempre al mejor no es efectiva a largo plazo, pues reduce la diversidad genética dejando al AG a merced de la suerte. Existen otros métodos más complejos que intentan hacer esta selección de un modo más "distribuido", dando siempre prioridad a los mejores individuos pero dejando cierto margen para los menos adaptados.

***Selección por torneo**:* es un método considerablemente más simple. Consiste en escoger n individuos al azar y hacerlos competir entre sí. La manera de escoger al vencedor puede ser muy simple (escogiendo al que tenga un mejor valor de *fitness*) o más aleatoria (asignando una probabilidad de selección a cada individuo en función de su valor de *fitness* y dejando que sea el azar quien decida. Evidentemente la probabilidad de ser elegido será mayor cuanto mejor sea el individuo, pero aun así todos tendrían posibilidades de ser vencedores).

La idea de este método es bastante simple: se trata de escoger al azar un número determinado de individuos (en general 2) y hacerlos competir entre sí para decidir cuál es el que se convertirá en uno de los progenitores de la nueva generación. En este punto el camino se bifurca. Por un lado, existe una versión más determinista en cuyo caso el individuo seleccionado es el que posea un nivel de adaptación mayor. Es decir, el que ofrezca unos valores más altos de la función de adaptación.

Lo más destacable de este método es su sencillez que se traduce en coste computacional muy bajo. Y el hecho de que permite suavizar los efectos propios de la presión selectiva. Tomando pocos individuos para realizar el torneo, se aumentan las opciones globales de los individuos menos adaptados, mientras que en torneos con muchos participantes sus opciones son más reducidas.
Selección por torneos de tamaño q=2 es análogo a la selección por ruleta, ya que ambos asignan dos copias al mejor individuo.

Esta selección se efectúa mediante un torneo (comparación) entre un pequeño subconjunto de individuos elegidos al azar desde la población. Los beneficios de este tipo de selección son la velocidad de aplicación (dado que no es necesario evaluar ni comparar la totalidad de la población) y la capacidad de prevenir, en cierto grado, la convergencia prematura. La principal desventaja es la necesidad de establecer el parámetro correspondiente al tamaño del subconjunto (véase Figura 1).

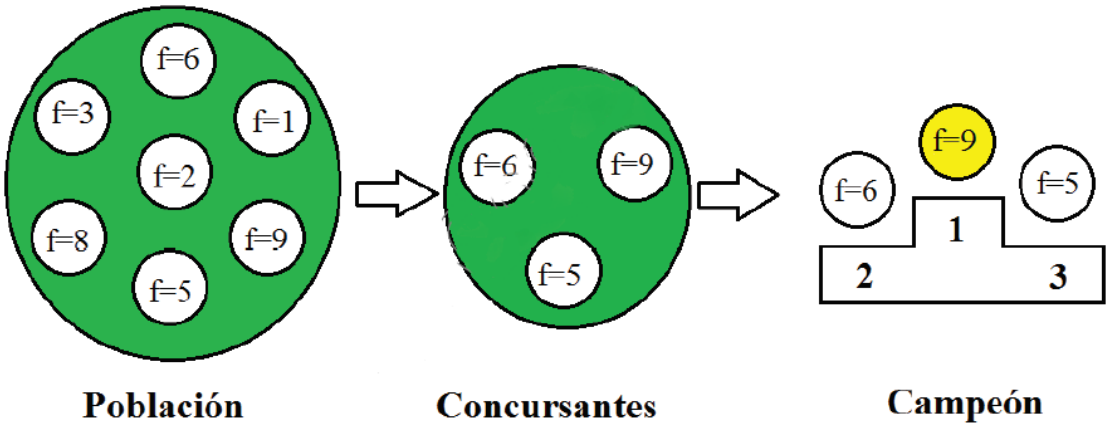


Figura 1. Ejemplo de selección por torneo.

