Ciencia y Tecnología, 11 (2), 179-199, 2011

Heurística aplicada a la asignación de recursos humanos en una Universidad

Marcelo Damián Parrino •

Abstract

Este trabajo presenta un algoritmo de asignación de recursos humanos en una Universidad, junto con un desarrollo asociado con el fin de evaluar objetivamente el rendimiento computacional de la solución planteada. Los resultados obtenidos por las diferentes pruebas dentro del prototipo permitieron analizar la utilidad y performance de la solución heurística propuesta en un caso real, permitiendo su futura comparación frente a otras soluciones posibles.

Index Terms: heurística, algoritmo, asignación de recursos, profesores, universidad

Fecha de recepción del original: 12/10/2010 | Fecha de evaluación del original: 18/10/2010

• Facultad de Ingeniería, Universidad de Palermo. Mario Bravo 1050, Buenos Aires, Argentina
mparri@palermo.edu

1. Introducción

Se investigaron algunos de los posibles modelos heurísticos aplicables a la asignación de recursos humanos (profesores) para cumplir satisfactoriamente con las variables del entorno, tales como horarios, materias que se deben dictar, profesores disponibles para dictarlas, la disponibilidad de horarios, etc.

La idea fundamental fue seleccionar el modelo algorítmico más adecuado, ajustarlo y mejorarlo para resolver eficientemente el problema, y finalmente desarrollarlo e implementarlo en una solución informática capaz de colaborar activamente en la asignación de recursos humanos dentro de un ambiente académico o universitario.

El primer objetivo específico fue investigar puntualmente algunos de los algoritmos heurísticos disponibles junto con sus posibles optimizaciones y mejoras, que permitieran obtener soluciones heurísticas para el problema particular de la asignación de recursos humanos en una universidad.

Como segundo objetivo específico, a partir de los algoritmos y modelos analizados en la etapa previa, se desarrolló una aplicación de software que implementó la solución planteada, incluyendo las interfaces con el usuario, la interacción con los orígenes de datos y la presentación de resultados.

El último objetivo fue la evaluación de los resultados obtenidos y la comprobación de la validez de las soluciones encontradas por el algoritmo heurístico a partir de la implementación de la aplicación de software. La evaluación del desempeño de la aplicación permitió valorar la utilidad concreta de la solución en el mundo real.

2. Heuristica

Dado que el problema de asignación de recursos humanos (docentes) para cubrir cierta oferta académica de cursos y horarios es del tipo NP-Completo y demandaría un tiempo de ejecución exponencial en relación a la entrada de datos, se decidió buscar una solución heurística aproximada.

Como alternativas posibles se presentaron dos opciones:

I. La adaptación del problema de asignación de recursos humanos para poder utilizar alguno de los algoritmos heurísticos existentes.
II. La creación de un nuevo algoritmo heurístico pensado específicamente para el problema a resolver, definiendo los niveles de aspiración que se intentarían alcanzar.

Se decidió optar por la segunda opción, ya que se consideró inapropiado tener que adaptar todos los datos representativos del problema a los de un algoritmo existente para que éste pudiera resolverlo.

2.1. Niveles de aspiración

Se definieron entonces los siguientes niveles de aspiración para el modelado y definición del algoritmo heurístico para el problema de asignación de recursos humanos:

- Como nivel deseado para el límite del problema, se requirió que el algoritmo sea capaz de dar solución a problemas de tamaño similar a la asignación de materias requerida cada cuatrimestre por la Facultad de Ingeniería de la Universidad de Palermo.
- Como nivel de aspiración para el tiempo de cómputo del algoritmo, se pretendió obtener buenas soluciones en el menor tiempo posible, buscando en lo posible que el proceso de búsqueda de soluciones del algoritmo sea "instantáneo" desde la perspectiva del usuario.
- Con respecto a los factores económicos, el algoritmo debía ser factible de ser implementado en un sistema informático estándar, sin necesidad de contar con poder de cálculo y procesamiento sólo disponible en grandes centros de cómputos.
- En relación a los atributos relacionados con el problema real típico, se requirió que el algoritmo se modelara respetando los objetos y datos existentes en el problema de asignación de recursos humanos.

2.2. Algoritmo de asignación

Se recurrió entonces al modelado de un algoritmo nuevo, diagramado a medida de los objetos que representarían los datos del problema. De esta forma se obtuvo un algoritmo heurístico, que cumpliendo con la premisa de hallar buenas soluciones al problema de asignación de recursos humanos, no tuviera la necesidad de contar con un tratamiento adicional previo y posterior de los datos con el fin de adaptarlos, desaprovechando información y malgastando recursos en funciones que no son especificas del problema.

