J. Ángel Velázquez Iturbide

Una Segunda Evaluación Cualitativa de la Comprensión de la Optimalidad

Número 2015-02

Serie de Informes Técnicos DLSI1-URJC
ISSN 1988-8074
Grupo Docente de Lenguajes y Sistemas Informáticos I
Universidad Rey Juan Carlos

Índice

Una Segunda Evaluación Cualitativa de la Comprensión de la Optimalidad

J. Ángel Velázquez Iturbide

Grupo Docente de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos,
C/ Tulipán s/n, 28933, Móstoles, Madrid
angel.velazquez@urjc.es

Resumen. OptimEx es un sistema para la experimentación interactiva con algoritmos de optimización. Este informe presenta un análisis de la comprensión de conceptos de optimización a partir de una sesión experimental con OptimEx celebrada en otoño de 2014. Se incluye el procedimiento y el enunciado usado, los resultados detallados y comentados, así como una discusión de los mismos. Los resultados son interesantes por los resultados, dificultades y opiniones de los alumnos. Asimismo, permiten tomar medidas correctoras para paliar en el futuro los resultados negativos.

Palabras clave: Algoritmos de optimización, experimentación interactiva, evaluación cualitativa, análisis de documentos.

1 Introducción

OptimEx es el acrónimo de "OPTIMization EXperimentation". Da nombre a un sistema interactivo para la experimentación con algoritmos de optimización. El objetivo genérico de OptimEx es permitir al alumno experimentar con distintos algoritmos para un mismo problema de optimización. El alumno debe comparar distintos algoritmos, determinando su optimalidad o suboptimalidad. En este último caso, también podrá determinar su desviación de la solución óptima, tanto en porcentaje de casos como en desviación media de la solución óptima.

Muchas de las características de OptimEx [1] están inspiradas en GreedEx [2]. Conviene usar ambos sistemas de forma combinada, primero GreedEx como introducción a los algoritmos voraces y otras técnicas de diseño de algoritmos de optimización, después OptimEx como sistema para la experimentación con algoritmos aproximados y quizá con técnicas de diseño exactas.

Previamente, se realizaron una evaluación de usabilidad [3] y una evaluación cualitativa [4] de los informes presentados por los alumnos para una práctica desarrollada con OptimEx. En este informe se presenta una segunda evaluación cualitativa de informes de los alumnos. La estructura del informe es la siguiente. El apartado 2 describe el protocolo utilizado. El apartado 3 presenta los resultados obtenidos y en el cuarto apartado se comentan. Finalmente, el apéndice contiene el enunciado de la práctica.

2 Protocolo

Esta evaluación se realizó en la asignatura obligatoria "Algoritmos Avanzados", de cuarto curso del Grado en Ingeniería Informática, en noviembre-diciembre del curso académico 2014-15.

En la asignatura se habían estudiado varias técnicas para la resolución de problemas de optimización: técnica voraz, vuelta atrás, ramificación y poda, programación dinámica y algoritmos aproximados. Habían realizado 4 prácticas:

1. Técnica voraz. Dado el problema de selección de actividades y pseudocódigo para su resolución voraz, se identifican dos funciones de selección óptima: orden creciente de instantes de fin (en adelante, representado F↑) y orden decreciente de instantes de inicio (en adelante, representado C↓). Se pide completar el algoritmo y ampliarlo para que no dependa de que las actividades estén ordenadas.
2. Técnicas de búsqueda. Se da el enunciado del problema de selección de actividades *ponderadas*. Se pide desarrollar un algoritmo de vuelta atrás y otro de ramificación y poda.
3. Eliminación de recursividad múltiple redundante. Se da una función recursiva f y se pide analizar su redundancia y eliminarla mediante memorización y tabulación.
4. Técnica de programación dinámica. Se da una solución recursiva para el problema de selección de actividades ponderadas y se pide convertirlo en un algoritmo de programación dinámica (tabulado).

La evaluación que presentamos se basó en la quinta práctica. Su objetivo era que los alumnos experimentaran con algoritmos exactos y aproximados. Participaron alumnos de dos grupos, presencial y online.

Se permitió realizar la práctica tanto individualmente como en parejas. Los alumnos se descargaron del campus virtual (con el que ya estaban familiarizados) el material necesario para realizarla: enunciado, clase Java con un algoritmo voraz y fichero con datos de entrada. También tenían disponible OptimEx y una pequeña guía de usuario. Debían entregar el informe sobre la práctica en el plazo de una semana. Los alumnos del grupo presencial asistieron a una sesión de prácticas de 2 horas de duración.

El enunciado de la práctica repetía la especificación del problema de selección de actividades ponderadas, ya conocido de las prácticas 2 y 4. Asimismo, la práctica 1 se había realizado sobre un problema muy parecido (selección de actividades).

A los alumnos se les pedía que compararan la optimalidad de 4 algoritmos al menos:

- Un algoritmo voraz basado en orden creciente de duración (proporcionado a los alumnos, representado D↑).
- Un algoritmo voraz desarrollado en la práctica 1, adaptado al nuevo problema.
- Un algoritmo de búsqueda entre los dos desarrollados (vuelta atrás, ramificación y poda) en la práctica 2 para este mismo problema.
- Algoritmo de programación dinámica desarrollado en la práctica 4 para este mismo problema.

Durante el tiempo de realización de la práctica se rectificó y se permitió que experimentaran con sólo 3 algoritmos debido a que algunos alumnos aún no habían terminado completamente la práctica 4.

Dadas las limitaciones de OptimEx para generación de datos con restricciones, se les proporcionaba un juego de 100 datos de entrada con los que realizar la experimentación.

El informe a entregar debía incluir:

- Identificación de los algoritmos utilizados, junto con una breve explicación de sus características principales de diseño, si debieron codificar algo y su código.
- Resultados de la experimentación en las tablas histórica y de resumen.
- Conclusiones.

Resumimos las principales diferencias con la práctica realizada un año antes [4]:

- Mejoras en el enunciado de la práctica:
  - Explicación mejor de qué algoritmos debían o podían compararse.
  - Porcentaje esperado de ejecuciones óptimas y subóptimas para D↑.
  - Petición de que en el informe se identificaran: algoritmos ejecutados y sus características principales, propuesta de algoritmos óptimos, y las tablas históricas y de resumen obtenidas.
- Se había ampliado la guía de usuario con una relación de resultados anómalos en una experimentación y sus posibles causas.
- Tras la corrección de la práctica y envío de comentarios por parte del profesor, se les permitió corregir errores y reenviar la práctica en el plazo de 4 días.

Una incidencia importante es que la versión disponible dio problemas a la mayoría de los alumnos durante la sesión, a veces por la versión de JDK instalada en el PC, otras por la generación de mensajes de compilación. Finalmente los problemas pudieron resolverse dentro o fuera de la sesión.

