J. Ángel Velázquez Iturbide

Una Evaluación Cualitativa de la Comprensión de la Optimalidad

Número 2014-03

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

Índice

1 Introducción 1
2 Protocolo 2
3 Análisis 3
4 Resultados 4
4.1 Prácticas Correctas e Incorrectas 101
4.2 Dificultades y Malentendidos 112
4.3 Actitudes 123
5 Discusión 134
6 Conclusiones 134
Agradecimientos. 145
Referencias 145
Apéndice A: Enunciado y Modelo de Informe de la Práctica 156

Una Evaluación Cualitativa de la Comprensión de la Optimalidad

J. Ángel Velázquez Iturbide

Departamento 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 diciembre de 2013. 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 hallazgos de errores, dificultades, malentendidos y actitudes de los alumnos. Asimismo, permiten tomar medidas correctoras para paliar en el futuro los resultados negativos.

Palabras clave: Algoritmos de optimización, experimentación interactiva, malentendidos, 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]. Asimismo conviene usar ambos sistema de forma combinada, primero GreedEx omo introducción a los algoritmos voraces y otras técnicas de diseño de algoritmos de optimización, después OptimEx como sistema para la experimentación con algoritmo aproximados y quizá técnicas de diseño exactas.

En este informe se presenta una evaluación cualitativa de los informes presentados por alumnos tras realizar una práctica con Optimex. 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 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: Se da el problema de selección de actividades y pseudocódigo para su resolución voraz, y 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 pedía completar el algoritmo y ampliarlo para que no independa 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 de poda.
3. Eliminación de recursividad múltiple redundante. Se da un algoritmo 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.

La evaluación se realizó a partir de la quinta práctica, en diciembre de 2013. Su objetivo era que los alumnos experimentaran con algoritmos exactos y aproximados. Participaron alumnos de dos grupos, presencial y online, más dos alumnos que asistían "de oyente".

La sesión de prácticas duraba 2 horas. Se permitió realizar la práctica tanto individualmente como en parejas. Debían entregar el informe sobre la práctica en el plazo de una semana.

Los alumnos se descargaron del campus virtual (con el que ya estaban familiarizados) el material necesario para realizarla: enunciado, clase Java con un algoritmo y fichero con datos de entrada (más un cuestionario de usabilidad). También tenían disponible OptimEx y una pequeña guía de usuario.

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 exigía que midieran la optimalidad de al menos 3 algoritmos. Se les proporcionaba un algoritmo voraz basado en orden creciente de duración (representado F↑); los otros dos debían tomarlos los propios alumnos de sus prácticas 1 y 2 (y 4 si querían). 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 debía incluir:

1. Identificación de los algoritmos utilizados, junto con su código.
2. Resultados en forma de tabla de la experimentación.

3. Conclusiones.

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 fuera de la sesión, suministrándose una nueva versión a los alumnos.

Se recogieron un total de 32 informes: 24 del grupo presencial (14 individuales y 10 parejas), 5 del grupo on-line (4 individuales y 1 pareja) y 1 de alumnos de asistencia informal (1 pareja).

Incluimos el enunciado de la práctica en el Apéndice.

3 Análisis

Los datos se analizaron utilizaron los principios de la teoría fundamentada [3]. Se construyó una tabla donde aparecieran los elementos que se consideraron importantes para este análisis. Se realizaron tres rondas de análisis:

1. Ronda primera (exploratoria). Se analizaron solamente los cinco informes del grupo on-line para identificar los elementos destacados de cada práctica. Se fue construyendo una tabla donde se iban registrando los elementos más destacados.
2. Segunda ronda. A partir de los elementos identificados en la ronda preliminar se diseñó una tabla con más precisión. Aunque el análisis procedió de forma básicamente secuencial, de vez en cuando se retrocedía a informes anteriores para comprobar que un elemento nuevo o importante se había tenido en cuenta, o para consultar en qué columna de la tabla se había anotado algún elemento.
3. Tercera ronda. A partir de la información contenida en la tabla, se elaboró una lista de preguntas que podían tener respuesta. También se reestructuró la tabla. Por último, se hizo una última ronda de análisis. Al igual que la segunda, fue secuencial pero con frecuentes retrocesos.

