Proyecto 1: Aproximación de la complejidad de Kolmogorov mediante Algoritmos Genéticos Eclécticos (EGA).

Eduardo David Martínez Neri

Instituto Tecnológico Autónomo de México

***Resumen*—Se presenta una forma de aproximar la complejidad de Kolmogorov, mediante Algoritmos Genéticos Eclécticos (EGA), codificando máquinas de Turing aleatorias como individuos en el algoritmo genético, y mediante la convergencia del EGA, obtener el mejor individuo que nos permita representar un texto.**

***Keywords*—EGA, Kolmogorov, complexity, Turing**

I. Introducción

En teoría de la información, la **complejidad de Kolmogorov** de un objeto, p.ej. un texto, se puede definir como la máquina de Turing más pequeña o un programa de computadora, que pueda producir ese texto como salida.
Es decir, se busca poder describir de forma lo más breve posible como está constituida una cadena. Tomemos como ejemplo las cadenas 'ABABABABABABAB' y '2AT3C45TRE2RE23', una descripción aceptable de la primer cadena sería: 'Escribe 7 veces AB', mientras que para la segunda cadena no hay una descripción breve fácil de observar.
Su nombre se debe a Andrey Kolmogorov, quien la publicó en 1963 [1]. Aunque es un concepto intuitivo, calcular la complejidad de Kolmogorov para una cadena s, es incomputable, es decir, es imposible escribir un programa al cual le pasemos como parámetro una cadena s, y este nos dé como resultado el programa más pequeño que describa a esa cadena.
De manera ingenua podríamos pensar en un programa que calculara la complejidad de Kolmogorov, haciendo pruebas sobre todos los posibles programas, iniciando del más corto y comparando si su salida es igual a s, como a continuación:

```
def KolmogorovComplexity(string s):
 for i = 1 to infinity:
  for each string p of length exactly i
   if isValidProgram(p) and
    evaluate(p) == s
    return i
```

Esto no funcionaría, debido a que algunos de los programas p, no terminarán, podrían ejecutar loops infinitos, esto nos conduciría al problema de la parada (Halting problem)[2], y por ende nuestro programa no terminaría. De hecho, el problema de calcular la complejidad de Kolmogorov con el problema de la parada son Turing-equivalentes.

I-A. Maquina de Turing

Una máquina de Turing es un dispositivo formado por un cabezal de lectura y escritura, una cinta infinita y una tabla de estados, como se puede observar en la figura 1. Fue definida por Alan Turing [2], como un concepto hipotético, que representa a una computadora, permitiendo así simular cualquier algoritmo de computación y ayuda a entender los límites del cálculo mecánico, introduciendo la noción de complejidad computacional.

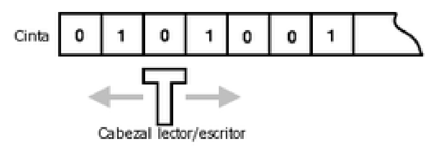

Figura 1. Representación gráfica del cabezal y la cinta de una máquina de Turing.

La tabla de estados, ejemplificada en la Tabla I, determina el cómputo de la máquina.

Estado, valor	Nuevo Valor	Direccion	Nuevo estado
Estado, (0/1)	(0/1)	(Izquierda/Derecha)	Estado

Cuadro I. Tabla de estados

I-B. Algoritmos genéticos

Son algoritmos basados en el proceso genético de los organismos vivos, los cuales a partir del teorema de Rudolph se sabe que convergen a un óptimo global[3].
Una gran ventaja es que no imponen precondiciones en la función a optimizar, p.ej. diferenciabilidad, continuidad, convexidad, forma cerrada, usuales en metodos clásicos. Es por esto que se han vuelto comunes en problemas de optimización.
Son meta-heurísticos, es decir, de acuerdo a sus parámetros podemos obtener toda una familia de algoritmos, en este caso nos enfocaremos en Algoritmos Genéticos Eclécticos (EGA's), ya que se ha comprobado que tienen un rendimiento superior al resto [4].

Los EGA se caracterizan por tener selección determinista, cruzamiento anular, mutación uniforme y elitismo total. De

forma muy breve esto se traduce en: Los $2n$ individuos de las últimas generaciones son ordenados de mejor a peor, y solo los mejores n sobreviven.
Posteriormente los individuos son seleccionados de forma determinista para cruzamiento, el mejor con el peor, el segundo mejor con el segundo peor, y así, hasta crear n nuevos individuos, este cruzamiento se realiza de forma anular (Fig 1) es decir se toma aleatoriamente un punto de corte del individuo1 y se intercambia con el mismo punto de corte del individuo2 con longitud $L/2$, como se puede observar en la figura 2.

