**INSTITUTO TECNOLÓGICO AUTÓNOMO DE MÉXICO**

**Fundamentos Matemáticos de la Computación**

# Proyecto Final

# Complejidad de Kolmogorov

# “Manual teórico”

**Equipo 5**

**Integrantes**

**Susana Muñoz Acosta 165889**

**Manuel Fernandez Verda 166496**

**Sebastián Valderrábano Cabrera 163791**

**Rubén Romero Ortega 174178**

**13 de mayo de 2020**

## Introducción

La teoría de la computabilidad es la parte de la computación que estudia los problemas de decisión que se pueden resolver con un algoritmo o de manera equivalente con una Máquina de Turing. Una Máquina de Turing (MT) es un dispositivo que a partir de una serie de reglas o estados y una cinta de entrada, genera una cinta de salida. Estas máquinas son diseñadas con el fin de representar cualquier función computable.

La Complejidad de Kolmogorov (CK) es una forma de referirse a la información contenida en un conjunto de datos sin apelar a conceptos probabilísticos. Su definición formal es la longitud (en bits) de la unidad de control de la MT binaria más compacta capaz de reproducir una colección de datos binaria. Esto se puede interpretar como la menor cantidad de líneas de código en un lenguaje de programación o bien la menor cantidad de estados en una máquina de Turing para producir un resultado determinado.

Una función es computable si y sólo si se puede representar con una MT, entonces la Complejidad de Kolmogorov se considera incomputable ya que su cálculo implicaría resolver la función del problema del paro. Por lo tanto, la definición de la CK es teórica porque no es posible asegurar que la MT aludida fuese la más compacta.

## Proyecto

Para este proyecto se nos pidió realizar un programa que calcule la Complejidad de Kolmogorov (CK) de una Máquina de Turing (MT). Aunque no se puede calcular CK por lo descrito anteriormente en la introducción, podemos obtener la mejor MT que minimice el número de estados necesarios para generar una cinta dada a partir de un programa basado en algoritmos evolutivos; es decir, que tenga la menor CK.

Los algoritmos evolutivos son métodos de búsqueda de soluciones y optimización basados en los postulados de la evolución biológica. En estos algoritmos se busca minimizar el valor de la función de ajuste partiendo de una colección inicial de soluciones o individuos plausibles que se evalúan por separado, posteriormente se selecciona un subconjunto adecuado de ellas y se “mezclan” dichas soluciones para después introducir variaciones o mutaciones aleatorias. Finalmente, después de n iteraciones de los pasos anteriormente descritos, el mejor individuo constituye la mejor solución numérica al problema.

En este proyecto los algoritmos evolutivos que se utilizaron para generar las posibles máquinas de Turing que nos dieran el resultado buscado fueron algoritmos genéticos.

La diferencia entre los algoritmos genéticos y los demás algoritmos evolutivos es que los operadores evolutivos operan no sobre los individuos (fenotipo), sino sobre el código en el que dichos individuos se expresan (genotipo).

Para encontrar el resultado se usó Random Mutation Hill Climber (RMH), un algoritmo iterativo que comienza con una solución arbitraria a un problema y luego intenta encontrar una mejor solución variando la solución. En el proyecto, se genera un individuo inicial con un cierto genoma que representaría la MT que buscamos. Dicho genoma tiene una longitud de 1024 bits dado que el número máximo de estados de nuestra Máquina de Turing es de 64 y cada uno de estos consta de 16 bits.

Una vez que este individuo inicial es generado se realizan una serie de evoluciones sobre dicho individuo generando nuevos individuos. Cada uno de estos nuevos genomas son evaluados en la máquina de turing para posteriormente calcular la complejidad de Kolmogorov de cada uno de ellos. Ante cada iteración o “generación” se escoge al individuo con menor CK y menor porcentaje de error; es decir, aquel con que genere una cinta resultante más parecida a la que se busca generar. Una vez que se ha realizado este proceso se imprime la máquina óptima que fue generada y se muestra la cadena que dicha máquina genera.

Para poder llevar esto a cabo la Maquina Universal de Turing evalúa en cada iteración el genoma que se le da y guarda la cinta que se produce con esta máquina de Turing.

## **Conclusiones**

Este proyecto tenía como objetivo realizar un programa que calculara la Complejidad de Kolmogorov de una Máquina de Turing y se logró con el código explicado en el “Manual técnico”. En este documento también se describen las pruebas que hicimos con nuestro código con las cuales llegamos a la conclusión de que para llegar a una aproximación más exacta sería necesario aumentar el tamaño de la cinta inicial, el número máximo de iteraciones de la Máquina de Turing y el número máximo de iteraciones que permita el análisis de Kolmogorov.

El problema en realidad es que cuando se simula una Máquina de Turing se deben acotar tanto el tamaño de la cinta y el número de transiciones, entonces por más que aumentemos el tamaño de los parámetros nunca será suficiente para que todas las pruebas sean resueltas de manera exacta. Como consecuencia podemos decir que la finitud de los sistemas físicos es la que hace que en la práctica nos encontremos con cuestiones incomputables como la Complejidad de Kolmogorov.

**Referencias**

Kuri, A. Cálculo Aproximado de la Complejidad de Kolmogorov [Presentación de PowerPoint].

Kuri, A. Optimización numérica usando algoritmos evolutivos (y más) [Presentación de PowerPoint].

Para elaborar el proyecto también se utilizaron los siguientes códigos elaborados por Ángel Kuri Morales:

- AGF.class
- AGH.java
- EGA20.class
- EGA.java
- N2bS.class
- N2bS.java
- RMF.class
- RMF.java
- RMH20a.class
- RMH20a.java