Un detalle menor que puede utilizarse para el análisis proviene de que los alumnos debían usar 100 juegos de datos fijos y el algoritmo voraz D↑. Por tanto, los resultados para este algoritmo pueden servir como control de los resultados:

Tabla 1. Análisis de los informes de los alumnos

Medida	**D↑**
Núm. ejecuciones	100
% subóptimas	82,00 %
% óptimas	18,00 %
% superóptimas	0,00 %
% desviación media	22,79 %
% desviación máxima superóptima	0,00 %
% desviación máxima subóptima	71,88 %

4 Resultados

Presentamos los resultados del cuestionario en la Tabla 2. 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".
- "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 destacadas son: si los datos fueron generados aleatoriamente (es así cuando se han usado más de 100 juegos de datos), si la experimentación se realizó marcando algún algoritmo como óptimo, si se realizó como si fuera un problema de minimización.
- "Resultados". Se indica si el grupo ha aportado, como evidencia, alguna tabla aparte de la de resumen y qué algoritmos han resultado óptimos, subóptimos o superóptimos. También se señala cuando obtiene los resultados esperados para D↑ (véase apartado anterior) y cuando el grupo ha dedicado atención a la desviación respecto a los algoritmos óptimos.
- "Propuesta". Si los alumnos han propuesto explícitamente algunos algoritmos como óptimos.
- "Dificultades y malentendidos experimentación". Explicación mía de los problemas o malentendidos que parecen tener.
- "¿Bien?". Si la propuesta del grupo ha identificado la optimalidad de alguno de los algoritmos aportados, obteniendo datos coherentes con lo esperado. Utilizamos los símbolos ✓ para indicar que identifica todos los algoritmos con los que experimenta que debieran ser óptimos y ½ para indicar que identifica a uno pero no a otro.
- "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 2. Análisis de los informes de los alumnos

Grupo	Algoritmos usados	Condiciones experimento	Resultados	Propuesta	Dificultades y malentendidos experimentación	¿Bien?	Comentarios abiertos	Otros
1. I	D↑ F↑ PD	100 datos (generados) PD marcado	Voraces superóptimos	PD	- Tras marcar PD, no ve problema en los resultados superóptimos		- Sugiere pedir complejidad para mejor valorar qué algoritmo usar	
2. I	D↑ F↑ RyP (Back)	100	Back óptimo Voraces subóptimos Control D↑ bien			✓	– Problemas JDK	Sin comentar
3. I	D↑ F↑ RyP	100 Minimizan	Tabla histórica 1. Todas subóptimas 2. RyP	RyP	- Tras ver los resultados, ha corregido para maximizar	✓	- Problemas JDK - OptimEx fácil de usar y útil - Valora evolución de técnicas en la asignatura (por complejidad y exactitud)	
4. P	F↑ RyP D↑	100	Todos subóptimos (D↑ en 100% casos)				- Problemas JDK - OptimEx fácil uso - Utilidad del manual - Satisfacción comparación final de prácticas	- Back es RyP - Calcula cota no incrementalmente
5. I	D↑ F↑ Back	100	Todos subóptimos (Back óptimo en 94%)		- Identifica "óptimo" con el mejor en un mayor % de casos - Considera que Back es peor respecto al criterio del % de casos subóptimos		- Problemas JDK - Sorpresa clase Java sin *main* - Icono ejecución intensiva - Satisfacción final prácticas mismo algoritmo	
6. I	Back D↑ F↑	1000	Tabla histórica Todos subóptimos (Back > D↑ > F↑)	Back	- Identifica "óptimo" con el mejor en un mayor % de casos		- Problemas JDK - Resto de práctica, sin problemas	
7. P	Back F↑ D↑	200	Todos subóptimos (Back > F↑ > D↑)				- Problemas JDK - OptimEx fácil uso y útil	