Se recogieron un total de 29 informes: 20 del grupo presencial (7 individuales y 13 parejas) y 9 del grupo on-line (6 individuales y 3 parejas). Solamente hubo 3 informes presentados en el segundo plazo: 2 del grupo presencial (corrigieron errores pero no los resultados de la experimentación) y 1 del grupo on-line (no pudo entregarlo en el primer plazo).

Incluimos el enunciado de la práctica en el Apéndice.

3 Análisis

Los datos se analizaron utilizando los resultados de la evaluación del curso anterior [4]. Se construyó una tabla donde aparecieran los elementos que se consideraron importantes para este análisis. Se realizaron dos series de rondas de análisis:

1. Ronda primera. Se construyó una tabla igual que la elaborada en la evaluación anterior.

2. Rondas posteriores. Se realizó una ronda para analizar cada aspecto identificado en el análisis anterior. Aunque coincidieron en muchos aspectos, hubo algunas diferencias importantes con respecto al anterior.

Conviene comentar que cada ronda no fue secuencial sino que implicó numerosas consultas a los informes y a las tablas construidas.

4 Resultados

Presentamos los resultados del cuestionario en la Tabla 1. La tabla incluye una fila por cada informe entregado. Cada columna tiene el siguiente significado:

- "Grupo". Asigna un número a cada grupo, indicando también si es individual "I" o una pareja "P".
- "Algoritmos usados". Identifica los algoritmos usados por el grupos en su experimentación. Puede ser voraz con función de selección "F↑", "C↓" o "D↑", vuelta atrás "Back", ramificación y poda "RyP" o programación dinámica "PD". También indica si se ha realizado una modificación menor ("adaptado", p.ej. la modificación de un algoritmo voraz para el problema de selección de actividades al problema de selección de actividades ponderadas) o mayor ("modificado", p.ej. añadir una fase inicial de ordenación al algoritmo de programación dinámica de la práctica 4)
- "Condiciones experimento". Da alguna información específica sobre el experimento realizado por el grupo. Al menos se incluye el número de datos de entrada utilizados. Otras condiciones son: si la experimentación se realizó marcando algún algoritmo como óptimo, y si se realizó como si fuera un problema de minimización.
- "Resultados". Se indica qué algoritmos han resultado óptimos, subóptimos o superóptimos, así como si se obtienen los resultados esperados para D↑ (véase apartado anterior).
- "Propuesta". Si los alumnos han propuesto explícitamente algunos algoritmos como óptimos.
- "¿Bien?". Utilizamos el símbolo ✓ para indicar que el grupo identifica todos los algoritmos, entre los que experimenta, que debieran ser óptimos.
- "Dificultades y malentendidos experimentación". Explicación mía de los problemas o malentendidos que parecen tener.
- "Comentarios abiertos". Comentarios realizados por los grupos en el apartado de conclusiones del informe.
- "Otros". Otros comentarios míos no relacionados directamente con la experimentación realizada.

Tabla 1. Análisis de los informes de los alumnos

Grupo	Algoritmos usados	Condiciones experimento	Resultados	Propuesta	¿Bien?	Dificultades y malentendidos experimento	Comentarios abiertos	Otros
1. I	D↑ F↑ adaptado C↓ adaptado RyP modificado PD adaptado	– 100 – PD marca	– RyP y PD óptimos – Voraces subóptimos – Control D↑ bien	RyP PD	✓	– RyP con nueva cota – PD con ordenación	– Problema con OptimEx (uso) – Sugiere medir complejidad para poder valorar el uso de los algoritmos voraces	– Ha hecho pruebas con un programa propio antes de usar OptimEx
2. P	D↑ C↑ adaptado RyP Back	– 100 – No marca	– RyP y Back óptimos – Voraces subóptimos – Control D↑ bien	RyP Back	✓		– Sugiere medir complejidad para conocer qué algoritmo es mejor – Valoración positiva de OptimEx, con sugerencia de mejora (medidas de tiempo o memoria) – Satisfacción práctica final	– Clasifica los algoritmos voraces en heurísticos y aproximados sin argumentos sólidos
3. P	D↑ F↑ adaptado RyP	– 100 – Minimizan – No marca	– Todos subóptimos – RyP con resultados aparentemente grandes y un resultado 0	–		– Inconsistencia en nombres de método – Detectan problemas pero no advierten ni corrigen la minimización	– Problemas con OptimEx, con sugerencia de mejora (vídeo) – Dicen que han modificado todas las prácticas pero no aclara cómo en RyP – Comentario ambiguo sobre optimidad y eficiencia – Importancia de elegir la técnica de diseño adecuada	
4. I	F↑ adaptado RyP D↑ PD	– 100	– Todos subóptimos (F↑ y RyP en 100% casos) – RyP y Back devuelven lo mismo en muchas ejecuciones	PD		– PD confuso: prevé ordenar pero no parece que lo haga – Uso inadecuado del término “óptimo” (mayor % de casos)	– Sin dificultad de uso de OptimEx “siguiendo instrucciones” – Comentario erróneo sobre eficiencia y optimalidad	– Usa variables globales de clase

2

5.	I	D↑ F↑ adaptado Back PD	– 100	– Todos subóptimos – F↑ con resultados 0 – PD lanza 2 excepciones (ejec. 3 por actividad con el mismo instante de inicio y fin; ejec. 49 quizá actividad que comienza en 0) – Back y PD dan valores menores que D↑	PD		– PD recursivo y sin ordenar – Identifica “óptimo” con el mejor en un mayor % de casos (“algoritmo más óptimo”)	– Problemas con OptimEx (JDK, carga datos, ejecución) – Comentario sobre ineficiencia del PD recursivo – ¡Propone comparar también los algoritmos de RyP y PD (tabulado)!	
6.	I	D↑ C↓ modificado Back modific. PD	– 100 – Minimizan – Back marca	– Voraces subóptimos y superóptimos – PD subóptimo	Back		– PD no ordena las actividades – Identifica “óptimo” con el mejor en un mayor % de casos (“algoritmo más óptimo”)	– Le sorprenden los resultados – Dificultad de adaptación de algoritmos a OptimEx	
7.	I	D↑ F↑ modificado Back modific.	– 100	– Back óptimo – Control D↑ bien	Back	✓	– Rehecho F↑ y Back	– Problemas con OptimEx y soluciones (JDK, tipos) – Describe su proceso de resolución de dificultades – Back es un “eje ficticio de los resultados para contrastar”	
8.	I	D↑ C↓ adaptado F↑ nuevo Back adaptado	– 100	– Back óptimo – Control D↑ bien	Back	✓	– Nuevo F↑ y Back reestruct.	– Incapacidad de desarrollar PD – Problema con OptimEx (marcar juegos de datos) – Satisfacción con carácter experimental de la práctica y OptimEx	