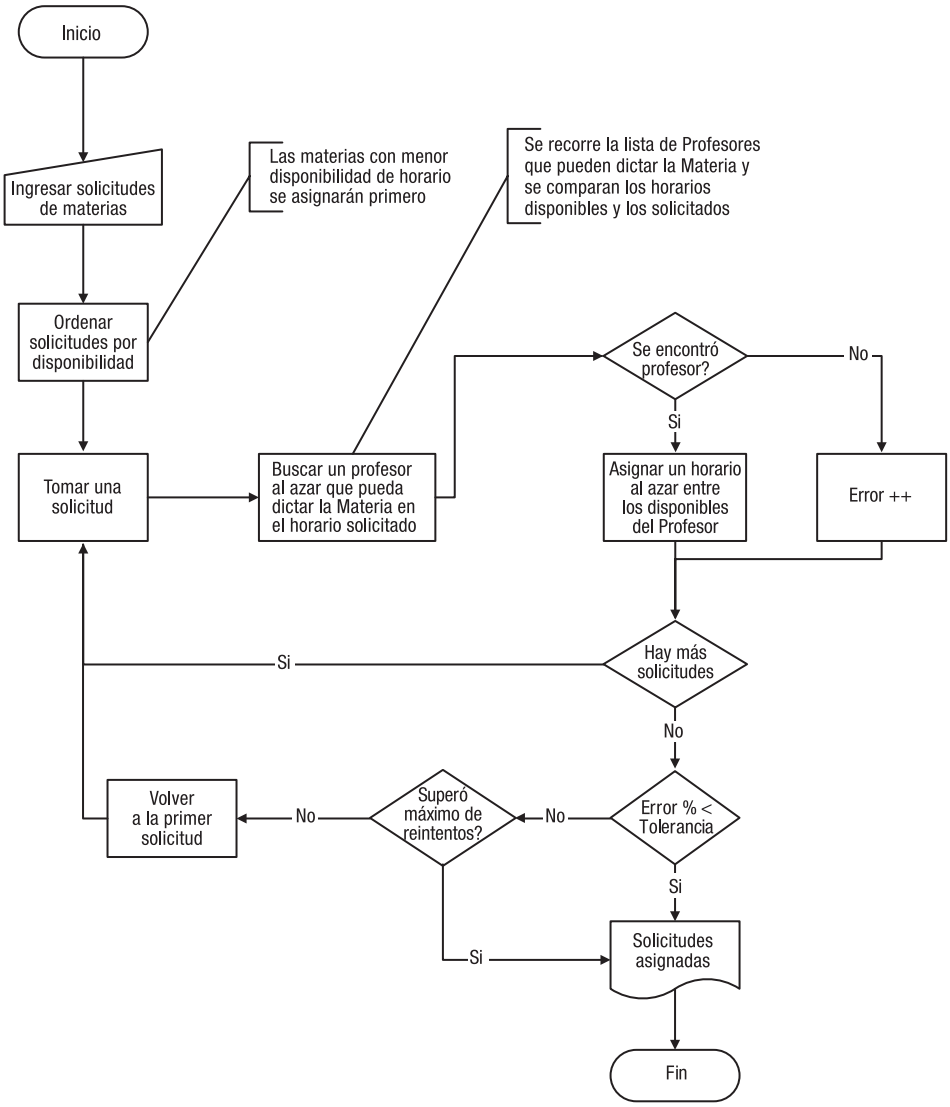

Fig. 1 | Diagrama de flujo – Algoritmo de asignación

El algoritmo planteado (figura 1) buscó reflejar de la forma más precisa posible el mecanismo intuitivo que una persona utilizaría para asignar profesores a las materias que deben ser dictadas.

A partir de una lista de solicitudes (requerimientos de las materias a dictar y los horarios solicitados) se busca primero asignar las solicitudes con menor disponibilidad de profesores y horarios. Para cada solicitud, se toma un profesor al azar entre aquellos profesores que pueden dictar esa materia y están libres en alguno

de los horarios solicitados, dando prioridad en la elección a aquellos profesores con mayor cantidad de horarios disponibles. Luego se asigna a la solicitud ese profesor junto con uno de los horarios libres (elegido al azar), y se marca en la lista de horarios del profesor ese horario como ocupado.

Si no se pudo encontrar un profesor o un horario disponible, se contabiliza la falla y se continúa con la siguiente solicitud, aplicando el mismo proceso.

Cuando no quedan más solicitudes para asignar, se evalúa si el porcentaje de errores no fue mayor a la tolerancia permitida; si fue menor se finaliza satisfactoriamente, o de lo contrario se vuelve a tomar la primera solicitud y se intenta asignar las solicitudes nuevamente.

2.3. Optimización y ajustes

Para la optimización del algoritmo planteado se decidió usar los diferentes bits de un tipo de dato entero estándar (Integer) para representar la lista de horarios disponibles, que sería utilizado tanto para las materias como para los profesores. Si el Bit de la posición que representa el horario estipulado se encuentra en 1, entonces ese horario está libre, mientras que si el valor es 0 entonces el horario no está disponible.

Tabla I: Relación de los bits que representan la lista de horarios

Bit	Horario representado
0	Lunes Mañana
1	Lunes Tarde
2	Lunes Noche
3	Martes Mañana
4	Martes Tarde
5	Martes Noche
6	Miércoles Mañana
7	Miércoles Tarde
8	Miércoles Noche
9	Jueves Mañana
10	Jueves Tarde
11	Jueves Noche
12	Viernes Mañana
13	Viernes Tarde
14	Viernes Noche
15	- No usado -

De esta forma se obtuvieron varias ventajas:

- Se eliminó la necesidad de crear un nuevo tipo de datos para representar una lista de horarios libres y ocupados.
- Se eliminó la necesidad de crear operaciones específicas en el sistema para trabajar con ese nuevo tipo de datos.
- Al valerse de los bits de un entero para representar los horarios, se pudieron utilizar las operaciones binarias AND, OR y XOR para realizar las diferentes evaluaciones y comparaciones de horarios. Como éstas operaciones se encuentran implementadas a bajo nivel en el procesador, se obtuvieron tiempos de respuesta más rápidos en todos los cálculos relacionados con las listas de horarios.
- Se eliminó el problema de conversión de la lista de horarios a un tipo de datos factible de manejar por un motor de bases de datos. Todo motor de bases de datos cuenta con un tipo de datos entero, en el cual se puede almacenar en tan solo una columna toda la información relacionada con la disponibilidad de horarios de un objeto.

