Formación Universitaria
E-ISSN: 0718-5006
citrevistas@gmail.com
Centro de Información Tecnológica
Chile

Sáez, Pablo D.; Monsalve, César E.
Aprendizaje Basado en Resolución de Problemas en Ingeniería Informática
Formación Universitaria, vol. 1, núm. 2, 2008, pp. 3-8
Centro de Información Tecnológica
La Serena, Chile

Formación Universitaria
Vol. 1(2), 3-8 (2008)
doi: 10.4067/S0718-50062008000200002

Aprendizaje Basado en Resolución de Problemas en Ingeniería Informática

Pablo D. Sáez[1] y César E. Monsalve[2]
(1) Calle Nueva 3, Población Versalles, San Pedro de la Paz, Concepción-Chile
(e-mail: pablosaezphd@gmail.com)
(2) Universidad de Concepción, Departamento de Ingeniería Informática, Casilla 160-C, Correo 3, Concepción-Chile (e-mail: cemonsal@udec.cl)

Resumen

Se analiza y aplica el sistema de aprendizaje basado en resolución de problemas, que es la base de los programas de estudio de varias universidades en distintos países. La idea es enfrentar al alumno a problemas concretos en una asignatura, en formato de "proyecto", dejando la "asimilación de información" a cargo del propio alumno con uso de libros. Se comentan algunas posibilidades de implementar este sistema sin pasar por un esquema tan radical como el planteado en la propuesta original, que podría fallar en las culturas latinoamericanas. Como ejemplo de la aplicación de estas ideas se presenta el desarrollo de un solucionador de Sudoku para estudiar los conceptos relacionados con la teoría de la NP-completitud en la ciencia de la computación. Se concluye que el proceso de enseñanza-aprendizaje es más eficiente en este tipo de enfoques que usando sistemas más tradicionales.

Palabras clave: solución de problemas, enseñanza-aprendizaje, NP-completitud, ciencias de la computación

Problem-Solving Based Learning in Computer Science Engineering

Abstract

The pedagogical method known as Problem-solving Based Learning (PBL), which is the basis for the curricula of a number of universities in other countries, is analyzed and applied to a particular situation. The idea is to face the student to a set of well defined problems during a course, on a project-based basis, letting the task of "information assimilation" to the student by means of books. Some guidelines that may help in the implementation of these pedagogical methods without the need of going through such a radical system as that originally proposed, and that could fail in Latin-American cultures, are outlined. An example of the application of these ideas the development of an automated Sudoku-solver to study the NP-completeness theory in computer science is presented. It is concluded that the teaching-learning process is more efficient using these new approaches than the traditional ones.

Keywords: problem-solving, teaching-learning, NP-completeness, computer science

INTRODUCCION

Se expone en este artículo brevemente el método de aprendizaje basado en resolución de problemas (Mills y Treagust, 2003; Prince y Felder, 2006), que consiste esencialmente en que el alumno se ve enfrentado a problemas concretos, prácticos, en el transcurso de las asignaturas, asumiendo el profesor un rol de asesor, o de guía más que el de "transmisor de información". La idea es un poco la del proverbio chino que dice "lo que escucho, lo olvido; lo que veo, lo recuerdo; pero lo que hago, lo aprendo".

La solución en su forma comúnmente adoptada puede resultar un tanto radical, dado que el alumno debe por cuenta propia leerse los contenidos en los libros correspondientes, consultando al profesor las dudas. Este enfoque en culturas como las nuestras puede tender a desvirtuarse, en la medida en que se presupone que el alumno va a, disciplinadamente, ocupar las tardes que tiene asignadas para estudiar en eso (estudiar).

Se propone en este artículo buscar un enfoque intermedio, que combine las ventajas de enfrentar al alumno a problemas concretos en el transcurso de la asignatura, sin que necesariamente disponga de una total libertad en el uso del tiempo. Mostramos un ejemplo concreto de estas ideas: la construcción por parte de los alumnos de un resolvedor automático de sudokus como ejemplo práctico en el estudio de los temas de complejidad computacional y NP completitud (Papadimitriou, 1994) en la ciencia de la computación. Para una aproximación ya no didáctica sino técnica de este problema se puede consultar en Lambert et al., (2006).

