Técnicas, Entornos y Aplicaciones de Inteligencia Artificial

Evaluación Práctica-4: Algoritmos Genéticos (Opt4J). 2022-2023.

Nombre: _____

1) **Subid a Poliformat todos los ficheros correspondientes a la práctica (habrá distintas versiones: versión original y cada uno de los ejercicios). Se puede subir un archivo .zip.**
2) **Contestad a las preguntas siguientes, rellenando los huecos con las respuestas. Se deben comentar los cambios realizados. Se debe partir de la práctica ya realizada.**
3) **La entrega fuera de plazo tendrá penalización en la nota.**
4) **La utilización de herramientas de mensajería está terminantemente prohibida.**

Tiempo: 75 minutos.

Nota 1: todos los ejercicios y apartados son incrementales. Es decir, el resultado del ejercicio 1 se utiliza como base para el ejercicio 2, el ejercicio 2 se usa como base para el ejercicio 3, y así sucesivamente.
Nota 2: todos los ejercicios se deberán realizar con 800 generaciones. El resto de los parámetros se deja a libertad del alumnado.

NOTA DE IMPLEMENTACIÓN. Recordad que las colecciones en Java comienzan con el índice 0.

1. **(2 *puntos, Tiempo estimado: 15'*)** Utilizando vuestro diseño original, añadir la siguiente nueva información:

- Existen dos nuevos ordenadores O11 y O12:

Ordenador	Nº procesos que puede atender por hora	Memoria RAM (GB)
O11	4	8
O12	3	12

- Existen tres nuevos procesos en la forma <RAM requerida, beneficio obtenido>: <3.2, 0.6>, <2.5, 1.1> y <2.4, 0.5>.
- Ahora interesa encontrar la distribución de procesos a ejecutar que maximiza el beneficio y minimiza la **mitad** de la memoria RAM consumida.

Indica las modificaciones realizadas y dos de los mejores resultados obtenidos en la forma <beneficio,RAM>:

Se añaden los datos en el archivo DatosCloud.java y se cambia el número de los procesos y ordenadores

En el archivo Evaluator se hace: `objectives.add("MIN RAM", Sign.MIN, 0.5*ram);`

Los mejores resultados obtenidos son <beneficio,RAM>: <77.52, 39.65>, <72.22, 34.55>, etc.

[0, 0, 0, 11, 3, 1, 4, 2, 9, 12, 0, 5, 2, 0, 0, 0, 0, 0, 0, 9, 1, 3, 0, 0, 7, 6, 0, 0, 8, 3, 10, 12, 4, 11, 0, 9, 0, 10, 8, 12, 0, 0, 0]

Realiza otras pruebas del algoritmo genético modificando los parámetros de "tamaño de la población" e “hijos por generación”. De acuerdo a las pruebas que has realizado, ¿resulta más adecuado trabajar con una población de mayor tamaño o generando más hijos por generación? Razona la respuesta en base a tus experimentos.

No hay resultados concluyentes y en función de cada implementación saldrán unos resultados u otros.

2. ***(3 puntos, Tiempo estimado: 15')*** **A partir de las modificaciones del ejercicio 1**, se han detectado ciertas restricciones sobre dos tipos de software específico a instalar. Por simplicidad, el SW1 está en los ordenadores impares y el SW2 en los ordenadores pares. Hay ciertas restricciones sobre el software que necesita cada proceso:

- Los procesos P1 y P2 necesitan el SW1.
- Los procesos P3 y P4 necesitan el SW2.
- El proceso P5 necesita el SW1 y el SW2.
- El resto de los procesos no necesita ningún software específico.

Indica las modificaciones realizadas y dos de los mejores resultados obtenidos en la forma <beneficio,RAM>:

En Evaluator:

```
// el índice de los procesos empieza en 0, por lo que es uno menos
switch(proceso)
{
// procesos P1 y P2
case 0:
case 1: if (ordenador % 2 == 0) // necesita SW1 -> debe asignarse a ordenador
impar
beneficio = Double.NEGATIVE_INFINITY; break;
// procesos P3 y P4
case 2:
case 3: if (ordenador % 2 != 0) // necesita SW2 -> debe asignarse a ordenador par
beneficio = Double.NEGATIVE_INFINITY; break;
// proceso P5
case 4:  // este proceso no puede ejecutarse en ningún ordenador (debería ser
par+impar)
beneficio = Double.NEGATIVE_INFINITY; break;
}
```

Los mejores resultados obtenidos son <beneficio,RAM>: <76.12, 39.2>, <74.37, 38.85>, etc.
¡P5 no se puede ejecutar porque ningún ordenador tiene instalados los 2 SW!

