Técnicas de inteligencia artificial aplicadas a la confección de horarios*

Fredy Cuenca Lucero

Universidad de Lima

Resumen

El presente artículo describe la estrategia que utiliza técnicas de inteligencia artificial para confeccionar horarios de clase de una entidad académica, de manera automática, con calidad igual o superior a la de los horarios confeccionados por expertos humanos en un tiempo razonable. La estrategia ha sido implementada como parte de un sistema de soporte de decisiones que, además de confeccionar automáticamente horarios de calidad dispone de herramientas para que estos horarios puedan ser modificados de manera manual pero asistida. La confección automática de horarios ha sido dividida en dos fases: la primera consiste en construir un horario inicial factible; la segunda fase es de optimización, en la cual el horario inicial es sometido a una serie de perturbaciones (modificaciones) hasta alcanzar un nivel de calidad deseado. La construcción del horario inicial factible ha sido modelado como un problema de satisfacción de restricciones, mientras que el proceso de optimización ha sido modelado como un problema de búsqueda. El sistema de soporte de decisiones ha sido implementado en Visual Basic 6.0, sobre una base de datos relacional Microsoft SQL Server 2000.

Palabras clave:
Horarios, optimización combinatoria, inteligencia artificial, problemas de satisfacción de restricciones, metaheurística, búsqueda local.

* Agradezco a la profesora Angélica Kamiyama, por haberme retado a resolver este problema.

1. Introducción

En todos los periodos académicos, la Facultad de Ingeniería de Sistemas tiene la tarea de confeccionar un horario de clases para la matrícula de sus alumnos. Esta labor es muy importante, pues la calidad del horario influirá fuertemente en el número de alumnos matriculados, en el nivel de satisfacción de los estudiantes y docentes y, consecuentemente, en el rendimiento académico del alumnado.

Un horario de clases es un listado que contiene todas las reuniones semanales que la facultad ofrecerá a los alumnos durante el periodo académico vigente como parte de su formación. Las reuniones que componen un horario de clases deben satisfacer un conjunto de restricciones, las cuales se clasifican en dos tipos:

- *Restricciones fuertes u obligatorias*: Son aquellas que definen la factibilidad o usabilidad del horario; para poder ser usado, un horario debe satisfacer todas las restricciones fuertes. Las principales restricciones fuertes consideradas en el problema son:
  - En un aula y una hora determinada no puede dictarse más de una reunión.
  - Un docente a una hora determinada no puede dictar más de una reunión.
  - Se deberá respetar la disponibilidad horaria de los docentes.
  - Se deberá respetar la disponibilidad horaria de las aulas.
  - Durante las horas de prácticas integradas no deberá programarse ninguna reunión.
- *Restricciones débiles o deseables*: Son aquellas que definen la calidad del horario. Un horario podría violar estas restricciones, pero debería hacerlo en la menor medida posible. Las principales restricciones débiles consideradas en el problema son:
  - Para cursos-sección que requieran más de una reunión semanal, se debe procurar que sus reuniones se inicien a la misma hora o lo más paralelamente posible.
  - Para cursos-sección que requieran más de una reunión semanal, se debe procurar que sus reuniones no sean dictadas en días muy próximos ni muy lejanos.
  - Las reuniones en un día determinado de un docente no deben estar muy separadas entre ellas.

– Si un docente tiene que dictar dos reuniones consecutivas, se debe procurar que estas se dicten en el mismo pabellón, para evitar demoras por cambio de aula.

El proceso de confección de horarios empieza varias semanas antes del inicio de un periodo académico, cuando la facultad solicita la disponibilidad horaria a sus docentes y la disponibilidad de aulas al área responsable. Paralelamente, la facultad elabora la programación de secciones, la cual consiste en determinar el número de secciones que se ofrecerá por cada curso y el número de secciones que tendrá a su cargo cada docente por curso durante el periodo académico venidero.

Luego de lo expuesto, el problema de confección de horarios puede ser formulado de la siguiente manera: Dadas las disponibilidades horarias de los docentes y aulas, y la programación de secciones, ¿cómo se puede elaborar —automáticamente y en un tiempo razonable— un horario que satisfaga todas las restricciones fuertes y minimice la violación de las restricciones débiles?

En la actualidad, el problema de confección de horarios es ampliamente estudiado en muchas de las principales universidades y centros de investigación a escala mundial. La dificultad para resolver este problema radica en el enorme número de horarios que podrían generarse. Por ejemplo, en nuestra facultad, y para datos reales de las disponibilidades horarias de docentes y aulas y la programación de secciones del periodo académico 2007-1, se puede demostrar que existe la posibilidad de confeccionar más de 10^{1800} horarios. Suponiendo que existiese una computadora que pueda generar y evaluar un billón de horarios por segundo, la exploración completa de este enorme espacio de soluciones requeriría un tiempo equivalente a varias veces la edad del Universo, que es tan solo del orden de 10^9 años (Griffith, 2003). Evidentemente, la exploración exhaustiva no es una opción viable para encontrar el horario óptimo.

