* 1. Manual técnico
		1. Diagramas a nivel de contexto

Diagrama a nivel de contexto del algoritmo genético de simulación.

# Estadístico

## Estadístico

Algoritmo genético de simulación

0

Información

Parámetros

Información

Gráfico

Impresión

Resultados

*Diagrama de contexto del algoritmo genético de simulación*

*Figura 4.23*Diagrama a nivel de contexto del algoritmo genético para el modelo exponencial o logístico dependientes del tiempo

# Estadístico

## Estadístico

Algoritmo genético del modelo

0

Información

Parámetros

Individuos

Resultados

β0, β1

*Diagrama de contexto del algoritmo genético del modelo dependiente del tiempo*

*Figura 4.24*

Diagrama a nivel de contexto del algoritmo genético para el modelo exponencial o logístico dependiente de varias variables.

# Estadístico

## Estadístico

Algoritmo genético del modelo

0

Información

Parámetros

Individuos

Resultados

*Diagrama de contexto del algoritmo genético del modelo dependiente de varias variables*

*Figura 4.25*

### Diagramas de nivel 0

### Diagramas de nivel 0 del algoritmo genético de simulación



*Diagrama cero del algoritmo genético de simulación*

*Figura 4.26*

### Diagramas de nivel 0 del algoritmo genético del modelo dependiente del tiempo

*Diagrama cero del algoritmo genético de los modelos dependientes del tiempo*

*Figura 4.27*

Diagramas de nivel 1 del proceso de cálculo del algoritmo genético de los modelos dependientes del tiempo.



*Diagrama de nivel 1 del proceso de cálculo del algoritmo genético de los modelos dependientes del tiempo para los beta*

*Figura 4.28*

Diagramas de nivel 0 del algoritmo genético de los modelos dependientes de varias varieables.

*Diagrama cero del algoritmo genético de los modelos dependientes de vaias variables*

*Figura 4.29*

Diagramas de nivel 1 del calculo del algoritmo genético del modelo de selección.



*Diagrama de nivel 1 del proceso de cálculo del algoritmo genético de los modelos dependientes de varias variables para los beta*

###### Figura 4.30

* + 1. Descripción de las variables y procesos de los algoritmos genéticos
			1. Descripción de las variables y procesos del algoritmo genético de simulación

Descripción de las variables globales

|  |  |
| --- | --- |
| *longicromo* | Longitud de los cromosomas |
| *int npob* | Tamaño de la población |
| *int genera* | Número de generaciones |
| *Double Pcro* | Probabilidad de crossover |
| *Double Pmu;* | Probabilidad de mutar |
| *int sele* | Tipo de selección |
| *int promejor* | Propagar el mejor? |
| *int limites* | Crossover solo en los limites? |
| *int Elitismo* | Usar elitismo? |
| *int tipocros* | Tipo de crossover |
| *int tipomuta* | Tipo de mutadcion |
| *int menor* | Mutar sólo los menores |
| *int exigir* | Exigir mutacion |
| *long iteracion* | Contador de las iteraciones |
| *double mayaptitud* | Mayor aptitud |
| *double proaptitud* | Aptitud promedio |
| *double proedad* | Edad promedio  |
| *double suma* | Suma de las aptitudes |
| *double nhombres* | Numero de hombres |
| *double nmujeres* | Numero de mujeres |
| *double nanterior* | Numero anterior de la poblacion |
| *double logi\_apti* | Poblacion para aptitud minima |
| *double nacea* | Límite inferior de las edades en las que se reproducen |
| *double naceb* | Límite superior de las edades en las que se reproducen |
| *double muere* | Edad en la que muere |
| *double nhijos* | Promedio de hijos |
| *double crecimiento* | Crecimiento porcentual año a año |

### Descripción de las estructuras utilizadas

*struct genetico {*

*char cromosoma[longicromo+1];*

*char cromosoma2[longicromo+1];*

*double geno;*

*double aptitud;*

*double dispo;*

*};*

*Estructura de tipo genético*

*Cuadro 4.1*

Estructura de caracter genetico

Contiene un individuo de tipo cadena de caracteres de longitud longicromo+1, la cual se le denomina cromosoma.

