Algoritmos Exactos y Metaheurísticas

Primer Semestre 2025

Universidad Diego Portales
Prof. Víctor Reyes Rodríguez

Objetivos

- Resumen clase anterior
- Otros algoritmos evolutivos

Introducción

- Comienzan desde una población inicial de soluciones P_0
- Luego, iterativamente se **genera** un grupo de individuos a través de una selección de padres, operadores de cruzamiento y operadores de mutación.
- Una vez generada una nueva población, se pasa a la siguiente generación seleccionando un subgrupo de tamaño P_0.

Pseudocódigo general

1. Generación de una población inicial N
2. Evaluación de todos los individuos de la población
3. Mientras no se cumpla el criterio de término (convergencia, generaciones, etc)
    a. Se define si se realizará mutación o cruzamiento, según probabilidad.
    b. Si es cruzamiento, elección de individuos según estrategia correspondiente.
    c. Algunas de las alternativas para generar la siguiente generación.
    d. No olvidar guardar el mejor individuo! (incluido su fitness)
4. Se entregan los resultados de la ejecución

Differential Evolution (DE)

- Corresponde a otro algoritmo evolutivo. Menos popular que GA, pero que muestra buenos resultados en optimización continua.

- Al igual que GA, DE genera una población inicial de soluciones P_0 de tamaño k. Cada individuo corresponde a un vector real x_{ij} de dimensión D.

- Cada individuo es codificado como un vector de números de punto flotante. Cada elemento del vector x_{ij} es generado aleatoriamente en el rango $[x^j_l, x^j_u]$ lo que representa el lower y upper bound de cada variable:

$$x_{ij} = x^l_j + \text{rand}_j[0,1] \cdot (x^h_j - x^l_j), i \in [1,k], j \in [1,D]$$

- rand_j es una variable aleatoria uniformemente distribuida en el rango [0,1].

Differential Evolution (DE)

- La recombinación o cruzamiento funciona de manera distinta a GA. Este se basa en un operador que realiza una combinación lineal.

Input: Parent i, three randomly selected individuals $r_1, r_2, r_3, i \neq r_1 \neq r_2 \neq r_3$.
$j_{rand} = int(rand_i[0, 1].D) + 1$;
For $(j = 1, j \leq D, j++)$ **Do**
  If $(rand_j[0, 1]) < CR)$ or $(j = j_{rand})$ **Then**
    $u_{ij} = v_{ij} = x_{r_3 j} + F.(x_{r_1 j} - x_{r_2 j})$;
  Else
    $u_{ij} = x_{ij}$;
Output: Offspring u_i.

- F (en general 0.8) y CR (en general 0.9) son parámetros en el intervalo [0,1]. La condición $j=j_{rand}$ asegura que al menos una variable del hijo será distinta al padre.

Differential Evolution (DE): Algoritmo

Input: Parameters: F (scaling factor), CR (crossover constant).
Initialize the population (uniform random distribution) ;
Repeat
  For $(i = 1, i \leq k, i++)$ **Do** /* Each individual */
    Mutate and Recombine:
      $j_{\text{rand}} = int(\text{rand}_i[0, 1] \cdot D) + 1$;
      For $(j = 1, j \leq D, j++)$ **Do**
        If $(\text{rand}_j[0, 1]) < CR)$ or $(j = j_{\text{rand}})$ **Then**
          $u_{ij} = v_{ij} = x_{r_3 j} + F \cdot (x_{r_1 j} - x_{r_2 j})$
        Else
          $u_{ij} = x_{ij}$
      Replace:

$$x_i(t+1) = \begin{cases} u_i(t+1) & \textbf{If } f(u_i(t+1)) \leq f(x_i(t)) \\ x_i(t) & \textbf{Otherwise} \end{cases}$$

  End For
Until Stopping criteria /* ex: a given number of generations */
Output: Best population or solution found.

¿Dónde se utiliza?

