Universidad de las Ciencias Informáticas

Facultad 15

Título: Implementación de una Biblioteca de Algoritmos para la generación automática de horarios de la aplicación web Ghordo.

Trabajo de Diploma para optar por el título de
Ingeniero en Ciencias Informáticas

Autores: Isabel González Flores

Humberto Cardoza Rodriguez

Tutores: Lic. Rolan Rober Bullain Dieguez

Lic. Bolívar Ernesto Medrano Broche

Ciudad de la Habana, Junio del 2010.

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año _____.

Isabel González Flores

_____

Firma del Autor

Humberto Cardoza Rodriguez

_____

Firma del Autor

Lic. Rolan Rober Bullain Dieguez.

_____

Firma del Tutor

Lic. Bolívar Ernesto Medrano Broche.

_____

Firma del Tutor

"Emplearse en lo estéril cuando se puede hacer lo útil, dedicarse a lo fácil cuando se tienen bríos para intentar lo difícil, es despojar de su dignidad al talento. Todo el que deja de hacer lo que es capaz de hacer, peca."

José Martí.

AGRADECIMIENTOS

De Humberto:

No existen palabras para agradecer a todos con los que he compartido en el transcurso de estos cinco años, desafortunadamente no tengo espacio para mencionarlos a todos pero quiero que sepan que estoy muy agradecido, en especial:

A mis padres: Tania y Humberto y a mis abuelos: Miriam y Manuel Amador, por estar siempre a mi lado, por guiarme por el buen camino, por darme los mejores consejos y la mejor educación, por su apoyo en todo momento, por confiar en mi, por dar tanto de ellos y sacrificar tanto por verme hoy en día graduado, por esas y infinitas razones los quiero mucho.

A mis abuelos América y Héctor por sus consejos y por todo lo bueno que aprendí de ellos y a mi abuelo Ilario que no esta presente pero se que estaría muy orgulloso de mi en estos momentos.

A mis tías, en especial a mi tía Caridad y a mis tíos de manera general, ya que todos me brindaron su apoyo incondicional en todo momento, estuvieron siempre para darme sus consejos, para ayudarme a salir adelante, por demostrarme que siempre voy a poder contar con cada uno de ellos, los quiero mucho a todos.

A mis hermanos por estar presentes en todo momento.

A mi esposa Ceila y a mi hijo querido Álvaro, qué seria de mi sin ustedes, los quiero mucho. A ti, mi amor, te agradezco tus consejos, tu compañía, tu dedicación, tu apoyo, tu amor, por estar cuando más lo he necesitado, por quererme como me quieres. Gracias por esperar por mí estos cuatro años que hemos estado separados, espero no tener que separarme de ti ni de nuestro hijo nunca más. A ti mi hijo querido espero enseñarte, educarte, quererte, apoyarte y darte todo lo que a mi me han dado, a los dos los quiero mucho.

A la familia de mi esposa por apoyarme en todo momento y confiar siempre en mí.

A mi familia en general.

A mis amigos y amigas en general, con todos con los que he compartido estos cinco años, les agradezco su amistad y acuérdense que no importa donde estemos siempre me acordaré de ustedes.

A mis tutores por todo el apoyo que me brindaron y la confianza que depositaron en mí.

A mi facultad, a mis profesores, a mi decana Yvonne por estar presente y brindarme su ayuda en todo momento desde que nació mi niño.

A la Universidad y a la Revolución por darme todo lo que soy hoy en día.

AGRADECIMIENTOS

De Isabel:

Quisiera agradecer a tantas personas que me han ayudado durante los cinco años de la carrera, e incluso antes de entrar a la UCI, que resulta imposible mencionarlos uno por uno, a pesar de que se lo merecen. Agradezco:

A las dos personas que más quiero en esta vida: mis padres, por estar siempre a mi lado a pesar de la distancia, por el apoyo incondicional, por el consejo oportuno, por el aliento y la esperanza que siempre me dan, incluso en los momentos más difíciles, por ser fuente de mi inspiración y mis ganas de salir adelante. Si no fuera por ustedes no hubiese podido llegar a escribir estas líneas y razones sobran para expresar lo orgullosa que estoy de tenerlos en mi vida.