Contiene un individuo seleccionado del mismo tipo que el individuo de trabajo, se denomina cromosoma2 y se utiliza en la selección.

El genotipo en la codificación a base decimal del cromosoma, y luego se hace la conversión a un intervalo. Esta es de tipo double.

La aptitud, es el genotipo evaluado en la función de aptitud, y es de tipo double.

La aptitud relativa es de tipo double, y es la aptitud en un intervalo de cero a uno.

*struct datos\_persona {*

*int edad\_actual;*

*int edad\_limite;*

*char sexo;*

*int reproa;*

*int reprob;*

*int num\_hijos;*

*int num\_tot\_hijos;*

*};*

*Estrctura de tipo datos de la persona*

*Cuadro 4.2*

Estructura con datos personales

Contine la edad actual del individuo

La edad de fallecimiento del individuo

El sexo del individuo, que puede ser masculino o femenino

El límite inferior del intervalo en que puede reproducirse un individuo

El límite superior del intervalo en que puede reproducirse un individuo

El número de hijos qe actualmente tiene

El número total de hijos que tendrá dicho individuo

*struct persona {*

*genetico gene;*

*datos\_persona datos;*

*};*

*Estructura de tipo persona*

*Cuadro 4.3*

Estructura de persona

Parte reproducitiva del individuo, que posee todas las cualidades del algoritmo genético.

Parte de los datos personales del individuo.

*struct poblacion {*

*persona info;*

*poblacion \*sig;*

*}\*Hpob,\*Mpob;*

*Estructura de tipo población*

*Cuadro 4.4*

Estructura de poblacion con personas

Contiene cada persona de la población enlazadas mediante un puntero al siguiente individuo y está separada entre hombres y mujeres.

*struct selec {*

*double acu;*

*int pos;*

*selec \*sig;*

*};*

*Estructura de tipo selección*

*Cuadro 4.5*

Estructura para el método de selección de la Rued de la Ruleta

Contiene la función acumulada de aptitudes de los individuos, la posición en la que será asignado el individuo y unPuntero al siguiente selecionado.