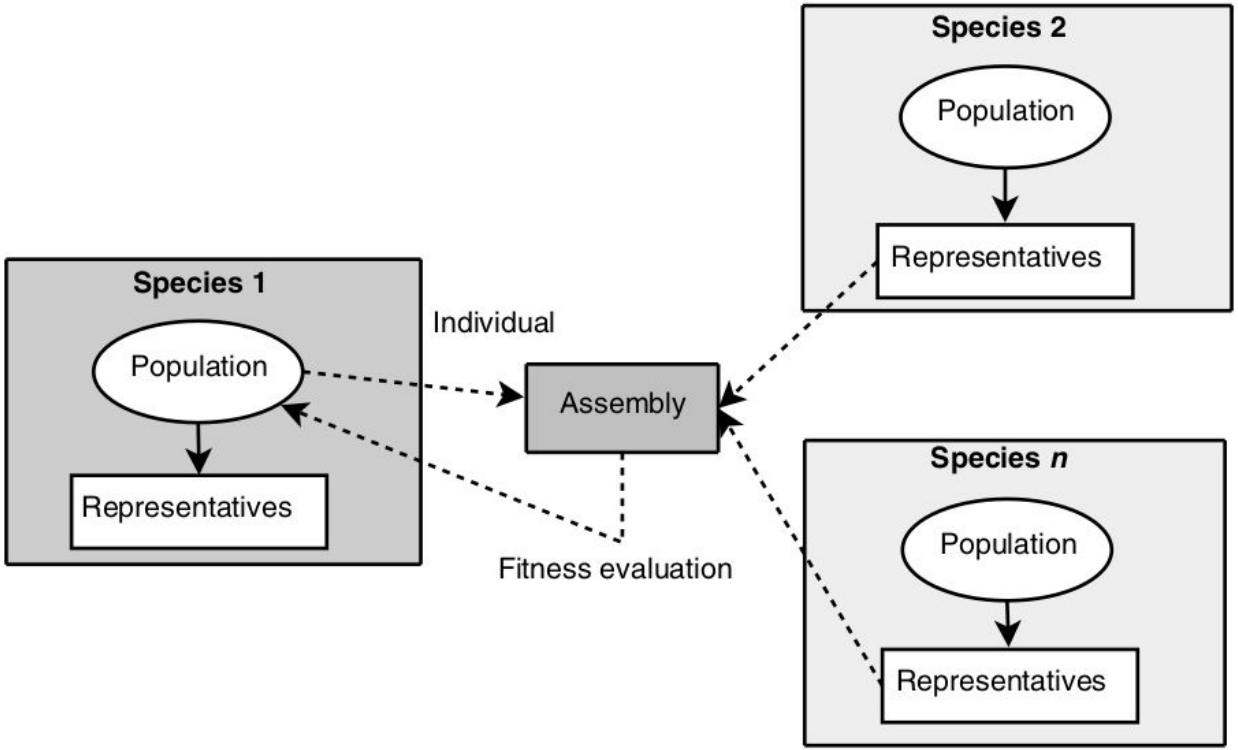
- Optimización continua con restricciones.
- Problemas en donde existan funciones no diferenciables.
- Problemas multi-objetivo.

Algoritmos de coevolución cooperativa

Algoritmos de coevolución cooperativa

- Coevolución es definida como una evolución complementaria de especies cercanas.
- Muchas observaciones de la naturaleza muestran la evolución conjunta de varias especies representadas por una colección de individuos similares en términos de su fenotipo.
- Por ejemplo: Plantas e insectos de polinización
- A diferencia de las estrategias evolutivas clásicas (la población es del mismo tipo de individuos), en los algoritmos coevolutivos cada población representará una especie en particular.
- Cada especie desarrolla un subcomponente de la solución, luego se integran los resultados en una solución global.

Algoritmos de coevolución cooperativa



Algoritmos de coevolución cooperativa

- La idea es dividir un problema grande en subproblemas, y resolverlos de manera independiente.
- Al final del algoritmo, los representantes de cada especie se unen (concatenación) para así obtener la solución final.
- Útil en problemas grandes de optimización y entrenamiento de redes neuronales.

Scatter Search

- Parte generado una población inicial (que raro no?). Luego se construye un conjunto de referencia (RefSet) de tamaño moderado, en la literatura significa 10 soluciones app .
- RefSet incluye soluciones con buen fitness y otras para mantener diversidad.
- Integra elementos de algoritmos basados en poblaciones comunes y de metaheurísticas de una solución.

Scatter Search