Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

Práctica 4. ALGORITMOS GENÉTICOS

Objetivo:

utilizar Opt4J para diseñar, resolver y evaluar un problema de optimización mediante AG

Opt4J está disponible en:

Poliformat, en M:\ETSINF\tia\alumnos\Practica-4 AG - Opt4J

y en

http://opt4j.sourceforge.net/

Práctica 4: Opt4J

Opt4J. Entorno libre

A Modular Framework for Meta-heuristic Optimization

Disponible en: http://opt4j.sourceforge.net/

Formulación sencilla de problemas utilizando librerías implementadas en Java

Existe un boletín completo que explica su instalación y uso

Algoritmos Genéticos

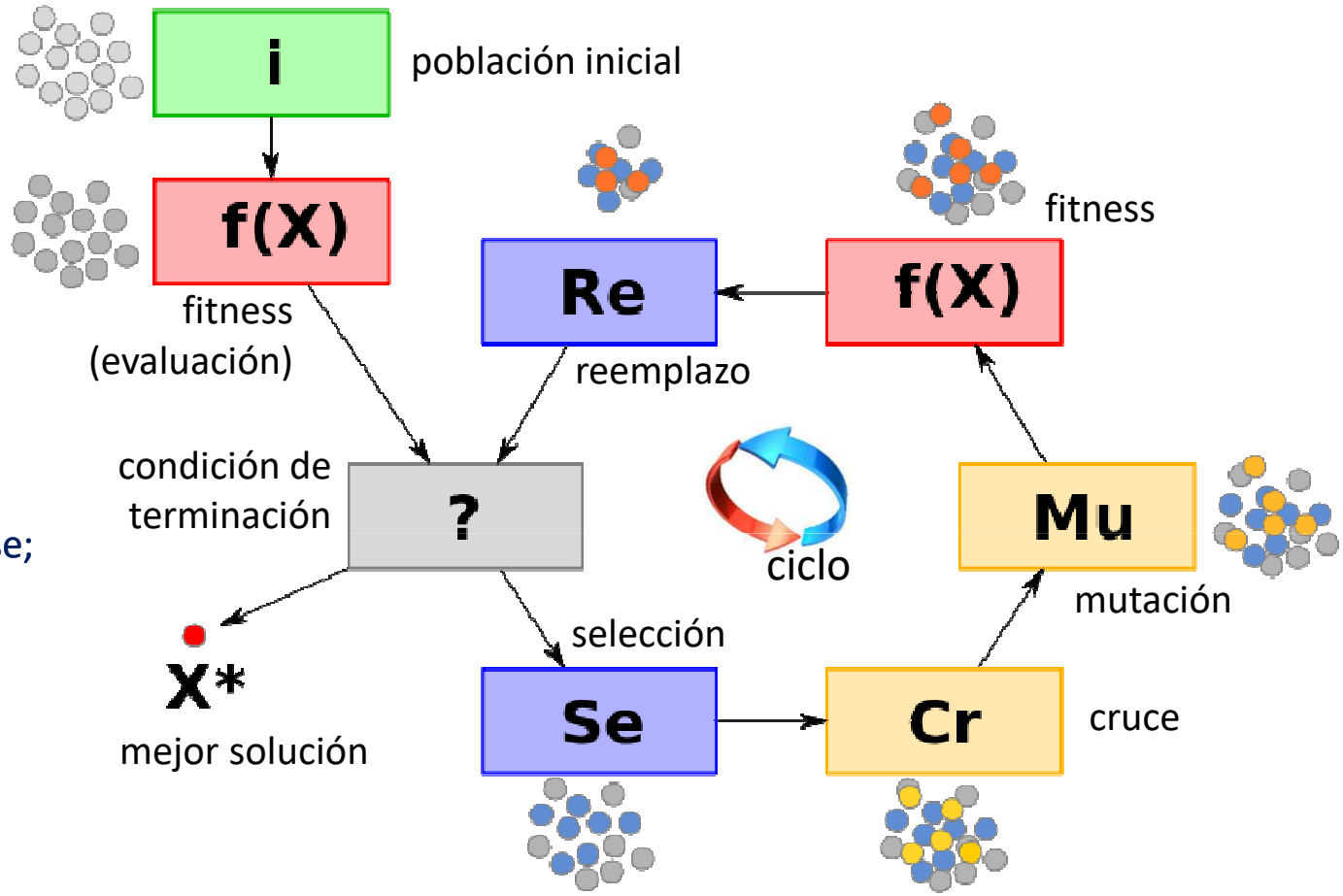
Diseño algoritmo genético

- Diseño del individuo. Codificación y decodificación.
