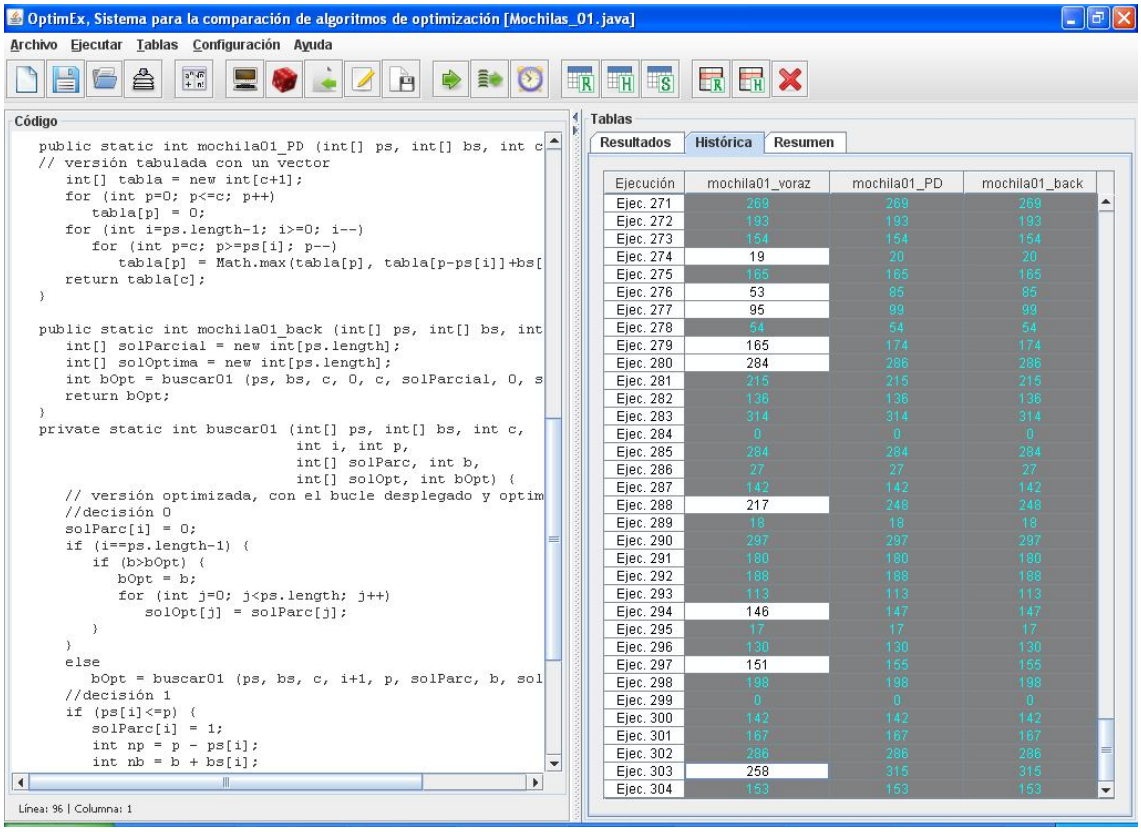
Guía rápida de uso de OptimEx

Introducción

El sistema OptimEx permite experimentar con la optimidad de algoritmos de optimización de forma parecida al sistema GreedEx. Puede observarse la similitud de la interfaz de usuario de ambos sistemas en la siguiente captura de pantalla de OptimEx:



Las principales diferencias entre ambos sistemas son:

– OptimEx es un sistema genérico, que permite comparar algoritmos de cualquier problema, mientras que GreedEx sólo soporta 6 problemas.
– GreedEx contiene una explicación teórica y genera visualizaciones y animaciones de los algoritmos soportado, mientras que OptimEx carece de estas características.

OptimEx es un fichero JAR, por lo que su instalación y uso sólo requiere tener instalado los JDK 6.0 ó 7.0.

Funcionamiento

OptimEx supone que van a compararse varios algoritmos contenidos en una misma clase Java, que tienen la misma signatura (tipos de los parámetros y del resultado) y que probablemente resuelven de forma distinta un mismo problema. Denominamos algoritmos aproximados a los algoritmos voraces o heurísticas, mientras que los algoritmos exactos se han desarrollado con las técnicas de vuelta atrás, ramificación y poda o programación dinámica.