A pesar de la dificultad del problema, la facultad logra confeccionar un horario en cada periodo académico apoyándose en los horarios de los periodos previos y el criterio y experiencia de las personas que realizan esta labor. Debido a que el tiempo disponible por la facultad para confeccionar un horario es (infinitamente) menor al que se requeriría para encontrar con certeza el horario óptimo, no debe extrañar que la calidad actual de los horarios confeccionados ofrezcan muchas oportunidades de mejora.

El objetivo de este trabajo es resolver el problema de confección de horarios mediante el desarrollo de un sistema informático que pueda actuar con el criterio inteligente de un usuario humano a la velocidad de un computador. Dicho sistema no sólo permitirá ahorrar tiempo y recursos sino que además permitirá obtener horarios de mejor calidad.

2. Estrategia de solución

El problema de confección de horarios pertenece al campo de la optimización combinatoria. La optimización combinatoria es una rama de las matemáticas cuyo objetivo consiste en hallar, dentro de un conjunto finito de elementos llamado espacio de soluciones, aquel que optimice una determinada función. En el caso particular del problema de confección de horarios, cada horario representa un elemento del problema y el conjunto de todos los horarios que podrían ser confeccionados para una determinada disponibilidad horaria de docentes, aulas y programación de secciones es el espacio de soluciones. Si definimos una función que cuantifique el costo de un horario a partir de las restricciones violadas de este, y evaluamos dicha función sobre cada horario del dominio, definiremos sobre este un paisaje de búsqueda (figura 1).

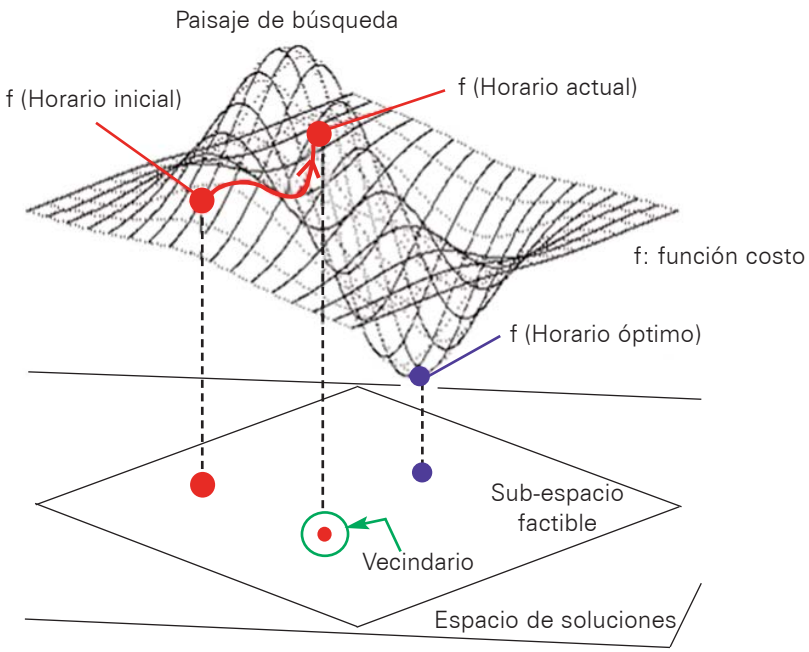

Figura 1. Problema de optimización combinatoria.

El problema de confección de horarios del gráfico anterior consiste en hallar sobre el espacio de soluciones del problema aquel horario que corresponda al punto más bajo del paisaje de búsqueda. Ese horario será justamente el de menor costo o mayor calidad. Debido a que existe un alto número de horarios infactibles en el espacio de soluciones (aquellos cuyo costo infinito no se aprecia en el paisaje de búsqueda) se ha considerado conveniente limitar la búsqueda al subespacio factible del problema. Para encontrar el horario óptimo se requieren dos fases:

- Construcción de un horario factible inicial H_{ini}, que consiste en seleccionar un horario del subespacio factible del problema.
- Optimización de un horario, que consiste en explorar el paisaje de búsqueda tratando de llegar hasta el horario óptimo a partir del horario H_{ini} seleccionado en la fase anterior.