En relación a los ajustes del algoritmo, se implementaron dos variables de ajuste: un límite máximo de errores (tolerancia) y un tope de reintentos de generación de asignaciones de profesores y horarios.

Un intento de generación de asignaciones se considera exitoso cuando el porcentaje de solicitudes de materias que no se pudieron asignar es menor al límite porcentual de errores permitido por el usuario. En caso contrario, la asignación falló y se hace un nuevo intento, ya que en el proceso al azar puede ocurrir que se encuentre una solución válida que cumpla con el límite establecido.

Para que el proceso no continúe infinitamente mientras no pueda hallar una solución, se estableció el tope de reintentos, el cual terminará el proceso si se alcanza el tope e informará al usuario del fracaso del algoritmo en la generación de una asignación de profesores y horarios para el conjunto de solicitudes ingresadas, con el límite de tolerancia establecido.

3. Diseño de la aplicación

Como caso de estudio para el modelado, diseño y desarrollo de la aplicación de software que implementa soluciones heurísticas para el problema de asignación de cursos, se tomó la institución académica de la Facultad de Ingeniería de la Universidad de Palermo. Su decano es el Ing. Esteban di Tada, MME.

3.1. Relevamiento de la Facultad

Los distintos departamentos que integran la Facultad de Ingeniería son:

Departamento de Tecnología de la Información: Sus áreas específicas son Programación, Lenguajes y Sistemas. Su directora es la Lic. Adriana Álvarez.

Departamento de Industrias y Servicios: Realiza la coordinación de las carreras de Ingeniería Industrial y Licenciatura en Producción Industrial tanto en los aspectos académicos como en lo que se refiere a las actividades de proyectos que realizan los alumnos. Su director es el Ing. Edgardo Tiscornia.

Departamento de Ciencias Exactas: Sus áreas específicas son Matemática, Estadística, Física, Química y Ciencias Naturales. Sus directoras son la Lic. Patricia González, MBA y la Lic. Gabriela Dussault.

Departamento de Informática Aplicada: Se dedica principalmente a la enseñanza de herramientas informáticas, cuya utilización aumenta la productividad de los profesionales de las más variadas áreas. Su coordinador es el Ing. Roberto Cerrina.

La Facultad de Ingeniería dispone de tres turnos (mañana, tarde y noche) para el dictado de clases, en los cuales se dictan las distintas materias de todas las carreras de la Facultad, en módulos únicos de 3 horas semanales. Sus horarios son:

- Turno mañana: 8:00 a 11:15 horas.
- Turno tarde: 14:00 a 17:15 horas.
- Turno noche: 19:00 a 22:15 horas.

Los cursos que dicta la Facultad de Ingeniería a sus alumnos son:

Área Informática

- Administración del Conocimiento
- Administración de Proyectos de Desarrollo
- Álgebra y Matemática Discreta
- Ambientación y Pensamiento Lógico
- Análisis de la Información y la Decisión
- Análisis de Sistemas I y II
- Análisis Matemático I, II, III y IV
- Análisis Numérico
- Aplicaciones en Internet
- Argumentación Jurídica
- Arquitectura de Computadores
- Auditoria y Seguridad en Sistemas
- Base de Datos
- Computación Aplicada

- Comunicaciones de Datos en Internet
- Data Mining
- DB2
- Derecho Aplicado a la Informática
- Diseño de Sistemas
- EJB (Enterprise Java Beans)
- Estadística I y II
- Estructura de Datos y Algoritmos
- Física I y II
- Inteligencia Artificial
- Introducción a las Comunicaciones
- Introducción a la Ingeniería del Software
- Introducción a la Programación
- Laboratorio I, II, III, IV y V
- Lenguajes Visuales I
- Management Científico I y II
- Modelos y Simulación
- Orientación a Objetos
- Planeamiento Estratégico de Sistemas
- Redes de Acceso
- Redes LAN y WAN
- Redes TCP/IP
- Seguridad en Internet
- Sistemas de Información Avanzados
- Sistemas Digitales I y II
- Sistemas Operativos
- Sistemas y Métodos
- Teoría de la Información

Área Industrias

- Álgebra
- Física A
- Matemática para Ingenieros I, II y III
- Sistemas de Representación
- Química General
- Química Industrial
- Seminario de Ingeniería
- Sistemas de Información
- Computación para Ingenieros
- Comunicaciones en Ingeniería
- Organización, Comercialización y Administración

3.2. Claustro académico (recursos humanos)

Se detallan a continuación las Autoridades y Profesores que integran el claustro académico de la Facultad de Ingeniería de la Universidad de Palermo.