2

8. I	C↓ RyP D↑	100	RyP óptimo Control D↑ bien	RyP	- Parece que asimila "subóptimo" con "menor" en todos los casos, no "menor o igual"	✓	- Problemas JDK - Utilidad OptimEx	
9. P	F↑ Back D↑	100	Back óptimo Control D↑ bien		- Comenta quien da más soluciones óptimas y subóptimas, como si importaran lo mismo - Contradicción entre resultados (100% óptimo el algoritmo de vuelta atrás) y conclusiones (ninguno es óptimo)		- Problemas JDK - Utilidad OptimEx	
10. P	C↓ D↑ Back	107	- Parte de tabla histórica - Todos subóptimos (Back > D↑ > F↑) - Control D↑ casi bien (supongo que en los 100 casos primeros) - Errores de ejec. por los 7 casos extra		- Atribuye los errores a su algoritmo de Back, sin más		- Problemas JDK - OptimEx fácil de usar y útil - Problemas para ejecutar los 100 casos ¿? - Conciencia quizá prácticas anteriores mal - Práctica fácil	No incluyen el código de D↑ (lo daba yo)
11. P	F↑ Back, D↑	100	- Back óptimo - Control D↑ bien	Back	- Errata: en la sección de D↑, dice que es de PD	✓	- Problemas JDK - OptimEx fácil de usar y útil - Práctica fácil	
12. I	D↑ F↑ PD	100 + 500	- Dos tablas de resumen - Voraz D↑ óptimo - PD superóptimo y subóptimo - Voraz F↑ superópt. 2º ejec - PD se desvía menos que F↑	D↑	- Toma como referencia D↑ - Considera "muy negativo" los resultados mucho mejores de PD - Ha entendido mal el objetivo o el proceso del experimento		- OptimEx fácil de usar y útil	
13. I	D↑ F↑ RyP	100	- RyP óptimo - Control D↑ bien	RyP		✓	- OptimEx útil	

14. P	F↑ Back D↑	100	- Back óptimo - Voraces superóptimos - Control D↑ porcentajes bien, muestra que Back calcula valores bajos	Back	- Confusión entre casos óptimos y superóptimos		- Problemas JDK - OptimEx útil	
15. P	F↑ Back D↑	100	- Tabla histórica - Back óptimo - D↑ > F↑ - Control D↑ bien		- Parece que identifica "óptimo" con mayor % de casos	✓	- Problemas JDK - OptimEx útil - Práctica fácil - Satisfacción comparación final de prácticas	No incluyen el código de D↑ (lo daba yo)
16. P	RyP C↓ D↑	100 Back marcado	- Back óptimo - Control D↑ mal, luego Back debe tener errores - Compara desviaciones	Back		✓	- OptimEx fácil de usar y útil - Sugerencias generación de datos aleatorios y elección max/min	Back es RyP
17. I	D↑ F↑ Back RyP		- No pudo ejecutar OptimEx				- Identifica objetivo de práctica	
18. I	F↑ RyP D↑	200	- Tablas de resultados e histórica (sólo las 38 ejecuciones primeras) - Todos subóptimos (D↑ > F↑ > RyP)		- No se extraña de los resultados subóptimos de RyP		- Problemas JDK - OptimEx fácil de usar y útil	Sin comentar
19. I	F↑ Back D↑	100 Back marcado	- D↑ superóptimo - Compara desviaciones	Back	- Achaca los resultados de D↑ a que está mal (no a Back)		- Adecuada comparación final de prácticas	

20. P	C↓ RyP D↑	100	- Tablas de resultados e histórica - Todos subóptimos (D↑ > RyP > F↑) - Incluye tablas histórica y de resultados				- Problemas JDK - OptimEx útil - Satisfacción comparación final de prácticas	
21. P	F↑ Back D↑	100 Minimizan	- Todos subóptimos - Tablas de resultados e histórica (F↑ la mejor)		- No se da cuenta de que minimiza		- Problemas JDK - OptimEx útil - Práctica fácil	
22. I	D↑ C↓ F↑ Back RyP	100	- Back óptimo - Control D↑ mal, luego Back debe tener errores (junto con RyP)	Back	- Comenta lo raro de los resultados de RyP	½	- OptimEx útil - Satisfacción comparación final de prácticas	
23. I	Back D↑ PD	100	- Tablas de resultados e histórica - Todos subóptimos (D↑ > F↑ > Back)		- Aunque espera que Back sea el óptimo, asume los resultados - No espera que PD sea óptimo		- OptimEx útil - Comenta la falta de tiempo para corregir prácticas pasadas (por eso eligió PD en vez de F↑)	
24. P	F↑ RyP D↑ PD	100	- Tablas de resultados e histórica - Control D↑ bien - D↑ > F↑ (su algoritmo F↑, quizá mal)	Back, PD	- Avisa de que está bien que no haya resultados superóptimos	✓	- Problemas JDK e inestabilidad OptimEx - Satisfacción comparación final de prácticas - Dificultad de la práctica por depender de que otras estén bien	No incluyen el código de D↑ (lo daba yo)
25. P	F↑ RyP D↑	100	- RyP subóptimo en 100%	RyP	- Comentan que han modificado RyP para que sea óptimo 100%. ¿Error en la entrega? - Comentan que habría estado bien comparar con PD: nada se lo impedía		- OptimEx útil - Problemas JDK - Satisfacción comparación final de prácticas	

