1er Cuatrimestre 2014

Facultad de Ingeniería

Universidad de Buenos Aires



75.19 – Teoría de la Comunicación

Cátedra Caram – Sarris

Trabajo Práctico – Algoritmos Genéticos

Integrantes

|  |  |  |
| --- | --- | --- |
| Chavar, Hugo | 90541 | hechavar@gmail.com |
| Meller, Diego | 91299 | mellerster@gmail.com |
| Zurita, Stephanie | 91809 | abigail.zurita@gmail.com |

Tabla de contenido

[Introducción a Algoritmos Genéticos 3](#_Toc392079624)

[Uso del Programa 5](#_Toc392079625)

[Código Fuente 6](#_Toc392079626)

# Introducción a Algoritmos Genéticos

Un algoritmo genético es un método adaptativo que usa la búsqueda y el aprendizaje para resolver problemas, basándose en los procesos de evolución genética de los organismos vivos. Su objetivo es encontrar una “buena” solución, no garantiza que sea la óptima.

Para operar, parte de una población inicial y evalúa la aptitud de cada individuo. Si no hay una solución satisfactoria, ejecuta un proceso iterativo de selección (elige los individuos más aptos para que su información permanezca en las siguientes generaciones), cruzamiento (permite que los con más aptitud combinen sus atributos) y mutación (fomenta la diversidad cambiando algunas características al azar); y lo repite las veces necesarias hasta cumplir con la condición de parada. Cada individuo representa una solución posible, y se elige la que tenga mayor aptitud.

El algoritmo toma parámetros para poder definir la población y los pasos de la población estos son:

* Cantidad Individuos En Población: Cuantos individuos se encuentran en la población original.
* Tasa De Selección: Sirve para saber qué cantidad de individuos mueren debido a la selección, son aquellos que no que tengan menor aptitud.
* Tasa De Mutación: Es la probabilidad de mutación de un individuo de la población, si se cumple esta probabilidad muta, sino no.
* Mínima Variación Entre Generaciones: Es la variación entre aptitudes de poblaciones en dos iteraciones seguidas. La misma se usa como condición de corte si es menor a este valor.

Selección

El operador de selección es el encargado de transmitir y conservar aquellas características de las soluciones que se consideran valiosas a lo largo de las generaciones. El principal medio para que la información útil se transmita es que aquellos individuos mejor adaptados (mejor valor de función de adaptación) tengan más probabilidades de sobrevivir.

El algoritmo utiliza la selección por ranking, que consiste en ordenar a los individuos de la población de manera ascendente según la aptitud de los mismos. Según la tasa de selección definida, se eliminan los individuos de menor aptitud, quedando los individuos que poseen mayor aptitud.

Reproducción

El operador de cruce permite realizar una exploración de toda la información almacenada hasta el momento en la población y combinarla para crear mejores individuos. Se aplica escogiendo aleatoriamente dos miembros de la población para intercambiar características.

En el algoritmo para calcular la aptitud del nuevo individuo, es el promedio entre la aptitud de los padres seleccionados.

Mutación

Consiste en una alteración de los valores de un cierto número de características de cada individuo, con la intención de fomentar la variabilidad dentro de la población.

El algoritmo utiliza la mutación simple, donde la probabilidad de mutación es constante durante toda la búsqueda.

# Uso del Programa

Para utilizar el programa se debe compilar usando una maquina virtual java

# Código Fuente

Main.java

package genetico**;**

**import** java**.**util**.**ArrayList**;**

public class Main **{**

public static void main**(**String**[]** args**)** **{**

System**.**out**.**println**(**"TP de Redes Complejas / Sistemas Multiagente"**);**