La forma ideal de trabajar con OptimEx es la siguiente:

1. Se carga la clase en el editor. Al mismo tiempo, se compila automáticamente. La clase podrá editarse y compilarse más adelante si es necesario.
2. Se selecciona la signatura de los algoritmos a comparar. En el mismo diálogo se debe indicar si es un problema de maximización o minimización.
3. Se generan aleatoriamente, se editan o se cargan de fichero los datos con los que se quiere experimentar. Si se generan aleatoriamente, podemos indicar cuántos juegos de datos queremos generar. Para repetir más adelante el experimento con unos datos concretos, pueden generarse muchos juegos de datos (múltiplos de 100) y almacenarlos en un fichero para cargarlos cuando queramos volver a usarlos.
4. Se realiza una ejecución sobre los datos de entrada, normalmente una ejecución intensiva sobre muchos juegos de datos. En el diálogo correspondiente hay que indicar sucesivamente la siguiente información:
    a. Los métodos a comparar (entre los que comparten la signatura seleccionada). Si se sabe que algún algoritmo es óptimo, puede marcarse como referencia para la comparación; ante la duda, es mejor no marcar ningún método.
    b. La forma de generar los datos (seleccionar datos cargados desde un fichero, generar aleatoriamente un número de juegos de datos o generar aleatoriamente durante un cierto tiempo). Si se van a utilizar datos cargados de un fichero, se pide seleccionar los juegos de datos a usar. Si se van a generar datos aleatoriamente, se pide especificar las restricciones de rango.
5. Comparar los resultados obtenidos en las tablas histórica y de resumen. Su formato es parecido al de las tablas de GreedEx, aunque la tabla de resumen contiene información nueva que facilita la comparación de algoritmos exactos y aproximados:
    – Número de ejecuciones realizadas.
    – Porcentaje de casos en que, para cada algoritmo, se han obtenido resultados óptimos, subóptimos (es decir, menores en problemas de maximización o mayores en problemas de minimización) o “superóptimos” (a la inversa, es decir, mayores en problemas de maximización o menores en problemas de minimización). Estos últimos casos se dan cuando algún algoritmo es incorrecto.
    – Desviación, como porcentaje, de los resultados no óptimos de cada algoritmo con respecto a las soluciones óptimas, tanto la desviación media como la desviación máxima.
6. Exportación de las tablas a ficheros de formato gráfico para redactar el informe del experimento.

Dificultades y resultados erróneos

Conviene estar atentos a tres dificultades prácticas que pueden encontrarse al usar la versión actual de OptimEx:

– La generación de datos en OptimEx es aleatoria, sin poder imponer más restricciones que los rangos de valores. Sin embargo, en algunos problemas los datos deben satisfacer otras restricciones. Por ejemplo, en el problema de selección de actividades, el instante de fin de cada actividad debe ser mayor que el instante de inicio o, en el problema de llenado de cajas, cada objeto debe tener un peso menor que la capacidad de las cajas, para poder meterlo en alguna de ellas. Esta limitación de OptimEx obliga

a generar datos aleatoriamente y almacenarlos en un fichero, retocar en el fichero los casos prohibidos con un editor externo y luego cargarlos en OptimEx.

- La selección de la signatura y la indicación de que el problema es de maximización o de minimización se piden en el mismo diálogo. Hay que tener cuidado de no olvidar la segunda parte de este diálogo.
- Los algoritmos a comparar deben estar depurados, ya que en otro caso los resultados obtenidos pueden ser inesperados: algoritmos aproximados de maximización que devuelven valores mayores que un algoritmo exacto, a la inversa para algoritmos de minimización, e incluso lanzamiento de excepciones por errores de ejecución que pueden aparecer en algunas celdas de las tablas. Si se da alguna de estas situaciones, hay que corregir las condiciones del experimento o depurar algunos algoritmos. Veamos varias situaciones comunes (suponemos que se trata de un problema de maximización; para minimización, es a la inversa):
    - Algunas celdas de la tabla histórica contiene errores de ejecución. Puede haber dos posibles razones: el algoritmo contiene errores o los datos usados para la experimentación no cumplen las restricciones del problema. En el primer caso, debe depurarse el algoritmo; en el segundo, debe revisarse el juego de datos y descartar o corregir los datos inválidos.
    - Se obtienen valores "superóptimos". Esto sólo sucede cuando se ha identificado algún algoritmo como óptimo. El problema puede deberse a que:
        - Se ha seleccionado un algoritmo subóptimo. Debe volverse a repetir el experimento marcando como óptimo un algoritmo exacto o no marcando ninguno.
        - Se ha seleccionado un algoritmo que debiera ser óptimo. Deben revisarse varios algoritmos porque alguno es incorrecto: bien el algoritmo exacto devuelve valores menores de lo esperado o el algoritmo que devuelve valores "superóptimos" calcula valores demasiado altos (y por tanto inválidos).
    - Ningún algoritmo obtiene un porcentaje del 100% de valores óptimos. Esto sólo sucede cuando no se ha identificado algún algoritmo como óptimo. Este resultado es correcto si ninguno de los algoritmos es exacto. En caso contrario, se deben revisar y corregir los algoritmos exactos para que siempre calculen valores óptimos.