Tabla II: Profesores del Depto. de Tecnología de la Información

Profesor	Materias
Alvarez, Adriana	Estructura de Datos y Algoritmos, Introducción a la Programación
Arambarri De Muir, Alejandra	Computación para Ingenieros
Arazi, Oscar	Planeamiento Estratégico de Sistemas, Sistemas de Información Avanzados
Argento, Diego	Base de Datos, Laboratorio IV
Arroyo Arzubi, Alejandro	Computación para Ingenieros, Sistemas de Información
Bolzonella, Sandra	Laboratorio III, Sistemas Operativos
Bursztyn, Rally	Base de Datos, Laboratorio II
Carriles, Oscar	Lenguajes de Scripting
Cerasulo, Paula	Introducción a la Programación
Coan, Mario	Sistemas de Información Avanzados
De María, Elio	Arquitectura de Computadores, Sistemas Digitales I y II
Ferrari, Pablo	Lenguajes Visuales I
Glinsky, Ezequiel	Laboratorio I, Orientación a Objetos, Inteligencia Artificial
Gouget, Marisa	Administración de Proyectos de Software, Análisis de Sistemas I
Lena, Pablo	Comunicaciones de datos en Internet, Trabajo Final de Grado
López, Daniela	Laboratorio I
Lozano, Cristina	Sistemas y Métodos
Martín, Ricardo	Arquitectura de Computadores
Martinez, Carlos	Sistemas y Métodos
Massino, Alejandro	Aplicaciones en Internet
Peña, Carlos Alberto	Argumentación Juridica, Derecho Aplicado a la Informática
Rey, Patricio	Auditoria y Seguridad en Sistemas
Rosenhain, Federico	Data Mining
Stein Millán, Andrés	Análisis de Sistemas II, Diseño de Sistemas, Estructura de Datos y Algoritmos
Steinberg, Beatriz	Análisis de la Información y la Decisión, Base de Datos
Taboada, Gabriel	Introducción a la Ingeniería del Software, Sistemas y Métodos
Zamozyck, Claudio	Lenguajes Visuales II, EJB

Tabla III: Profesores del Depto. de Ciencias Exactas

Profesor	Materias
Acero, Fernando	Análisis Matemático IV, Modelos y Simulación
Arnal, Patricia	Análisis Matemático I, Estadística I y II, Management Científico I
Baquero, Mariana	Estadística I y II
Bozzalla, Analía	Ambientación y Pensamiento Lógico
Cattaneo, Maricel	Análisis Matemático I, Estadística I
Coccolo, Beatriz	Análisis Matemático I y II
Diez, Stella Maris	Estadística II
Dussault, Gabriela	Análisis Matemático III, Análisis Numérico
Esperón, Gabriela	Análisis Matemático I y II, Análisis Numérico, Matemática para Ingeniería III
Fuertes, Beatriz	Ambientación y Pensamiento Lógico, Management Científico II
Garrido, Graciela	Química General, Química Industrial
Guzmán, Juan Ricardo	Análisis Matemático II
Latorre, Adriana	Análisis Matemático I y II
López, Claudia	Matemática para Ingeniería I
López, Fabio	Física A, I y II
López Sardi, Estela	Química General, Química Industrial
Riccitelli, Marina	Álgebra y Matemática Discreta
Wilder, Marcela	Álgebra, Álgebra y Matemática Discreta

Tabla IV: Profesores del Depto. de Comunicaciones

Profesor	Materias
Castillo Asenjo, Gerardo	Introducción a las Comunicaciones, Laboratorio V, Seguridad en Internet, Redes TCP/IP
Valle, Luis	Introducción a las Comunicaciones
Schmidberg, Eduardo	Ingeniería en Redes, Redes de Acceso, Redes LAN y WAN

3.3. Diseño orientado a objetos

La aplicación heurística de asignación de recursos humanos fue diseñada utilizando el paradigma de programación orientado a objetos, con lo cual se consiguió una modelización precisa del sistema a representar.