### Descripción de los procedimientos y funciones

|  |  |
| --- | --- |
| *AGpob(void);* | Constructor del Agpob, algoritmo genético de simulación poblacional. Inicializa el número de iteraciones. |
| *double normal(double,double);* | Función para generar números aleatorios normales N(μ,σ2). |
| *void lee\_configuracion(void);* | Lee la configuración ingresada del algoritmo genético. |
| *void lee\_parametros(void);* | Lee los parametros ingresados del modelo poblacional. |
| *void generainicial(void);* | Genera la poblacion inicial, incluyendoles un genotipo y una aptitud a cada uno de los individuos. |
| *double aptitud(double);* | Calcula la aptitud de cada individuo de la población. |
| *long random(long);* | Genera un número aleatorio uniforme U(0,b). |
| *int muerte(double);* | Genera un número aleatorio con la distribución de edad límite de vida. |
| *double U01(void);* | Genera un número aleatorio de la distriubución uniforme U(0,1). |
| *int converge(void);* | Verifica si termina la ejecución del Algoritmo genético. |
| *int puede\_reproducirse(poblacion \*);* | Verifica si un indivduo puede reproducirse. |
| *long num\_reproduce(int);* | Halla el número de individuos que pueden reproducirse. |
| *void halla\_aptitud01(void);* | Halla las aptitudes entre 0 y 1. |
| *void seleccion\_Torneo(long,long);* | Selecciona los padres por medio del método del Torneo. |
| *void seleccion\_Ruleta(int,long);* | Selecciona los padres por medio del método de la Rueda de la Ruleta. |
| *void crossover1(void);* | Realiza el crossover en un punto. |
| *void crossoverU(void);* | Realiza el crossover Uniforme. |
| *void mutacion1(int);* | Realiza la mutacion en un punto. |
| *void mutacionU(int);* | Realiza la mutacion Uniforme. |
| *void pasa\_tiempo(void);* | Función que envejece los individuos de la población. |
| *void elimina\_viejos(void);* | Elimina los individuos que pasan el limite de edad. |
| *double genotipo(char []);* | Genotipo del cromosoma. |
| *void reemplazo(void);* | Remplazo la antigua poblacion. |
| *void halla\_mayproaptitud(void);* | Halla la aptidud promedio y la mayor. |
| *void guardar\_resultados(int);* | Guarda los resultados de una iteracion. |
| *void crear(void);* | Crea una poblacion con individuos. |
| *void destruir(void);* | Destruye una poblacion. |
| *void quitar\_primeroH(void);* | Elimina el primero de la poblacion. |
| *void quitar\_primeroM(void);* | Elimina el primero de la poblacion. |
| *void agregar\_hombre(persona);* | Aagrega una persona. |
| *void agregar\_mujer(persona);* | Agrega una persona. |
| *void ordenar(int,int);* | Ordena la poblacion. |
| *char conjuncion(char,char);* | Operador de conjunción. |
| *char disyuncion(char,char);* | Operador de disyunción. |
| *char no(char);* | Operador de negacion. |
| *char igual(char,char);* | Operador de igualdad. |
| *char diferente(char,char);* | Operador de diferencia. |
| *int \_sele(void);* | Retorna el método de selección. |
| *int \_tipomuta(void);* | Retorna el tipo de mutación. |
| *int \_tipocros(void);* | Retorna el tipo de crossover. |
| *int \_genera(void);* | Retorna el número de generaciones. |
| *int Logistica\_permite(void);* | verifica si puede seguir creciendo. |

* + - 1. Descripción de las variables y procesos del algoritmo genético de Optimización para los modelos dependientes del tiempo

### Descripción de las variables globales

|  |  |
| --- | --- |
| *MAXLON* | Número maximo de la longitud de los cromosomas |
| *double Y[3]* | Vector que contiene sumas propias de la función de adaptación |
| *double SCR* | Suma cuadratica de refresión |
| *double R2* | Coeficiente de Determinación |
| *long n* | Número de datos |
| *long iter* | Iteraciones |
| *long tambus* | Tamaño de búsqueda paralela |
| *double pcross* | Probabilidad de crossover |
| *double pmuta* | Probabilidad de mutacion |
| *tope* | Límite superior del creciemiento poblacional |
| *double b0a* | Límite inferior del intervalo de búsqueda del parámetro β0 |
| *double b0b* | Límite superior del intervalos de búsqueda del parámetro β0  |
| *double b1a* | Límite inferior del intervalos de búsqueda del parámetro β1 |
| *double b1b* | Límite superior del Intervalo de búsqueda del parametro β1 |
| *int tcross* | Tipo de crosover |
| *int tmuta* | Tipo de mutación |
| *int sele* | Tipo de selección |
| *BOOL extremos*  | Utilizar crossover únicamente en los extremos |
| *BOOL menor* | Utilizar mutación únicamente en los de menor aptitud |
| *BOOL mejor* | Promover el mejor individuo |
| *BOOL elitismo* | Utilizar elitismo |
| *int longitud* | Longitud de los cromosomas |
| *double mayor* | Mayor aptitud |
| *double prome* | Aptitud promedio |
| *double mayor\_a* | β0 de la mayor aptitd |
| *double mayor\_b* | β1 de la mayor aptitd |

### Descripción de las estructuras utilizadas

*struct persona {*

*int cromo1[MAXLON+1];*

*int cromo2[MAXLON+1];*

*double ada;*

*double dis;*

*double a;*

*double b;*

*};*

*Estructura de tipo persona*

*Cuadro 4.6*

Estructura de tipo persona, nodos de búsqueda

Contiene al individuo o nodo que lo describo como cromo1, y el individuo seleccionado para la reproduccón en el crossover, que lo identifico como cromo2. La adaptación del individuo, que en este caso es la suma de cuadrados del error, y la disposición del individuo, que es la función acumulada de las aptitudes. Además del β0 del individuo y el β1 del individuo.

