Algortimo UMDA (POO)

Pablo Antonio Stack Sánchez
Programación y Algoritmos

18 de noviembre de 2019

1. Introducción

1.1. Programación orientada a objetos

Es un paradigma de programación que usa objetos y sus interacciones, para diseñar aplicaciones y programas. Está basado en varias técnicas, incluyendo métodos, herencia, abstracción, polimorfismo y encapsulamiento.

- Clase: Una clase se puede definir de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ella.
- Herencia: Es la facilidad mediante la cual la clase B hereda en ella cada uno de los atributos y operaciones de una clase A, como si esos atributos y operaciones hubiesen sido definidos por la misma B. Por lo tanto, puede usar los mismos métodos y variables registrados como "públicos".
- Objeto: Instancia de una clase.
- Métodos: algoritmo asociado a un objeto (o a una clase de objetos), es lo que el objeto puede hacer.
- Abstracción: Denota las características esenciales de un objeto, donde se capturan sus comportamientos.
- Polimorfismo: Comportamientos diferentes, asociados a objetos distintos, pueden compartir el mismo nombre, al llamarlos por ese nombre se utilizará el comportamiento correspondiente al objeto que se esté usando.
- Encapsulamiento: Significa reunir a todos los elementos que pueden considerarse pertenecientes a una misma entidad, al mismo nivel de abstracción.

1.2. Algoritmo UMDA

UMDA es un algoritmo genético de estimacion de distribución (EDA). Los EDA son algoritmos heurísticos de optimización que basan su búsqueda al igual que los algoritmos genéticos en el caracter estocástico de la misma. Tambien al igual que los algoritmos genéticos los EDA están basados en poblaciones que evolucionan. Sin embargo a diferencia de los algoritmos genéticos en los EDAs la evolución de las poblaciones no se lleva a cabo por medio de los operadores de cruce y mutación. En lugar de ello la nueva población de individuos se muestrea de una distribución de probabilidad, la cual es estimada de la base de datos conteniendo al conjunto de individuos seleccionados de entre los que constituyen la generación anterior.

2. Metodología

Algoritmo UMDA:
1) Inicializar una población binaria de nbits.
2) Convertir a su representación decimal y evaluar la función a minimizar.
3) Ordenar los individuos respecto al valor de la evaluación.
4) Tomar los mejores individuos.
5) Calcular las nuevas probabilidades.
6) Repetir hasta convergencia.

Se programó el UMDA utilizando clases y el paradigma orientado a objetos, se crearon tres clases diferentes:

1. Individuo: Clase que contendrá la secuencia de bits (genes) de cada individuo.
   a) Métodos de la clase:
      1) get_individuo() // regresa los genes del individuo
2. Población: Vector de individuos.
   a) Métodos de la clase:
      1) show() //muestra en consola la población
      2) get_nbits() //regresa el número de bits de los individuos.
      3) get_elementos() // regresa el número de elementos de la población.
      4) get_poblacion() // regresa el vector de individuos correspondiente a la población.
3. UMDA: Clase que se encargará de realizar el algoritmo.
   a) Métodos de la clase:
      1) convierte() //Pasa de una representación binaria a una decimal en los reales.
      2) evalua() // Hace la evaluación a la función que se desea minimizar.
      3) ordena() // Ordena la población de menor a mayor.

4) Probabilidad() // Calcula la probabilidad.
5) get_mapa() // Regresa la población ordenada.
6) get_probabilidad() // Regresa el vector de probabilidades.
7) run() //Ejecuta el algoritmo.

Se fijó la función a minimizar como $f(x) = 2(x-2)^2 + 4$ y el número de iteraciones en 200.

3. Resultados

Se modificó algunas veces la función a minimazar directamente en el código y se probó con distintos números de bits para los individuos, en el cuadro siguiente se muestran los resultados obtenidos.

f(x)	**[a,b]**	**N de bits**	**Tamaño de la población**	**Resultado**
$2(x-2)^2+4$	[-3,3]	4	400	2.25
	[-3,3]	20	400	1.99951
	[-10,10]	20	400	2.00018
$2(x-4)^3+4$	[-5,5]	10	400	-5
	[-10,10]	4	400	-10
	[-3,3]	20	400	-3

Cuadro 1: Resultados obtenidos con UMDA

En la tabla se observa que el algoritmo programado fue capaz de encontrar correctamente el mínimo de las funciones. Entre más grande el n de bits mayor resolución tiene el resultado.

4. Conclusiones

El algoritmo programado es eficiente al minimizar funciones sencillas. El código se podría mejorar haciendo una clase de algoritmos genéticos que tenga los métodos de ordenamiento, cálculo de probabilidades y selección. La clase UMDA sería hija de esta superclase y heredaría todas estas funciones. También se puede crear una clase de funciones de prueba.