26. I	D↑ F↑ Back		- No pudo ejecutar OptimEx						
27. I	F↑ Back PD D↑	100	- Tabla histórica (sólo las 35 ejecuciones primeras) - Todos subóptimos (D↑ > Back, PD > F↑)		- Concluye que los algoritmos aproximados son mejores que los exactos				
28. I	Back PD D↑	100	- Tabla histórica	Back	- Ha incluido el código de Back en lugar de PD	½	- OptimEx fácil de usar - Poca fiabilidad de los resultados de OptimEx		
29. I	F↑ RyP D↑	100	- Todos subóptimos (F↑, D↑ > RyP) - Hace 4 pruebas, sin seleccionar óptimo y seleccionando cada algoritmo				- Dificultad en conocer el proceso de experimentación		
30. P	F↑ RyP D↑	100	- Todos subóptimos		- Los ordena (por %) "de más a menos óptimo" - Han modificado RyP para subir su %		- Problemas JDK - Práctica sencilla		

Cuando queramos referirnos a algún grupo utilizaremos la notación G*nn*, donde *nn* es su número identificativo. Hay que notar que dos individuos (G17, G26) no pudieron ejecutar OptimEx, por lo que limitamos nuestro análisis a los 28 grupos restantes.

4.1 Prácticas Correctas e Incorrectas

Veamos primero los resultados globales.

Tabla 3. Resultados de la experimentación

Resultado	#(%) grupos	Grupos
Bien	8 (28'6%)	G2, G3, G8, G11, G13, G15, G16, G24
A medias	2 (7'1%)	G22, G28
Mal	18 (64'3%)	G1, G4, G5, G6, G7, G9, G10, G12, G14, G18, G19, G20, G21, G23, G25, G27, G29, G30

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

- G03 realizó la experimentación sin darse cuenta de marcar la casilla de maximizar. Al comprobar que los resultaos eran inversos, se dio cuenta del error y lo corrigió.
- G08 y G15 parecen tener malentendidos sobre el concepto de optimalidad.
- G16 y G22 obtienen resultados distintos de los esperados para el algoritmo de control D↑. Probablemente se debe a que el algoritmo óptimo no es correcto del todo.

Los dos grupos de la categoría "a medias" (G22, G28) incluyen dos algoritmos de optimización exacta, pero sólo da resultados óptimos e de vuelta atrás. Los algoritmos subóptimos aportados son de ramificación y poda (G22) y programación dinámica (G28). G22 comentó que no debiera ser así.

Veamos ahora varios factores que confluyen en los grupos que han realizado mal la experimentación:

- Los 6 grupos que no respetaron el uso de los datos de entrada (G01, G06, G07, G10, G12, G18) la realizaron mal. Un grupo (G10) tiene casi bien los resultados para el algoritmo de control D↑. Probablemente obtuvo los resultados esperados para los 100 datos dados. El grupo informa de problemas con la carga de datos, que pudo estar en la base de generar 7 juegos más. Algunos de estos datos produjeron errores en tiempo de ejecución.
- Un grupo (G21) no marcó la casilla de maximizar pero no se dio cuenta de lo sorprendentes de los resultados.

Veamos la razón de haber realizado mal el experimento:

- Un grupo (G09) realizó bien la experimentación pero no identificó como óptimo a ningún algoritmo, ni siquiera al mejor en el 100% de los casos. Claramente, es un malentendido conceptual.

- 4 grupos marcan un algoritmo que no es óptimo como tal (G01, G12, G14, G19), obteniendo datos mayores ("superóptimos") con otros algoritmos. No interpretan que estos resultados no son admisibles o no saben cómo corregirlos. En 3 casos (G01, G14, G19) se ha marcado como óptimo un algoritmo exacto y resulta mejor algún algoritmo voraz. En el caso restante (G12), se parte de seleccionar a un algoritmo voraz como óptimo.
- 13 grupos han realizado una experimentación abierta en la que ningún algoritmo ha arrojado unos resultados óptimos (G04, G05, G06, G07, G10, G18, G20, G21, G23, G25, G27, G29, G30). Ningún grupo revisó porqué los algoritmos exactos no eran óptimos. En algunos casos, la evidencia era aún mayor: un algoritmo es subóptimo siempre (G04, G25), un algoritmo exacto se queda cerca de un 100% de casos mejores (G05), un algoritmo exacto obtenga peores resultados que uno o varios voraces (G10, G20, G24).