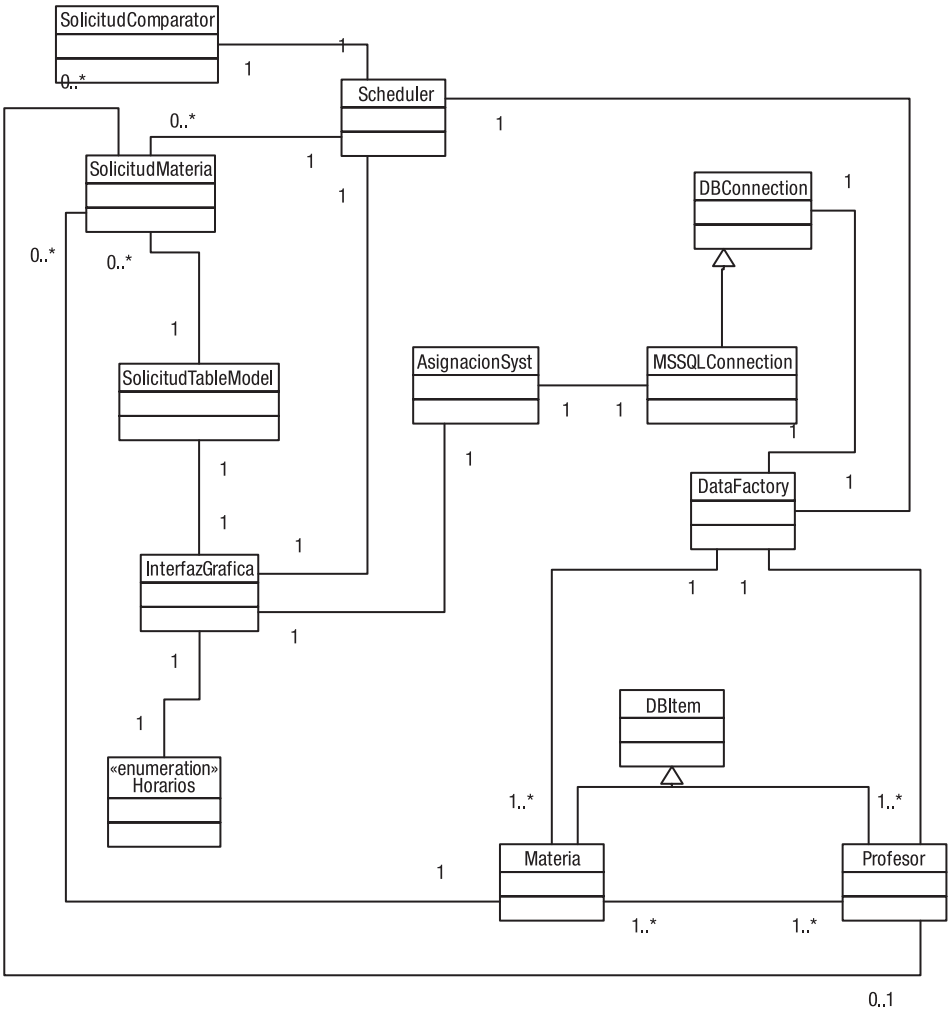

Fig. 2 | Diagrama de clases del Sistema

En la figura 2 se representa el diagrama de clases y las relaciones existentes entre los objetos modelados.

4. Desarrollo de la aplicación

Para el desarrollo de la aplicación se utilizó el lenguaje de programación Java, y el motor de bases de datos Microsoft SQL Server 2000.

Se eligió Java como lenguaje de programación porque ya había sido utilizado profundamente con anterioridad en diferentes asignaturas de la Facultad de

Ingeniería, habiendo demostrado gran versatilidad para modelar sistemas orientados a objetos. Además, Java es un lenguaje multiplataforma y se encuentra respaldado por empresas de la talla de IBM, Oracle y Sun Microsystems.

Dentro de sus librerías, Java cuenta con clases gráficas como awt y swing, las cuales permiten crear objetos gráficos comunes, altamente configurables y con una arquitectura independiente de la plataforma.

Java también permite acceder a bases de datos fácilmente con JDBC, independientemente del motor de bases de datos utilizado, y el manejo de las bases de datos es uniforme, transparente y simple.

Con respecto al motor de bases de datos, se decidió utilizar Microsoft SQL Server 2000 por varios factores, entre ellos:

- Ser un motor de bases de datos ampliamente probado.
- Tener herramientas integradas de administración, depuración y análisis.
- Contar con un lenguaje de consultas difundido y estandarizado como es SQL.
- Permitir la creación de procedimientos almacenados (Stored Procedures) que facilitan la interacción entre las aplicaciones y los datos.

A continuación se presentan los detalles del desarrollo de las interfaces gráficas, la lógica del sistema, el algoritmo heurístico y la interacción con la base de datos.

4.1. Interfaces gráficas

A partir del diagrama de clases, la interfaz gráfica del sistema se desarrolló aplicando el concepto de "MVC" (Model – View – Controler). La vista es la clase «InterfazGrafica», que recibe las entradas del usuario y envía las solicitudes al controlador «Scheduler» para realizar las acciones sobre los datos del modelo.

Así mismo, la interfaz gráfica toma los datos del modelo (las solicitudes de materias), y por medio de la clase «SolicitudTableModel» muestra la vista de los datos según los requerimientos del sistema.

En la clase «InterfazGrafica» los tres métodos más relevantes a codificar fueron:

I. cargarComboMaterias(): tomando la lista de materias previamente cargada por «DataFactory» desde la base de datos, carga el "combo box" de la interfaz gráfica para ingresar nuevas solicitudes de materias al sistema.
II. agregarSolicitud(): toma los datos ingresados por el usuario desde la interfaz gráfica y solicita a la clase «Scheduler» dar de alta una nueva "Solicitud de Materia", informando luego al usuario el resultado de la operación.
III. generarAsignacion(): Solicita al «Scheduler» generar una nueva asignación de horarios y profesores para las solicitudes existentes, notificando luego al usuario el resultado de la operación.

4.2. Lógica y algoritmos heurísticos

Se desarrollaron las clases «Scheduler», «SolicitudMateria» y «SolicitudComparator» que componen la lógica del sistema y el algoritmo heurístico de asignación de recursos humanos.

La clase «SolicitudMateria» contiene la información de la materia solicitada, el horario solicitado y si tiene o no profesor asignado. Además cuenta con un método para verificar que el horario solicitado sea válido (debe existir al menos un profesor que pueda dictar la materia en ese horario).

La clase «Scheduler» administra una colección de objetos «SolicitudMateria», y se encarga de dar de alta nuevas solicitudes, y asignar profesores y horarios aplicando el algoritmo planteado cuando el usuario lo requiera.

Por último, la clase «SolicitudComparator» implementa la interfaz Java «Comparator» para poder comparar objetos «SolicitudMateria», con el fin de poder generar ordenaciones según lo requieran otros objetos como «Scheduler».

Se explican a continuación los métodos codificados más destacados de cada clase:

«Scheduler»

- addSolicitud(): recibe por parámetros la materia y el horario solicitado, verifica que el horario requerido sea válido y si es así agrega la nueva solicitud a la lista de solicitudes de materias.
- asignarProfesor(): asigna un profesor disponible para la materia solicitada, tomado al azar entre los profesores que pueden dictar esa materia en los horarios requeridos.
- asignarHorario(): asigna un horario específico para el profesor asignado a la solicitud de materia, tomado al azar dentro de los horarios libres del profesor.
- generarSchedule(): realiza la asignación de profesores y horarios para la lista de solicitudes de materias existente en el sistema, aplicando el algoritmo heurístico definido previamente.

