# GENETIC ALGORITHM TEMPLATE DESCRIPCIÓN

Leslie Pérez Cáceres

---

El template del algoritmo que utilizará en este curso le permitirá aplicar el algoritmo para resolver instancias del problema del vendedor viajero (TSP) en formato TSPlib. La carpeta `ga_solver` contiene:

- código: clases Java en la carpeta `src/`
- instancias: instancias en formato TSPlib en la carpeta `instances`

El código posee las siguientes clases:

- `TSPlibReader.java`: clase que implementa funciones para leer las instancias de TSP en el formato de la librería de instancias TSPlib.
- `TSP.java`: clase que representa la instancia de TSP. Contiene los siguientes métodos:
  - `compute_tour_length`: calcula el costo de un tour t
  - `tsp_check_tour`: revisa si un tour t es correcto
  - `random_tour`: genera un tour aleatorio
  - `greedy_nearest_n`: genera un tour utilizando la heurística del vecino más cercano
- `Tour.java`: clase que implementa una solución del TSP. El constructor de clase permite definir si la solución inicial es construida aleatoriamente o utilizando la heurística del vecino más cercano. Contiene las siguientes variables y métodos:
  - `current`: solución del TSP que se representa con un arreglo de enteros de tamaño $n+1$, donde $n$ son los nodos (ciudades) y la última ciudad del tour corresponde siempre a la primera ciudad. Ejemplo para TSP $n = 5$: 243102
  - `cost`: costo del tour de la solución
  - `swap`: método que aplica el movimiento swap a dos nodos en el tour actual
  - `twoOptSwap`: método que aplica el movimiento 2-opt a dos nodos en el tour actual
- `GeneticAlgorithm.java`: clase que implementa el algoritmo genético
  - `search`: método que inicia la búsqueda comenzando de una solución inicial generada aleatoriamente
  - `terminationCondition`: método que revisa si la condición de término (temperatura mínima o número de evaluaciones) se ha cumplido
- `Population.java`: clase que implementa todos los métodos para manipular las soluciones de la población de un algoritmo genético
  - `mutation`: método que aplica un operador de mutación a la población

    - `crossover`: método que aplica un operador de crossover a los padres proporcionados (ids)
    - `selectParents`: método que selecciona dos padres aplicando un operador de selección
    - `selectPopulation`; método que selecciona la población reduciendo su tamaño en base a un operador de selección
- `Runner.java`: clase que ejecuta el algoritmo genético (main)
- `AlgorithmOptions.java`: clase que permite pasar opciones (parámetros) al algoritmo a través de argumentos

**Parámetros del algoritmo**

Los parámetros del algoritmo se pueden cambiar en la clase `AlgorithmOptions.java` o si el algoritmo es ejecutado directamente desde la línea de comando utilizando pasándolos como argumentos

- `filename`: Ruta al archivo de la instancia de TSP a resolver, por defecto `instances/burma14.tsp`. (`--instance <instance_path>`)
- `seed`: Semilla para el generador de números aleatorios. (`--seed <int>`)
- `pop_size`: Tamaño de la población. (`-p ]2,INT_MAX]` )
- `offspring_size`: Tamaño de la población de hijos. (`-o ]2,INT_MAX]`)
- `pselection_type`: Operador de selección de padres. Los valores posibles son: `Population.SelectionType.BEST`, `Population.SelectionType.RANDOM`, `Population.SelectionType.ROULETTE` y `Population.SelectionType.TOURNAMENT`. (`-ps [ random | best | roulette | tournament ]`)
- `crossover_type`: Operador de cruzamiento. Los valores posibles son `Population.CrossoverType.OX`, `Population.CrossoverType.PMX` y `Population.CrossoverType.OPX`. (`-c [ ox | opx | pmx ]`)
- `mutation_type`: Operador de mutación. Los valores posibles son: `Population.MutationType.SWAP` y `Population.MutationType.TWO_OPT`. (`-m [swap | 2-opt]`)
- `mutation_prob`: Probabilidad de mutación. (`-mp [0,1]`)
- `selection_strategy`: Estrategia de selección de población. Los valores posibles son: `GeneticAlgorithm.SelectionStrategy.MULAMBDA` y `GeneticAlgorithm.SelectionStrategy.MUPLUSLAMBDA`. (`-g [ mu,lambda | mu+lambda]`)
- `gselection_type`: Operador de selección de población. Los valores posibles son: `Population.SelectionType.BEST`, `Population.SelectionType.RANDOM`, `Population.SelectionType.ROULETTE` y `Population.SelectionType.TOURNAMENT`. (`-gs [ random | best | roulette | tournament ]`)
- `max_evaluations`: número máximo de funciones de evaluación calculadas. (`-e <int>`)

- `max_iterations`: número máximo de iteraciones.
  (`-t <int>`)

### Compilación y ejecución

Si utiliza Eclipse IDE, revise este video

- Para compilar en la consola:
  `javac -d bin/ -sourcepath src/ -cp lib/commons-cli-1.2.jar src/algorithms/*java`
- Para mostrar la ayuda en la consola:
  `java -cp bin/:lib/commons-cli-1.2.jar algorithms.Runner --help`
- Para ejecutar en la consola:
  `java -cp bin/:lib/commons-cli-1.2.jar algorithms.Runner --instance instances/burma14.tsp`