Javier Barbero Gómez
Rafael Barbudo Lunar
Álvaro Andrés Belmonte Pérez
Aurora Esteban Toscano

Metaheurística

Informe de la práctica 1

Contenidos:

1. Presentación de la codificación
2. Pruebas
3. Análisis
4. Avances en el problema de la práctica 5

Presentación de la codificación

Las prácticas de la asignatura las desarrollaremos en Java 7. Hemos creado un modelo de datos bastante genérico para la representación de problemas de Metaheurística con el propósito de seguir ampliándolo en futuras prácticas. Con el fin de agilizar las pruebas, los parámetros de entrada, tales como la semilla, el número de iteraciones o la instancia a cargar en este caso; se recibirán de un fichero de configuración específico para cada problema.

En esta primera práctica hemos aplicado la búsqueda aleatoria sobre una serie de instancias a fin de encontrar soluciones mejores. La bondad de la solución se mide con el parámetro fitness, cuyo dominio y función objetivo dependerán del problema concreto. Como se ha dicho antes, la búsqueda se aplicará un número de iteraciones específico (1 000 para TSP y 100 000 para KP). Con cada iteración se genera una solución aleatoria. Si su fitness es mejor que el de la solución actual, se mostrará una salida por pantalla especificando la iteración en la que se encuentra, la representación de dicha solución y el instante de tiempo en el que se encontró. Así mismo, los datos se almacenan en un fichero para su posterior procesado.

Para implementar el problema de KP hemos diseñado una estructura de datos *KPObject* que almacena el peso y el valor de un objeto a introducir en la mochila. Para instanciar el problema tenemos una clase *InstanceKnapsack* que recibe el número de objetos, el tamaño de la mochila y la lista de objetos que proceden del fichero de instanciación. La clase tiene un método *evaluate* que recibe una posible solución, representada por un vector de booleanos donde 1 significa que el objeto se ha metido en la mochila. Este método comprueba lo buena que es la solución cuantificándola con el parámetro fitness, que es negativo si la solución sobrepasa el tamaño de la mochila.

Para desarrollar el problema de TSP hemos creado dos estructuras de datos: *TSPNode,* para guardar las coordenadas de cada nodo, y *TSPGraph,* que almacena las distancias euclídeas de cada nodo con el resto de nodos; está implementado con una matriz aunque es transparente a los métodos usados. Hemos implementado una clase *InstanceTSP* que sigue la misma filosofía que la expuesta en el problema de KP.

Enlace al repositorio del código en GitHub

Pruebas

Vamos a realizar una serie de ejecuciones de cada problema en los que variaremos parámetros como son la semilla y, en el caso de KP, la probabilidad de meter o no los objetos. Cabe decir que al ser un proceso de búsqueda aleatoria no podemos influir demasiado en los resultados obtenidos.

Prueba 1: Evaluación KP

En el caso del problema de KP hemos seleccionado cuatro instancias de las proporcionadas por el profesor. Ejecutamos varias búsquedas aleatorias sobre cada instancia variando la probabilidad de introducir un objeto en la mochila. En cada ejecución la probabilidad de meter o no un objeto se reduce a la mitad empezando en 0.5 hasta 1e-4. Para cada probabilidad se han hecho 10 pruebas con el fin de obtener la media de la función fitness para esa probabilidad. Cada prueba genera 1 000 soluciones aleatorias y seleccionamos la mejor.

Prueba 2: Evaluación TSP

Para el problema de TSP hemos probado las tres instancias (*berlin52*, *ch150*, *d2103*) y hemos ejecutado una búsqueda aleatoria sobre cada una cuatro veces con diferentes semillas. La búsqueda aleatoria está configurada para generar 100 000 soluciones. Durante la búsqueda recogemos los instantes en los que mejora la solución actual.

