Tarea 12 - Implementación de Algoritmo UMDA usando el Paradigma Orientado a Objetos

Oscar Esaú Peralta Rosales

Objetivo

- Programar el Algoritmo de UMDA (Univariate Marginal Distribution Algorithm) en el lenguaje C++ a través del paradigma orientado a objetos.

Algoritmo de UMDA

El algoritmo UMDA pertenece a los algoritmos de estimaciones de distribución (EDA's) que basados en modelos probabilísticos ayudan a encontrar óptimos de soluciones.

El principio base de este algoritmo es usar la frecuencia de las componentes de cada individuo de la población que pertenece al grupo candidato de mejores soluciones. Las probabilidades determinadas por cada frecuencia sobre el total de individuos determina la selección o no de algún componente para los nuevos candidatos.

El algoritmo es el siguiente:

```
bits_num <- Número de componentes de los individuos
population_size <- Tamaño de la población
selection_size <- Número de mejores individuos a seleccionar en cada
generación

Population <- InitializePopulation(bits_num, population_size)
EvaluatePopulation(Population)
S_best <- GetBestSolution(Population)

while(!stopCondition())
    selected <- select_kbest(Population, selection_size)
    F <- getFrequencies(selected)
    next_population <- [selected...]
    for i to population_size - size(selected):
        I <- ProbabillisticalSolutionConstruction(frecs);
        add(next_population, I)
    EvaluatePopulation(next_population)
    S_best <- GetBestSolution(Population)
    population <- next_population

return S_best
```

Implementación y Resultados

Se implementaron 4 clases distintas:

- Clase Individual: Abstracción para un individuo de la población
- Clase Population: Abstracción de la entidad población, es una *Clase Compuesta* formada por varios individuos.
- Clase Selection: Abstracción de los métodos de selección y funciones de coste, es una clase formada por métodos estáticos puesto que no hay interacción con atributos de la misma.
- Clase UMDA: Abstracción para el algoritmo *Univariate Marginal Distribution Algorithm*.

Se planteó encontrar soluciones para expresiones matemáticas, se implementaron dos en específico:

