Algoritmos Bioinspirados: Evolución Diferencial

Alberto García y Diego Martínez

4 de abril de 2020

1. Introducción

En este trabajo hemos implementado el algoritmo diferencial desarrollado en el paper de Tian, Gao y Dai[1]. Una de las principales características de este algoritmo es la capacidad de autogestión y adaptabilidad a la hora de elegir entre diversidad y convergencia.

A continuación describimos los mecanismos de mutación y recombinación, pasando por alto detalles más concretas del algoritmo como el cálculo de F_1 o $\vec{d}_{r2,G}$.

1.1. Mutación

Los individuos mutados se generan mediante la siguiente fórmula:

$$\vec{v}_{i,G} = \begin{cases} \vec{x}_{\text{rand},G} + F_1(\vec{x}_{g,G} - \vec{x}_{\text{rand},G}) + F_2(\vec{x}_{r1,G} - \vec{d}_{r2,G}) & \text{si rand} < \xi_1 \\ \vec{x}_{\text{cur},G} + F_1(\vec{x}_{g,G} - \vec{x}_{\text{cur},G}) + F_2(\vec{x}_{r1,G} - \vec{d}_{r2,G}) & \text{caso contrario} \end{cases} \tag{1}$$

Donde las variables tienen el siguiente significado:

- ξ_1: Constante definida por el usuario, ≤ 1.
- rand: Variable aleatoria con probabilidad uniforme entre 0 y 1.
- G: Generación a la que pertenecen los individuos.
- $\vec{v}_{i,G}$: Individuo mutado.
- $\vec{x}_{\text{cur},G}$: Individuo a mutar.
- $F_1,\ F_2$: Constantes determinadas por el fitness.
- $\vec{x}_{\text{rand},G}$: Individuo seleccionado al azar.
- $\vec{x}_{g,G}$: *Guiding individual*. Individuo seleccionado al azar entre los individuos con mejor *fitness*.
- $\vec{d}_{r2,G}$ es un vector aleatorio dentro del espacio de búsqueda.

La predisposición hacia la convergencia o la diversidad viene dada por el valor que demos a ξ_1.

El primer término favorece la convergencia: "sustituye" la posición del individuo original por uno aleatorio, forzando que los puntos se mantengan donde está la mayoría.

El segundo término favorece la exploración: "mantiene" la posición original y luego le suma dos perturbaciones: una que lo lleva hacia el *guiding individual* y otra aleatoria.

Nótese que no se están redifiniendo las posiciones de la población original, denotada por $\vec{x}$, sino definiendo una nueva población mutada $\vec{v}$.

2. Estructura del programa

Para escribir el programa hemos modificado bastante el programa inicial dado en clase. El programa se sigue llamando desde `lanzador.R`, que inicializa el problema mediante las funciones del directorio `funciones` y el script `inicia.R` (aunque este ha sido renombrado como `inicializador.R`).

A partir de aquí los scripts se han directamente sustituido o eliminado. Una vez inicializado el problema, `lanzador.R` llama a `evolutivo.R`. Este es el script central del programa, y se encargará de llevar a cabo todos los pasos descritos en el paper de Tian, Gao y Dai[1]. Para ello hará uso de los siguientes scripts:

- `mutacion.R`: este script recibe como argumento una población de individuos y genera una población mutada, sin modificar la población original.
- `crossover.R`: este script toma la población original y la mutada mediante `mutacion.R` y las combina generando un individuo *trial*, en base a la variable *CR*, que determina la probabilidad de un individuo de mutar. El *trial* generado es un individuo que puede tener componentes tanto del individuo original como del individuo mutado.
- `seleccion.R`: este script selecciona los individuos eligiendo entre los individuos originales o los *trials*. Para ello tiene en cuenta el *fitness* de los individuos.

Referencias

[1] M. Tian, X. Gao, and C. Dai, "Differential evolution with improved individual-based parameter setting and selection strategy," *Applied Soft Computing*, vol. 56, pp. 286–297, 2017.