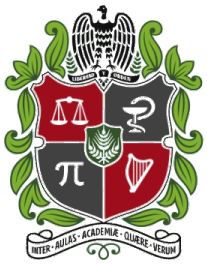

UNIVERSIDAD
NACIONAL
DE COLOMBIA

Computación Evolutiva (2019766) II Semestre 2019

Proyecto de Programación Genética

Profesores: *Edwin Camilo Cubides & Jonatan Gómez*
{eccubidesg,jgomezpe}@unal.edu.co

1. Descripción del problema

Dados los parámetros que se describen más adelante, diseñar un programa que permita inducir un programa funcional escrito en el lenguaje de programación $\mathcal{SIPRES}$ que use algoritmos evolutivos para obtener una solución al problema de generar un programa que permita deducir los ejemplos positivos que son especificados como un parámetro del programa.

Los parámetros estarán especificados en un archivo de texto de la siguiente manera:

i) Un entero que indique la cantidad máxima de ecuaciones que componen el programa.

Ejemplo 1.

```
3
```

ii) Un entero que indique la cantidad máxima de términos que pueden componer las ecuaciones del programa.

Ejemplo 2.

```
10
```

iii) Una cadena que contiene la información de los ejemplos positivos, separados por saltos de linea (\n o ↵), donde cada ejemplo contiene la función a inducir junto con los argumentos en los cuales se evalúa la función y el resultado de la evaluación de la función.

Ejemplo 3.

```
String examples =
"
geq(0,1) = false
geq(0,0) = true
geq(1,0) = true
geq(1,1) = true
geq(1,2) = false
geq(2,1) = true
geq(2,5) = false
geq(5,2) = true
geq(3,3) = true
";
```

El aspecto del archivo con la especificación anterior es el siguiente

```
3
10
geq(0,1) = false
geq(0,0) = true
geq(1,0) = true
geq(1,1) = true
geq(1,2) = false
geq(2,1) = true
geq(2,5) = false
geq(5,2) = true
geq(3,3) = true
```

Otros ejemplos de archivos de especificación de parámetros son:

```
2
5
then(true,true) = true
then(true,false) = false
then(false,true) = true
then(false,false) = true
```

```
3
10
min(0,0) = 0
min(0,1) = 0
min(1,0) = 0
min(1,1) = 1
min(5,2) = 2
min(2,5) = 2
min(3,3) = 3
min(4,1) = 1
min(1,4) = 1
```

Como ayuda para la inducción utilizar el siguiente proyecto NetBeans el cual es el interprete del lenguaje *SIPRES* que se puede descargar desde el siguiente enlace

http://www.alife.unal.edu.co/~eccubidesg/computacion-evolutiva/SIPRESInterpreter.zip

Para obtener las funciones, aridades, terminales y para evaluar los programas obtenidos (funciones) se pueden utilizar las siguiente clase junto con sus métodos Java:

- Para obtener las funciones (definidas y constructoras), los terminales y las variables, se puede utilizar una instancia de la clase `Extractor` del lenguaje Java definida en el paquete `sipres.interpreter` y cuyo único construtor tiene un argumento que es una cadena en la cual se especifican los ejemplos positivos dados en el archivo de entrada.

  Ejemplo 4.

  ```
  Extractor ext = new Extractor(examples);
  ```

- Para obtener las funciones definidas se puede utilizar el método `getTableMainFunctors()` que retorna un conjunto (`HashSet<String>`).

  Ejemplo 5.

  ```
  ext.getTableMainFunctors() = [geq]
  ```

- Para obtener las funciones definidas y constructoras junto con su respectiva aridad se puede utilizar el método `getTableFunctors()` que retorna un mapa (`HashMap<String, Integer>`).

  Ejemplo 6.

  ```
  ext.getTableFunctors() = {geq=2, s=1}
  ```

- Para obtener los terminales se puede utilizar el método `getTableTerminals()` que retorna un conjunto (`HashSet<String>`).

Ejemplo 7.

```
ext.getTableTerminals() = [0, 1, 2, 3, 5, false, true]
```

- Para generar el posible conjunto de variables a utilizar por el método evolutivo se puede utilizar el método `getTableVariables()` que retorna un conjunto (`HashSet<String>`).

  Ejemplo 8.

```
ext.getTableVariables() = [A, B]
```

2. Entrega y sustentación del proyecto

Para la entrega y la sustentación del proyecto es necesario diseñar una presentación de diapositivas en donde se describa el algoritmo genético construido, es decir, hay que especificar las distintas componentes del algoritmo, como son:

1. La representación de los individuos (fenotipo y genotipo).
2. La generación de la población inicial.
3. La estrategia de selección.
4. La estrategia de reemplazo.
5. Los operadores genéticos.
6. La función de aptitud o de ajuste (*fitness function*).

Después de realizar la descripción del algoritmo mediante una exposición y defensa individual, es necesario presentar la ejecución del algoritmo sobre el conjunto de datos de entrenamiento y luego sobre el conjunto de datos de prueba.