3. Fase I: Construcción de un horario factible inicial

El horario factible generado en esta fase servirá como punto de acceso al subdominio factible del problema sobre el cual se encuentra el horario óptimo (figura 1). La construcción del horario factible inicial se ha modelado como un problema de satisfacción de restricciones y ha sido resuelto utilizando técnicas pertenecientes al campo de la inteligencia artificial, como consistencia de dominio y búsqueda sistemática.

3.1 Problemas de satisfacción de restricciones (CSP)

Los problemas de satisfacción de restricciones, llamados CSP por sus iniciales en inglés (*Constraint Satisfaction Problem*), son problemas matemáticos en los cuales se debe asignar un conjunto de valores a igual número de variables, respetando las restricciones existentes del problema. Los CSP son tema de un intenso estudio en inteligencia artificial e investigación operativa.

Formalmente, un CSP es una terna (V, D, C), donde V es un conjunto $\{V_1, V_2, \ldots, V_n\}$ de variables que representan aspectos fundamentales del problema; D es el dominio del problema conformado por los dominios $D(V_i) = \{v_{i1}, v_{i2}, \ldots, v_{in}\}$ de todas las variables $V_i \in V$, y el conjunto C representa las restricciones del problema. Una solución de un CSP es un conjunto de asignaciones $V_i \leftarrow v_{ik}$ donde se respetan todas las restricciones C del problema. Las principales técnicas para resolver un CSP se clasifican de la siguiente manera:

- Técnicas de consistencia de dominio, cuyo objetivo es reducir el dominio de un CSP eliminando para ello aquellos valores que no podrían formar parte de una solución factible. Dentro de estas técnicas destacan la consistencia de nodo, de arco y de camino.
- Métodos de búsqueda sistemática, que tratan de hallar una solución explorando ordenadamente el espacio de búsqueda. Los métodos más destacados son *BackTracking*, *BackJumping*, *BackMarking* y *Forward Checking*.

3.2 Construcción de un horario factible como CSP

El subproblema de construcción de un horario inicial factible ha sido modelado como CSP, definiendo como variables a cada una de las reuniones R_1, R_2,..., R_n que se desea programar. Cada reunión por programar R_k puede ser definida con un código curso, sección y reunión. Adicionalmente, debido a la programación de secciones, se puede saber qué docente dictará dicha reunión.

Por otro lado, el dominio $D(R_i)$ de cada reunión R_i será el conjunto de combinaciones sección-aula-hora, donde dicha reunión puede programarse. Finalmente, el conjunto C estará conformado por las restricciones fuertes del problema de confección de horarios. Formalmente:

$R = \{(c, s, r, d) /$ la reunión r del curso-sección c-s será dictado por el docente d$\}$

$D(R_i) = \{(sx, a, h) /$ la reunión R_i será dictada en la sección sx, el aula a y la hora h$\}$

$C(V_1,..., V_k) = \{$restricción obligatoria k-aria entre $R_1,....$ y $R_k\}$

donde: $D = \cup_i D(R_i)$ y $C = \cup_{V1,..., Vk} C(R_1,..., R_k)$

Una solución de este CSP, es decir un horario factible, tendría la siguiente forma:

H = {(1392, 1, 1, jsanchez) ← (301, W44, 1), (1392, 1, 2, jsanchez) ← (301, W43, 31),...
..., (7131, 1, 1, jnunez) ← (801, G207, 27),... , (2284, 4, 1, lchavez) ← (1002, W24, 42)}

Para resolver este CSP, primero se ha reducido el dominio de este utilizando una técnica de consistencia de nodo y luego se ha construido un horario factible inicial sobre dicho dominio reducido utilizando el algoritmo *Forward Checking*.

3.2.1 Consistencia de dominio

El presente trabajo ha aplicado la consistencia de nodo sobre el dominio del problema. Un proceso de consistencia de nodo debe revisar cada variable del problema y eliminar de su dominio aquellos valores que violen alguna restricción unaria. Las restricciones unarias del problema son:

- Durante las horas de prácticas integradas no deberá programarse ninguna reunión.
- Un docente debe estar disponible durante todas las horas de una reunión.
- Un aula debe estar disponible durante todas las horas de una reunión.