4.2 Dificultades y Malentendidos

El malentendido principal es el propio concepto de "óptimo". Esta falta de comprensión e manifiesta de dos formas principales:

- Que un algoritmo obtenga resultados mejores que un algoritmo "óptimo" para el mismo problema.
- Que ningún algoritmo obtenga resultados mejores en el 100% de los casos, cuando durante la asignatura se han identificado los algoritmos exactos y aproximados.

La manifestación primera no suele ser comentada, pero sí la segunda:

- (G05) "Según se puede ver en la tabla, el algoritmo recursivo (con poda) es un 94% óptimo en comparación con los otros dos algoritmos, sin embargo en el porcentaje de subóptimas sale perdiendo."
- (G06) "Muestra además los porcentajes de medidas, por ejemplo, el mayor porcentaje de óptima es, como se vio en la tabla anterior, para ‘actividades’, porque de las 1000 ejecuciones de estos algoritmos, fue óptima el 80,90 % de las veces, seguida de lejos por ‘selecActividades3’ con un 22,00%."
- (G30) "(...) esto ha hecho que suba el porcentaje de soluciones óptimas de esta función."

Este último comentario sugiere que quizá haya que extender el vocabulario de forma que se pueda distinguir entre lo óptimo en general y lo mejor en un caso concreto. Esta intuición se repite en otro grupo: "Como podemos observar el algoritmo que más soluciones óptimas obtiene es el segundo que se corresponde con el algoritmo de backtracking. Sin embargo el algoritmo que más soluciones sub-óptimas tiene es el tercero que es el algoritmo de selección de actividades cuando las actividades están ordenadas en orden creciente de duración" (G09). En este caso, se dio la circunstancia de que habiendo un 100% de mejor resultado para un algoritmo de vuelta atrás, no seleccionaron ninguno: "Podemos comprobar que no hay un

algoritmo que sea mejor que otro para todos los casos si no que estamos hablando de más eficiente en tantos casos o en menos".

Incluso grupos que han realizado bien su práctica utilizan el idioma de forma sospechosa, en cuanto a lo asimilado de su comprensión del concepto de óptimo: "OptimEx nos ha parecido un programa muy útil a la hora de saber, cuál de los diferentes algoritmos puede ser el más óptimo, (...)" (G15).

Las palabras parecen cambiar de significado para el siguiente grupo: "Y vemos que solo el 22% y 18 % son las desviaciones con respecto al algoritmo optimo" (G14). Estos porcentajes corresponden a los casos mejores de dos algoritmos, y los comparan con resultados superóptimos.

Finalmente veamos algunos comentarios adicionales que, de forma distinta, parecen reproducir esta falta de asimilación:

- (G08) "... aunque también tenemos que comentar que en algunos ejemplos, los resultados que se obtienen son equivalentes a los óptimos, porque coincide que el resultado óptimo se consigue". Parece como si se esperara que un algoritmo subóptimo siempre tuviera que calcular un valor menor, no igual.
- No podemos descartar que en algún caso no se hay comprendido el objetivo de la práctica o el experimento en sí. El grupo G12 tomó como referencia el algoritmo D↑ para después afirmar: "Por su parte, vemos cómo en el tercero de los algoritmos, de programación dinámica, se dispara el porcentaje de soluciones superóptimas. Esto quiere decir que son soluciones que sobrepasan el beneficio máximo de las actividades compatibles, y lo hace casi en un 86% de las ocasiones, un dato muy negativo".

4.3 Actitudes

Repasemos las actitudes de los grupos. No ha sido frecuente encontrar grupos que han advertido problemas. Entre estos, sólo estamos seguros de un grupo que haya corregido los problemas. En efecto, G03 advirtió que OptimEx estaba minimizando y cambió el análisis para maximización.

Otros dos grupos informan de sus mejoras pero no hay constancia de ellas en sus informes:

- (G25) "El mayor problema encontrado fue quizás con el algoritmo de backtracking, pues en un primer instante no devolvía un 100% de éxito en cuanto a la devolución de resultados, y hemos tenido que modificarlo con el fin de conseguir que de lo que tiene que dar." Sin embargo, en su tabla de resumen, el algoritmo de vuelta atrás da un 100% de resultados subóptimos.
- (G30) "Hemos agregado una condición en el algoritmo de ramificación y poda para el cálculo de la cota y esto ha hecho que suba el porcentaje de soluciones óptimas de esta función". Sin embargo, sólo calcula el resultado máximo en el 37% de los casos, inferior a los resultados de D↑.