Análisis

Resultados de la prueba 1

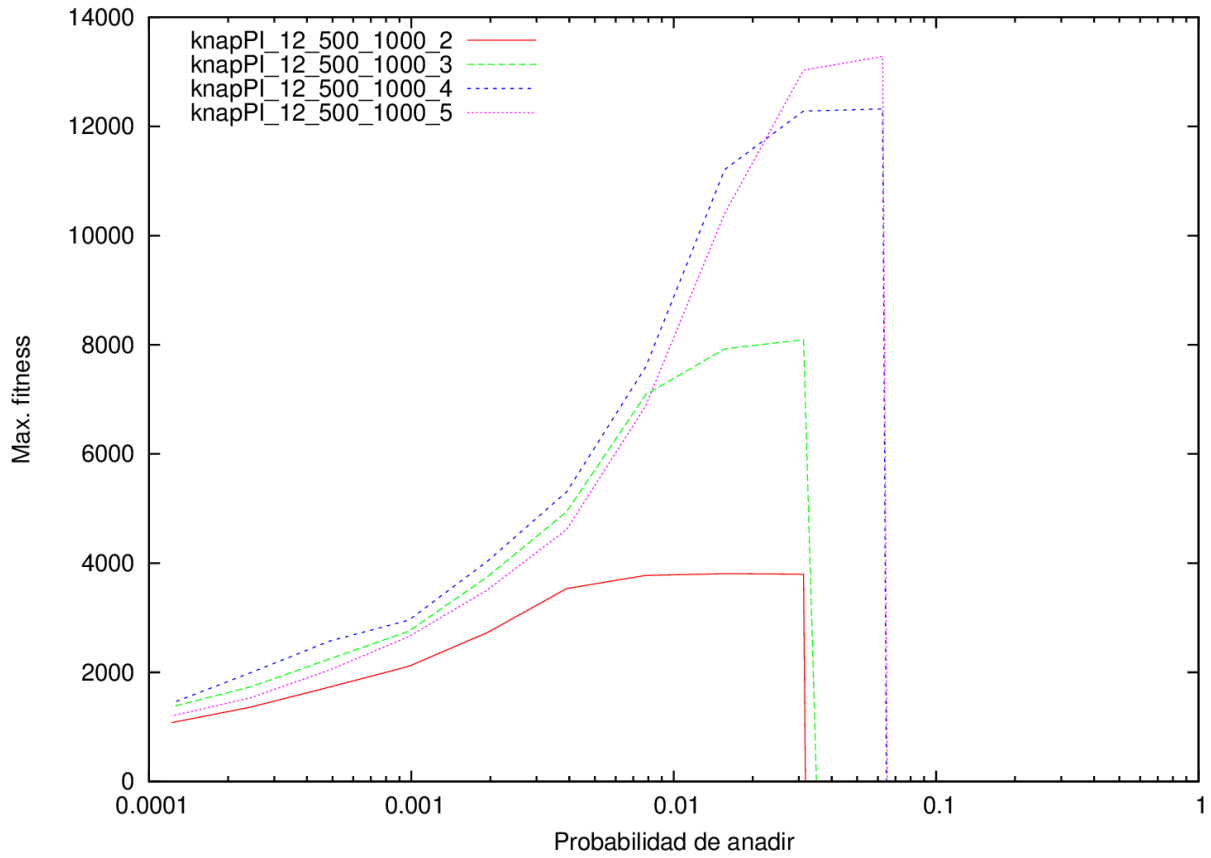

Figura 1: Resultados para el KP.

** Nota: cada una de las líneas representa cada una de las instancias (referenciadas en la leyenda).*

- **Observaciones:**
  - Con probabilidades muy altas la mochila se llena demasiado rápido y se obtienen soluciones inválidas (fitness negativo).
  - El punto crítico a partir del cual se empiezan a obtener soluciones válidas depende de la capacidad de la mochila y el peso total de los objetos.
  - Para las instancias proporcionadas, esta probabilidad es baja debido al gran peso total de los objetos comparado con la capacidad de la mochila.