9. P	D↑ F↑ Back PD	– 100 – D↑ marca	– Todos superóptimos de D↑ – Dos algoritmos con resultados óptimos y superóptimos de D↑			– PD recursivo y sin ordenar – Experimentación progresiva de 2, 3 y 5 algoritmos – Falta de información sobre las técnicas y decisiones de diseño de los algoritmos – Falta de visibilidad de los identificadores de los métodos para su correcta identificación en las tablas – Uso de “más óptimo” con dos significados distintos	– Problemas con OptimEx (JDK, entrada datos, resultados de métodos, tablas) – Satisfacción con carácter experimental de la práctica	
10. P	D↑ C↓ adaptado Back PD	– 100	– Todos subóptimos	Back		– PD no ordena las actividades	– Satisfacción práctica final y carácter experimental – Satisfacción y problema con OptimEx (JDK)	
11. I	C↓ adaptado Back PD	– 107 – D↑ no se compara	– Todos subóptimos – PD lanza excepciones – No da datos de PD			– PD no ordena las actividades – No ha querido comparar PD	– Problemas con OptimEx (JDK, ejecuciones)	
12. I	F↑ adaptado Back PD modificado D↑	– 100	– RyP y Back óptimos – Voraces subóptimos – Control D↑ bien	Back PD	✓	– Explica identificación de error en PD y modificación	– Satisfacción con carácter experimental de la práctica	
13. I	D↑ F↑ o C↓ adapt. Back PD	– 100 – Datos generados	– Todos subóptimos			– Imprecisión algoritmo voraz – Imprecisión modificación Back – PD no ordena las actividades	– Satisfacción y problema con OptimEx (JDK)	
14. I	F↑ adaptado RyP D↑ PD modificado	– 100 – RyP o PD marca	– RyP y Back óptimos – Voraces subóptimos – Control D↑ bien	RyP PD	✓	– PD con ordenación		

15. P	D↑ F↑ adaptado Back PD	– 100	– Todos subóptimos – Control D↑ bien (curiosamente)			– Descartaron RyP y adaptaron Back – PD no ordena las actividades	– Satisfacción práctica final y su carácter experimental – Satisfacción (fácil de usar, útil) y problema con OptimEx (JDK)	
16. P	D↑ F↑ adaptado Back RyP PD modificado	– 100	– PD óptimo – Voraces subóptimos – Control D↑ bien	PD	✓	– Back y RyP dan resultados distintos con OptimEx y NetBeans (ejecs. 30 y 43) – Tras eliminarlos, los resultados de los otros 3 algoritmos están bien		
17. P	F↑ adaptado Back D↑ PD	– 100	– Todos subóptimos – Back lanza una excepción (ejec. 3)			– Identifican el problema en Back pero no lo depuran – PD no ordena las actividades – Identifica “óptimo” con el mejor en un mayor % de casos (“algoritmo más óptimo”)	– Satisfacción y problemas con OptimEx (JDK, ejecuciones) – Satisfacción práctica final	
18. I	D↑ F↑ adaptado Back adaptado PD modificado	– 100	– Back y PD óptimos – Voraz subóptimo – Control D↑ bien	Back PD	✓	– PD con ordenación	– Satisfacción práctica final – OptimEx fácil de usar y útil	– Dice que su algoritmo de Back es de Back o de RyP
19. P	F↑ adaptado Back modific. PD modificado D↑	– 100	– Todos subóptimos	PD		– PD no ordena las actividades – Informan de problemas – Uso inadecuado del término “óptimo” (mayor % de casos)	– Problemas con OptimEx (JDK, ejecuciones)	
20. P	F↑ adaptado RyP D↑ modificado PD	– 100 – RyP marca	– Todos subóptimos y superóptimos de RyP			– PD no ordena las actividades – Confusión en posibles modificaciones de D↑ y PD para ordenar las actividades	– Satisfacción práctica final – Propuestas de mejora de OptimEx (manual, guía)	

21. I	D↑ F↑ adaptado Back adaptado PD modificado	– 100 – Minimiza	– Back siempre da 0 – Resultados muy dispares entre los dos algoritmos voraces – PD siempre lanza excepción				– Problemas con OptimEx (ejecuciones) – Sin tiempo para corregir los algoritmos	
22. P	RyP F↑ D↑	– 100	– RyP óptimo – Voraces subóptimos – Control D↑ bien	RyP	✓		– Satisfacción y problemas con OptimEx (JDK, compilación, manual) – Práctica muy cercana a la anterior	
23. P	F↑ adaptado Back D↑	– 100	– Back óptimo – Voraces subóptimos – Control D↑ bien	Back	✓	– Comentario raro: “En la taba de historificación de resultados, podemos ver el beneficio de cada actividad y las veces que se extrae una solución óptima”	– Problemas con OptimEx y soluciones (clase única, cargar datos)	
24. P	F↑ adaptado RyP modificado D↑ PD modificado	– 100	– RyP y PD óptimos – Voraces subóptimos – Control D↑ bien	RyP PD	✓	– Cree necesario ordenar las actividades en RyP (“según enunciado de pr.2”) – PD con ordenación – Identifica “óptimo” con el mejor en un mayor % de casos (“algoritmo más óptimo”)	– Satisfacción (manual) y problemas con OptimEx (compilación) – Satisfacción con carácter experimental de la práctica	– RyP en realidad es Back
25. P	F↑ adaptado Back modific. D↑	– 100 – Minimiza – Back marca	– Voraces superóptimos – F↑ subóptimo – Control D↑ bien pero invertido	Back		– Comentan que los resultados de D↑ no coinciden con los del enunciado	– Problema con OptimEx (JDK) – Práctica sencilla	

26. P	F↑ adaptado Back D↑ PD modificado	– 100	– Back y PD óptimos – Voraces subóptimos – Control D↑ bien	Back PD	✓	– Considera PD “más óptimo” que Back por su eficiencia – Identifica “óptimo” con el mejor en un mayor % de casos (“algoritmo más óptimo”)	– Satisfacción con la práctica – Satisfacción (interfaz, comparación, exportación) y problemas con OptimEx (JDK)	
27. P	F↑ adaptado RyP D↑ PD modificado	– 100	– RyP y PD óptimos – Voraces subóptimos – Control D↑ bien	RyP PD	✓	– RyP ordena las actividades en F↑	– Esfuerzo en corregir errores de prácticas anteriores – Propone repartir esta práctica a lo largo de la asignatura	
28. I	D↑ F↑ adaptado C↓ nuevo RyP PD modificado	– 100	– RyP y PD óptimos – Voraces subóptimos – Control D↑ bien	RyP PD	✓			
29. P	F↑ adaptado PD modificado Back D↑	– 100	– Back y PD óptimos – Voraces subóptimos – Control D↑ bien	Back PD	✓		– Satisfacción práctica final y su carácter experimental – Satisfacción con OptimEx	

En el análisis que presentamos a continuación, utilizamos la notación G*nn* para referirnos al grupo con número identificativo *nn*.