A mi hermanito, fuente inagotable de alegría e ingenio, yo termino este año en la UCI y tú apenas vas a comenzar, te deseo muchos éxitos y espero que puedas aprovechar tu tiempo como estudiante, esta es la mejor etapa de nuestras vidas.

A mi abuela Belkis, mi abuelo, mis tíos y tías, mis primos y primas… A toda mi familia que siempre ha estado pendiente de mí y que constituyen mi apoyo, mi fuerza y mi deseo de ser mejor cada día.

A mi gente buena de Aguilera que tanto confiaron en mí y que me brindaron esperanzas y apoyo en todo momento.

A mis vecinos del Guaso, que tanto preguntan y se preocupan por mí.

A mis tutores, gracias por la dedicación, apoyo y la confianza depositada.

A mis amigos, mis compañeros de estudio y aventura, gracias por su compañía.

A Omarito, por ayudarme incondicionalmente a entrar, mantenerme y terminar exitosamente la carrera.

A mi novio, que me dio fuerzas cuando más la necesitaba y por su cariño… y por tener una familia maravillosa y generosa que me ayudó más de lo que imaginé.

A mis profesores y a todos los que estuvieron a mi lado de una u otra forma.

A la Revolución, a nuestro comandante Fidel y a la UCI por darme la oportunidad de formarme como profesional.

DEDICATORIA

De Humberto:

A mis padres.

A mis abuelos.

A mis tías y a mis tíos.

A mis hermanos.

A mi esposa.

A mi hijo.

De Isabel:

A mis padres.

A mi hermano y mi familia.

RESUMEN

En este trabajo se propone la implementación de una biblioteca de algoritmos para la generación automática del horario docente correspondiente a la Facultad 15 de la Universidad de las Ciencias Informáticas. Surge a partir de la necesidad de automatizar el proceso de planificación docente que actualmente se realiza de forma manual y no se ajusta al dinamismo de la universidad. La misma constituirá una nueva funcionalidad al ser integrada, en un futuro cercano, a la aplicación web Gestor de Horario Docente (Ghordo).

Se realiza la investigación y construcción de heurísticas para la obtención de diferentes horarios factibles, así como la optimización de los resultados iniciales a través de la metaheurística implementada. Generándose, finalmente, el horario docente de acuerdo a las restricciones y condiciones de la Facultad 15, teniendo en cuenta la variabilidad de los recursos humanos y de infraestructura, permitiendo reducir el tiempo y los recursos invertidos en la realización de este complejo proceso.

PALABRAS CLAVE

Biblioteca de algoritmos, horario, planificación, heurística, metaheurística.

TABLA DE CONTENIDO

TABLA DE CONTENIDO

TABLA DE CONTENIDO

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Desde el inicio de la humanidad el hombre se ha propuesto hacer un uso racional y eficiente de los recursos y especialmente del tiempo. Existen distintos mecanismos de planificación entre los que toman un gran predominio los planes de intervalos cortos, tiempo durante el cual se desarrolla habitual o regularmente una acción o se realiza una actividad, también conocido como horario (Real Academia Española, 2010).

En la arista social todos los elementos están íntimamente relacionados a la planificación pero indiscutiblemente el proceso docente-educativo es uno de los primeros beneficiados, siendo la planificación del horario docente un proceso importante.

Toda institución educativa debe poseer suficiente cantidad de recursos en infraestructura y medios educativos que le permita impartir una educación de alta calidad. Estos recursos deberán ser lo suficientemente planificados y estudiados, a fin de que no se presente escasez o sobreoferta de ellos, de lo contrario ocasionaría el deterioro en la calidad de la educación. El deterioro de la educación también se pone de manifiesto cuando la planificación del proceso docente no es del agrado de los profesores y alumnos implicados en ella (Franco Baquero, y otros, 2008).

La literatura científica recoge con el nombre de *Timetabling Scheduling* a la variedad de problemas de Optimización Combinatoria que sirven para modelar de forma general los procesos organizativos de los centros de enseñanza. La complejidad de los mismos es elevada, lo que ocasiona que muchos de los trabajos en el tema se desarrollan tomando datos y problemas creados artificialmente (Murray, y otros, 2007).

