Práctica 2.

Optimización Local de Soluciones

Metaheurísticas – Grado de Ingeniería Informática

Universidad de Córdoba

2015 / 2016

El objetivo de esta práctica es el de iniciar al alumnado en la programación y análisis de metaheurísticas basadas en trayectorias y particularmente, en el caso de métodos de búsqueda local. Para ello, se presentará un guión de actividades a realizar para generar y analizar diversos métodos de búsqueda local en los problemas de optimización comentados en la práctica anterior, el problema de la mochila (KP) y el del viajante de comercio (TSP)

Tareas a realizar

Se debe (leer todo antes de programar):

1. Implementar una clase `NeighOperator<nombreProblema>` que tenga un método tal que recibiendo un objeto `Solution<nombreProblema>` y unos valores para sus parámetros de funcionamiento produzca un nuevo objeto `Solution<nombreProblema>` que represente una solución vecina de la original, generada según los valores de los parámetros.
    a. Se debe implementar al menos un operador para el KP y otro para el TSP, aunque el grupo podría decidir implementar más de una operador (por ejemplo, inversión de bits y intercambio de objetos en KP, o 2-opt y 2-swap en TSP). Si este es el caso, habría que adaptar los nombres de las clases de forma correspondiente.
    b. Podría ser deseable que el objeto `NeighOperator<nombreProblema>` informase a quien lo utilizase (método de otra clase) el número de posibles valores para sus parámetros de ejecución o el propio conjunto de posibles valores.
2. Implementar dos clases `<X??>NeighExplorator` que pueda inicializarse con un operador de vecindario que se le pase en el constructor y que tenga un método que, recibiendo una solución, implemente una estrategia de exploración de dicho vecindario y devuelva una solución vecina de la original.
    a. El alumnado debe deducir lo que debe ser "`<X??>`" (dos opciones y hay que implementar las dos opciones)

   b. Opcionalmente, el alumnado puede especializar estas clases para cada problema, incluyendo el nombre del problema en el nombre de la clase. En otro caso, sería necesario crear clases abstractas para `Solution`, `Evaluator`, `NeighOperator`...

   c. Podría ser interesante que recibiese un objeto de la clase `Random` en su constructor para su inicialización.

3. Implementar una clase `LocalSearch` que pueda inicializarse con un operador de vecindario y una estrategia de exploración de vecindarios (constructor) y que tenga un método que, recibiendo una solución, aplique el proceso de optimización local correspondiente sobre dicha solución, devolviendo el óptimo local correspondiente.

   a. Opcionalmente, el alumnado puede especializar esta clase a cada problema. Más aún si lo desea, puede implementar la exploración y la generación de vecinos dentro de esta clase (no siendo necesaria la implementación de las clases anteriores).

4. Implementar dos programa para cada problema (dependiendo del valor de `<X??>`) que, recibiendo un fichero instancia del problema, genere 1000 soluciones aleatorias, les aplique un método de optimización correspondiente (random + local / random + local / ...) e imprima:

   a. Una fila por cada una de las 1000 soluciones iniciales y optimizadas

   b. En cada fila, el mejor valor de aptitud encontrado hasta el momento (tras cada generación de una nueva solución)

   c. Al final, otra línea con la configuración de la mejor solución y su coste

5. Desarrollar los pasos necesarios para la optimización local para el problema seleccionado, diferente al KP y TSP.

6. Elaborar un informe con la tarea realizada, resultados obtenidos y análisis. Se sugiere presentar algunas tablas con los resultados obtenidos en las diferentes instancias y gráficas de convergencia de las ejecuciones.

   a. Opcionalmente, podría ser interesante analizar cuántas iteraciones realiza cada proceso de optimización local, el tiempo que tarda, la magnitud de la mejora...

7. Subir el informe a moodle **JUNTO CON la evaluación del mismo por otros dos grupos**, quienes muy brevemente indicarán los puntos fuertes y débiles del trabajo realizado y le asignarán una calificación numérica entre 1 y 10.