4.1 Resultados de la Experimentación

Veamos primero los resultados globales, que se encuentran en la Tabla 2.

Tabla 2. Resultados de la experimentación

Resultado	**# (%) grupos**	**Grupos**
Bien	15 (51'7%)	G01, G02, G07, G08, G12, G14, G16, G18, G22, G23, G24, G26, G27, G28, G29
Mal	14 (48'3%)	G03, G04, G05, G06, G09, G10, G11, G13, G15, G17, G19, G20, G21, G25

No todos los grupos que han realizado bien la práctica están exentos de incidencias:

- G16 comprobó que obtenía resultados distintos para los algoritmos de vuelta atrás y ramificación y poda utilizando OptimEx o NetBeans. Suprimieron ambos algoritmos y los otros tres algoritmos ya dieron resultados correctos.
- G24 y G26 utilizan un lenguaje inapropiado sobre el concepto de optimalidad.
- G24 y G27 ordenan las actividades en $F\uparrow$ para sus algoritmos de ramificación y poda. G24 cree que el enunciado de la práctica 2 suponía que las actividades estaban ordenadas. G27 las ordena para que los valores calculados sean óptimos.

En cuanto a los grupos que han realizado mal su experimentación, algunos indicios indicaban la existencia de problemas:

- Ejecuciones de algún algoritmo con resultado 0 (G03, G05, G21). Destaca G21, con todos los resultados del algoritmo de vuelta atrás igual a cero.
- Ejecuciones de algún algoritmo con lanzamiento de excepciones (G05, G11, G17, G21). De nuevo, G21 destaca al tener todas las ejecuciones del algoritmo de programación dinámica lanzando una excepción.
- Muchas ejecuciones de algún algoritmo con el mismo resultado (G04).
- Se marca como óptimo un algoritmo exacto y se obtienen resultados superóptimos con algún algoritmo voraz (G06, G20, G25). Deberían revisarse los algoritmos para analizar dónde está el error: si el algoritmo exacto no es óptimo o si el algoritmo voraz devuelve valores inválidos.
- Se marca como óptimo un algoritmo exacto y se obtienen resultados subóptimos parar otro algoritmo exacto (G06, G20). Deberían revisarse los algoritmos exactos para analizar cuál es incorrecto.
- Se realiza una experimentación abierta en la que ningún algoritmo arroja resultados óptimos (G03, G04, G05, G10, G11, G13, G15, G17, G19, G21). Debería revisarse porqué los algoritmos exactos no son óptimos. En algunos casos los resultados eran especialmente llamativos: algoritmos subóptimos en casi el 100% de los casos (G04).

- Resultados para D↑ distintos de los dados en el enunciado. Destaca G09 que marca D↑ como óptimo, obteniendo el algoritmo de programación dinámica los porcentajes previstos para D↑ (los subóptimos previstos para D↑ se obtienen como porcentajes superóptimos del algoritmo de programación dinámica). Por otro lado, G25 obtiene los mismos porcentajes que en el enunciado de la práctica, pero el porcentaje previsto de resultados subóptimos aparece en superóptimos (estaba minimizando en lugar de maximizar).

Veamos las razones de que los resultados sean incorrectos. En algunos grupos se dan varios factores (G03, G06, G25):

- Dos grupos (G11, G13) no respetaron las condiciones del experimento sobre los juegos de datos. Un grupo (G11) utilizó 107 juegos de datos, según su explicación probablemente debido a que OptimEx daba problemas para hacer exactamente 100 ejecuciones. Otro grupo (G13) generó los 100 datos aleatoriamente.
- Un grupo (G09) marcó el algoritmo de D↑ proporcionado, considerando que este era el objetivo del experimento, aunque aparentemente los algoritmos están bien.
- Cuatro grupos (G03, G06, G21, G25) no marcaron la casilla de maximizar, lo condujo a resultados erróneos o de difícil interpretación. Dos de ellos (G03, G25) detectaron que obtenían resultados distintos para D↑ de los indicados en el enunciado de la práctica, pero no lo arreglaron. G06 marcó como óptimo el algoritmo de vuelta atrás. Parece estar mal porque todos los demás algoritmos dan resultados subóptimos (que serían superóptimos si se maximizara); el resto de algoritmos parece estar bien.

  El caso de G25 es algo especial porque realizaron dos entregas, siendo conscientes de los resultados de control en ambas entregas. En la primera entrega ningún algoritmo resultó óptimo. En la segunda entrega el algoritmo de vuelta atrás fue óptimo porque lo marcaron pero al minimizar, los algoritmos voraces daban resultados superóptimos en casi todos los casos. Los porcentajes de D↑ eran correctos, sólo que inversos (al minimizar). Sin embargo, F↑ daba resultados subóptimos, que corresponderían a superóptimos, es decir tenía algún error.
- Algún algoritmo es incorrecto (G03, G04, G05, G06, G10, G15, G17, G19, G20, G21, G25) y por tanto da resultados incorrectos, invalidando su comparación con otros algoritmos, quizá correctos. Los grupos de esta categoría intersectan con la tercera categoría, no lo hacen con la segunda y no sabemos si intersectan con la primera. Ya hemos comentado que algunos indicios de que los algoritmos son erróneos son el lanzamiento de excepciones, la obtención de resultados cero o la obtención del mismo resultado en varias ejecuciones de un mismo algoritmo.

  Es muy frecuente el caso de grupos que incluyeron un algoritmo de programación dinámica (12 de 14: G04, G05, G06, G09, G10, G11, G13, G15, G17, G19, G20, G21) pero no se dieron cuenta de que el algoritmo desarrollado en la práctica 4 presuponía que las actividades venían ordenadas en orden creciente de fin. Al no existir esta precondición en la práctica 5,

debían modificar el algoritmo de programación dinámica añadiendo un primer paso de ordenación de las actividades. Algunos casos atípicos son:

- G04 afirma que el algoritmo ordena inicialmente las actividades pero no hay rastro de ello en el código incluido y los resultados no son óptimos en el 100% de los casos (de hecho, todas las ejecuciones devuelven unos pocos valores iguales).
- G05 y G09 utilizan la versión recursiva en lugar de la tabulada. G09 marca D↑ como algoritmo óptimo, obteniendo en el algoritmo de programación dinámica los porcentajes previstos para D↑ (en filas distintas).