*struct pobla {*

*persona individuo;*

*pobla \*sig;*

*}\*pob;*

*Estructura de tipo población*

#### Cuadro 4.7

Estructura de tipo población

Contiene al individuo que pertenecie a la población y un puntero al siguiente individuo de la población. Esta lista contendrá n individuos y no va a variar su tamaño durante toda la ejecución.

Descripción de los procedimientos y funciones del algoritmo genético de optimización para los parámetros del modelo dependiente del tiempo.

|  |  |
| --- | --- |
| *void sacar(void)* | Elimina el primero de la lista. |
| *void ruleta(void)* | Método de la ruleta |
| *void torneo(void)* | Método del torneo |
| *void crossover1(double,BOOL)* | Realiza el crossover en un punto |
| *void crossoverU(double)* | Realiza el crossover de tipo uniforme |
| *void mutacion1(double,int)* | Realiza la mutación en un punto |
| *void mutacionU(double,int)* | Realiza la mutación de tipo uniforme |
| *double genotipo(int [])* | Base 10 del cromosoma |
| *double adaptacion(double,double);* | Funcion de adaptacion de los individuos |
| *Double genotipo1(int A[]);* | Genotipo de β0 |
| *Double genotipo2(int A[]);* | Genotipo de β1 |
| *long random(long x);* | Número aleatorio uniforme discreto |
| *long preferencia\_menor(double w);* | Número aleatorio con preferencia en los menores |
| *void crear(void);* | Crea la lista para los individuos iniciales |
| *void destruir(void);* | Destruye la lista |
| *void lee\_configuracion(void);* | Lee la configuración ingresada |
| *void inicial(void);* | Genera la población inicial |
| *void halla\_Sumas(void);* | Halla las sumas para la función de aptitud |
| *void ordenar(void);* | Ordena la lista de menor a mayor |
| *void crossover(void);* | Realiza el crossover |
| *void mutacion(void);* | Realiza la mutacion |
| *void seleccion(void);* | Realiza la selecció |
| *void reemplazo(void);* | Reemplaza la poblacion anterior con la nueva |
| *void maypro(void);* | Halla el mayor y el promedio |
| *double halla\_sce(void);* | Halla la suma de cuadrados del error; |
| *void halla\_resultados(void);* | Halla el Coeficiente de determinación |
| *BOOL converge(long);* | Verifica si la población converge |
| *double \_a1(void);* | Retorna el estimador de β0 |
| *double \_b1(void);* | Retorna el estimador de β1 |
| *double \_SCE1(void);* | Retorna la suma de cuadrados del error |
| *long \_iter(void);* | Retorna el número de iteraciones |
| *double \_SCR(void);* | Retorna la suma de cuadrados de regresión |
| *double \_R2(void);* | Retorna el coeficiente de determinación |

* + - 1. Descripción de las variables y procesos del algoritmo genético de Optimización para los modelos dependientes de varias variables.