El problema de Asignación de Horario (PAH) consiste básicamente en asignar una serie de actividades o eventos en períodos de tiempo, bajo ciertas restricciones, de manera tal que se satisfagan lo mejor posible un conjunto de objetivos (Wren, A., 1996).

Existe un considerable cuerpo de desarrollo de este campo, durante los últimos años muchos son los trabajos que se reportan sobre el tema desde el punto de vista teórico y práctico. Siendo el origen de un importante grupo de trabajo: *Working Group on Automated Timetabling (WATT)*, de la Universidad de Nottingham, la cual auspicia una conferencia bianual asociada: *Practice and Theory of Automated Timetabling (PATAT)*, donde investigadores ponen a prueba sus algoritmos para resolver problemas de prueba.

La Universidad de las Ciencias Informáticas (UCI), centro donde se desarrolla la investigación, desde sus orígenes ha sido atípica tanto estructural como funcional, dinámica, debido a su participación activa en las actividades políticas, culturales y recreativas convocadas tanto por la Revolución como por la propia universidad, requiere un elevado número de profesores y personal calificado para cubrir las materias del plan docente, influyendo en la necesidad de recurrir a

profesores externos, adjuntos y de prestación de servicio al centro, que pueden estar vinculados o no a la producción. Actualmente el número de locales para llevar a cabo el proceso docente es insuficiente, dificultando la planificación. En este centro los procesos docente y productivo están estrechamente relacionados, provocando la participación de gran cantidad de recursos humanos en dos procesos diferentes de forma simultánea. Para que la calidad de los mismos no se vea afectada debe existir gran sincronización. Establecer niveles aceptables de sincronización, requiere de una planificación flexible y eficiente de las actividades docentes debido a que el proceso productivo se integra con el docente.

Durante la planificación docente se manejan grandes volúmenes de información y son numerosas las restricciones a cumplir. Lo anterior implica que el proceso de planificación del horario docente sea complejo, susceptible a errores, vulnerable a cambios de última hora y su confección de forma manual se torna en ocasiones impracticable. Varias facultades han realizado intentos para resolver o atenuar este problema, como el caso de la Facultad 7 que actualmente recurre al uso del la herramienta Microsoft Office Excel 2007 a la cual se le añadió la funcionalidad de evitar las colisiones en el horario.

Las Facultades 9 y 10 han avanzado más en materia de informatización, disponen de un asistente de planificación con dos componentes fundamentales: HorrPlanner, herramienta de escritorio para la creación del horario, que contribuye a evitar las colisiones en el horario, y HorrPublisher, aplicación web para la publicación.

La Facultad 15 cuenta con una aplicación web: Gestor de Horario Docente (Ghordo), creado con el propósito de informatizar y elevar la calidad del proceso de planificación del horario docente, sin embargo, al igual que el empleado en las Facultades 9 y 10, constituye un asistente para la planificación del horario docente, por lo tanto el proceso continúa siendo engorroso.

Existen otras facultades, en las que el personal encargado de acometer esta tarea la ejecuta de forma manual. A raíz de lo antes planteado se evidencia la **situación problemática** de la investigación.

Definiéndose el siguiente **problema**: La complejidad en la confección manual del horario docente provoca que el mismo sea susceptible a errores, rígido ante posibles cambios y en ocasiones se torna impracticable.

Teniendo como **objeto de estudio** la gestión de la planificación de horarios el **campo de acción** está enmarcado a la implementación de algoritmos para la generación de horarios docente.

Para darle solución al problema planteado se define como **objetivo general** implementar una Biblioteca de Algoritmos que pueda ser integrada a la aplicación web Ghordo y permita la generación automática del horario docente.

Para dar cumplimiento al objetivo general planteado se proponen los siguientes **objetivos específicos**:

- Modelar el Problema de Asignación de Horarios Docentes de la Facultad 15.
- Diseñar heurísticas que permitan resolver el Problema de Asignación de Horarios de Clases.
- Implementar los algoritmos heurísticos diseñados.
- Validar la biblioteca.

La **hipótesis** que guía la investigación se define como: si se elabora una Biblioteca de Algoritmos que pueda ser integrada a la aplicación web Ghordo que permita la generación automática del horario docente, entonces se perfeccionará el proceso de planificación del horario docente.