4.2 Terminología

Podemos distinguir entre dos significados de la palabra "óptimo":

- "Resultado óptimo" (o valor o beneficio) significa el máximo valor (o mínimo si el problema fuera de minimización) que puede obtenerse para unos datos dados.
- "Algoritmo óptimo" es un algoritmo que devuelve el resultado máximo (u óptimo) siempre, es decir, para datos de entrada cualesquiera.

Estos significados los han mantenido correctamente diversos grupos de ambas categorías. Otros grupos no comentan sus hallazgos o los describen de manera trivial, de forma que no es posible conocer su uso del lenguaje.

Sin embargo, hay algunos grupos que utilizan el término "algoritmo óptimo" para referirse a un algoritmo que calcula un resultado óptimo en un porcentaje mayor de casos. Este significado suele encontrarse en grupos que hicieron la práctica mal (G04, G05, G06, G09, G17, G19), pero también en algunos grupos que la hicieron bien (G24, G26). Veamos este uso para un algoritmo que obtiene resultados óptimos es un mayor número de casos, pero no en todos:

- "Como era de esperar, el algoritmo óptimo es el de la práctica 4 (algoritmo exacto). La técnica de programación dinámica combinada con la tabulación genera el resultado óptimo en el 84% de los casos de los ejemplos del test." (G04).
- "En nuestro caso, nos sale como algoritmo óptimo, el algoritmo de la práctica 4 (programación dinámica). Vemos cómo tiene un 60% de soluciones óptimas." (G19).

Este uso de la palabra "óptimo" suele venir expresado como "más óptimo" o "menos óptimo". Esta expresión se utiliza como sinónimo de "con mayor porcentaje de resultados óptimos" o simplemente de "mejor":

- "De estos datos deducimos que el *algoritmo más óptimo* es el implementado con el método de programación dinámica, ya que obtenemos un 64% de ejecuciones óptimas. (...) También deducimos que el algoritmo implementado con la técnica de vuelta atrás posee un porcentaje de ejecuciones óptimas bastante bajo, con un 17% mientras que el algoritmo voraz con orden creciente de momento de finalización es *el menos óptimo* con un 0%." (G05)

– “Como se puede deducir con los datos generados por Optimex los *algoritmos más óptimos* son los de los algoritmos de ramificación y poda ya que calcula la solución óptima en el 100% de los casos de prueba, de la misma manera el algoritmo de programación dinámica nos brinda un resultado idéntico, estos algoritmos son algoritmos exactos.“ (G24)

Aunque este uso de “más óptimo” es más frecuente con al segundo significado de óptimo antes señalado, también lo encontramos para el primer significado:

– “Si por ejemplo comparamos el algoritmo ofrecido con el de la práctica 4 (recordemos que ya ha sido ordenado, por lo que el algoritmo es válido) observamos que nuestro algoritmo supera al algoritmo voraz ofrecido por el profesor en un 82%, lo que significa que en 82 casos nuestro algoritmo encuentra *una solución más óptima* que el voraz. (..) De igual manera podemos realizar comparaciones con *un algoritmo más óptimo* (obteniendo resultados superóptimos en comparación con el facilitado) y seguir mostrándolo bajo las mismas condiciones, tal y como muestran las siguientes figuras.” (G09).

Finalmente, un grupo habla de “eficiencia” en términos de optimalidad (G03). Aunque es correcto usar esta terminología, puede conducir a confusión dado que la eficiencia de algoritmos suele referirse al uso del tiempo o de la memoria:

– “A la vista de los resultados creo que tenemos que tener algo mal en el algoritmo porque creemos que el más eficiente y el que tendría que tener más resultados óptimos es el RyP pero nos ha salido mejor el voraz creciente fin de actividad. (..) En lo relacionado con la práctica he podido comprobar lo importante que es elegir una buena estrategia para desarrollar un algoritmo eficiente.” (G03)

De hecho, otro grupo (G04) hace un comentario claramente erróneo:

– “Queda patente que los algoritmos exactos, además de alcanzar una solución óptima en una amplia mayoría de pruebas, además ejecuta menos cálculos.” (G04).

4.3 Trabajo Realizado en la Práctica y Autoconocimiento

Repasemos el trabajo realizado y el autoconocimiento de los grupos. Los grupos que desarrollaron bien la práctica han advertido las modificaciones que debían realizar en los algoritmos para su comparación y los resultados que eran aceptables. Aquellos que compararon el algoritmo de programación dinámica han añadido una fase inicial de ordenación (10 de 15 grupos: G01, G12, G14, G16, G18, G24, G26, G27, G28, G29).

Estos grupos a veces dan una relación detallada de las dificultades que han encontrado y cómo las han resuelto (G07, G08, G12, G16). Así, G07 informa de las modificaciones que debió realizar en los algoritmos voraz y de vuelta atrás. G08 no había realizado el algoritmo de programación dinámica, por lo que debió desarrollar otro algoritmo voraz; también tuvo que adaptar la estructura de clases del algoritmo de vuelta atrás. G12 informa de un error que corrigió en su algoritmo de

programación dinámica. Otro grupo (G16) que detectó que los algoritmos de búsqueda daban resultados distintos al ejecutarlos con NetBeans y con OptimEx. Ante la imposibilidad de otra acción posible, los descartaron de la comparación.

Sin embargo, son numerosos los grupos que hicieron la práctica mal y simplemente incluyen sus resultados, sin más comentario (8 de 14 grupos: G04, G05, G06, G09, G10, G13, G15, G20).

Los restantes 6 grupos que hicieron mal su práctica detectaron que algo fallaba:

- Tres grupos (G11, G17, G21) han detectado el lanzamiento de excepciones en algunas ejecuciones pero no los corrigen. Un grupo (G17) incluso aventura la razón de las excepciones. Otro grupo (G21) indica que no tiene tiempo para indagar en los errores y corregirlos; se propone hacerlo en la ampliación de plazo pero finalmente no lo hace.
- Dos grupos (G03, G25) simplemente advierten que los resultados obtenidos no son los esperados.
- Un grupo (G19) comenta "En nuestro caso, hemos tenido muchos problemas a la hora de cambiar los algoritmos ya que, siempre que comparábamos juntos los 4 algoritmos, no nos daban las soluciones óptimas de forma correcta." No obtienen resultados óptimos para ningún algoritmo pero se dan por satisfechos al ser el algoritmo de programación dinámica el que obtiene resultados mejores en un número mayor de casos (60%).

En cuanto a la percepción de la dificultad de la práctica, la Tabla 3 muestra los grupos que expresaron alguna opinión.

Tabla 3. Percepción de la facilidad de la práctica

Resultado	Práctica fácil	Práctica menos fácil de lo que parecía
Correcto		G03
Incorrecto	G25	G11, G17

4.4 Valoración Subjetiva

El enunciado de la práctica animaba a los alumnos a realizar comentarios de valoración en el apartado de conclusiones de la memoria de la práctica. Sus comentarios suelen valorar la facilidad de la práctica y su utilidad, e identifican puntos positivos y negativos de OptimEx. La Tabla 4 clasifica las respuestas de valoración de la práctica.