La teoría de la NP-completitud forma parte en la actualidad de los currículums de la carrera de ingeniería informática en muchas universidades. Tiene su origen en el trabajo de Cook (1971) y los aportes posteriores de Karp (1972) y otros autores (Adleman y Manders 1977; Levin, 1973). El estudio de las reducciones de Karp planteado en forma estrictamente teórica resulta normalmente árido y difícil de entender para los estudiantes. Por lo tanto una forma de lograr que ellos trabajen en forma práctica en estos temas consiste en interesarlos en el problema de satisfacibilidad de fórmulas proposicionales (Cook, 1971), más aun considerando la existencia de herramientas computacionales ad-hoc, los llamados SAT-solvers (Bordeaux et al., 2006), que han sido extensamente estudiadas (Bordeaux et al., 2006; Moskewicz et al., 2001; Simon et al., 2005).

APRENDIZAJE BASADO EN RESOLUCION DE PROBLEMAS

Se sabe en el área de la biología que algunas especies de hormigas usan una técnica conocida como "andar en tándem" para mostrarle a otra hormiga cómo ir desde el nido a una fuente de comida (Franks y Richardson, 2006). La hormiga "profesora" guía a la "alumna", llevándola y corrigiéndole el rumbo. En contraste, es conocido el comportamiento de la abeja, que cuando sabe de la ubicación de algún alimento llega a la colmena y realiza una danza a la cual "asisten" las otras abejas, pasivamente. Hay en este caso tan sólo una "transmisión de información". Y la pregunta que surge entonces para el docente es: ¿estoy siendo "abeja" u "hormiga"?, es decir: se limita el docente a realizar una "danza" adelante, en el pizarrón, pretendiendo "transmitir información" a los alumnos, o está realmente involucrado en su proceso de aprendizaje y trata de guiarlos en su actuar, corrigiéndolos y orientándolos?

No es difícil darse cuenta de que en la academia la gran mayoría de los profesores somos "abejas", sumergidos en un afán por "transmitir conocimiento", considerado éste como un cúmulo de datos. Sin embargo al momento de enseñar ciencia, o ingeniería, lo que debería propiciarse es más bien el razonamiento autónomo en los alumnos, la habilidad de reflexionar en forma crítica y el autoaprendizaje; pero la pregunta evidente es: ¿cómo?

Existen distintos modelos educacionales: aquellos basados en escuchar y repetir, aquellos basados en observar y reproducir, y aquellos basados en percibir y comprender. El modelo que se muestra en este artículo, brevemente, por las limitaciones del espacio, es el modelo PBL (Mills y Treagust, 2003; Prince y Felder, 2006), por su sigla en inglés: Problem-solving Based Learning (aprendizaje basado en resolución de problemas), en donde la idea central es enfrentar a los

alumnos a problemas concretos, prácticos, en el transcurso de un ramo. El ramo pasa a ser una especie de proyecto durante el cual el profesor hace de guía para el equipo de trabajo, ayudándolos u orientándolos. Desde luego hay un proceso de estudio, de "asimilación de contenidos", pero de esta parte del proceso de aprendizaje es responsable el propio alumno, quien debe leerse los capítulos correspondientes del libro correspondiente en el transcurso del ramo, estando el profesor evidentemente encargado de aclarar las dudas que se le presentan al alumno en su lectura del texto; pero ya no encargado de "danzar" en el pizarrón, transmitiendo información. Porque no es difícil darse cuenta de que si el alumno pierde el hilo de esta "transmisión de información" van a ser vanos los esfuerzos por pretender "transmitirle más información" a un receptor que ya no está "procesando". En contraste con esta situación, reuniones de trabajo o conversaciones grupales siempre son más eficientes al momento de entender conceptos o relacionarlos.

En general se pueden clasificar los distintos estilos de enseñanza en cuatro categorías:

1) Sólo exposición, sin ejercitación (estilo "decimonónico").
2) Exposición de contenidos como hilo conductor de la asignatura, con ejercitación (este estilo es el que podríamos considerar usual).
3) Ejercicios o proyectos como hilo conductor de la asignatura, con exposición de contenidos (esto correspondería a PBL).
4) Sólo ejercicios o proyectos (éste sería el caso del aprendizaje en un taller por ejemplo).

Desde luego los modelos de aprendizaje tipo PBL no son bienvenidos por todos los alumnos. Estadísticamente un 25% de los alumnos aproximadamente prefiere en realidad el sistema tradicional, de "entrega de información". Un 25% prefiere PBL y un 50% dice que puede aprender indistintamente con uno u otro método (Mauffette, 2007). Hay por lo tanto un grado de resistencia al cambio al momento de introducir este tipo de metodologías, lo cual es natural por lo demás, esperable. La distribución semanal de las actividades por ejemplo es muy distinta a la usual. El trabajo en proyectos, que es lo que finalmente hace el alumno, es muy demandante. Y se le asigna bastante tiempo libre al alumno, de modo que disponga del tiempo necesario para leerse los capítulos de los libros que corresponde. Típicamente un alumno cursa solamente *dos* ramos al mismo tiempo, los ramos son de duración variable etc.