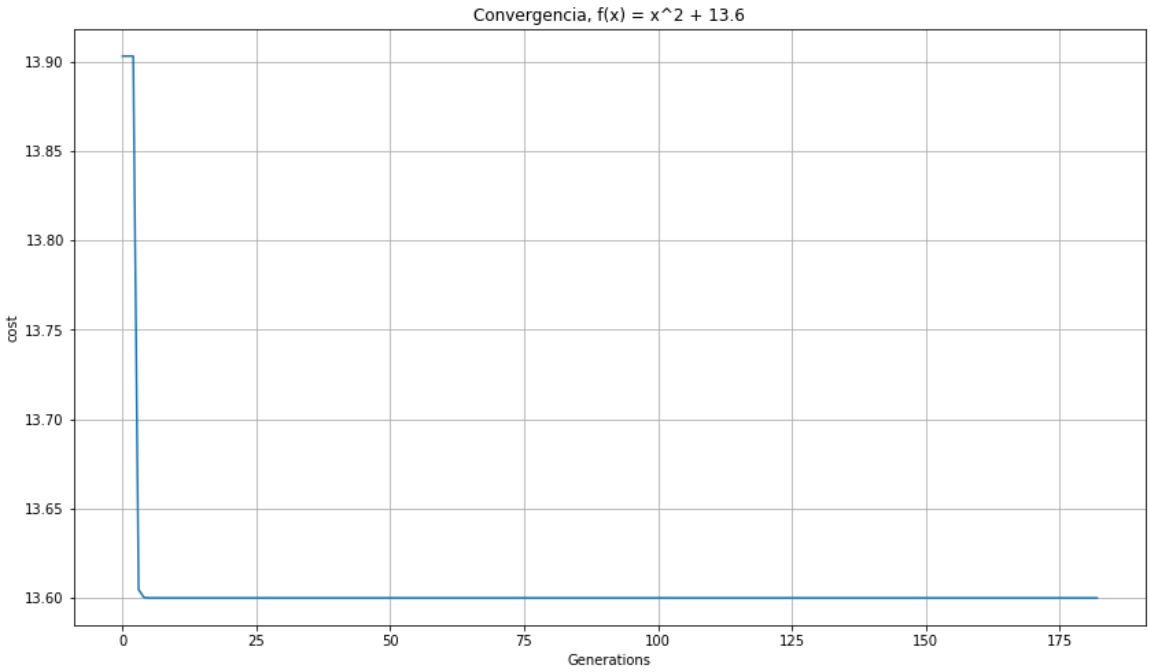
$$f(x) = x^2 + 13.6 \tag{1}$$

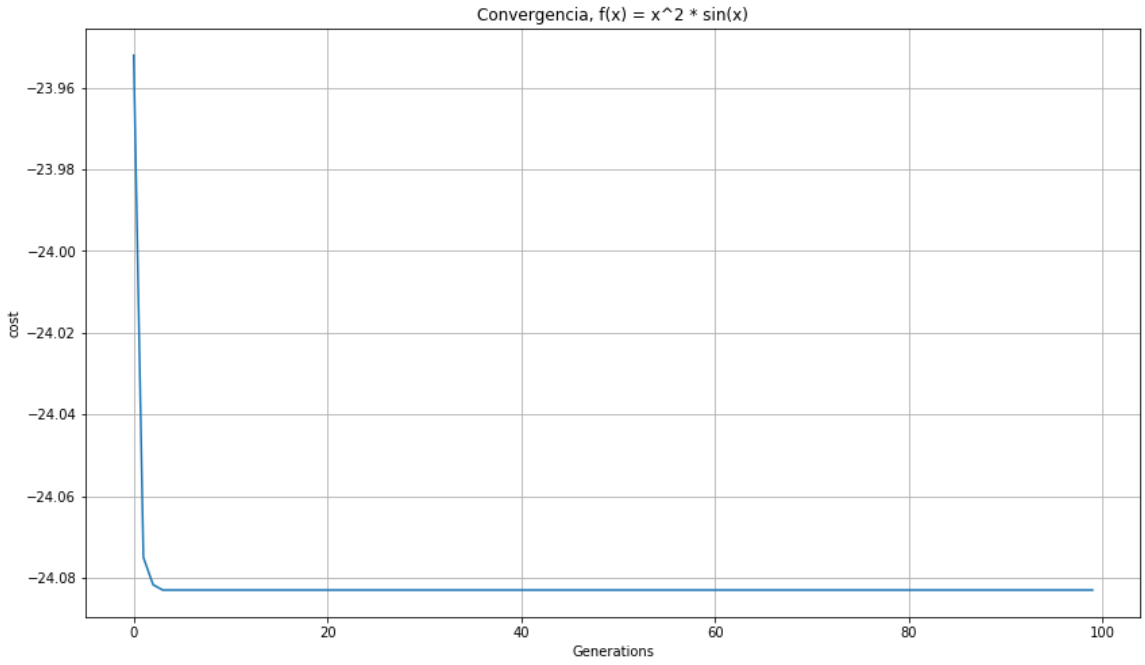
$$f(x) = x^2 sin(x) \tag{2}$$

Dónde cada individuo representa un número dentro de un rango definido de las funciones dónde se espera encontrar un mínimo. Por tanto se realiza un mapeo del valor entero representado por la expresión en bits a el rango correspondiente de la función de interés. En este ocasión los rangos establecidos son de $[-30, 30]$ y $[-4, 7]$ respectivamente.

Las soluciones obtenidas fueron los valores de 13.6 y -24.083 las funciones anteriores.

Las siguientes figuras muestran la convergencia a lo largo de las generaciones:





Conclusiones

La convergencia de las soluciones fue sumamente rápida; 6 generaciones para la primer función y 4 generaciones para la segunda. Se realizaron 3 pruebas más modificando el tamaño de la población, el tamaño de las características de los individuos y el tamaño de la selección.

Tanto para el tamaño de la población (mayor a 10) no se observó ningún cambio pero para tamaño de las características de los individuos (entre 4 y 20) y el número de mejores soluciones a seleccionar (entre 10 y 30) si se afectó en la soluciones. Evidentemente mientras menos mejores soluciones hay la tabla de frecuencias está no contendrá información suficiente y mientras menos características tienen los individuos menor es la precisión de representación y mapeo de los números reales dentro del rango de cada función.

Hay que tomar en cuenta que el problema solucionado es muy sencillo (no se espera tener resultados sorprendentes) y solo es sujeto a prueba para ver la implementación del algoritmo usando el paradigma orientado a objetos.