- Función de evaluación (fitness)
- Generación población inicial.
- Selección. Cruce (individuos inválidos). Mutación. Reemplazo.

Evaluación algoritmo genético

- Criterios de evaluación: Fitness versus Soluciones generadas, Tiempo cómputo.
- Tamaños del problema
- Parámetros de evaluación: Población, Selección, Cruce, Mutación, etc.

Práctica 4: Opt4J



https://commons.wikimedia.org/wiki/File:Evolutionary_algorithm.svg

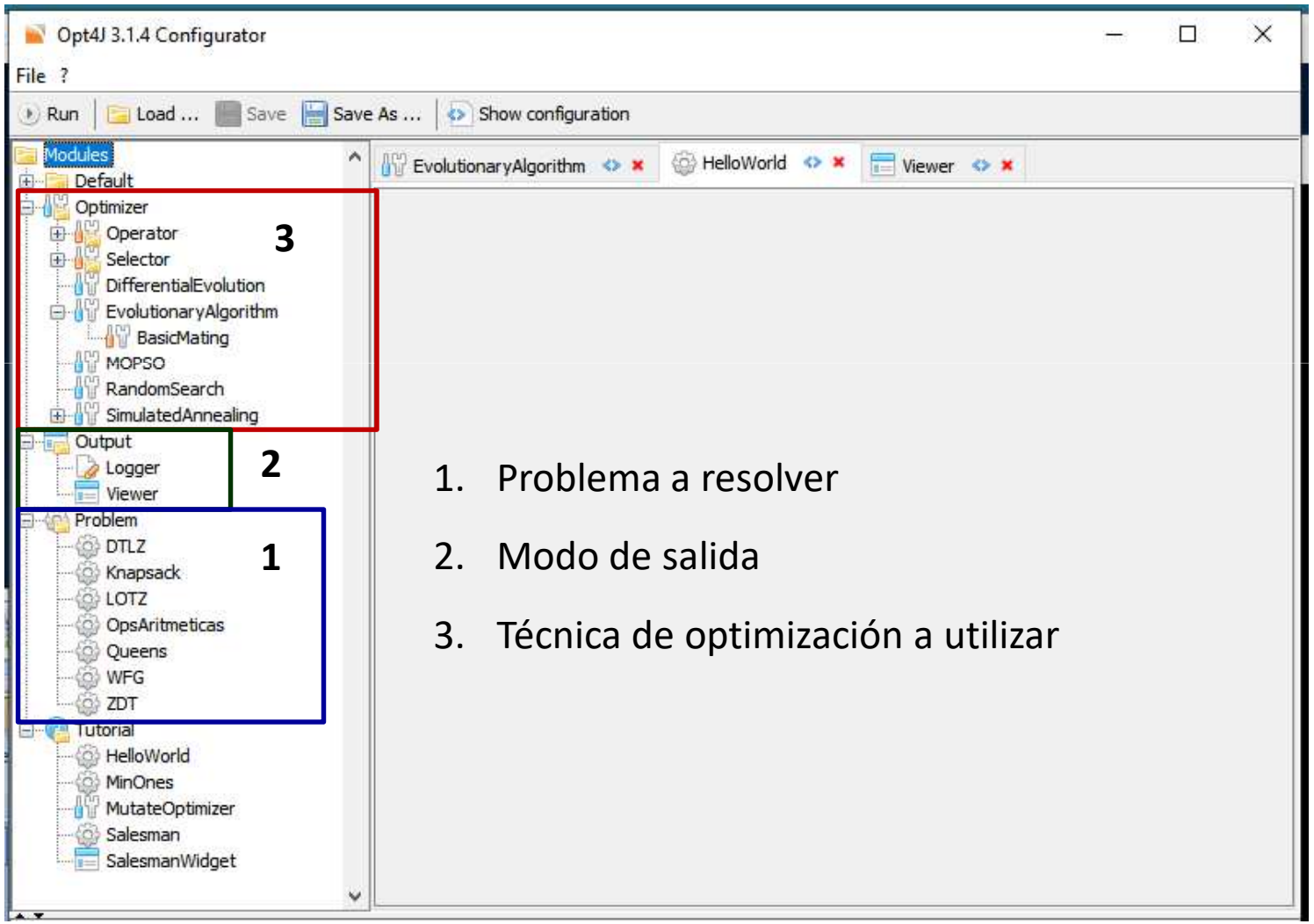
```
t ← 0;    terminar ← False;
inicializar P(0);
while not(terminar)
   S ← Selección [P(t)];
   Q ← Cruce [S];
   Q' ← Mutación [Q];
   P(t+1) ← Reemplazo [P(t), S, Q'];
   t ← t+1
   terminar ← (Convergencia [P(t+1)]) OR (t>límite)
end_while
```