### Descripción de las variables globales

|  |  |
| --- | --- |
| *double SCE;* | Suma de cuadrados del error |
| *double MCE;* | Media de cuadrados del error |
| *double R2;* | Coeficiente de Determinación |
| *double Femp;* | Valor de la prueba F de Fisher |
| *long n;* | Número de datos |
| *double tope;* | valor maximo de la funcion |
| *long iter;* | Iteraciones |
| *long tambus;* | tamaño de búsqueda paralela |
| *double pcross;* | Probabilidad de Crossover |
| *double pmuta;* | Probabilidad de Mutacion |
| *int tcross;* | Tipo de crossover |
| *int tmuta;* | Tipo de mutación |
| *int sele;* | Tipo de selección |
| *BOOL extremos;* | Crossover en los extremos |
| *BOOL menor;* | Mutar los menores |
| *BOOL mejor;* | Propagar el mejor |
| *BOOL elitismo;* | Utilizar elitismo |
| *int longitud;* | Longitud de los cromosomas |
| *double mayor;* | Mayor aptitud |
| *BOOL mayor\_cro[MAXLON];* | Cromosoma de mayor aptitud |
| *double prome;* | Aptitud promedio |

### Descripción de las estructuras utilizadas

*struct persona {*

 *BOOL cromo[MAXLON];*

 *double ada;*

 *double dis;*

*};*

*Estructura de tipo persona*

*Cuadro 4.8*

Estructura de tipo persona, nodos de búsqueda

Contiene al individuo o nodo que lo describo como cromo1, y el individuo seleccionado para la reproduccón en el crossover, que lo identifico como cromo2. La adaptación del individuo, que en este caso es la suma de cuadrados del error, y la disposición del individuo, que es la función acumulada de las aptitudes. Además del β0 del individuo y el β1 del individuo.

*struct pobla {*

 *persona individuo;*

 *pobla \*sig;*

*}\*pob2;*

*Estructura de tipo población*

*Cuadro 4.9*

Estructura de tipo población

Contiene al individuo que pertenecie a la población y un puntero al siguiente individuo de la población. Esta lista contendrá n individuos y no va a variar su tamaño durante toda la ejecución.

Descripción de los procedimientos y funciones del algoritmo genético de optimización para los parámetros del modelo dependiente de varias variables.

|  |  |
| --- | --- |
| *void sacar(void);* | Elimina el primero de la lista |
| *void ruleta(void);* | Método de la ruleta |
| *void torneo(void);* | Método del torneo |
| *void crossover1(double,BOOL);* | Crossover en 1 punto |
| *void crossoverU(double);* | Crossover Uniforme |
| *void mutacion1(double,int);* | Mutación en 1 punto |
| *void mutacionU(double,int);* | Mutación Uniforme |
| *double adaptacion(BOOL A[]);* | Funcion de adaptacion |
| *long random(long x);* | Número aleatorio discreto en tre 0 y x |
| *long preferencia\_menor(double w);* | Número aleatorio con preferencia en los menores |
| *void crear(void);* | Crea la lista |
| *void destruir(void);* | Destruye la lista |
| *void lee\_configuracion(void);* | Lee la configuración ingresada por el usuario |
| *void inicial(void);* | Genera la población inicial |
| *void halla\_Sumas(void);* | Halla las sumas necesarias para la regresión |
| *void ordenar(void);* | *Ordena la población de mayor a menor aptitud* |
| *void crossover(void);* | Realiza el crosover |
| *void mutacion(void);* | Realiza la mutación |
| *void seleccion(void);* | Realiza la selección |
| *void reemplazo(void);* | Realiza el reemplazo de la población |
| *void maypro(void);* | Halla el mayor y el promedio |
| *double halla\_sce(void);* | Halla la suma de cuadrados del error |
| *void halla\_resultados(void);* | Halla los resultados |
| *BOOL converge(long);* | Verifica si la población a convergido |
| *double \_SCE1(void);* | Retorna la suma de cuadrados del error |
| *double \_MCE(void);* | Retorna la media de cuadrados del error |
| *double \_R2(void);* | Retorna el coeficiente de determinación |
| *double \_Femp(void);* | Retorna el valor de la prueba F de Fisher  |
| *long \_iter(void);* | Retorna el número de iteraciones |