Tabla 4. Clasificación de las respuestas de valoración personal

Resultado	Satisfacción por práctica final	Satisfacción por práctica experimental	Otros comentarios
Correcto	G02, G18, G29	G08, G24, G26	G22, G27
Incorrecto	G10, G12, G15, G17, G20	G09	

En líneas generales, no hay grandes diferencias en el número de comentarios recibidos de grupos que realizaron la práctica correcta e incorrectamente.

Los alumnos que están satisfechos con la práctica como colofón de la asignatura valoran que engloba todas las técnicas de diseño anteriores por medio de las prácticas realizadas a los largo de la asignatura. Por ejemplo: “Me ha parecido muy interesante esta práctica, ya que he podido evaluar la eficacia de casi todos los algoritmos desarrollados en las prácticas de esta asignatura.” (G10); “Con esta práctica la sensación que nos queda es la de saber apreciar la diferencia entre todos los tipos de algoritmos propuestos durante el curso, puesto que siempre estamos trabajando con los mismos ejemplos. En un principio puede parecer repetitivo, pero finalmente creemos que ha sido la mejor forma de apreciar las pequeñas y grandes diferencias que aportan los distintos algoritmos y las consecuencias de implementar uno u otro.” (G17).

Otros grupos valoran el carácter experimental o comparativo de la práctica (frente a otras de codificación). Por ejemplo, “Para finalizar, he de decir que me ha gustado bastante hacer esta práctica porque normalmente sólo se pide implementar los algoritmos y en base a la teoría nos tenemos que creer que son más óptimos. Con OptimEx se nos permite hacer una valoración objetiva de la optimalidad de los algoritmos corroborando lo que aprendemos en teoría.” (G08).

Ambos aspectos no son excluyentes pero hemos priorizado el criterio en el que más incide el grupo. Por ejemplo, los grupos que resaltan que se comparan algoritmos se clasifican en el segundo aspecto pero si resaltan que se comparan las prácticas de toda la asignatura, se clasifican en el primero: “Con esta práctica hemos podido comparar los algoritmos que hemos ido desarrollando a lo largo de la asignatura entre sí, viendo su optimalidad. Además, nos ha servido de repaso de lo visto en la asignatura y para mejorar ciertos aspectos en algunos puntos (como en el backtracking).” (G15).

Otras dos comentarios contienen una queja de la cercanía de esta práctica con la anterior, cuando había que utiliza el algoritmo de programación dinámica desarrollado en esta, y una propuesta de repartir las comparaciones a lo largo de todo el curso, simultaneando desarrollo de un algoritmo y comparación con otros: “Por tanto, una posible mejora en las prácticas para evitar esta carga de trabajo en la última proponemos que en cada práctica, aparte de que el resultado sea el mismo al ejemplo que se expone en el enunciado de dicha práctica, se hubiese presentado la herramienta OptimEx para poder probar que los algoritmos propuestos eran óptimos en el caso de ramificación y poda y programación dinámica, o un porcentaje de resultados óptimos en el caso de algoritmos voraces.” (G27).

En cuanto a OptimEx, se recibieron comentarios de 23 grupos (79’3%, todos los grupos salvo G06, G12, G14, G21, G27, G28). Existen valoraciones positivas pero también problemas encontrados, soluciones adoptadas y propuestas de mejora. Podemos utilizar los resultados de la evaluación de usabilidad anterior [3] para clasificar las respuestas recibidas. Vemos en primer lugar los comentarios positivos. Se recibieron respuestas positivas de 13 grupos (44’8%). Solamente incluimos algunas citas aclaratorias:

- Utilidad para comparar algoritmos de optimización (13 respuestas: G02, G04, G07, G08, G10, G13, G15, G17, H18, G20, G24, G26, G29).

- Facilidad de uso (4 respuestas: G02, G13, G15, G26).
- Manejo de datos, incluyendo su generación y su exportación (1 respuesta: G26): “Y el permitir exportar las tablas, es un detalle que se agradece”.
- Aspectos de la interfaz de usuario (1 respuesta: G24): “(...) contiene unos manuales de ayuda muy bien explicados con pantallazos sobre las operaciones que podemos realizar con la herramienta, aunque a primera vista su interfaz gráfica no se veía muy atractiva, si nos pareció muy útil la información que nos brinda en cuanto a optimización de nuestros códigos”.

Han sido más numerosas las quejas o sugerencias de mejora (23 grupos, 79’3%). Utilizamos las mismas categorías que en [3] pero variamos las subcategorías donde resulta conveniente. Sólo incluimos las respuestas que pueden ser más explicativas o inesperadas:

- Soporte de configuración de Java (12 respuestas):
  - Dificultad de funcionar con una JDK o de selección del .BIN adecuado (11 respuestas, G05, G07, G10, G11, G13, G15, G17, G22, G24, G25, G26).
  - Opción de configuración de JDK (1 respuesta, G09).
- Datos de entrada (4 respuestas):
  - Selección de juegos de datos en cada ejecución intensiva (G05, G08, G19, G23). La peor experiencia fue la siguiente: “Otro inconveniente que vi fue a la hora de ejecutar las funciones es que no conseguí que el programa ejecutara simultáneamente las funciones con todos los datos del XML, por lo que tuve que ejecutar el programa cien veces seleccionando cada vez los datos de entrada de la tabla.” (G05)
- Soporte de ejecución (4 respuestas):
  - Resultados poco fiables (3 respuestas, G07, G09, G16): “Pero el gran inconveniente apareció una vez usamos ya la versión correcta. Al intentar conseguir los resultados mediante los datos de prueba aportados, podían ocurrir dos cosas sin ningún patrón aparente, que la aplicación no hiciese nada o que mostrase el mismo resultado para los cien casos de los 3 métodos, es decir, el mismo número 300 veces. Indagando y acotando se consiguió determinar que el problema era el algoritmo de vuelta atrás, y para intentar solventar el error se decidió eliminar las colecciones de java (era el único elemento que no tenían las otras dos implementaciones) con lo que ello supuso” (G07); “El programa, si bien cumple su función, tiene algunos fallos que deberían corregirse, como el que se ha reportado por correo electrónico al profesor por el que existen conjuntos de datos que arrojan resultados diferentes si se ejecutan con una u otra máquina virtual” (G09); “Tras la experimentación con OptimEx nos ha surgido un problema con los algoritmos de vuelta atrás y ramificación y poda, ya que no obtenemos los mismos resultados que al ejecutar los algoritmos en NetBeans, obteniendo así unos resultados óptimos para esos algoritmos.” (G16).
  - Ejecución del número de veces deseadas (1 respuesta, G11): “Una vez funcionando, el programa no rendía y no llegaba a hacer las 100

iteraciones, teniendo así que realizar repetidas veces las ejecuciones de los algoritmos".