El trabajo se encuentra estructurado en tres capítulos:

- **Capítulo 1: Fundamentación Teórica**, se exponen los principales conceptos, enfoques y técnicas de solución aplicadas a este tipo de problema, así como las metodologías, lenguajes de programación y herramientas convenientes para la solución, seleccionando las más apropiadas y argumentando el por qué de su utilización.
- **Capítulo 2: Solución propuesta**, a partir de las características propias del problema a resolver, se abordan los principales aspectos que describen la solución propuesta, las restricciones y la estructura de la Biblioteca de Algoritmos.
- **Capítulo 3: Análisis de los resultados**, se muestran los casos de prueba realizados, los resultados obtenidos y se realiza el estudio de factibilidad económica, en el que se determina si es factible o no el desarrollo de la Biblioteca de Algoritmos.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

En este capítulo se abordan los conceptos indispensables para la compresión del presente trabajo. Se realiza una revisión del estado del arte sobre los enfoques de modelación, técnicas de solución y algunas aplicaciones reales del Problema de Asignación de Horarios de Clases (PAHC). Finalmente se realiza una valoración de las metodologías de desarrollo de software, los lenguajes de programación y las herramientas existentes, con el objetivo de seleccionar lo más adecuado para la solución del problema.

1.2. Definición de términos

A continuación se definen una serie de términos utilizados en la Universidad de las Ciencias Informáticas (UCI), los mismos presentan especial importancia para la comprensión de la naturaleza del PAHC.

Curso: Período de estudios de una especialidad, enmarcado en un tiempo de duración descrito en meses (aproximadamente 10 meses).

Asignatura: Materia o disciplina desarrollada en actividades docentes que pertenece a un año específico de la carrera. Existen asignaturas optativas que son planificadas durante el semestre y tienen una determinada frecuencia semanal.

Actividad: La actividad (turno de clase) es impartida por el profesor y es referente a una asignatura, y puede ser: conferencia, clase práctica, clase teórica, clase teórico-práctica, seminario, taller o laboratorio.

Grupo Docente: Estudiantes que pertenecen al mismo año e intervienen en las mismas actividades.

Grupo Docente Mixto: Grupo de estudiantes que cursan la misma asignatura optativa.

Local: Sala destinada a desarrollar una actividad docente. Las aulas tienen determinada capacidad, según su denominación (Salón de Conferencias, capacidad aproximada de 90 estudiantes, Aula, capacidad de 30).

Laboratorio: Local dotado de computadoras personales utilizado para impartir actividades de laboratorio de las asignaturas que lo requieran, con capacidad de 30 estudiantes.

Turno: Espacio de tiempo que demora una clase, 90 minutos.

Sesión: Espacio de tiempo compuesto por tres turnos y sus respectivos recesos. El día se divide en dos sesiones, mañana y tarde.

Semanas: Marco de tiempo compuesto por seis turnos, distribuidos de lunes a sábado, se clasifican en: par e impar. Tomando como referencia el inicio del curso.

Semestre: Un año se divide en dos semestres. Un semestre tendrá aproximadamente 20 semanas.

Profesor: Encargado de dictar las actividades. Son agrupados en: internos, externos, prestación de servicios y pueden estar vinculados o no a la producción.

- Internos: profesores con residencia en la Universidad.
- Externos: profesores que residen fuera de la Universidad.
- Vinculados a la producción: pertenecen a un grupo productivo.

1.3. Elementos del Problema de Asignación de Horarios

Los horarios están presentes en muchas áreas de la vida del hombre, se utilizan en eventos deportivos *(Scheduling and Timetabling in Sport Tournaments)*, estaciones de trenes *(Train Timetabling)*, aeropuertos *(Aircraft Routing and Scheduling)*, en la educación *(Course Timetabling y Examination Timetabling)*, entre otras. Estos ejemplos conforman una de las variedades de los bien conocidos en la literatura científica como *Scheduling Problem*.

