Pseudocódigo Gramatica Evolutiva (problema Hormiga)

Evolución Gramatical o Gramática Evolutiva (GE)

- Forma de programación genética basada en gramáticas.
- Los individuos se codifican a través de cromosomas con codificación entera (como sucede en AGs).
- Permite aplicar operadores de recombinación mutación de AGs.
- La expresión de un genotipo se realiza decodificando por medio de la gramática.
- Los fenotipos (programas) se evalúan como se hace en PG.

```
/*
 * <S> = <exp>
 * <exp> = <SiComida> | <Progn2> | <Progn3> | <avanza> | <derecha> | <izquierda>
 * <SiComida> = if(comida) <exp> else <exp>
 * <Progn2> = <exp> <exp>
 * <Progn3> = <exp> <exp> <exp>
 */

public class Gramatica {
    private int wraps;
    private int maxWraps;

    public Gramatica(int maxWraps){
        this.wraps = 0;
        this.maxWraps = maxWraps;
    }

    public void S(List<Double> codones, Hormiga hormiga) {
        int instruc = codones.get(0).intValue() % 6;

        int i = decode(codones, hormiga, 0, true);
    }

    private int progn2(List<Double> codones, int i, boolean operativa, Hormiga hormiga) {
        i++;
        i = decode(codones, hormiga, i, operativa);

        i++;
        return decode(codones, hormiga, i, operativa);
    }

    private int progn3(List<Double> codones, int i, boolean operativa, Hormiga hormiga) {
        i++;
        i = decode(codones, hormiga, i, operativa);

        i++;
        i = decode(codones, hormiga, i, operativa);

        i++;
        return decode(codones, hormiga, i, operativa);
    }

    private int sicomida(List<Double> codones, int i, boolean operativa, Hormiga hormiga) {
        i++;
        if (operativa) {
            if (!hormiga.hayComidaDelante()) {
```

```
                i = decode(codones, hormiga, i, false);
                i++;
                return decode(codones, hormiga, i, true);
            } else {
                i = decode(codones, hormiga, i, true);
                i++;
                return decode(codones, hormiga, i, false);
            }
        } else {
            i = decode(codones, hormiga, i, operativa);
            i++;
            return decode(codones, hormiga, i, operativa);
        }
    }


private int decode(List<Double> codones, Hormiga hormiga, int i, boolean operativa) {
        if (i >= codones.size()){
            i = 0;
            this.wraps++;
        }
        if(this.wraps >= this.maxWraps)
            return i;
        int instruc = codones.get(i).intValue() % 6;

        switch (instruc) {
            case 0:
                return sicomida(codones, i, operativa, hormiga);

            case 1:
                return progn2(codones, i, operativa, hormiga);

            case 2:
                return progn3(codones, i, operativa, hormiga);

            case 3:
                if (operativa)
                    hormiga.avanza();
                return i;

            case 4:
                if (operativa)
                    hormiga.girar(Hormiga.Giro.DERECHA);
                return i;

            case 5:
                if (operativa)
                    hormiga.girar(Hormiga.Giro.IZQUIERDA);
                return i;

            default:
                return i;
        }
    }


}
```