Resultados de la prueba 2

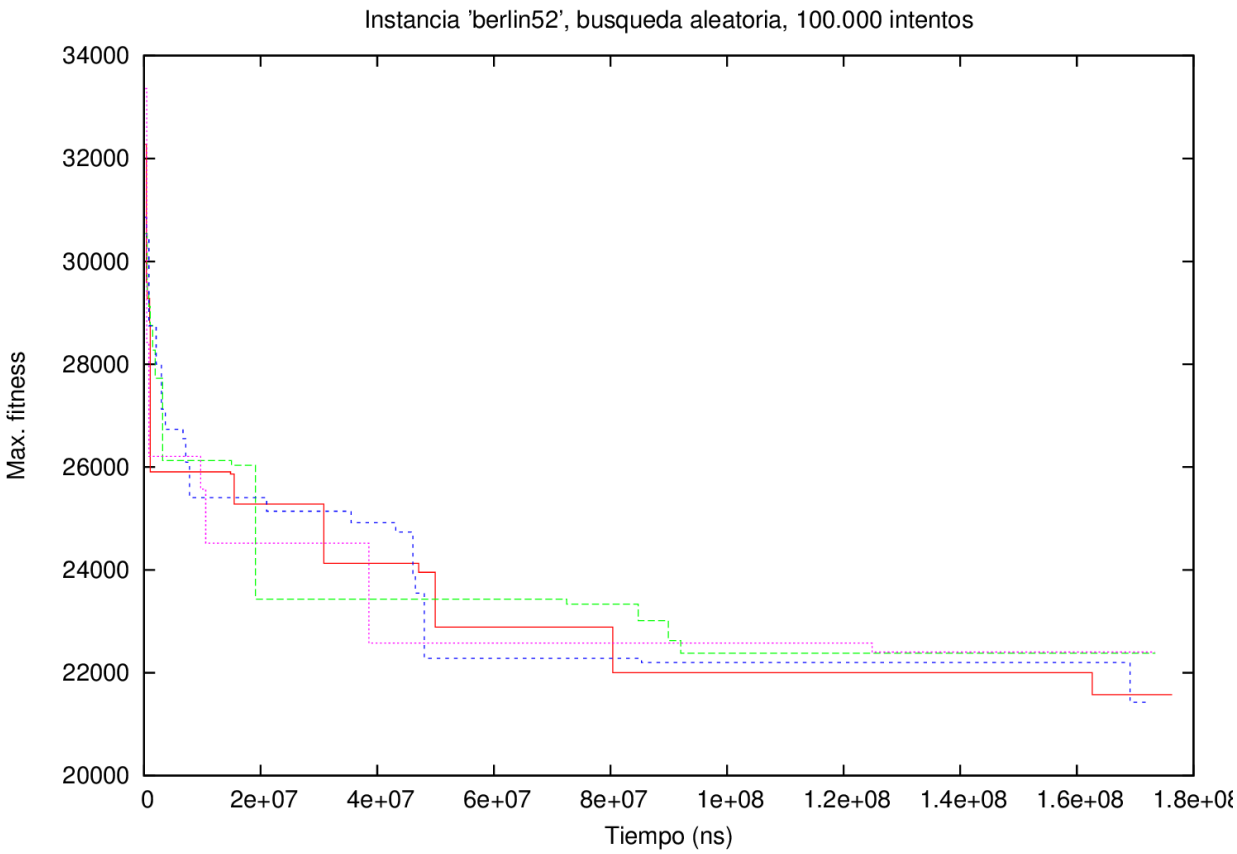

Figura 2: Resultados para TSP (berlin52)