Estas restricciones son consideradas unarias porque su definición involucra la programación de una sola reunión. Contrariamente, las restricciones que evitan los cruces de docentes y aulas son restricciones binarias. Luego de ejecutar el proceso de consistencia de nodo sobre todas las variables, se dice que el dominio del problema ha quedado nodo-consistente. Esto significa que cualquier sección-aula-hora que asignemos a una reunión respetará todas las restricciones mencionadas anteriormente. Consecuentemente, durante la posterior construcción del horario factible ya no debemos preocuparnos de verificar las restricciones unarias del problema.

```
PROC NodoConsistencia (CSP = (R, D, C))
    REPETIR PARA CADA Ri ∈ R
        REPETIR PARA CADA rj ∈ D (Ri)
            SI C(Ri) ∈ C / rj ∉ C(Ri) ENTONCES
                D (Ri) ←D (Ri) – { ri }
            FIN SI
        FIN REPETIR
    FIN REPETIR
FIN PROC
```

3.2.2 Búsqueda sistemática

Después que la consistencia de nodo ha reducido el dominio del CSP corresponde buscar dentro de este dominio reducido un horario factible que sirva como punto de partida para hallar el horario óptimo. La búsqueda deberá encontrar un conjunto de asignaciones $\{R_1 \leftarrow r_1, R_2 \leftarrow r_2 \ldots, R_n \leftarrow r_n\}$ donde cada reunión $R_i \in V=\{R_1, R_2, \ldots, R_n\}$ del problema haya sido

asignada con algún valor $r_i \in D(R_i)$ de su dominio, de modo que se respeten las restricciones fuertes del problema. Así, el conjunto $\{R_1 \leftarrow r_1, R_2 \leftarrow r_2 \ldots, R_n \leftarrow r_n\}$ sería la solución del CSP, o sea el horario factible inicial. Para hallar esta solución se ha utilizado el algoritmo *Forward Checking* (FC). En cada paso de su ejecución, FC asigna a la reunión R_i el primer valor r_i disponible de su dominio $D(R_i)$ y luego poda de los dominios $\{D(R_{i+1}), D(R_{i+2}), \ldots, D(R_n)\}$ de las reuniones R_j $(j>i)$ aún no asignadas aquellos valores r_j que resulten conflictivos con la asignación $R_i \leftarrow r_i$, o sea donde $R_i \leftarrow r_i$ y $R_j \leftarrow r_j$ violen alguna restricción binaria. Si este podado eliminase por completo algún dominio $D(R_k)$, los efectos de la asignación $R_i \leftarrow r_i$ deberán ser revertidos y se deberá buscar el siguiente valor $r_{i'}$ disponible en $D(R_i)$. Este proceso se repite con la siguiente reunión R_{i+1} y así sucesivamente, hasta conseguir asignar la última reunión R_n. En ese momento el conjunto $H_{ini} = \{R_1 \leftarrow r_1, R_2 \leftarrow r_2 \ldots, R_n \leftarrow r_n\}$ es el horario factible inicial buscado.

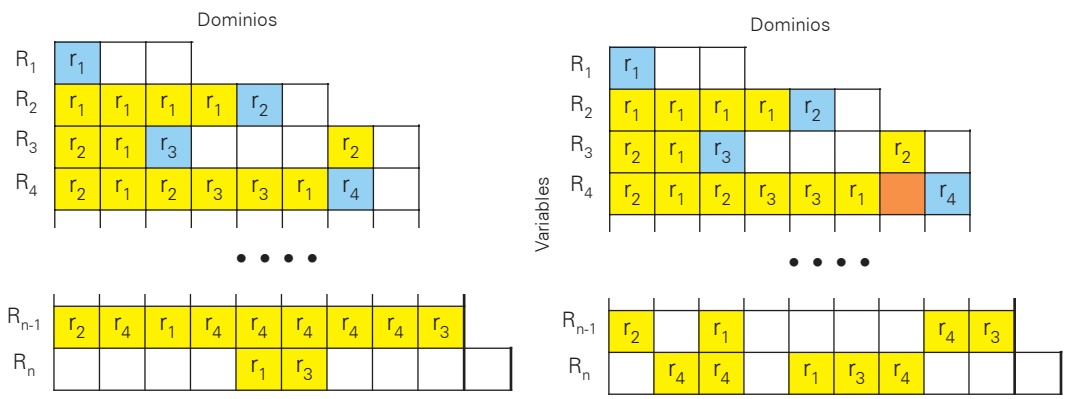

Figura 2. Algoritmo *Forward Checking* (FC).

En el gráfico anterior se observa cómo el podado realizado luego de la asignación $R_4 \leftarrow r_4$ anula el dominio $D(R_{n-1})$. La verificación pospodado que realiza FC le permite detectar esta situación inmediatamente y por ello devuelve al dominio del problema los valores que han sido podados por causa de $R_4 \leftarrow r_4$ y trata de asignar a R_4 el siguiente valor disponible de su dominio $D(R_4)$. Luego, el algoritmo continuará tratando de asignar una por una las variables restantes, hasta finalmente asignar R_n o concluir que no existe solución.