System**.**out**.**println**(**"Alumnos:"**);**

System**.**out**.**println**(**" Stephanie Abigail Zurita"**);**

System**.**out**.**println**(**" Hugo Chavar"**);**

System**.**out**.**println**(**" Diego Meller"**);**

System**.**out**.**println**();**

System**.**out**.**println**();**

// Defino parametros y veo si hay archivo de configuraci󮍊 int cantidadIndividuosEnPoblacion **=** 100**;**

double tasaDeSeleccion **=** 0.1**;** // tasa de reproduccion es el mismo nro

double tasaDeMutacion **=** 0.01**;**

double minimaVariacionEntreGeneraciones **=** 0.00000001**;**

// Creo la generacion 0 de la poblacion

Poblacion pobl **=** **new** Poblacion**(**cantidadIndividuosEnPoblacion**,**

tasaDeSeleccion**,** tasaDeMutacion**,**

minimaVariacionEntreGeneraciones**);**

ArrayList**<**Double**>** aptitudPoblaciones **=** **new** ArrayList**<**Double**>();**

ArrayList**<**String**>** estAptitudPoblaciones **=** **new** ArrayList**<**String**>();**

**while** **(true)** **{**

pobl**.**seleccion**();**

pobl**.**selectReproduccion**();**

pobl**.**mutacion**();**

aptitudPoblaciones**.**add**(**pobl**.**getAptitudPoblacion**());**

pobl**.**calcAptitudPoblacion**();**

double indMedio **=** pobl**.**getAptitudPoblacion**();**

double indMin **=** pobl**.**getLeastIndividuo**();**

double indMax **=** pobl**.**getBestIndividuo**();**

String res **=** String**.**format**(**"%9s"**,** String**.**format**(**"%.5f"**,** indMedio**))**

**+** "\t "

**+** String**.**format**(**"%9s"**,** String**.**format**(**"%.5f"**,** indMax**))**

**+** "\t "

**+** String**.**format**(**"%9s"**,** String**.**format**(**"%.5f"**,** indMin**));**

estAptitudPoblaciones**.**add**(**res**);**

**if** **(!**pobl**.**esCondicionDeFin**())** **{**

**break;**

**}**

**}**

Individuo mejor **=** pobl**.**getMejor**();**

System**.**out**.**println**(**"El mejor individuo tiene un valor de: "

**+** mejor**.**getValor**());**

System**.**out**.**println**();**

System**.**out

**.**println**(**"Gen.\t Aptitud Media\t Aptitud Peor\t Aptitud Mejor\t Individuo"**);**

int i**;**

**for** **(**i **=** 0**;** i **<** estAptitudPoblaciones**.**size**();** i**++)** **{**

System**.**out**.**println**(**i **+** "\t " **+** estAptitudPoblaciones**.**get**(**i**));**