- Compilador (3 respuestas):
  - "Y como tenía tres clases implementadas para este algoritmo he buscado una opción para agregar clases dentro de otra clase (clases internas o *inner* clases)" (G08).
  - "Al principio no nos compilaba porque teníamos el *package* en la práctica." (G19).
  - "Como conclusión indicar que, la herramienta OptimEx nos ha ocasionado problemas, ya que, al principio no sabíamos que todos los métodos tenían que estar en la misma clase" (G23).
- Preparación de ejecución (3 respuestas):
  - Imposibilidad de seleccionar métodos (2 respuestas, G22, G26): "Al cargar y compilar un archivo .java con OptimEx, creaba automáticamente una línea al principio del código que impedía el funcionamiento correcto del código. La clase se compilaba correctamente, pero no encontraba ningún método dentro de la misma" (G22).
  - Tener que repetir los mismos pasos en cada ejecución (G17): "Una vez hecho funcionar el programa nos encontramos con que hay que repetir una y otra vez los mismos pasos para hacer la misma prueba, seleccionando el tipo de argumentos de los métodos que queremos probar, seleccionando los datos con los que se quieren hacer las pruebas, etc".
- Documentación (3 respuestas):
  - Ampliar la guía de usuario o la ayuda interactiva (2 respuestas, G20, G22).
  - Proporcionar un vídeo (G03): "La práctica parecía fácil en un principio pero me he encontrado varios problemas al empezar a usar el programa OptimEx, creo que ha faltado un video explicativo y algún ejemplo para ver cómo se usaba, me ha costado un día adaptarme a él".
- Presentación de resultados (2 respuestas):
  - Contenido de los datos (G02): "A modo de sugerencia, incluiríamos indicadores referidos a consumos de tiempo de ejecución o espacio en memoria usado para poder soportar con mayor firmeza la elección final de un algoritmo para un problema concreto".
  - Formato de las tablas (G09): "También se ha detectado, aunque no es grave, que al arrastrar una columna en la vista de 'histórico' los colores de las celdas permanecen en la posición antigua, cuando debieran ser movidos junto con la columna".
- Interfaz de usuario (1 respuesta, G23): "También indicar que sería conveniente que la aplicación ofreciese más feedback, porque a la hora de cargar la colección de datos, es necesario seleccionarlos y no se indica nunca".
- Otras (2 respuestas): "En cuanto a las dificultades encontradas para realizar la práctica, la mayor ha sido utilizar la herramienta OptimEx, cosa que no es extraña porque era algo nuevo para mí" (G01); "(...) de igual manera, hay ciertos conjuntos de acciones con los que el programa queda inestable y hay que volver a cargar todo el experimento (...)" (G09).

5 Discusión

Podemos resumir los hallazgos realizados:

- Es prácticamente igual el número de grupos que han realizado la experimentación bien o mal. La frontera entre prácticas bien y mal realizadas es más clara que en la evaluación anterior. Esta mejora de resultados parece deberse a las intervenciones y mejoras realizadas.
- Los resultados erróneos obtenidos en la experimentación se han debido a varias razones: no respetar las condiciones del enunciado, marcar opciones equivocadas sobre la función objetivo o sobre un algoritmo de referencia, y sobre todo por errores en los algoritmos desarrollados o por no interpretar adecuadamente los datos de las tablas. Algunos errores de funcionamiento o fallos de usabilidad de OptimEx también dificultan la realización de la práctica.
- Es frecuente el uso del término “más óptimo”.
- Entre los grupos que hicieron mal la práctica, está tan extendida la falta de percepción de resultados incorrectos como su percepción sin ninguna acción correctora.
- Los grupos han señalado bastantes problemas de manejo de OptimEx, algunos graves.
- Los grupos están satisfechos por la organización de las prácticas a lo largo del curso y por el carácter experimental de esta práctica. También recibimos propuestas para su reorganización.

Todavía podrían introducirse algunas mejoras en la organización actual de prácticas:

- Advertir en el enunciado que los algoritmos desarrollados en algunas prácticas presuponen que las actividades vienen ordenadas pero en esta práctica ya no es así. Por tanto, deben adaptarlos.
- Dejar al menos 2 semanas entre la sesión de laboratorio de las prácticas 4 y 5. De esta forma, podrían realizar y mejorar la práctica 4 antes de realizar la 5.

Otra posibilidad es realizar cambios más drásticos:

- Repartir parte del trabajo de comparación de algoritmos con OptimEx entre las prácticas anteriores. Esto facilitaría el ajuste de los algoritmos y aprender una correcta interpretación del contenido de las tablas.
- Incluir una sesión al comienzo de la asignatura sobre conceptos de optimización. Se incluirían conceptos técnicos (problema de optimización combinatoria, solución óptima frente a optimal, etc.) y también cuestiones lingüísticas (incorrección de “más o menos óptimo”). Debería fomentarse la participación en la sesión o realizar una prueba final.

Una posible reorganización de la asignatura que distribuye las comparaciones entre algoritmos e incluye un capítulo inicial es:

1. Conceptos básicos. Se incluirían conceptos de optimización. También podrían presentarse las dificultades para la resolución eficaz y eficiente de problemas de optimización combinatoria y otros. Esto permitiría motivar los conceptos de algoritmos exactos y aproximados, así como la aleatorización.
2. Algoritmos voraces. Incluye la práctica 1.
3. Algoritmos aproximados. Puede utilizarse GreedEx para comparar varias funciones de selección para el problema de selección de actividades.
4. Vuelta atrás. La práctica podría desglosarse en dos:
   - Práctica 2. Especificación del contraejemplo y desarrollo del algoritmo de vuelta atrás.
   - Práctica 3. Desarrollo del algoritmo de ramificación y poda, codificación de $F\uparrow$ y comparación con OptimEx de estos dos algoritmos y el de vuelta atrás.
5. Eliminación de la recursividad redundante. Incluye una práctica 4 (la 3 actual).
6. Programación dinámica.
   - Práctica 5 (4 actual). Añadir la ordenación en el algoritmo tabulado y compararlo con los 3 algoritmos anteriores.

La actual práctica 5 pierde su sentido y desaparecería. Aunque deja de haber una práctica dedicada específicamente a algoritmos exactos y aproximados, se enfatizan al adelantar su capítulo y al comparar algoritmos en dos prácticas (3 y 5).