La *selección por torneo*, constituye un procedimiento de selección de padres muy extendido y en el cual la idea consiste en escoger al azar un número de individuos de la población, tamaño del torneo, (con o sin reemplazamiento), seleccionar el mejor individuo de este grupo, y repetir el proceso hasta que el número de individuos seleccionados coincida con el tamaño de la población. Habitualmente el tamaño del torneo es 2, y en tal caso se ha utilizado una versión probabilística en la cual se permite la selección de individuos sin que necesariamente sean los mejores.
Una posible clasificación de procedimientos de selección de padres consistirá en: *métodos de selección dinámicos*, en los cuales las probabilidades de selección varían de generación a generación, (por ejemplo, la selección proporcional a la función objetivo), frente a *métodos de selección estáticos*, en los cuales dichas probabilidades permanecen constantes (por ejemplo, la selección basada en rangos).
Si se asegura que todos los individuos tienen asignada una probabilidad de selección distinta de cero el método de selección se denomina *preservativo*. En caso contrario se acostumbra a denominarlo *extintivo*.

DESARROLLO

En este método de selección no se basa en valores esperados y no requiere por lo tanto de un algoritmo de muestreo. El algoritmo es como sigue:

1. Escoger el tamaño de torneo q (típicamente q=2).
2. Crear una permutación aleatoria de M enteros.
3. Comparar la adaptación de los próximos q- miembros de la población y seleccionar el mejor.
4. Si se acaba la permutación, generar una nueva permutación.
5. Repetir hasta llenar la población.

Algoritmo Pseudocódigo Selección por Torneo

```
Definir tamaño del subconjunto
Calcular el fitness acumulado, fitness total (P_i)
mientras tamaño del subconjunto >0 hacer
  Elegir concursante
fin mientras
Ordenar los concursante de acuerdo a su fitness y elegir el ganador del torneo
Devolver el cromosoma ganador del torneo
```

En el proceso de emular la selección por torneo, puede seguir:

```
t := 0
Inicia-Población P(t)
Evalúa-Población P(t)
Mientras t < N-Generaciones hacer
```

```
P1 := Selección por torneo de (1-r)·p individuos de P(t)
P2 := Selección por torneo de (r·p) individuos de P(t)
P3 := Cruza P2
P4 := Union de P1 y P3
P(t+1) := Muta P4
Evalua-Población P(t+1)
t:= t+1
Fin-Mientras
Devolver el mejor de P(t)
```

Deberá diseñar su propio programa y mostrar pantallas de la ejecución correspondiente. Toda la documentación elaborada deberá subirla a su portafolio de SEDUCA en la fecha indicada.

Conclusiones

Anote de manera breve las principales conclusiones obtenidas al término de esta práctica

_____

_____

_____

_____

Acervo bibliográfico

Básico:

Hilera Gonzales, J. R. & Martinez Hernando V. J,(Eds.2005) Redes Neuronales Artificiales, Fundamentos, Modelos y Aplicaciones, ra-ma (Libro),ISBN 84-7897-155-6, Madrid, España

Anderson, J. A. & Rosenfeld, E. (Eds.) (1990). Neurocomputing: Fundations of Research, Cambridge: MIT Press.

Hopfield, J.J. (1982). Neural networks and physical systems with emergent collective computational abilities, Proceedings of the National Academy of Sciences, 79, 2554-2558.

Freeman, J.A. & Skapura, D. M (1992). Neural Networks: Algorithms, Applications, and Programming Techniques, Addison-Wesley, Massachusetts.

Bibliografía Complementaria

An introduction to Genetic Algorithms. Autor: Melanie Michell. Editorial: MIT Press

Practical Genetic Algorithms. Randy l Haup, sue Ellen Haup Ed.: Wiley

Holland, J.H., "Adaptation in Natural and Artificial Systems", University of Michigan Press, 1975, 211 p.


ESTADOS UNIDOS MEXICANOS
UNIVERSIDAD AUTONOMA
PATRIA • CIENCIA Y TRABAJO
ESTADO DE MEXICO