4. Fase II: Optimización de un horario

Una vez obtenido el horario factible inicial H_{ini}, se requiere explorar el subdominio factible del problema de confección de horarios para tratar de llegar hasta el horario de mejor calidad en un tiempo razonable. Para llevar a cabo esta tarea se ha desarrollado un algoritmo metaheurístico. Un metaheurístico es una estrategia de optimización basada en la búsqueda local que intenta aunque no garantiza llegar al punto óptimo del paisaje de búsqueda a partir de cualquier punto inicial. Para lograr este propósito, en cada paso de su ejecución solicita a alguna heurística subordinada hallar dentro del vecindario del punto actual otro punto que podría acercarnos hacia el óptimo del paisaje (figura 1). Mientras la búsqueda en el vecindario del punto actual o búsqueda local es ejecutada por un heurístico, es el metaheurístico el que finalmente decide la aceptación o rechazo del punto propuesto por el heurístico, basándose en la historia de la búsqueda y en sus propios criterios.

Para la optimización de un horario de clases se ha diseñado un metaheurístico que cuenta con el apoyo de varios heurísticos subordinados que se encargan de realizar el cambio de aula, cambio de hora, cambio de sección, etcétera, sobre una reunión determinada. Este metaheurístico elige aleatoriamente una reunión que esté violando alguna restricción débil para reprogramarla en otra aula, hora o sección, procurando eliminar la violación cometida. El seudocódigo de dicho metaheurístico es el siguiente:

0) $H_{act} \leftarrow H_{ini}$ // horario factible hallado en Fase I.
1) Seleccionar aleatoriamente en el horario H_{act} una reunión R cuya programación en (sx, a, h) viole alguna restricción débil C_k
2) Seleccionar una heurística h que pueda eliminar la penalización identificada en 1).
3) Aplicar la heurística h sobre la reunión R del horario H_{act} generando un nuevo horario H_{new} donde R $\leftarrow$ (sx', a', h').
4) Si H_{new} es mejor que H_{act} entonces $H_{act} \leftarrow H_{new}$
5) Si la calidad de H_{act} es suficiente entonces terminar.
6) Volver al paso 1).

Durante todo el proceso de optimización se busca mantener la factibilidad del horario H_{act}. Para un mayor detalle de las restricciones $C_{k,}$ las heurísticas h, la función de calidad y la condición de parada consultar Cuenca Lucero, Fredy. "Sistema de soporte de decisiones para la elaboración de horarios", pp. 82-86, 93-105.

5. Sistema de soporte de decisiones

Para que el proceso de confección de horarios pueda realizarse en forma automática se requiere desarrollar un sistema informático que implemente los algoritmos descritos anteriormente. Para ello se deben agrupar los algoritmos en procesos, crear las estructuras de datos necesarias y diseñar las interfaces de usuario.

5.1 Procesos

Los procesos que componen el sistema desarrollado son:

- Cierre de Programación. En el caso de nuestra facultad, las reuniones que se ofrecerán durante un periodo académico quedan indirectamente determinadas por la programación de secciones. Luego, el proceso de Cierre de Programación debe generar un conjunto que contenga todas las reuniones que se van a programar a partir de la programación de secciones (número de secciones que se van a abrir por curso) y los requerimientos específicos de cada curso (duración y número de reuniones por semana).
- Carga de Dominio. Durante este proceso se construye el dominio, es decir el conjunto de sección-aula-hora de las reuniones generadas anteriormente. Para ello se utilizan las disponibilidades horarias de docentes y aulas registradas por el usuario.
- Construcción del Horario Factible. Este proceso resuelve el CSP cuyas variables y dominios fueron definidas por el Cierre de Programación y la Carga de Dominio, respectivamente. Durante este proceso se ejecutan los algoritmos de consistencia de nodo y *Forward Checking*. El horario factible obtenido es mostrado al usuario para que pueda modificarlo manualmente o someterlo a un proceso posterior de optimización automática.
- Optimización de Horario. Aquí se mejora la calidad de horario factible hasta que el usuario considere conveniente. Durante este proceso se ejecuta el metaheurístico mostrado en el acápite 4.