**}**

**}**

**}**

Poblacion.java

package genetico**;**

**import** java**.**util**.\*;**

public class Poblacion **{**

public int getTamano**()** **{**

**return** tamano**;**

**}**

private ArrayList**<**Individuo**>** listaIndividuos**;**

private int tamano**;**

private double porcSeleccion**;**

private double porcMutacion**;**

private double aptitudPoblacion**;**

private double aptitudAnterior**;**

private double condicionDeFin**;**

// Constructor

Poblacion**(**int tamano**,** double porcSeleccion**,** double porcMutacion**,**

double condDeFin**)** **{**

**this.**tamano **=** tamano**;**

**this.**porcSeleccion **=** porcSeleccion**;**

**this.**porcMutacion **=** porcMutacion**;**

**this.**aptitudPoblacion **=** 0**;**

**this.**condicionDeFin **=** condDeFin**;**

// crear poblacion

listaIndividuos **=** **new** ArrayList**<**Individuo**>();**

int i**;**

**for** **(**i **=** 0**;** i **<** tamano**;** i**++)** **{**

listaIndividuos**.**add**(new** Individuo**(this.**porcMutacion**));**

**}**

**this.**calcAptitudPoblacion**();**

**}**

public double getAptitudPoblacion**()** **{**

**return** **this.**aptitudPoblacion**;**

**}**

public double getCondicionDeFin**()** **{**

**return** **this.**condicionDeFin**;**

**}**

public void setAptitudAnterior**(**double valor**)** **{**

aptitudAnterior **=** valor**;**

**}**

public double getAptitudAnterior**()** **{**

**return** aptitudAnterior**;**

**}**

public void calcAptitudPoblacion**()** **{**

setAptitudAnterior**(this.**getAptitudPoblacion**());**

int i**;**

double result **=** 0**;**

**for** **(**i **=** 0**;** i **<** listaIndividuos**.**size**();** i**++)** **{**

result **=** result **+** listaIndividuos**.**get**(**i**).**getAptitud**();**

**}**

**this.**aptitudPoblacion **=** result **/** listaIndividuos**.**size**();**

**}**

// Seleccion: eliminar las individuos con la menor aptitud o sea con la

// aptitud maxima

public void seleccion**()** **{**

// calcular el numero de individuos de borrar

double aux **=** tamano **\*** porcSeleccion**;**

int elim **=** **(**int**)** aux**;**

Collections**.**sort**(**listaIndividuos**);**

Iterator**<**Individuo**>** ite **=** listaIndividuos**.**iterator**();**

// elimino los"elim" elementos maximos de listaIndividuo,

**while** **(**ite**.**hasNext**())** **{**

ite**.**next**();**

ite**.**remove**();**

elim**--;**

**if** **(**elim **==** 0**)** **{**

**break;**

**}**

**}**

**}**

// eligo individuo al azar por la reproduccion

public void selectReproduccion**()** **{**

int noReproduccion **=** tamano **-** listaIndividuos**.**size**();**

int i**;**

**for** **(**i **=** 0**;** i **<** noReproduccion**;** i**++)** **{**

double random1 **=** Math**.**random**()** **\*** **(**listaIndividuos**.**size**()** **-** 1**);**

double random2 **=** Math**.**random**()** **\*** **(**listaIndividuos**.**size**()** **-** 1**);**