El modelo PBL se ha aplicado con éxito en distintos lugares del mundo, incluyendo Canadá, Australia, Dinamarca, México etc.

Algunas desventajas del modelo PBL en su forma usual son que un alumno puede tener por ejemplo un día a la semana entero más tres tardes enteras libres, de modo tal de disponer del tiempo necesario para estudiar en la biblioteca. Es decir que se presupone un cierto grado de disciplina y responsabilidad del alumnado que en nuestra cultura tiende a faltar. Se propone por lo tanto en este artículo un enfoque intermedio, mixto: es decir que sin necesidad de cambiar completamente los métodos de enseñanza, las mallas curriculares y el sistema en general, el docente puede encauzar el desarrollo de las asignaturas por caminos de corte más práctico, interactuar con los alumnos, planteándoles problemas a resolver, corriendo el riesgo de "perder" una o dos horas de clase completas con los alumnos estudiando algún problema a veces sencillo, manteniéndose en un papel de asesor del equipo de trabajo. Y una ventaja extra de este tipo de actividades es que el docente se entera de si los alumnos están captando *algo* de lo que se está viendo en clases. De lo contrario, al "actuar como abeja", es probable que la hora o el par de horas hayan sido de todos modos tiempo perdido.

Por lo tanto el profesor pasa de tener un papel de científico a asumir además el de líder, en cierto modo. Entonces aquellos docentes que tengan un carácter demasiado tímido pueden en realidad preferir seguir usando esquemas tradicionales de enseñanza; pero el otro extremo tampoco es favorable a PBL: el profesor demasiado dominante también va a ser un líder deficiente. Y por el lado del alumnado el sistema también puede fallar, sobre todo en la cultura actual de la televisión, que tiende a producir jóvenes pasivos. Por lo tanto como regla general se puede decir que el docente puede hacer un esfuerzo razonable por interesar a los alumnos en actividades de

carácter práctico, pero si a pesar de los esfuerzos el sistema no da los frutos esperados siempre es posible volver a esquemas docentes de “transmisión de información”, y en este sentido se plantea en este artículo una propuesta no tan radical como en el modelo PBL en su forma usual, que sí exige la modalidad de proyecto siempre. Ahora, ciertamente que el alumno siempre resuelve problemas de carácter práctico en los distintos ramos, por ejemplo al momento de rendir certámenes o tests, pero el sentimiento tiende a ser más bien negativo, es decir “me fue más o menos mal (o más o menos bien), hice lo que pude en tal o cual problema”. En la clasificación presentada más arriba, el enfoque propuesto sería una suerte de “enfoque 2.5”.

Se expone entonces brevemente a continuación un ejemplo de actividad práctica por parte de alumnos: la construcción de un resolvedor automático de sudokus como ejemplo en el estudio de la teoría de la NP-completitud en ciencia de la computación.

METODOLOGÍA

Sin entrar en detalles técnicos excesivos, el problema de Satisfacibilidad de Fórmulas Proposicionales Booleanas (SAT) fue el primer problema identificado como perteneciente a la clase de complejidad NP-completo (Cook, 1971), es decir que hay una clase importante de problemas (la clase NP) que puede ser “eficientemente traducida” a una instancia de SAT, es decir a una fórmula proposicional que resuelve el problema original si es que se puede encontrar un conjunto de valores de verdad para las proposiciones que haga verdadera a la fórmula (Karp, 1972). Ver por ejemplo (Papadimitriou, 1994) para una exposición general del tema. Se podría decir que los programas que resuelven SAT constituyen una tecnología de razonamiento automático, que ha encontrado muchas aplicaciones a nivel industrial en las últimas décadas en variadas áreas tales como verificación de diseño de circuitos digitales, optimización e inteligencia artificial por nombrar algunos. Los problemas SAT son resueltos por un algoritmo comúnmente llamado solver, el cual automáticamente busca la solución al problema planteado (Bordeaux et al., 2006). El sistema de resolución de un SAT-solver se podría ver como una caja negra, en la cual no se necesita una mayor interacción entre el usuario y el solver (Bordeaux et al., 2006; Moskewicz et al., 2001; Simon et al., 2005).