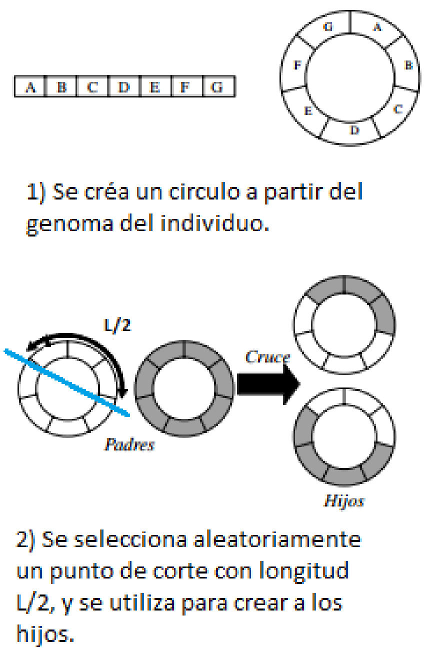

Figura 2. Cruzamiento anular

Los mejores individuos son retenidos y mezclados con los peores, y por varias generaciones los sobrevivientes se convierten en una elite de tamaño n para todo el proceso. El cruzamiento y mutación son probabilistas y se definen a través de los parámetros Pc y Pm.

El algoritmo correspondiente a los EGA es:

1. Generate a random population of size n
2. Evaluate the population.
3. For i=1 to n/2 perform Annular Crossover between individuals i and n-i+1
4. Perform Mutation
5. Steps 3 and 4 yield n new individuals; evaluate them.
6. Sort the accumulated 2n individuals by their fitness, ascending
7. If G is reached return the best individual and stop
8. Discard the worst n individuals
9. Duplicate the Population
10. Go to 3

II. Metodología

Para solucionar el problema y escribir un programa que aproxime la complejidad de Kolmogorov usando el algoritmo genético ecléctico, se utilizó el siguiente algoritmo:

1. Expresar una Máquina de Turing (MT) como una cadena de 1,024 bits, p.ej. tabla II
2. Alimentar la cadena deseada (sea ”T”)
3. Preparar una cinta en ceros de longitud 10,000 (sea ”S”)
4. Evolucionar la MT que maximice el parecido entre ”Sz ”T”

Estado	Valor Cinta	Escribe	Mueve	Siguiente Estado
0	0	1 bit	1 bit	6 bits
0	1	1 bit	1 bit	6 bits
1	0	1 bit	1 bit	6 bits
1	1	1 bit	1 bit	6 bits
2	0	1 bit	1 bit	6 bits
2	1	1 bit	1 bit	6 bits
...	0	...	...	...
...	1	...	...	...
62	0	1 bit	1 bit	6 bits
62	1	1 bit	1 bit	6 bits
63	0	1 bit	1 bit	6 bits
63	1	1 bit	1 bit	6 bits

Cuadro II. Tabla de estados

Como se puede apreciar por la codificación utilizada, existen 2^{64} (10^{19}) MTs posibles para este problema.

Como función de fitness en el EGA, se utilizó distancia de Hamming [5], que es el número de posiciones en la que los símbolos entre 2 cadenas son diferentes.

p.ej. La distancia de Hamming entre ’010**1**010**1**’ y ’010**0**010**0**’ es 2.
La distancia de Hamming entre ’**00**0**1**11’ y ’**11**0**0**11’ es 3.

Algoritmo para calcular la distancia de Hamming, para cadenas de igual longitud.

```
int hamming=0;
for(int j=0;j<cadena1.length;j++){
    if(cadena1[i] != cadena2[i])
    {
        hamming++;
    }
}
```

III. Resultados

En la ejecución de algoritmo con cadena: ’AAAAAAAAAAAAAAAAAAAA’, utilizando un algoritmo genético con 500 individuos, 500 generaciones, Pc:0.9, Pm:0.01 y una cinta de 3000 posiciones. Se obtiene una complejidad de Kolmogorov de 160 bits, con 10 estados de la MT y un fitness de 0, es decir la cadena resultante es exactamente igual a la requerida, figura 4.

Una cadena un poco más complicada: ’12345’, con los mismos parametros, excepto que se utilizaron 2000 generaciones. Se obtiene una complejidad de Kolmogorov de 128 bits, con 8 estados de la MT, y un fitness de 6, es decir, la cadena resultante es diferente a la requerida por 6

```
Mejor Fitness: 0.0
Cadena objetivo: AAAAAAAAAAAAAAAAAAAA
Cadena original, codificada:
0100000101000001010000010100000101000001010000010100000101000001010000010100000101000001010000010100000101000001010000010
Cadena final, decodificada: AAAAAAAAAAAAAAAAAAAA
Cadena final, codificada:
0100000101000001010000010100000101000001010000010100000101000001010000010100000101000001010000010100000101000001010000010
La máquina resultante es:
Estado: 0, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 54, Valor cinta: 1, Escribe: 1, Mueve: I, Sig. estado: 59
Estado: 54, Valor cinta: 0, Escribe: 1, Mueve: D, Sig. estado: 62, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 37
Estado: 62, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 52, Valor cinta: 1, Escribe: 1, Mueve: I, Sig. estado: 1
Estado: 52, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 57, Valor cinta: 1, Escribe: 1, Mueve: D, Sig. estado: 29
Estado: 57, Valor cinta: 0, Escribe: 1, Mueve: D, Sig. estado: 38, Valor cinta: 1, Escribe: 0, Mueve: I, Sig. estado: 41
Estado: 38, Valor cinta: 0, Escribe: 1, Mueve: D, Sig. estado: 25, Valor cinta: 1, Escribe: 1, Mueve: D, Sig. estado: 2
Estado: 25, Valor cinta: 0, Escribe: 1, Mueve: I, Sig. estado: 55, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 44
Estado: 55, Valor cinta: 0, Escribe: 0, Mueve: I, Sig. estado: 47, Valor cinta: 1, Escribe: 0, Mueve: I, Sig. estado: 47
Estado: 47, Valor cinta: 0, Escribe: 0, Mueve: I, Sig. estado: 35, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 0
Estado: 37, Valor cinta: 0, Escribe: 1, Mueve: D, Sig. estado: 0, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 41

Estados de la MT: 10
La complejidad de Kolmogorov, es de: 160 bits
```

