Práctica 3: búsqueda aleatoria y paralela de máximo de función

Nuevas tecnologías de la programación, 2014-2015

Contenido:

1 Descripción del problema **1**

2 Trabajo a realizar **2**

3 Notas **2**

1 Descripción del problema

Se desea implementar un método de búsqueda del máximo valor para funciones compleja para las que las técnicas analíticas de optimización no funcionan. Hay diferentes tipos de métodos para realizar esta tarea, como por ejemplo búsqueda aleatoria simple, búsqueda aleatoria multicomienzo, ascensión de colinas, recocido simulado, algoritmos genéticos, etc.

Para aprovechar la existencia de varios núcleos en los procesadores actuales y hacer la búsqueda más eficiente el sistema permitirá que haya varias objetos (quizás ejecutando diferentes algoritmos de búsqueda) trabajando a la vez de forma coordinada. La forma de coordinación usada implica la existencia de un único objeto de una clase especial (**EstadoBusqueda**) que representa el estado global de la búsqueda hasta el momento actual (es decir, teniendo en cuenta las mejores soluciones obtenidas por todos los objetos que trabajan simultáneamente).

De esta forma, los objeto de las clases que realizan la búsqueda comunicarán su mejor resultado (según lo encuentren) al objeto de la clase **EstadoBusqueda**. Este, a su vez, notificará de los cambios de interés a todos los objetos de búsqueda. Para que el diseño sea más flexible se desea además que todos los objetos de búsqueda puedan cambiar su estrategia de trabajo de forma dinámica.

Una posible función a optimizar es la siguiente:

$$f(x_1, x_2) = 21.5 + x_1 sin(4\pi x_1) + x_2 sin(20\pi x_2)$$

con $x_1 \in [-3, 12.1]$ y $x_2 \in [4.1, 5.8]$. Conviene diseñar el código de forma que el procedimiento de búsqueda pudiera realizarse sobre otras funciones diferentes sin necesidad de cambios en las clases de las que hemos hablado antes (**EstadoBusqueda** y aquellas que realizan el trabajo en sí).

2 Trabajo a realizar

Se pide:

- un diagrama UML que indique el diseño adoptado para resolver este problema. Deberían guardarse todos los diagramas que hayáis usado como forma de documentar el proceso completo, desde el principio hasta el fin.
- una explicación de los elementos claves del diseño y de las razones por las que se adopta esta solución. Debe justificarse el uso de alguno de los patrones vistos en clase, si se considera oportuno.
- la prueba de que el sistema funciona comporta la implementación de dos clases que representen diferentes estrategias de búsqueda (búsqueda aleatoria y recocido simulado, por ejemplo).
- también debe entregarse un directorio que contenga todo los archivos java desarrollados.
- la fecha de entrega se anunciará más adelante.

3 Notas

Para permitir el funcionamiento concurrente de los objetos que implementan la búsqueda se hará uso de las posibilidades de las hebras en Java. Aspectos concretos sobre este tema se tratarán en clase. Se puede ir trabajando en el diseño sin tener en cuenta este aspecto.

En el enlace

`http://www.theprojectspot.com/tutorial-post/simulated-annealing-algorithm-for-beginners/6`

puedes encontrar una explicación de este método de búsqueda así como la implementación java del mimo, que puede servir de base para el que hagáis vosotros.

Esta es la práctica final de la asignatura (y examen). En este documento se describe la funcionalidad básica, pero puede considerarse la aplicación de cualquier otro patrón de diseño o principio de orientación a objetos que se crea pertinente y que hayamos visto en clase (o que hayáis investigado por vuestra cuenta). Puede realizarse en parejas.