En el primer resultado que has indicado, identifica en qué ordenador se ejecuta cada uno de los siguientes procesos:

Proceso	Ordenador
P1	NINGUNO
P2	NINGUNO
P3	NINGUNO
P4	O4
P5	NINGUNO
P6	O11
P7	O9

3. ***(2.5 puntos, Tiempo estimado: 10')*** Se ha detectado que hay ciertas combinaciones de procesos que, si se ejecutan en el mismo ordenador, producen una sobrecarga en la red que repercute negativamente en el rendimiento de los ordenadores de la empresa. Esto reduce el beneficio en un 5%. Es decir, si el beneficio obtenido es X y esos procesos se ejecutan en el mismo ordenador, el beneficio final es solo de 0.95*X. Esto ocurre cuando en el mismo ordenador se ejecutan “P6 y P8” o “P6 y P9”.

Indica las modificaciones realizadas y dos de los mejores resultados obtenidos en la forma <beneficio,RAM>:

En Evaluator:

```
// Si P6 y P8 o P6 y P9 están en el mismo ordenador se reduce el beneficio
// el índice de los procesos empieza en 0, por lo que es uno menos
if ((fenotipo.get(5) != 0) &&
 ((fenotipo.get(5) == fenotipo.get(7)) || (fenotipo.get(5) == fenotipo.get(8))))
{
beneficio *= 0.95;
}
```

Los mejores resultados obtenidos son <beneficio,RAM>: <68.92,32.6>, <65.82,30.5>, etc.

4. ***(1.5 puntos, Tiempo estimado: 15')*** Se desea probar un software automático de paralelización de procesos en los ordenadores O1 y O2. Esto será equivalente a contar con un ordenador O1+O2 que es capaz de atender a 1 proceso por hora y con una memoria RAM de 4 GB, sin afectar a las características individuales iniciales de O1 y O2. Es decir, ahora será posible ejecutar un proceso en O1, otro en O2 y otro paralelizado en O1+O2, teniendo en cuando las restricciones dadas.

Indica las modificaciones realizadas y dos de los mejores resultados obtenidos en la forma <beneficio,RAM>:

Basta con modificar DatosCloud.java, añadiendo un nuevo ordenador O1+O2 "paralelizado". Hay 13 ordenadores y 43 procesos

```
	// numero procesos que puede atender por hora cada ordenador
	public static final int[] capacidad =
		{
			3, 2, 4, 5, 2, 1, 3, 2, 4, 3, 4, 3, 1  // el último ordenador
es uno paralelizado
		};

	// memoria RAM de cada ordenador
	public static final int[] memoria =
		{
			4, 6, 8, 8, 6, 2, 4, 16, 32, 8, 8, 12, 4  // el último
ordenador es uno paralelizado
	};
```

Los mejores resultados obtenidos son <beneficio,RAM>: <68.92,32.65>, <66.92,32.1>, etc.

En el primer resultado que has indicado, identifica si algún proceso se ha ejecutado de forma paralelizada:

El proceso 38 está en el ordenador nuevo (el 13):

[0, 0, 0, 6, 0, 4, 5, 8, 9, 8, 0, 11, 3, 0, 0, 0, 0, 0, 0, 0, 4, 10, 9, 0, 0, 12, 0, 0, 9, 4, 7, 2, 11, 3, 0, 12, 0,

13, 0, 12, 0, 0, 0]

5. ***(1 punto, Tiempo estimado: 10')*** Explica (no es necesario implementar nada) cuál sería el mejor genotipo si simplemente se deseara obtener una solución que indicara si un proceso se ha ejecutado o no. ¿Y cuál sería el mejor genotipo si deseamos conocer si el proceso se ha ejecutado ya, si está en una cola de ejecución o si no se ejecutará? **Razona las respuestas**.

En el primer caso, bastaría con un BooleanGenotype que representa valores T/F para indicar si se ejecuta o no.
En el segundo caso, bastaría con un IntegerGenotype que representa tres posibles valores (ej. 0, 1, 2) para indicar sí ejecutado, en cola, no se ejecutará.
También podría ser un SelectGenotype