El Problema de Asignación de Horarios de Clase (PAHC) se define como la búsqueda de una secuencia fija de encuentros, en locales, entre profesores y estudiantes en un prefijado período de tiempo satisfaciendo varios tipos de restricciones. Las restricciones se clasifican en dos grupos, restricciones fuertes y débiles (Schaerf, 1999):

- **Restricciones fuertes *(Hard constraints)*:** Son de estricto cumplimiento, definen la factibilidad o validez del horario.
- **Restricciones débiles *(Soft constraints)*:** No son de estricto cumplimiento, pero se deben satisfacer lo mejor posible. El grado de satisfacción de estas define la calidad del horario.

Las restricciones y los criterios que determinan si un horario es mejor que otro, son factores que varían considerablemente dependiendo de los requerimientos específicos de cada institución, razón por la cual se hace muy difícil diseñar una solución que se adapte a todas las instancias (B. Cooper, y otros, 1995). Motivo por el cual no se pueden asimilar otros sistemas de probada calidad que han sido desarrollados para modelos educacionales diferentes al nuestro.

1.3.1. Complejidad del Problema de Asignación de Horarios.

Este tipo de problema pertenece a la categoría de problemas NP-Completos, en los cuales existe una proporción directa entre el tiempo y la calidad de la solución, lo cual impulsa a utilizar métodos de búsqueda que entreguen una solución cercana al óptimo en un tiempo adecuado, ya que, existe la posibilidad de no encontrar la solución óptima global; por lo tanto, una solución exacta sólo se puede lograr para pequeñas instancias del problema (Schaerf, 1999) (Araya, 2007) (Astaiza A., 2005).

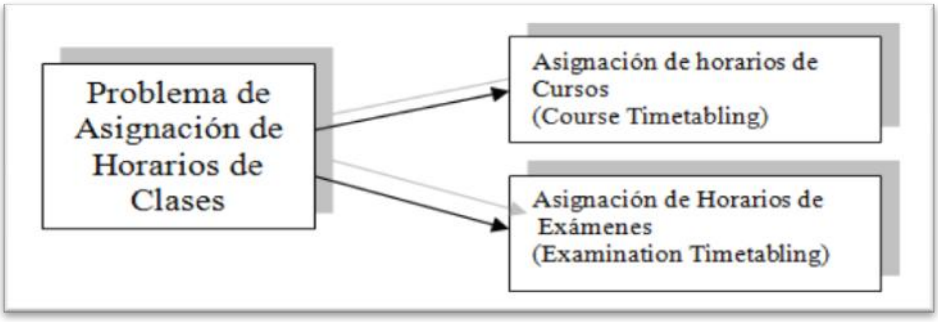
1.3.2. Variantes

Diferentes variantes han sido propuestas en la literatura, las cuales difieren del tipo de evento, institución involucrada, y de sus restricciones. De acuerdo a esto se pueden identificar tres grupos, los cuales se describen a continuación (Hernández, y otros, 2008):

- Programación de horarios de evaluaciones y exámenes *(Examination Timetabling)*.
- Programación de horarios de clases para colegios *(School Course Timetabling)*.
- Programación de horarios de clases para instituciones de educación superior o universidades *(University Course Timetabling)*.

La programación de horarios de clases puede ser dividida en dos categorías principales: calendario de exámenes y horarios de curso.

Figura 1. Clasificación del Problema de Asignación de Horarios de Clases.



Fuente: Elaboración propia

El gran número de eventos (cursos o exámenes) a ser programados y la gran variedad de restricciones impuestas a los horarios hace que el conjunto de todas las posibles soluciones, (es decir, el espacio de búsqueda del problema) sea muy grande. La construcción del horario docente es una tarea extremadamente difícil y su solución manual requiere mucho esfuerzo. Este tipo de problema ha atraído la atención de la comunidad científica de una serie de disciplinas como la

Inteligencia Artificial (K. Burke, y otros, 2002). El presente documento hace énfasis en el PAHC para la Universidad de las Ciencias Informáticas (UCI), enfocado a horarios de curso.

El desafío en la construcción de estos horarios radica en proponer una planificación que cumpla con todas las restricciones impuestas y obtenga una solución de calidad aceptable en un tiempo razonable. Conseguir este propósito no es fácil, por la cantidad de restricciones impuestas a estos problemas se convierten en difíciles de resolver y pueden ser extremadamente consumidores de tiempo (Araya, 2007).