Práctica 4: Opt4J en ejecución

http://opt4j.sourceforge.net/download.html

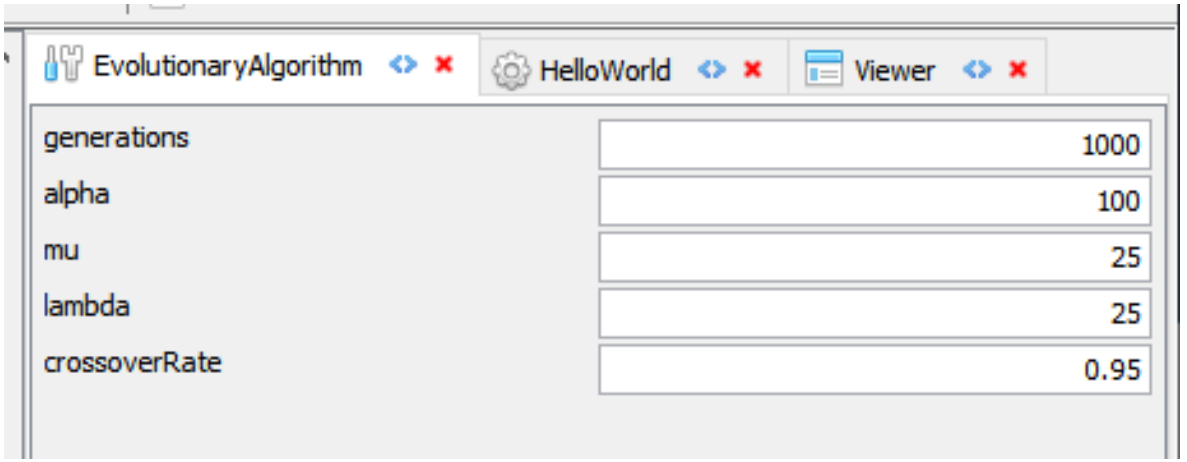
M:\ETSINF\tia\alumnos\Practica-4 AG - Opt4J