«SolicitudMateria»

- setProfesor(): establece el profesor asignado a la materia solicitada, removiendo el horario asignado de la lista de horarios libres del profesor, para evitar asignaciones de materias en horarios duplicados.
- isHorarioValido(): el método verifica que el horario requerido por el usuario para dictar la materia sea válido, es decir, que exista al menos un profesor que dicte esa materia y tenga disponibilidad en ese horario.

«SolicitudComparator»

- compare(): compara dos objetos «SolicitudMateria» según un campo elegido y devuelve un valor. Este valor dependerá de que los objetos sean iguales, el primero menor al segundo, o el primero mayor al segundo. Los campos posibles para realizar la comparación pueden ser "Disponiblidad", "Nombre de la Materia" o "Nombre del Profesor".

4.3. Interacción con bases de datos

En el desarrollo de la interacción del sistema con la base de datos se asumió que las tablas requeridas de la base y sus respectivos datos existen y son válidos, ya que se consideró no relevante a la investigación la codificación de módulos ABM (alta, baja y modificación) de profesores y materias, usuales en cualquier sistema de gestión.

La interacción del sistema con la base de datos se desarrolló en dos partes:

I. Desde el lado del "cliente" se codificaron las clases «DataFactory», «DBConnection», «DBItem» y «MSSQLConnection», que componen el acceso a los datos por parte del sistema.
II. Desde el lado del servidor de bases de datos SQL Server 2000 se codificaron los procedimientos almacenados «Select_Profesores», «Select_Materias», «Select_ProfesorMateria_1» y «Select_ProfesorMateria_2» para devolver las filas de las tablas según lo necesite la aplicación cliente, separando de esta forma las consultas SQL del sistema y sus clases en Java.

La clase abstracta «DBConnection» define la interfaz de acceso a los datos de la base, independientemente del motor de bases de datos que se utilice en la implementación del sistema.

La clase «MSSQLConnection» implementa los métodos definidos en «DBConnection» y realiza las llamadas correspondientes a SQL Server 2000 utilizando el driver JDBC «SQLServerDriver» desarrollado por Microsoft.

La clase abstracta «DBItem» modela los datos comunes (id, disponiblidad) de los diferentes objetos (Profesores y Materias) que se encuentran en la base de datos.

Por último, la clase «DataFactory» es la encargada de construir, a partir de los datos que devuelve una instancia de «MSSQLConnection» los objetos «Materia» y «Profesor» junto con sus relaciones asociadas.

5. Evaluación y Análisis

En este capítulo se presentan los detalles de la implementación del sistema heurístico de asignación de recursos humanos, junto con la evaluación y sus resultados.

5.1. Implementación del sistema

Para realizar la implementación del sistema se incorporaron los datos relevados oportunamente en la Facultad de Ingeniería de la Universidad de Palermo. Con la información sobre profesores y materias (junto con sus relaciones) se generaron las tablas de las bases de datos necesarias para el funcionamiento del sistema. La disponibilidad de horarios de los profesores fue asumida al azar ya que no se contaban con datos al respecto de cada uno de los profesores del claustro docente de la Facultad.

Toda esta información fue incorporada por medio del Administrador Corporativo a un servidor Microsoft SQL Server 2000, donde se creó un usuario "up/up" que tuviera permisos de acceso a datos y ejecución de procedimientos almacenados en la base de datos "Universidad".

Posteriormente se configuró el sistema desarrollado para que pueda acceder a la base de datos "Universidad" del servidor SQL Server 2000 utilizando el usuario y clave asignados.

El resultado del proceso de implementación sobre el caso real de la Facultad de Ingeniería de la Universidad de Palermo fue exitoso, ya que el sistema pudo cargar todos los datos desde el servidor SQL Server 2000, y generar los objetos para cada uno de los profesores existentes en la Facultad y para cada una de las materias que se dictan. Además en la implementación se verificó que el sistema fue capaz de mantener en todo momento las relaciones entre cada uno de los objetos «profesores» y «materias» creados en tiempo de ejecución.

5.2. Análisis de los resultados

Para analizar los resultados de la generación de asignaciones de horarios y profesores sobre una implementación real por parte del algoritmo heurístico diseñado, primero se definió un criterio de evaluación correspondiente, y luego se realizaron las pruebas necesarias para comprobar la calidad de la solución presentada por el presente trabajo al problema de asignación de recursos humanos.

Se presentan a continuación los criterios utilizados, para luego realizar el posterior análisis y evaluación de resultados correspondiente.

5.3. Criterios de evaluación

Los programas heurísticos no se justifican porque obtengan una solución óptima verificable por procedimientos analíticos, sino porque se prueba experimentalmente que son útiles en la práctica. Como la experimentación en el problema real es usualmente impráctica, se deben utilizar otros métodos de prueba más prácticos. Las alternativas posibles incluyen:

I. Comparación de los resultados del programa heurístico con aquellos de los métodos exactos.
II. Solución de problemas con soluciones conocidas (problemas previamente resueltos por métodos exactos).
III. Generación de problemas de gran tamaño con soluciones conocidas para que sirvan como casos de prueba.

Los métodos (I) y (II) son los que se encuentran en la mayoría de las publicaciones que se han revisado. Se argumenta que estos métodos no pueden ser utilizados ni para validar ni para rechazar un esquema heurístico, porque la heurística está usualmente destinada a la solución de problemas de tamaño real, mientras que los métodos exactos están casi invariablemente restringidos a problemas irrealmente pequeños. Es decir que no tiene más sentido evaluar el desempeño ("performance") de una heurística diseñada para problemas de miles de variables, contra un método exacto apropiado únicamente para problemas de unas docenas de variables.