La interrelación de los procesos que componen el sistema de soporte de decisiones se muestra en el siguiente gráfico:

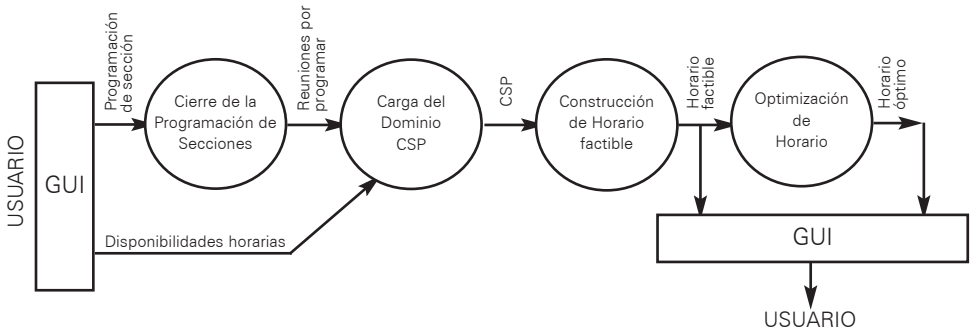

Figura 3. Procesos del sistema.

5.2 Base de datos relacional

En el modelo del problema de construcción de horario factible como CSP (acápite 3.2), las reuniones y los dominios son relaciones matemáticas y por ello su implementación sobre una base de datos relacional ha resultado bastante natural. Esta implementación no solo nos permite guardar el dominio del problema sin importar su tamaño (las implementaciones con arreglos están limitadas por la memoria RAM, que es mucho menor que la capacidad de un disco duro) sino que nos facilita la implementación de varias funciones importantes. Muchas implementaciones de CSP guardan cada restricción del problema como una matriz booleana, donde las filas y columnas representan los dominios de las variables involucradas en la restricción y cada elemento indica la existencia o ausencia de conflictos entre los valores que representan su fila y columna. En el problema de confección de horarios se requeriría un gran número de matrices y por ende una gran cantidad de espacio. Por ejemplo, para almacenar la restricción de cruces de aula en un horario de N reuniones se requerirían N x N-1 matrices. Con la implementación del *Forward Checking* sobre CSP solo se requiere implementar una función que actúe luego de programar una reunión, eliminando (lógicamente) del dominio de las reuniones por programar aquellos valores que compartan la misma aula y hora asignadas recientemente. Esta función se implementa con una consulta simple basada en el lenguaje SQL. Del mismo modo, la consistencia de nodo y la restauración de dominios pueden ser implementados fácilmente (Cuenca, 2007: 93-105, 135-145).

En cuanto al proceso de optimización, este utiliza consultas SQL para identificar fácilmente las reuniones que están violando alguna restricción y para identificar los valores posibles donde se pueda reprogramar una reunión manteniendo la factibilidad del horario.

5.3 Soporte a las decisiones

Finalmente, a pesar de que el sistema intenta buscar el horario óptimo global, debido al enorme dominio del problema y el carácter heurístico del proceso de optimización generalmente tendremos que conformarnos con horarios subóptimos. No obstante ser los horarios subóptimos de buena calidad podrían ofrecer algunas oportunidades de mejora al usuario. Por esta razón el sistema añade a su principal tarea (la confección automática de horarios), la posibilidad de efectuar modificaciones manuales por parte del usuario. Sin embargo, se debe considerar que las modificaciones manuales que realiza el usuario podrían introducir nuevas penalizaciones al horario, disminuyendo así su calidad o, peor aún, destruyendo su factibilidad. Para eliminar este riesgo el sistema apoya al usuario en la toma de decisiones, sugiriéndole un conjunto de posibles cambios exentos del riesgo de introducir cruces o programar docentes y/o aulas fuera de su disponibilidad horaria. Además, el sistema ofrece al usuario la posibilidad de administrar hasta un máximo de 100 horarios y puede comparar objetivamente su calidad.