…/bin/opt4j.bat

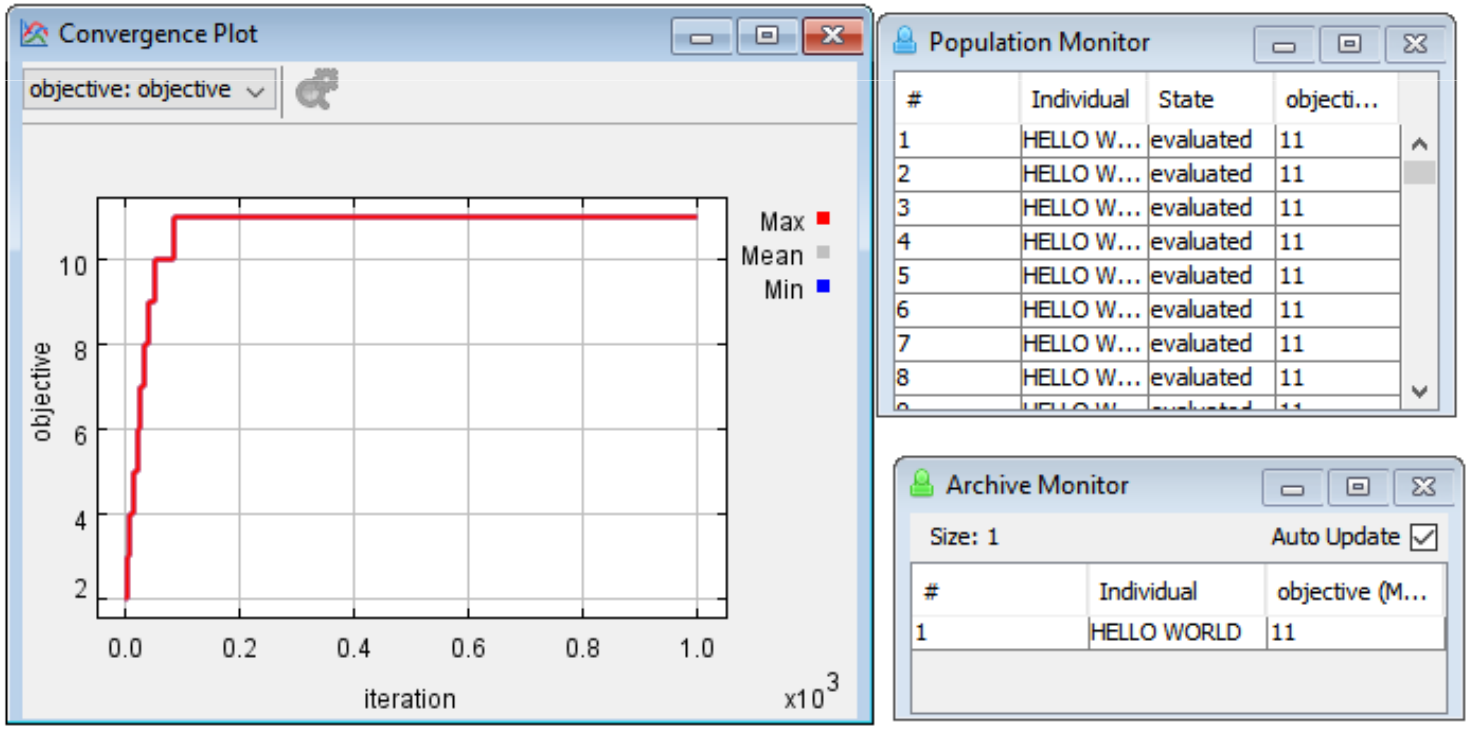


Práctica 4: Opt4J

Parametrización del AG

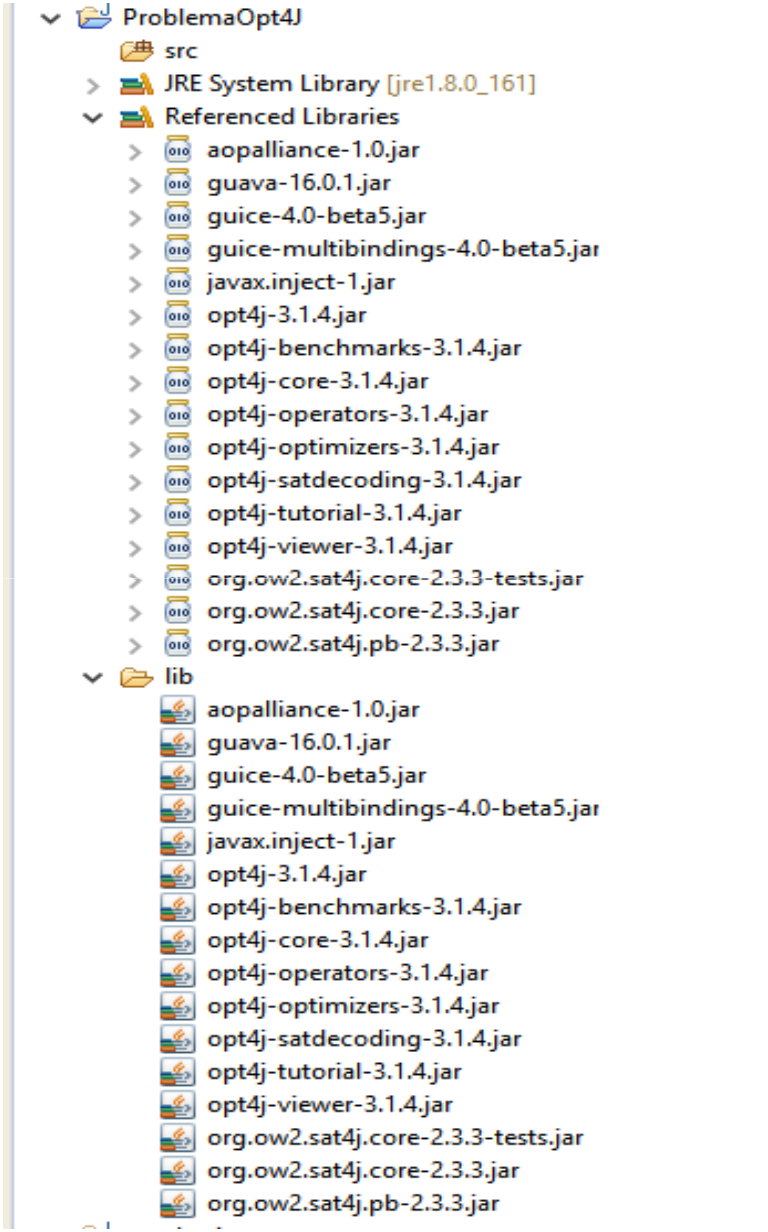


Resultados:



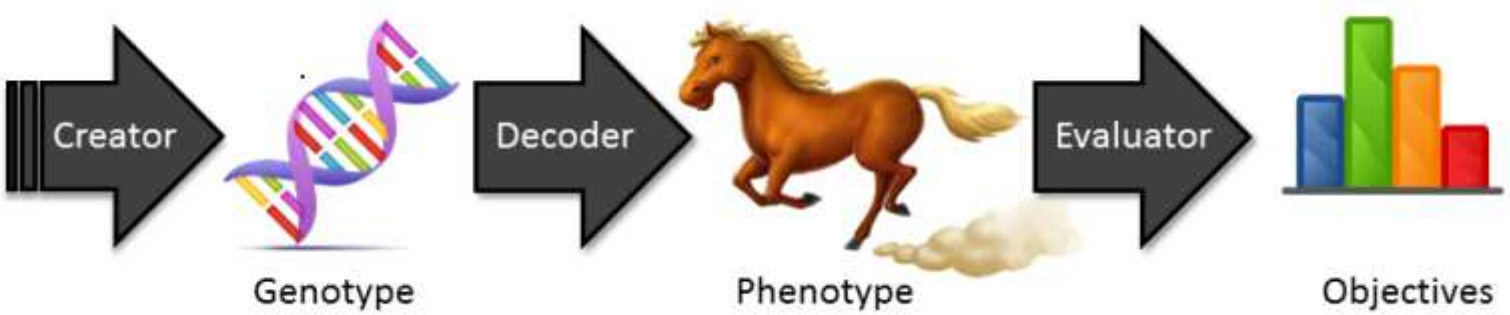
Práctica 4: Integración Opt4j en ECLIPSE

- Opt4J permite la importación y resolución de problemas previamente modelados en Java.
- Por simplicidad, utilizaremos el **entorno Eclipse**.
- ***Configuración de ECLIPSE en Boletín.***
- Modelado del problema en Java (creator, decoder, evaluator)



Práctica 4: diseñando el problema en Opt4J

Básicamente, hay que implementar tres clases



public class NombreClaseCreator **implements** Creator<**GENOTIPO**>

public class NombreClaseDecoder **implements** Decoder<**GENOTIPO**, **FENOTIPO**>