6 Conclusiones

Hemos presentado de forma detallada un análisis de la comprensión de conceptos de optimización. La evaluación se realizó con OptimEx en diciembre de 2014. Se ha incluido el procedimiento y el enunciado usados, los resultados detallados y comentados, así como una discusión de los mismos. Los resultados son interesantes por los resultados, dificultades y opiniones de los alumnos. Asimismo, permiten tomar medidas correctoras para paliar en el futuro los resultados negativos.

Agradecimientos. Este trabajo se ha financiado con el proyecto TIN2011-29542-C02-01 del Ministerio de Economía y Competitividad.

Referencias

1. Velázquez Iturbide, J.Á., Martín Torres, R., González Rabanal, N. OptimEx: un sistema para la experimentación con algoritmos de optimización. En: SIIE13 XV International Symposium on Computers in Education – Proceedings, Maria José Marcelino, Maria Cristina Azebedo Gomes y António José Mendes (eds.) (2013) 30-35
2. Velázquez Iturbide, J.Á., Debdi, O., Esteban Sánchez, N., Pizarro, C.: GreedEx: A visualization tool for experimentation and discovery learning of greedy algorithms. IEEE Transactions on Learning Technologies 6, 2 (abril-junio 2013) 130-143, DOI 10.1109/TLT.2013.8

3. Velázquez Iturbide, J.Á. Una evaluación de usabilidad de OptimEx. En: Serie de Informes Técnicos DLSI1-URJC, no. 2014-02, Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos, 2014.
4. Velázquez Iturbide, J.Á. Una evaluación cualitativa de la comprensión de la optimalidad. En: Serie de Informes Técnicos DLSI1-URJC, no. 2014-03, Departamento de Lenguajes y Sistemas Informáticos I, Universidad Rey Juan Carlos, 2014.

Apéndice: Enunciado de la Práctica

Grado en Ingeniería Informática
Asignatura *Algoritmos Avanzados*
Curso 2014/2015
Práctica nº 5

Objetivo
El objetivo de la práctica es que el alumno profundice en su conocimiento de los algoritmos aproximados y su diferencia con técnicas de diseño de algoritmos exactos.

Carácter
La práctica es voluntaria (aunque el alumno debe obtener una calificación de 5 en las prácticas para aprobar). Puede realizarse individualmente o en pareja.

Enunciado
El *problema de selección de actividades ponderadas* ya se planteó en las prácticas 2 y 4. Su objetivo es encontrar un subconjunto de actividades compatibles que proporcionen beneficio máximo.

Por ejemplo, sea el siguiente conjunto de 6 actividades:

i	0	1	2	3	4	5
c_i	0	2	4	1	7	6
f_i	3	7	6	5	9	8
b_i	2	7	4	4	2	2

Una solución óptima es el subconjunto $\{a_1, a_4\}$, con beneficio 9.

El objetivo de la práctica es comparar el rendimiento de algoritmos aproximados con algoritmos exactos, medido como el porcentaje de casos para los que su ejecución da una solución óptima. Deben compararse los resultados de 4 algoritmos, como mínimo, aunque es deseable comparar el mayor número posible de ellos:

- Un algoritmo voraz con orden creciente de duración (D↑) como función de selección. Su código está disponible en un fichero Java anexo.
- Un algoritmo voraz desarrollado en la práctica 1, con orden creciente de fin (F↑) u orden decreciente de inicio (I↓) como función de selección. Se adaptará el algoritmo al problema de selección de actividades ponderadas.
- Uno de los algoritmos de búsqueda desarrollados en la práctica 2 (vuelta atrás o ramificación y poda).
- El algoritmo de programación dinámica desarrollado en la práctica 4.

Solamente deben incluirse algoritmos de las prácticas realizadas por el grupo. Si se desea, también pueden compararse otros algoritmos desarrollados en las prácticas 1 (adaptados) y 2. Todos los algoritmos medidos deben tener la misma signatura:

```
public static int selecActividades (int[] cs, int[] fs, int[] bs)
```

donde *cs*, *fs* y *bs* son los vectores que contienen los instantes de comienzo, los instantes de fin y los beneficios, respectivamente.

Idealmente, puede usarse OptimEx de la siguiente forma: se carga una clase con los algoritmos a comparar, se selecciona su signatura, se realiza una ejecución intensiva y se comparan los resultados obtenidos en la tabla resumida. El problema es que los instantes de fin deben ser superiores a los de comienzo, por lo que unos datos generados aleatoriamente no cumplirán esta restricción.

La práctica consiste en una experimentación modesta: realizar una ejecución intensiva en OptimEx con los 100 casos de prueba del fichero anexo. Se documentarán los resultados obtenidos siguiendo el modelo de informe indicado a continuación.

La guía de uso de OptimEx identifica diversos resultados que son contradictorios con los conceptos presentados sobre las distintas técnicas de diseño. Conviene leerla atentamente para, en su caso, identificar y corregir estas situaciones. Si es necesario, hay que corregir los algoritmos realizados en prácticas anteriores que presenten un rendimiento inaceptable respecto a la optimalidad.

Entrega

El equipo debe entregar un informe elaborado siguiendo el índice detallado a continuación. El informe debe enviarse por medio del apartado de Evaluación del campus virtual. Si la práctica se ha realizado en pareja, sólo debe hacerse un envío con el nombre de los dos alumnos escrito en el informe. Si se tienen dificultades, puede enviarse por el correo del campus virtual con el asunto “Práctica 5”. El plazo de entrega es el jueves 4 de diciembre de 2014, incluido.

Informe

El equipo debe entregar un informe con la siguiente estructura:

1. **Experimento.** Se identificarán los algoritmos medidos y se incluirá, para cada algoritmo, la siguiente información:
    a. Técnica de diseño con la que ha desarrollado. Si es necesario, también debe incluirse algún elemento diferenciador (p.ej. función de selección para un algoritmo voraz o definición de cota para un algoritmo de ramificación y poda).
    b. Indicar si se ha necesitado modificar el algoritmo durante esta práctica, explicando por qué se realizó dicha modificación y en qué consistió.
    c. Código del algoritmo.
2. **Resultados.** Se identificarán los algoritmos que son óptimos según los resultados de la experimentación. La propuesta irá respaldada con dos tablas obtenidas en OptimEx: la tabla histórica y la tabla de resumen. Como orientación, el algoritmo

voraz D↑ debe calcular la solución óptima en el 18% de los casos de prueba (y, obviamente, soluciones subóptimas en el restante 82% de los casos).

3. **Conclusiones.** Se explican las conclusiones obtenidas tras realizar la práctica. Estas conclusiones pueden consistir en una valoración del proceso de experimentación o de OptimEx o cualquier otro comentario sobre la práctica. Por ejemplo, pueden describirse las incidencias que han dificultado la realización de la práctica, sus aspectos más atractivos o más difíciles, sugerencias sobre cómo mejorar la práctica, etc.

Evaluación

Se evaluará la calidad del trabajo realizado y la claridad del informe.