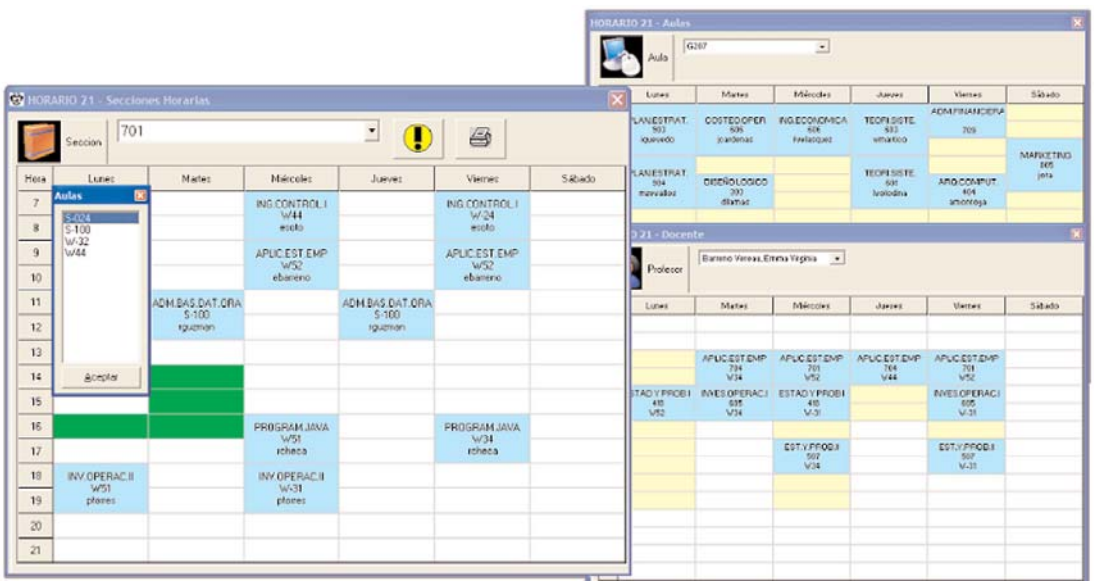

Figura 4. Formularios de modificación manual de horarios.

Todas estas características ayudan al usuario a que pueda tomar decisiones en dos niveles: en el nivel de un grupo de horarios (¿cuál horario es el mejor?, ¿cuál horario merece ser sometido a un proceso de reoptimización?, ¿cuán mejor es un horario respecto de otro?), y en el nivel de reuniones dentro de un horario (¿a qué otra aula podría mover esta reunión?, ¿podría trasladar esa reunión?, ¿podría trasladar esa reunión al sábado?, etcétera).

Para una descripción detallada de las herramientas brindadas por el sistema de soporte de decisiones consultar Cuenca Lucero, Fredy. "Sistema de soporte de decisiones para la elaboración de horarios", pp. 146-184.

6. Resultados

Los resultados mostrados en este acápite se refieren al análisis de eficiencia y eficacia de los algoritmos involucrados en la construcción y optimización de horarios. Una mayor variedad de pruebas puede ser consultada en Cuenca Lucero, Fredy. "Sistema de soporte de decisiones para la elaboración de horarios", capítulo 8. Las pruebas mostradas han sido ejecutadas en un computador con procesador Intel (R) Core ™ 2 CPU 1.66 GHz. Los resultados mostrados a continuación son el promedio de una decena de pruebas.

Proceso de cierre de programación

Cursos por programar	56
Cursos-sección por programar	293
Docentes por programar	114
Reuniones por programar generadas	528
Tiempo del proceso	1 seg

Proceso de carga de dominio

Reuniones por programar	528
Tamaño del dominio inicial del CSP	3,522,620
Número de soluciones completas totales	14E+1800 horarios
Tamaño del dominio nodo-consistente	2,274,761
Porcentaje de reducción por consistencia de nodo	35.4%
Tiempo de generación del dominio del CSP	5 mins

Proceso de construcción de horario factible

Reuniones por programar	528
Reuniones que no se pudieron programar	0
Total valores podados en el proceso	28,300
Promedio valores podados por variable	54
Dominio promedio de una variable	4232
Número de *BackTracking* requeridos para hallar una solución	27
Tiempo del proceso	23 mins

Proceso de optimización de un horario

De lo observado en una decena de pruebas se puede afirmar que el proceso de optimización sobre un horario factible reduce su costo inicial de este en un 80% en 90 minutos en promedio. A continuación se muestran los resultados de una de las pruebas realizadas.

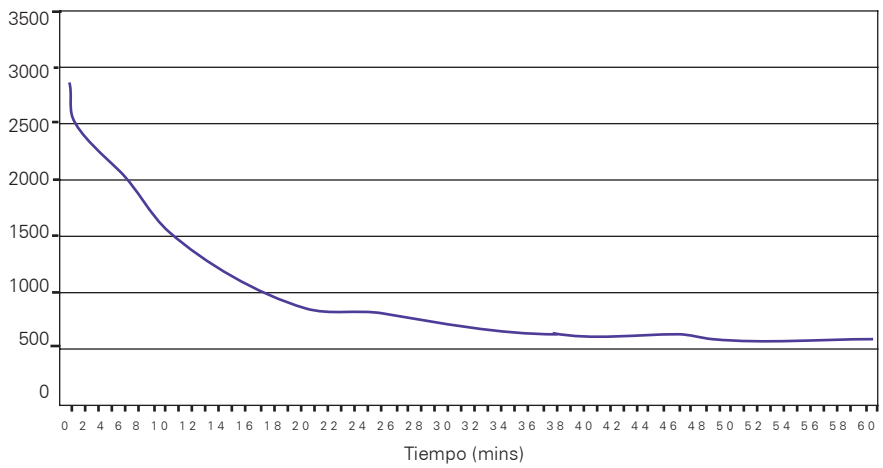

Figura 5. Costo horario vs tiempro (mins)

7. Conclusiones

- La estrategia de resolver el problema de confección de horarios en dos fases (construcción y optimización) ofrece muchas ventajas, como garantizar al menos la factibilidad del horario final, eliminar la preocupación por las restricciones débiles durante la construcción y por las restricciones fuertes durante la optimización y permitir explorar un subconjunto muy reducido del dominio original del problema.