Otros 4 grupos (G10, G19, G22, G23) detectan resultados erróneos pero los atribuyen a un algoritmo suyo (G10, G22), al del profesor (G19) o simplemente cambia lo aprendido en clase para que se adapte al nuevo hallazgo: "El algoritmo que

encuentra menos veces la solución óptima es el de Búsqueda (Backtracking), para mi sorpresa, ya que teóricamente pensaba que sería el mejor de todos. Esto demuestra que para hallar una solución óptima no es recomendable un algoritmo por fuerza bruta ya que 'pensando un poco' podemos encontrar algoritmos rápidos e inteligentes" (G23).

5 Discusión

Podemos resumir los hallazgos realizados:

- Son mayoría los grupos que han realizado mal la experimentación. Sin embargo, la frontera entre prácticas bien y mal realizadas es difusa. Algunos grupos que la hicieron bien utilizan un lenguaje equívoco, otros grupos han encontrado un algoritmo óptimo pero no otro, y algunos grupos cuyos resultados son erróneos tienen soluciones cercanas a la correcta.
- Algunos grupos realizaron mal la experimentación por no respetar las condiciones del enunciado.
- La mayor parte de las prácticas mal realizadas lo han sido por no interpretar los resultados mostrados en las tablas o por partir de algoritmos subóptimos.
- Hay un uso difuso de los términos, que conviene fijar.
- Hay una falta bastante común de percepción de resultados incorrectos. Incluso cuando se detectan, no se corrigen.

Una forma de intentar remediar estos resultados consiste en no dejar el enunciado tan abierto, sino dejar más claro lo que deben documentar en el informe:

- Junto con el algoritmo D↑, podrían darse sus (tablas histórica y resumen) para que les sirva de control de sus resultados.
- Deben identificarse los algoritmos óptimos. e incluir las tablas histórica y de resumen.
- Si hay errores de ejecución o resultados inaceptables (resultados superóptimos o todos subóptimos), deben determinar su origen y corregir los algoritmo o el proceso experimental.

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 2013. 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 hallazgos de errores, dificultades, malentendidos y actitudes 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. Glaser, B., Strauss, A. The Discovery of Grounded Theory: Strategies for Qualitative Research. 1967. Chicago, IL, USA, Aldine

Apéndice A: Enunciado y Modelo de Informe de la Práctica

Grado en Ingeniería Informática
Asignatura *Algoritmos Avanzados*
Curso 2013/2014
Práctica nº 5

Objetivo
El objetivo de la práctica es que el alumno profundice en su conocimiento de los algoritmos aproximados.

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* se planteó en la práctica 2. Su objetivo consiste 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

La 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. Como mínimo, deben compararse los resultados de 3 algoritmos para este problema:
- Un algoritmo voraz con orden creciente de duración (D↑) como función de selección.
- Un algoritmo voraz con orden creciente de fin (F↑) u orden decreciente de inicio (I↓) como función de selección (parecido al desarrollado en la práctica 1),
- Uno de los algoritmos de búsqueda desarrollados para este problema en la práctica 2.

El código del primer algoritmo está disponible en un fichero Java anexo. Para los otros dos algoritmos, puede usarse el código desarrollado por los alumnos en las prácticas 1 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 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 juegos de datos del fichero anexo. Se documentarán los resultados obtenidos siguiendo el modelo de informe indicado a continuación.

Como actividad voluntaria, puede rellenarse un cuestionario de usabilidad de OptimEx. La realización de este cuestionario se puntuará, según las normas de la asignatura, como una actividad voluntaria puntuable con 0'25 puntos sobre la nota final (sólo si la asignatura está aprobada).

Entrega
El alumno 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". En caso de entregar el cuestionario de usabilidad de OptimEx, puede comprimirse con el informe para su envío conjunto. El plazo de entrega es el jueves 12 de diciembre de 2013, incluido.

Informe
El alumno debe entregar un informe con la siguiente estructura:
1. **Algoritmos medidos.** Se explicarán brevemente las características de los algoritmos medidos por el alumno y se incluirá su código.
2. **Resultados.** Se incluirá una tabla de resumen con los resultados obtenidos, donde se refleje el total de ejecuciones realizadas y las estadísticas calculadas por OptimEx.
3. **Conclusiones.** Se explican las conclusiones obtenidas tras realizar la práctica. Por ejemplo, estas conclusiones pueden consistir en una valoración del proceso de experimentación o de OptimEx. También pueden incluirse otros comentarios cualesquiera sobre la práctica, p.ej. 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, así como la claridad del informe.