La entrada al solver, o input, es una fórmula proposicional booleana expresada en Forma Normal Conjuntiva (CNF), la cual es una conjunción (Y lógico (∧)) de una o más cláusulas. Una cláusula es una disyunción (O lógico (∨)) de uno o más literales. Por su parte, un literal es una instancia positiva o negativa de una letra proposicional. Por ejemplo, las siguientes formulas están en CNF:

$$A \wedge B \qquad \neg A \wedge (B \vee C) \qquad (A \vee B) \wedge (\neg B \vee C \vee \neg D) \wedge (D \vee \neg E) \qquad (1)$$

Como output el solver responde sí o no, dependiendo de la satisfacibilidad de la formula. Si la instancia es satisfacible, algunos SAT-solver pueden entregar una valuación que hace que la fórmula ingresada como input sea verdadera. Para modelar el problema, el principal desafío es expresar todas las restricciones del problema original en fórmulas proposicionales booleanas en CNF.

El problema a ser resuelto por los alumnos mediante SAT-solvers fue el del conocido Sudoku, que es un juego de ingenio en el cual se da una grilla de 9X9 celdas con algunos números entre el 1 y el 9 ya puestos, y de lo que se trata es de completar la grilla con enteros del 1 al 9 de modo que no se repitan ni en las filas ni en las columnas ni al interior de los nueve sub-cuadrados de 3X3 que hay en la grilla. Los alumnos modelaron este problema, llegando a una forma normal conjuntiva que expresa todas las características del juego de Sudoku.

RESULTADOS Y DISCUSIÓN

Para codificar Sudoku como un problema de satisfacibilidad, los alumnos llegaron al siguiente modelo: para cada celda de la grilla *S* de 9x9, se asocian 9 posibles valores, así tenemos 9x9x9 = 729 variables proposicionales. Se asume la notación S_{xyz} para referirse a estas variables, en

donde *x* representa una fila de S, *y* representa una columna de *S* y *z* representa el valor de la celda. Por ejemplo, S_{254} significa que la celda de la fila 2 y columna 5 tiene el valor 4, es decir, S[2, 5] = 4. La forma de plantear la notación y las reglas del Sudoku como CNF es la siguiente:

Hay al menos un número en cada celda:

$$\bigwedge_{x=1..9} \bigwedge_{y=1..9} \bigvee_{z=1..9} S_{xyz} \quad (2)$$

Cada número aparece a lo más una vez en cada fila:

$$\bigwedge_{y=1..9} \bigwedge_{z=1..9} \bigwedge_{x=1..8} \bigvee_{i=x+1..9} (\neg S_{xyz} \vee \neg S_{iyz}) \quad (3)$$

Cada número aparece a lo más una vez en cada columna:

$$\bigwedge_{x=1..9} \bigwedge_{z=1..9} \bigwedge_{y=1..8} \bigvee_{i=y+1..9} (\neg S_{xyz} \vee \neg S_{xiz}) \quad (4)$$

Cada número aparece a lo más una vez en cada sub-grilla de 3x3:

$$\bigwedge_{z=1..9} \bigwedge_{i=0..2} \bigwedge_{j=0..2} \bigwedge_{x=1..3} \bigwedge_{y=1..3} \bigwedge_{k=y+1..3} (\neg S_{(3i+x)\,(3j+y)\,z} \vee \neg S_{(3i+x)\,(3j+k)\,z}) \quad (5)$$

$$\bigwedge_{z=1..9} \bigwedge_{i=0..2} \bigwedge_{j=0..2} \bigwedge_{x=1..3} \bigwedge_{y=1..3} \bigwedge_{k=x+1..3} \bigwedge_{m=1..3} (\neg S_{(3i+x)\,(3j+y)\,z} \vee \neg S_{(3i+k)\,(3j+m)\,z})$$

Esta codificación consta de una formula CNF de 8.829 cláusulas, de las cuales 81 son nueve-arias (restricción 1) y 8.748 son binarias (restricciones 2 a 4). Como resultado de esta implementación se obtiene un archivo que contiene todas estas restricciones. El formato de ese archivo es el siguiente:

```
p cnf 729 8.829
111 112 113 114 115 116 117 118 119 0
-111 -112 0
-111 -113 0
-111 -114 0
```

Esta notación significa que la codificación está en CNF, consta de 729 variables y 8.829 cláusulas. La primera cláusula dice que S111 y S112 no pueden se verdaderas al mismo tiempo, es decir que la casilla (1,1) no puede tener al mismo tiempo los valores 1 y 2, etc. Una vez ingresado a un SAT-solver, en este caso MiniSAT, la respuesta obtenida es:

```
SAT
-1 -2 ... -99 -100 ... -541 542 -543 ... 999 0
```

En estos dos archivos los números que están negados significan que S_{xyz} es falso. Para el archivo de salida los valores positivos son los valores que se consideran como respuesta para completar el tablero. Además, la primera línea del archivo de salida con la palabra SAT significa que el problema es satisfacible y se entrega la valuación que lo hace verdadero. Sino se obtendría solamente UNSAT.