La heurística solo puede ser juzgada analizando su efectividad en la solución de problemas reales. Como un medio de llevar a cabo esta comparación a veces es posible generar problemas grandes con soluciones conocidas y exactas.

Sin embargo, no siempre es posible generar esos problemas de grandes dimensiones, con soluciones conocidas. De modo que uno tiene que basarse en un programa heurístico por cualquiera de los métodos (I) o (II) indicados arriba o desechar la idea de utilizar un programa heurístico. Esto último no resulta apropiado si estamos ante un problema real que tiene que ser resuelto. La aceptación de una heurística se genera a partir de la "razonabilidad" del programa heurístico y se refuerza si los resultados del programa heurístico están de acuerdo (hasta cierto grado deseado) con los resultados de métodos exactos de problemas pequeños y/o si los resultados del programa heurístico son satisfactorios en pequeños problemas que han sido resueltos previamente. Dicho criterio puede ser condenado como subjetivo y anticientífico, pero no podemos esperar hasta que los teóricos desarrollen

un método verdaderamente exacto y, como ingenieros, debemos proceder con la tarea de resolver el problema con los recursos que existen.

Para el caso particular del sistema desarrollado en el presente trabajo, se decidió:

- Experimentar en el caso real de la Facultad de Ingeniería de la Universidad de Palermo, para evitar errores de evaluación al tomar casos irrealmente reducidos.
- Intentar la solución de un problema de asignación real, teniendo una solución exacta y conocida. Se tomó como solución conocida la asignación de cursos y profesores de la Facultad de Ingeniería para el primer cuatrimestre de 2006.
- Comparar los resultados del programa heurístico con la asignación de profesores, cursos y horarios conocida.

5.4. Prueba piloto y evaluación

Para el análisis del sistema se utilizó la oferta académica de la Facultad de Ingeniería para el primer cuatrimestre de 2006.

Los cursos solicitados al sistema fueron los siguientes:

- Arquitectura de Computadores: 3 cursos, mañana, tarde y noche
- Estructura de datos y algoritmos: 3 cursos, mañana, tarde y noche
- Base de Datos: 3 cursos, mañana, tarde y noche
- Análisis Matemático II: 3 cursos, mañana, tarde y noche
- Diseño de Sistemas: 1 curso, noche
- Orientación a objetos: 1 curso, noche
- Laboratorio I: 3 cursos, mañana, tarde y noche
- Estadística I: 3 cursos, mañana, tarde y noche
- Sistemas Operativos: 3 cursos, mañana, tarde y noche
- Análisis de la Información y la Decisión: 2 cursos, noche
- Laboratorio II: 3 cursos, mañana, tarde y noche
- Análisis de Sistemas I: 2 cursos, mañana y noche
- Laboratorio III: 2 cursos, mañana y noche
- Introducción a las Comunicaciones: 2 cursos, mañana y noche
- Estadística II: 3 cursos, mañana, tarde y noche
- Álgebra y Matemática Discreta: 3 cursos, mañana, tarde y noche
- Análisis de Sistemas II: 1 curso, noche
- Auditoria y Seguridad en Sistemas: 1 curso, noche
- Administración de Proyectos del Software: 1 curso, noche
- Management Científico I: 1 curso, noche
- Fisica I: 1 curso, noche

- Laboratorio IV: 1 curso, noche
- Management Científico II: 1 curso, noche
- Análisis Matemático III: 1 curso, noche
- Sistemas Digitales I: 1 curso, noche
- Sistemas Digitales II: 1 curso, noche
- Laboratorio V: 1 curso, noche
- Planeamiento Estratégico de Sistemas: 1 curso, noche
- Seguridad en Internet: 1 curso, noche
- Sistemas de Información Avanzados: 1 curso, noche
- Computación Aplicada: 2 cursos, mañana y noche
- Introducción de Programación: 4 cursos, mañana, tarde y noche
- Introducción a la Ingeniería del Software: 3 cursos, mañana, tarde y noche
- Lenguajes Visuales II (ASP.Net): 1 curso, noche
- Lenguajes Visuales I (Visual Basic.Net): 1 curso
- Recursos Informáticos II: 1 curso, noche
- Ingeniería en Redes: 1 curso, noche
- Sistemas y Métodos: 3 cursos, mañana, tarde y noche
- Ambientación y Pensamiento Lógico: 3 cursos, mañana, tarde y noche
- Análisis Matemático I: 4 cursos, mañana, tarde y noche
- Derecho Aplicado a la Informática: 3 cursos, mañana, tarde y noche

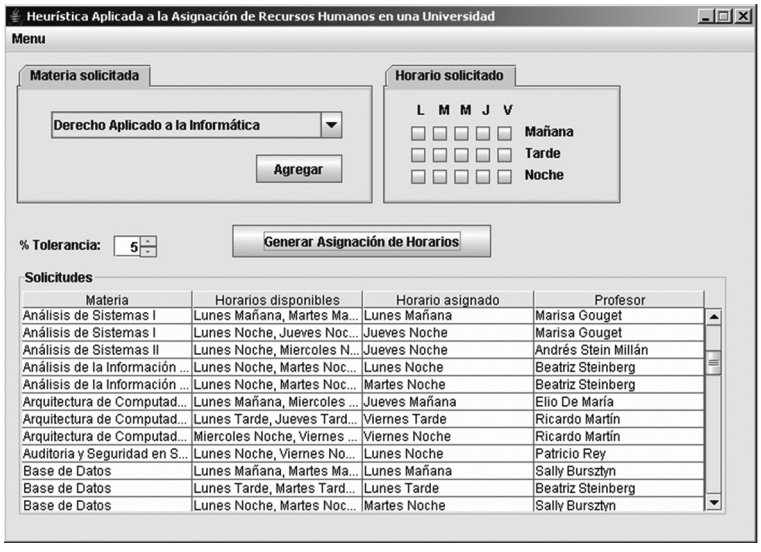

Fig. 3 | Sistema de Asignación de Recursos Humanos

La evaluación del algoritmo fue completamente satisfactoria en relación a los niveles de aspiración que se pretendieron alcanzar en el desarrollo del algoritmo heurístico de asignación de recursos humanos.