Figura 3. Ejecucion del programa con 20A's

posiciones.

Para la cadena 'ABCDABCDABCD', con 5000 generaciones en el algoritmo genético, se obtuvo una complejidad de 256 bits, con 16 estados de la MT, y una diferencia de 9 posiciones entre las cadenas.

```
Mejor Fitness: 9.0
Cadena objetivo: ABCDABCDABCD
Cadena original, codificada:
010000010100001001000011010001000100000101000010010000110100010001000001010000100100001101000100
Cadena final: C@C@C@C@C@C@
Cadena final, codificada:
010000110100000001000011010000000100001101000000010000110100000001000011010000000100001101000000
La máquina resultante es:
Estado: 0, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 7, Valor cinta: 1, Escribe: 1, Mueve: I, Sig. estado: 15
Estado: 7, Valor cinta: 0, Escribe: 1, Mueve: D, Sig. estado: 25, Valor cinta: 1, Escribe: 0, Mueve: I, Sig. estado: 31
Estado: 25, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 58, Valor cinta: 1, Escribe: 0, Mueve: I, Sig. estado: 15
Estado: 58, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 55, Valor cinta: 1, Escribe: 1, Mueve: I, Sig. estado: 5
Estado: 55, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 28, Valor cinta: 1, Escribe: 0, Mueve: I, Sig. estado: 51
Estado: 28, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 59, Valor cinta: 1, Escribe: 1, Mueve: D, Sig. estado: 32
Estado: 59, Valor cinta: 0, Escribe: 1, Mueve: D, Sig. estado: 27, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 14
Estado: 27, Valor cinta: 0, Escribe: 1, Mueve: D, Sig. estado: 30, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 13
Estado: 30, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 35, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 53
Estado: 35, Valor cinta: 0, Escribe: 1, Mueve: I, Sig. estado: 30, Valor cinta: 1, Escribe: 1, Mueve: D, Sig. estado: 3
Estado: 3, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 2, Valor cinta: 1, Escribe: 1, Mueve: D, Sig. estado: 26
Estado: 2, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 4, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 2
Estado: 4, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 12, Valor cinta: 1, Escribe: 1, Mueve: I, Sig. estado: 12
Estado: 12, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 48, Valor cinta: 1, Escribe: 0, Mueve: I, Sig. estado: 12
Estado: 48, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 11, Valor cinta: 1, Escribe: 0, Mueve: D, Sig. estado: 18
Estado: 11, Valor cinta: 0, Escribe: 0, Mueve: D, Sig. estado: 0, Valor cinta: 1, Escribe: 0, Mueve: I, Sig. estado: 31

Estados de la MT: 16
La complejidad de Kolmogorov, es de: 256 bits

BUILD SUCCESSFUL (total time: 2 minutes 58 seconds)
```

Figura 4. Ejecucion del programa con ABCDABCDABCD

IV. Conclusion

Se ha logrado obtener un buen estimado de la complejidad de Kolmogorov utilizando algoritmos genéticos eclecticos, y se han realizado pruebas con diferentes cadenas, hay diferentes parametros para configurar en el algoritmo, como el número de individuos y el número de generaciones. Es interesante como incrementar estos parametros nos acerca más al objetivo e incluso variar la probabilidad de mutación. Solo se realizaron pruebas con cadenas de longitud pequeña, pero realizando diferentes pruebas con el algoritmo se puede llegar a aproximar cualquier cadena.

Referencias

[1] Andrei N Kolmogorov. On tables of random numbers. *Sankhyā: The Indian Journal of Statistics, Series A*, pages 369–376, 1963.

[2] Alan Mathison Turing. On computable numbers, with an application to the entscheidungsproblem. *J. of Math*, 58(345-363):5, 1936.

[3] Günter Rudolph. Convergence analysis of canonical genetic algorithms. *IEEE transactions on neural networks*, 5(1):96–101, 1994.

[4] Angel Fernando Kuri-Morales, Edwin Aldana-Bobadilla, and Ignacio López-Peña. The best genetic algorithm ii. In *Mexican International Conference on Artificial Intelligence*, pages 16–29. Springer, 2013.

[5] Wikipedia. Hamming distance — wikipedia, the free encyclopedia, 2016. [Online; accessed 19-October-2016].