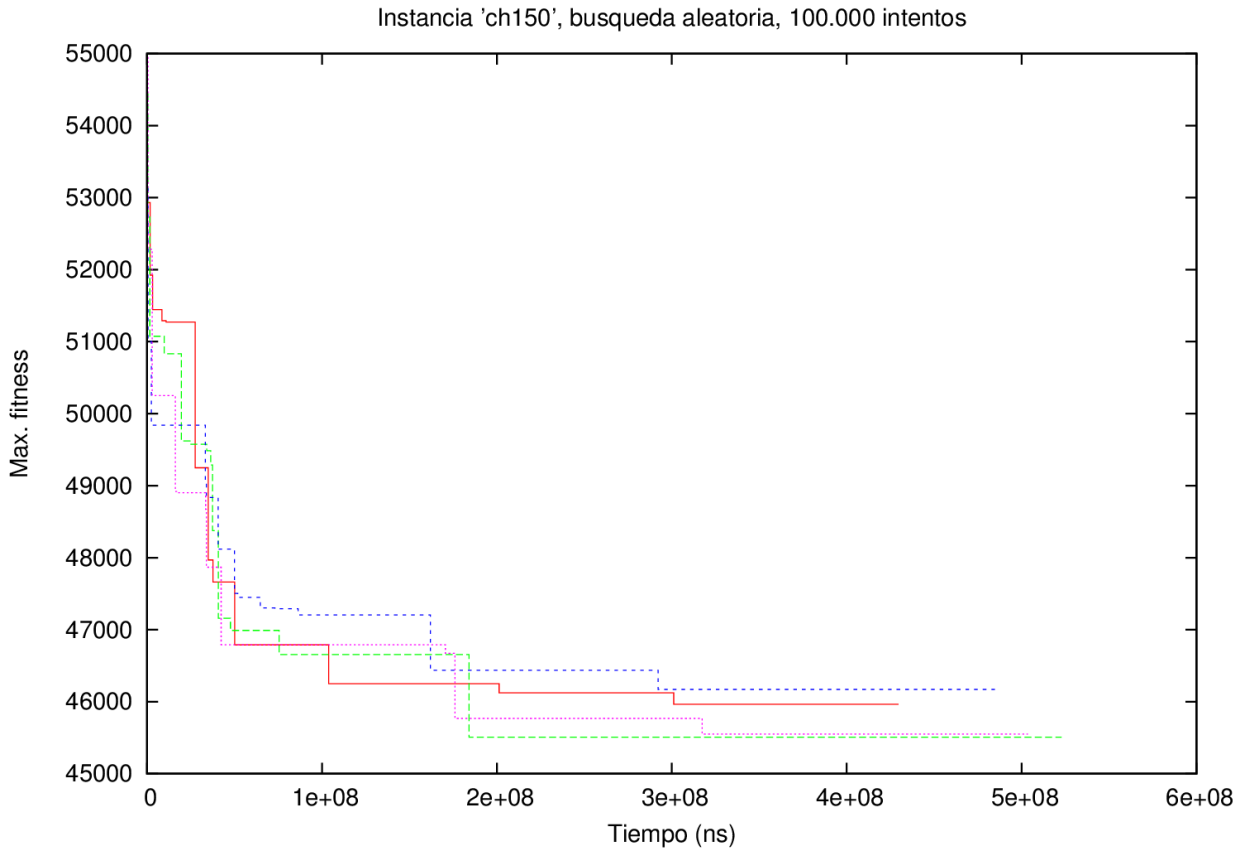

Figura 3: Resultados para TSP (ch150)