En primer lugar, el algoritmo se demostró capaz de trabajar en un entorno real, con mayores volúmenes de datos que los que se pueden utilizar en pruebas de desarrollo. En el caso concreto de la oferta académica del primer cuatrimestre de 2006, el algoritmo tuvo que generar la asignación de profesores y horarios para un total de 79 cursos.

En segundo lugar, el tiempo de respuesta siempre fue instantáneo, aún en la última prueba de asignación de 79 cursos para la Facultad de Ingeniería.

En tercer lugar, y como factor más importante, el algoritmo logró dar una solución valida y satisfactoria a partir de las solicitudes ingresadas, ya que fue capaz de asignar 78 de los 79 cursos solicitados. Los dos puntos anteriores se volverían intrascendentes si el algoritmo hubiera sido incapaz de realizar su tarea de asignación de profesores y horarios a los cursos solicitados.

Como detalle final, en el caso de evaluación anterior el límite de tolerancia también demostró su utilidad: la única falla de asignación (1/79) fue menor que el 5% permitido, con lo cual el algoritmo asumió que la solución encontrada era válida. La falla de asignación ocurre por simple lógica, ya que se solicitaron "Sistemas Digitales I" y "Sistemas Digitales II" ambas por la noche, y dado que el único profesor capaz de dictar esas dos materias (Elio De María) solo tenía disponible el miércoles por la noche, una de las dos materias quedó indefectiblemente sin asignarse.

6. Conclusión

En el transcurso del presente trabajo se lograron alcanzar los tres objetivos planteados al inicio de la investigación.

En primer lugar, se investigaron diferentes algoritmos heurísticos existentes, buscando definir un algoritmo que permitiera obtener soluciones heurísticas para el problema particular de la asignación de recursos humanos en una universidad. A partir de las diferentes alternativas estudiadas, se optó por definir un algoritmo heurístico nuevo que resolviera específicamente el problema planteado, sin tener que adaptar la realidad existente a un algoritmo y modelo preestablecido.

Como segundo objetivo específico, a partir del algoritmo y modelo definidos por el autor, se desarrolló una aplicación de software que implementó la solución planteada, incluyendo las interfaces con el usuario, la interacción con

los orígenes de datos y la presentación de resultados. La aplicación de software así obtenida cumplió los lineamientos establecidos con respecto a los casos de uso, las interfaces gráficas, los menús, pantallas, informes y reportes, el acceso a los datos almacenados en la base de datos y a la implementación y codificación del algoritmo heurístico propuesto.

El último objetivo se alcanzó durante la evaluación del desempeño de la aplicación de software codificada. A partir de la implementación de la aplicación de software desarrollada por el autor, se realizó la evaluación de los resultados obtenidos y se pudo comprobar la validez de las soluciones encontradas por el algoritmo heurístico propuesto.

Como conclusión final, se destaca que el sistema propuesto demostró su utilidad concreta al encontrar buenas soluciones heurísticas en un problema de asignación de recursos humanos tomado del mundo real.

7. Futuras Líneas de Investigación

A partir de la investigación desarrollada en el presente trabajo, el autor plantea las siguientes líneas de investigación futuras:

- La extensión del algoritmo y el sistema para analizar y administrar el cupo de aulas disponibles en cada turno y las sedes del establecimiento educativo.
- La optimización del algoritmo para distribuir más uniformemente en la semana los días asignados según los requerimientos del usuario.
- El desarrollo de un sistema que permita la asignación diaria de aulas a partir de las materias que se deben dictar ese día, los alumnos inscriptos en cada curso y la capacidad de alumnos que posee cada aula del edificio.
- La adaptación del algoritmo para colaborar con los alumnos en la selección de materias a cursar durante la inscripción de cada ciclo lectivo. El sistema permitiría idealmente que un alumno seleccione las materias que tiene interés en cursar en ese ciclo, y generaría diferentes alternativas de cursos, profesores y horarios combinando las materias elegidas por el alumno.

Referencias

[1] Grossman D. (Diciembre 2004) – Information Retrieval: Algorithms and Heuristics, 2º edición – Ed. Springer

[2] Jaworski J. (2002) – Java 1.2 al descubierto – Editorial Prentice Hall

[3] Michalewicz Z., Fogel D. (Abril 2005) – How to Solve It: Modern Heuristics, 2º edición – Ed. Springer

[4] Papadimitriou C., Steiglitz K. (Julio 1998) – Combinatorial Optimization: Algorithms and Complexity – Ed. Dover Publications

[5] Riel A. (Abril 1996) – Object Oriented Design Heuristics, 1º edición – Ed. Addison-Wesley Professional

[6] "Heuristic Algorithms", Gaston H. Gonnet. http://www.inf.ethz.ch/personal/gonnet/CAII/HeuristicAlgorithms/

[7] "TSP Algorithms in Action", Stephan Mertens. http://www-e.uni-magdeburg.de/mertens/TSP/index.html