int res1 **=** **(**int**)** random1**;**

int res2 **=** **(**int**)** random2**;**

Individuo ind1 **=** listaIndividuos**.**get**(**res1**);**

Individuo ind2 **=** listaIndividuos**.**get**(**res2**);**

Individuo indHijo **=** ind1**.**reproducirCon**(**ind2**);**

listaIndividuos**.**add**(**indHijo**);**

**}**

**}**

// calcular condicion de fin

public boolean esCondicionDeFin**()** **{**

double aptitud **=** **this.**getAptitudPoblacion**();**

double aAnterior **=** getAptitudAnterior**();**

double limit **=** getCondicionDeFin**();**

double check **=** Math**.**abs**(**1 **-** **(**aptitud **/** aAnterior**));**

**return** **(**check **>=** limit**);**

**}**

public void mutacion**()** **{**

int i**;**

**for** **(**i **=** 0**;** i **<** listaIndividuos**.**size**();** i**++)** **{**

listaIndividuos**.**get**(**i**).**mutar**();**

**}**

**}**

public void mostrarIndividuos**()** **{**

Collections**.**sort**(**listaIndividuos**);**

Iterator**<**Individuo**>** ite **=** listaIndividuos**.**iterator**();**

**while** **(**ite**.**hasNext**())** **{**

Individuo s **=** ite**.**next**();**

System**.**out**.**println**(**s**);**

**}**

**}**

public double getBestIndividuo**()** **{**

int i**;**

double max **=** listaIndividuos**.**get**(**0**).**getAptitud**();**

double act**;**

**for** **(**i **=** 1**;** i **<** listaIndividuos**.**size**();** i**++)** **{**

act **=** listaIndividuos**.**get**(**i**).**getAptitud**();**

**if** **(**act **>** max**)** **{**

max **=** act**;**

**}**

**}**

**return** max**;**

**}**

public double getLeastIndividuo**()** **{**

int i**;**

double min **=** listaIndividuos**.**get**(**0**).**getAptitud**();**

double act**;**

**for** **(**i **=** 1**;** i **<** listaIndividuos**.**size**();** i**++)** **{**

act **=** listaIndividuos**.**get**(**i**).**getAptitud**();**

**if** **(**act **<** min**)** **{**

min **=** act**;**

**}**

**}**

**return** min**;**

**}**

public Individuo getMejor**()** **{**

Collections**.**reverse**(**listaIndividuos**);**

Individuo ind **=** listaIndividuos**.**get**(**0**);**

**return** ind**;**

**}**

**}**

Individuo.java

package genetico**;**

**import** java**.**util**.**Random**;**

public class Individuo **implements** Comparable**<**Object**>** **{**

private double aptitud**;**

private float valor**;**

private double probabilidadMutacion**;**

public static Random rand **=** **new** Random**();**

public double getAptitud**()** **{**

**return** aptitud**;**

**}**

public void setValor**(**float valor**)** **{**

**this.**valor **=** valor**;**

**}**

public float getValor**()** **{**

**return** valor**;**

**}**

public Individuo**(**double probabilidadMutacion**)** **{**

**this.**probabilidadMutacion **=** probabilidadMutacion**;**

**this.**valor **=** rand**.**nextFloat**()** **\*** 200 **-** 100**;** // ramdom entre -100 y 100

calcularAptitud**();**

**}**

public Individuo**(**float valor**,** double probabilidadMutacion**)** **{**

**this.**valor **=** valor**;**

**this.**probabilidadMutacion **=** probabilidadMutacion**;**

calcularAptitud**();**

**}**

public void calcularAptitud**()** **{**

**this.**aptitud **=** Math**.**pow**(this.**valor**,** 2**)** **+** 4 **\*** **this.**valor **-** 4**;**

**}**

public void mutar**()** **{**

**if** **(**rand**.**nextDouble**()** **<** **this.**probabilidadMutacion**)** **{**

**this.**realizarMutacion**();**

calcularAptitud**();**

**}**

**}**

public void realizarMutacion**()** **{**

// realizo una intercambio de 3 bits de los 32 que tiene un float

// los la posicion de los bits a cambiar se elije de manera aleatoria

// con una distribucion normal

int valorComoBitsDeEntero **=** Float**.**floatToIntBits**(this.**valor**);**

int aux **=** 0**;**

int cambiosValidos **=** 3**;**

**while** **(**cambiosValidos **>** 0**)** **{**

// los bits mas probables de ser cambiados

// son los centrales

int posicion **=** **(**int**)** **(**3 **\*** rand**.**nextGaussian**()** **+** 15**);**

aux **|=** 1 **<<** posicion**;**

cambiosValidos**--;**

**}**

valorComoBitsDeEntero ^= (aux & 0xffffffff);

this.valor = Float.intBitsToFloat(valorComoBitsDeEntero);

calcularAptitud();

}

@Override

public String toString() {

return "Individuo [ valor=" + String.format("%.5f", this.valor)

+ ",aptitud= " + String.format("%.5f", this.aptitud) + "]";

}

public Individuo reproducirCon(Individuo other) {

Individuo hijo = new Individuo(

(this.getValor() + other.getValor()) / 2,

this.probabilidadMutacion);

return (hijo);

}

@Override

public int compareTo(Object obj) {

final int BEFORE = -1;

final int EQUAL = 0;

final int AFTER = 1;

if (this == obj) {

return EQUAL;

}

if (obj == null) {

return EQUAL;

}

if (!(obj instanceof Individuo)) {

return BEFORE;

}

Individuo other = (Individuo) obj;

if (this.getAptitud() > other.getAptitud())

return BEFORE;

if (this.getAptitud() < other.getAptitud())

return AFTER;

return EQUAL;

}

}