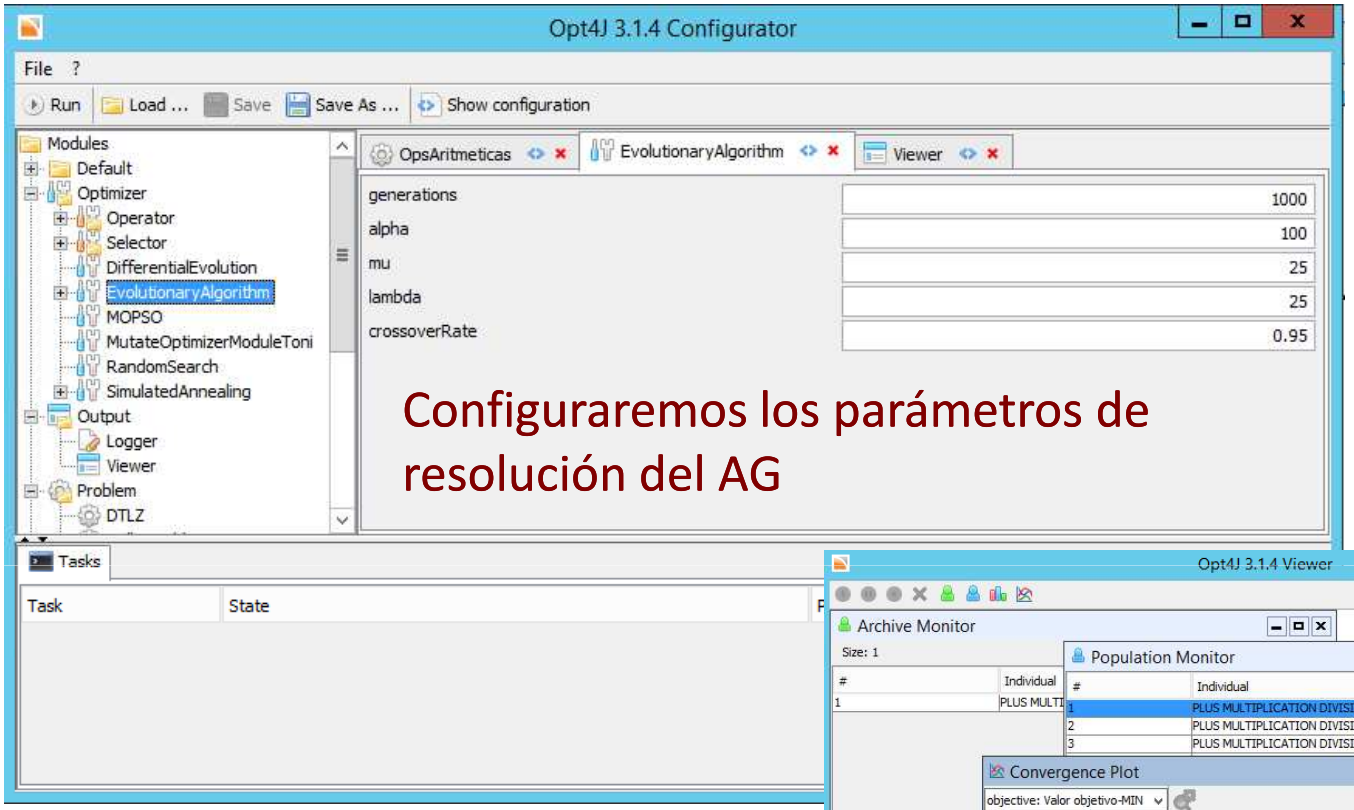
public class NombreClaseEvaluator **implements** Evaluator<**FENOTIPO**>

…más la clase Module que las referencia:

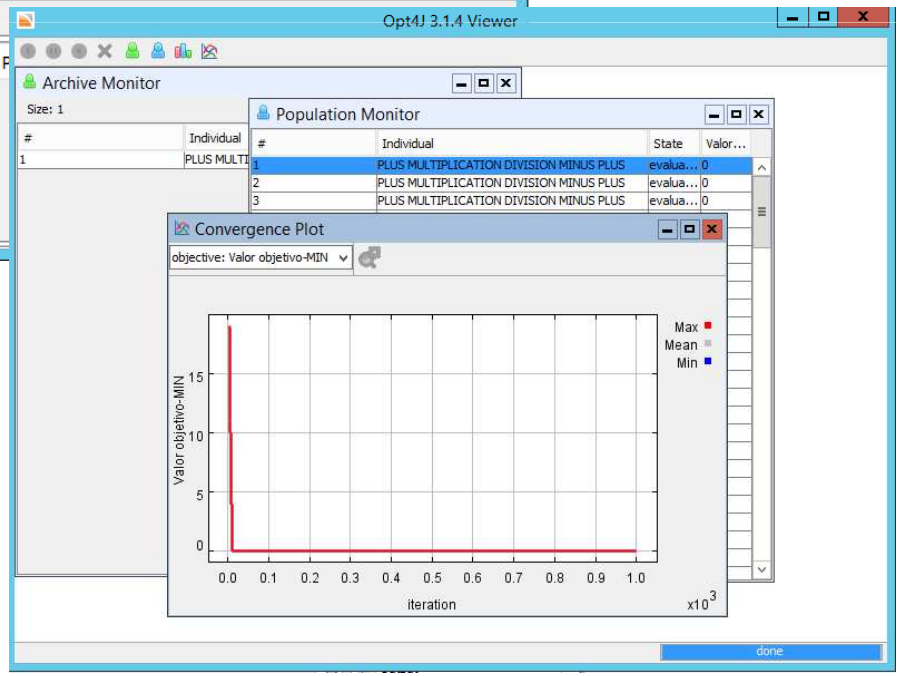
public class ClaseModule **extends** ProblemModule

Ver ejemplos en el Boletín!!

Práctica 4: resolviendo el problema en Opt4J



Configuraremos los parámetros de resolución del AG



Y analizaremos los resultados

Práctica 4: Opt4J

Evaluación:

- Realizar el ejercicio propuesto (se necesitará para el día de la evaluación, en el que se planteará una breve ampliación)

Calendario:

Sem	LABORATORIO	Evaluación
26-XI	Opt4J	
3-XII	Opt4J	
		P4: ***Aplicac. Opt4J***

Aplicación y evaluación de Algoritmos Genéticos (15%) P4

Problema a resolver:

- Distribuir 10 cuadrillas de trabajadores entre los turnos de trabajo T1-T3 de un día concreto.
- Cada cuadrilla tiene un coste de asignación a cada turno.
- Deben haber la menos 3 cuadrillas por turno

	Cuad. 1	**Cuad. 2**	**Cuad. 3**	**Cuad. 4**	**Cuad. 5**	**Cuad. 6**	**Cuad. 7**	**Cuad. 8**	**Cuad.9**	**Cuad.10**
Coste Turno1	1000	1100	1500	2500	3200	2500	2100	2100	1450	2200
Coste Turno2	1400	2100	1900	2400	2050	2000	1850	3050	1600	1500
Coste Turno3	2100	3100	2050	2100	2100	1400	1900	2050	2150	1900

Ejemplo:

Cuad.1	**Cuad. 2**	**Cuad. 3**	**Cuad. 4**	**Cuad. 5**	**Cuad. 6**	**Cuad. 7**	**Cuad. 8**	**Cuad. 9**	**Cuad. 10**
T1	T3	T2	T3	T1	T2	T3	T1	T2	T3
1000	3100	1900	2100	3200	2000	1900	2100	1600	1900

Coste: 20.800 *(Mejor solución obtenida ha sido de coste 16.000€.)*

Adicionalmente,

Considerar la productividad de cada cuadrilla en cada turno (m^2 reasfaltados/hora)

	Cuad. 1	Cuad. 2	Cuad. 3	Cuad. 4	Cuad. 5	Cuad. 6	Cuad. 7	Cuad.8	Cuad.9	Cuad.10
Product. Turno1	50	42	42	40	50	50	40	52	50	44
Product. Turno2	52	50	53	40	30	50	40	50	45	50
Product. Turno3	45	30	35	25	50	30	30	42	35	45

Objetivos contrapuestos: Productividad ⇔ Coste

Por ejemplo, <coste, productividad>: <16000,401>, <18500,465>, <19350,472>, etc.

En poliformat se ha dejado un archivo Datos.java con los valores de las dos matrices anteriores.