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	3	6
O12	5	8

- Existen tres nuevos procesos en la forma <RAM requerida, beneficio obtenido>: <2.5, 1.6>, <3.5, 1.3> y <3.4, 1.1>.
- Ahora interesa encontrar la distribución de procesos a ejecutar que maximiza el beneficio y minimiza el **doble** 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, 2*ram);`

Los mejores resultados obtenidos son <beneficio,RAM>: <70.20, 142.2>, <66.84, 130.8>, etc.

[0, 0, 10, 8, 3, 5, 10, 0, 12, 3, 0, 8, 3, 0, 0, 12, 0, 0, 0, 0, 5, 0, 0, 0, 0, 2, 0, 0, 0, 7, 4, 11, 9, 4, 0, 2, 0, 9, 9, 9, 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 desea analizar el consumo eléctrico de los ordenadores. Por simplicidad, los ordenadores impares tienen un consumo de 1 unidad por proceso, y los pares tienen un consumo de 2 unidades por proceso. Por ejemplo, si el ordenador 4 ejecuta 3 procesos, tendrá un consumo de 2*3=6 unidades. Ahora interesa encontrar la distribución de procesos a ejecutar que maximiza el beneficio y minimiza el consumo eléctrico total.

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

Actualizamos DatosCloud.java con los consumos eléctricos de los ordenadores

En Evaluator calculamos el consumo total:

```
// actualizamos el consumo
consumo += DatosCloud.consumo[ordenador];
```

Y cambiamos a la nueva métrica:

```
objectives.add("MIN consumo", Sign.MIN, consumo);
```

Los mejores resultados obtenidos son <beneficio,consumo>: <72.39, 27.0>, <71.19, 30>, etc.

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

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	O1
P5	O11
P6	O5
P7	O4

3. ***(2.5 puntos, Tiempo estimado: 10')*** Se ha detectado que la ejecución de ciertos procesos produce un sobrecalentamiento excesivo de los ordenadores, siendo necesario incrementar el consumo eléctrico necesario para refrigerar la sala en la que se encuentran. Esto puede incrementar el consumo eléctrico total en un 2% o 5%. Los procesos que se han detectado son P6, P9 y P10. Si uno o dos de estos tres procesos se ejecuta, el consumo eléctrico total pasa a ser 1.02*X. Si los tres procesos se ejecutan, el consumo eléctrico total es 1.05*X.

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

En Evaluator:

```
            // Procesos P6, P9 y P10
            // Si uno o dos de estos tres procesos se ejecuta, el consumo
eléctrico total pasa a ser 1.02*X
            // Si los tres procesos se ejecutan, el consumo eléctrico total es
1.05*X
            // el índice de los procesos empieza en 0, por lo que es uno menos
            if ((fenotipo.get(5) != 0) &&
                        (fenotipo.get(8) != 0) &&
                        (fenotipo.get(9) != 0))
            {
                  consumo *= 1.05;  // los tres procesos se ejecutan
            }
            else
                  if ((fenotipo.get(5) != 0) ||
                              (fenotipo.get(8) != 0) ||
                              (fenotipo.get(9) != 0))
                  {
                        consumo *= 1.02;  // al menos uno de los tres procesos se
ejecuta
                  }
```

Los mejores resultados obtenidos son <beneficio,consumo>: <74.92, 31.5>, <71.14, 27.3>, etc.

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

4. ***(1.5 puntos, Tiempo estimado: 15')*** Se desea probar un software automático de paralelización de procesos en los ordenadores O1+O2 y O3+O4. Esto será equivalente a contar con dos ordenadores "O13=O1+O2" y "O14=O3+O4" capaces de atender a 1 proceso por hora y con una memoria RAM de 4 GB, sin afectar a las características de los ordenadores individuales iniciales. Por ejemplo, ahora será posible ejecutar un proceso en O1, otro en O2 y otro paralelizado en O13, teniendo en cuando las restricciones dadas. El consumo eléctrico se mantiene igual, tanto para ordenadores impares como pares.

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

Basta con modificar DatosCloud.java, añadiendo dos nuevos ordenadores O1+O2 y O3+O4 "paralelizados". Hay 14 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, 3, 5, 1, 1  // los dos últimos
ordenadores son paralelizados
            };

      // memoria RAM de cada ordenador
      public static final int[] memoria =
            {
                  4, 6, 8, 8, 6, 2, 4, 16, 32, 8, 6, 8, 4, 4  // los dos últimos
ordenadores son paralelizados
            };

      // consumo eléctrico de cada ordenador
      public static final int[] consumo =
            {
                  1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2  // los dos últimos
```

```
ordenadores son paralelizados
                };
```

Los mejores resultados obtenidos son <beneficio,consumo>: <76.22, 32.55>, <72.74, 28.35>, etc.

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

Los procesos 25 y 33 están en los ordenadores nuevos paralelizados (14 y 13, respectivamente):

[0, 0, 4, 11, 5, 11, 5, 0, 9, 3, 0, 3, 0, 0, 0, 0, 0, 8, 0, 0, 1, 1, 0, 0, 14, 11, 0, 0, 9, 10, 8, 7, 13, 9, 0, 10, 0, 2, 9, 4, 0, 0, 0]

5. ***(1 punto, Tiempo estimado: 10')*** Explica (no es necesario implementar nada) qué ventajas e inconvenientes tendría utilizar como genotipo para este problema el tipo IntegerMapGenotype<Integer>. ¿Se te ocurre otro genotipo más sencillo para este problema? **Razona las respuestas**.

Ventaja: la conversión al fenotipo sería sencilla pues tendríamos un mapa <x,y> de "el proceso x se ejecuta en el ordenador y"

Inconveniente: hace falta mapear explícitamente cada proceso x.

Es mucho más sencillo utilizar un IntegerGenotype. El propio índice del vector nos da el proceso y ya está implícito.