Sin entrar a dar detalles técnicos extra, o a abordar temas teóricos relacionados con la teoría de la NP-completitud en ciencia de la computación, lo que se destaca aquí es el hecho de que los estudiantes, al desarrollar un ejemplo práctico en el cual efectivamente un problema de carácter computacional es llevado a una forma normal conjuntiva que entrega (cuando es satisfacible) una solución al problema original, pueden captar la idea de la reducción de problemas a SAT en forma más vívida que si meramente se expusieran los conceptos en la pizarra o en las transparencias

(un poco la idea del proverbio chino). El alumno efectivamente desarrolla una transformación concreta de un problema a otro, *reflexiona* sobre el tema, es decir que no solamente lo lee, y por lo tanto el proceso de aprendizaje resulta más eficiente. Nos comentan los alumnos involucrados que este proceso de reflexión los hace en algunos casos tratar de visualizar otros problemas de otros tipos como un problema a ser planteado mediante una forma normal conjuntiva.

CONCLUSIONES

En este artículo se dio una reseña del método de aprendizaje basado en resolución de problemas y se sugirieron formas de aplicación práctica de este método que parecen más adaptadas a la realidad latinoamericana que la propuesta original del método. Los alumnos responden, se sienten motivados en general, y quedan con un sentimiento positivo de *logro.* El proceso tiende a ser más eficiente en general, y en la terminología de la metáfora inicial, se trata para los docentes de ser más “hormigas” y menos “abejas”.

La tarea de introducir cambios en los métodos de enseñanza en la línea de las ideas presentadas en este artículo puede resultar ardua, sobre todo si se adopta el sistema PBL en su forma original. Posturas intermedias tales como las que se presentan en este artículo pueden ayudar a hacer más eficientes los procesos de enseñanza-aprendizaje.

REFERENCIAS

Adleman, L. y K. Manders; *Reducibility, Randomness and Intractability.* Proceedings of the 9th ACM Symposium on the Theory of Computing, pp. 151-163 (1977).

Bordeaux, L., Y. Hamadi y L. Zhang; *Propositional Satisfiability and Constraint Programming: A comparative Survey*. ACM Computing Surveys, ISSN:0360-0300 (en línea), 38(4), 2006. http://portal.acm.org/citation.cfm?id=1177354

Cook, S.; *The Complexity of Theorem-Proving Procedures.* Proceedings of the 3rd IEEE Symposium on the Foundations of Computer Science, pp. 151-158 (1971).

Franks, N. y T. Richardson; *Teaching in tandem-running ants*. Nature: 439(7073), 153 (2006).

Karp, R.; *Reducibility among Combinatorial Problems.* En Complexity of Computer Computations, J. W. Thatcher y R. E. Miller, editores, pp. 85-103. Plenum Press, New York (1972).

Lambert, T., E. Monfroy, y F. Saubion; *A Generic Framework for Local Search: Application to the Sudoku Problem*. Proceedings of the International Conference on Computation Science ICCS’2006 pp. 641-648 (2006). Lecture Notes in Computer Science 3991, Springer Verlag.

Levin, L.; *Universal Sorting Problems.* Problems of Information Transmission 9, pp. 265-266 (1973).

Mauffette, Y.; Comunicación personal (2007).

Moskewicz, M. y otros cuatro autores; *Chaff: Engineering an Efficient SAT-solver*. Proceedings of ICCAD 2001 pp. 530-535 (2001).

Mills, J. y D. Treagust; *Engineering Education - Is Problem-Based or Project-Based Learning the Answer?* Australasian Journal of Engineering Education, ISSN 1324-5821 (en línea), 2003-04. http://www.aaee.com.au/journal/2003/mills_treagust03.pdf

Papadimitriou, C., *Computational Complexity*. Addison Wesley (1994).

Prince, M. J. y R. Felder; *Inductive Teaching and Learning Methods: Definitions, Comparisons and Research Bases.* Journal of Engineering Education 95(2) (2006).

Simon, L., D. Le Berre y E. Hirsch; *The SAT 2002 competition*. Annals of Mathematics and Artificial Intelligence: 43(1), 308-342 (2005).