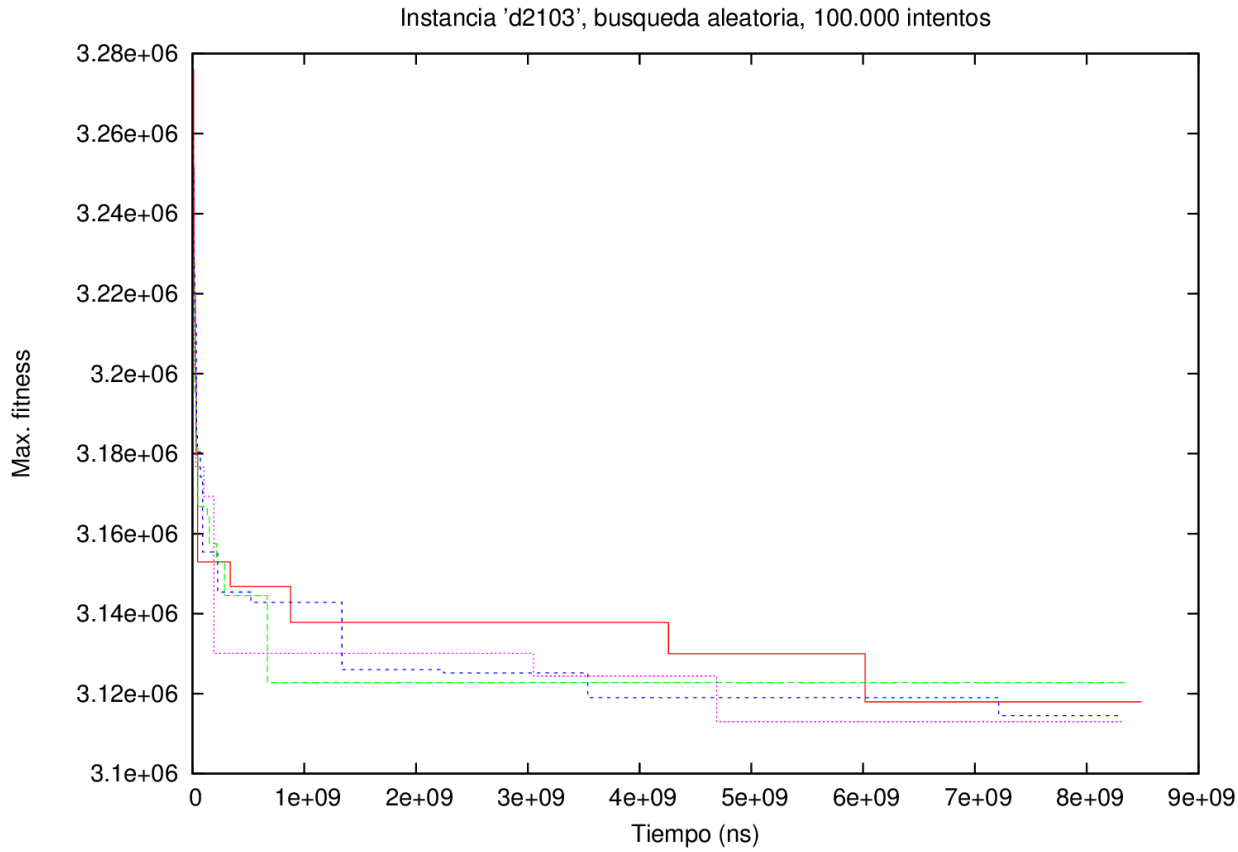

Figura 4: Resultados para TSP (d2103)

** Nota: Cada una de las líneas representa una ejecución de la búsqueda*

- **Observaciones:**
  - Con cada mejora, el tiempo medio transcurrido hasta la siguiente mejora aumenta ya que la probabilidad de mejorar la solución disminuye.
  - Si configuramos la búsqueda para que genere más soluciones, llega un momento en el que este tiempo es tan grande que las mejoras iniciales no se apreciarían en una gráfica.

Avances en el problema de la práctica 5

Hemos decidido abordar el problema de la mochila múltiple cuadrática (QMKP) de cara a la última práctica. Trataremos de aprovechar las estructuras de datos de la mochila simple y hemos pensado en representar las posibles soluciones mediante un vector de enteros de longitud el número de objetos. El valor almacenado en la posición i-ésima representa la mochila en la que se introduce el material i-ésimo (cero significa que no se introduce y cualquier otro valor *j* mayor que cero significa que se introduce en la mochila j-ésima).

Por otro lado, los beneficios conjuntos de los materiales se almacenarán en una matriz, tal y como se especifica en el fichero de entrada de datos.