- Las técnicas de inteligencia artificial ayudan a implementar la estrategia planeada y resolver el problema de confección de horarios. La fase de construcción es modelada y resuelta como CSP y la fase de optimización como un problema de búsqueda.
- El sistema informático desarrollado reducirá notoriamente el tiempo de elaboración del horario de clases para matrícula. El tiempo que toma la confección del horario de clases actualmente se reducirá de una semana en promedio (Kamiyama, 2004) a unas pocas horas.
- El uso del lenguaje relacional SQL facilita la implementación de funciones importantes. Las funciones de podado y restauración de dominio utilizadas por el algoritmo *Forward Checking* son fácilmente implementadas. De modo similar, la identificación de las reuniones que penalizan el costo del horario y la ejecución de perturbaciones que mantengan la factibilidad del horario facilitan la implementación de la fase de optimización cuando se utiliza lenguaje relacional.

8. Bibliografía

Bail, Rubin; Burke, Edmund; Kendall, Graham y Barry McCollum. "A simulated Anealing Hyper-heuristic for University Course Timetabling". School of Computer Sciencies & IT, University of Nottingham, 2006.

Burke, Eckersley, McCollum, Petrovic, "Hybrid Variable Neighbourhood Approaches to University Exam Timetabling", School of Computer Sciencies & IT, University of Nottingham, 2006.

Cuenca Lucero, Fredy. "Sistema de soporte de decisiones para la elaboración de horarios". Tesis de titulación. Lima: Universidad de Lima, Facultad de Ingeniería de Sistemas, setiembre del 2007.

Díaz, Adenso; Glover, Fred; Ghaziri, Hassan; González, J. L.; Laguna, Manuel; Moscato, Pablo y Fan T. Tseng. *Optimización heurística y Redes Neuronales.* Madrid: Paraninfo, 1996.

Dorigo, M.; Birattari, M. y T. Stutzle. "Ant colony optimization". IEEE Comp. Intell. Mag., Vol. 1, núm. 4. Noviembre del 2006, pp. 28-39.

Eley, Michael. *Ant algorithms for the exam timetabling problem.* Aschaffenburg: University of Aplied Science Logistics Laboratory, 2006.

Erben, Wilhelm y Jurgen Keppler. *A Genetic Algorithm Solving a Weekly Course-Timetabling Problem.* Department of Computer Science, Konstanz: Fachhochschule Konstanz, 1995.

Escolano Ruiz, Francisco; Cazorla Quevedo, Miguel Ángel; Galipienso, Mª Isabel; Colomina Pardo, Otto y Miguel Ángel Lozano Ortega. *Inteligencia artificial*. *Modelos, técnicas y áreas de aplicación.* Madrid: Thomsom Editores, 2003.

Ferland, Jacques A. y Charles Fleurent, "SAPHIR: A decision support system for Course Scheduling". *Interfaces Magazzine*.

Freuder, Eugene C. y Richard J. Wallace. *Partial Constraint Satisfaction.* Computer Science Department, Kingsbury Hall. Durham: University of New Hampshire, 1992.

Griffith, Susan. "Physicists Set Lower Age of Universe at 11.2 Billion Years, New Age Provides Evidence for Presence of a Dark Energy", 01/2003. [en línea] <http://www.universe.nasa.gov/press/2003/030103a.html>. Julio del 2007

Johnsonbaugh, Richard. *Matemáticas discretas.* Pearson Educación, 2005.

Kamiyama M., Angélica. "Software para la generación de horarios para la propuesta docente". Lima: Universidad de Lima, IDIC, 2004.

Larrosa, Javier y Pedro Meseger. *Razonamiento con restricciones.* Tutorial. Madrid: Iberamia, 2002.

Marriott, Kim y Peter Stuckey. *Programming with Constraints. An introduction.* The MIT Press, 1998.

Russell, Stuart y Peter Norvig. *Inteligencia artificial. Un enfoque moderno.* 2.ª Edición. Pearson/Pretince Hall, 2004.

Stallaert, Jan. *Automated Timetabling Improves Course Scheduling at UCLA.* University of Texas.

Tsang, Edward. "A Glimpse of Constraint Satisfaction". University of Essex, UK. [en línea] <http://www.essex.ac.uk/CSP/>. Junio del 2007.