1.4. Estudio de Estado del Arte

Existen varios artículos que recogen el quehacer científico desde el punto de vista teórico y dan una amplia panorámica en cuanto a Problemas de Asignación de Horario (PAH) se refiere, (Shaerf, 1999), (Burke, 1997) y más reciente encontramos a (Lewis, 2007).

Son varios los enfoques de solución propuestos en la literatura, Welsh y Powell, en 1967, señalaron la similitud entre este problema y el de coloración de un grafo, donde los vértices corresponden a los cursos, y los arcos entre ellos representan los conflictos (Astaiza A., 2005). En "*Automated University Timetabling: The State of the Art"*, de E. Burke, 1997, se reporta la utilización de este enfoque por G. A. Neufeld, *"Graph coloring conditions for the existence of solutions to the timetable problem"* en 1974, D. De Werra, "*An introduction to timetabling",* en 1985 y M. W. Carter, *"A Survey of Practical Applications of Examination Timetabling Algorithms"*, en 1986. En las décadas de los 70, 80 e inicio de los 90 aparecen trabajos que resolvieron el problema mediante métodos de programación matemática. Es válido resaltar que el enfoque de programación matemática se vio limitado por la capacidad de cómputo, pero ya en el 2000 la aparición de las librerías IPLOG CLEX han permitido que el enfoque vuelva a cobrar fuerza, pues esta herramienta permite resolver instancias con gran número de variables. Inicialmente esta librería estaba bajo licencia GNU pero recientemente fue adquirida por la IBM.

Los enfoques heurísticos y el uso de metaheurísticas son los que más han proliferado, por su versatilidad y eficiencia probada. La mayoría de las metaheurísticas han sido utilizadas para solucionar el Problema de Asignación de Horarios. En la conferencia bianual *(Practice and Theory of Automated Timetabling (PATAT)*[1]*)*, se realiza una competición de algoritmos clásicos e híbridos para resolver problemas de pruebas generados artificialmente. Por la gran cantidad de trabajos existentes sobre este enfoque sólo nos remitimos a citar (Lewis, 2007), donde se realiza un estudio pormenorizado de los resultados obtenidos por la comunidad científica en el uso de las metaheurísticas en los problemas de asignación de horarios.

[1] http://www.asap.cs.nott.ac.uk/patat/patat-index.shtml

La programación basada en restricciones es otro enfoque de solución para el problema, que consiste en resolver problemas mediante la declaración de restricciones sobre el área del problema y consecuentemente encontrar soluciones que satisfagan todas las restricciones que optimicen unos criterios determinados (Barber, y otros, 2003). Los enfoques más novedosos son los basados en Hiperheurísticas y Sistemas de Agentes (Burke, y otros, 2004).

En el mundo existen varios sistemas para confección de horarios basados en tecnología libre como FET y UniTime, los cuales podrían ser asimilados, pero las transformaciones que se deben hacer para adaptarlos a las necesidades de la UCI serían muy engorrosas.

En nuestro país se han realizado pocos trabajos en esta materia, se tiene la referencia de (Echevarría Cartaya, y otros, 2010) los cuales aplican Búsqueda Tabú para solucionar el PAHC de la facultad de informática de la Universidad de Cienfuegos.

1.5.Formulación del problema

El PAHC puede ser formulado como un Problema de Búsqueda con el objetivo de encontrar una solución factible que satisfaga las restricciones impuestas. En otros casos se puede enunciar como un Problema de Optimización Combinatoria (POC), que consiste en encontrar un horario que cumpla estrictamente las restricciones fuertes y minimice, o maximice, una función objetivo, la cual está relacionada a las restricciones débiles, para luego realizar un conjunto de transformaciones con el fin de obtener una solución factible mejor que la anterior.

1.6. Enfoques de solución

El PAHC se puede resolver aplicando diversos enfoques de solución, tanto los ofrecidos por el área de Investigación de Operaciones como por los de Inteligencia Artificial (Araya, 2007), en la literatura especializada ambos enfoques tienen tanto adeptos como detractores.

Cuando se resuelven problemas para situaciones reales hay que tener en cuenta los efectos indeseables que puedan causar a la usabilidad del sistema, la configuración ideal de los parámetros de la metaheurísticas usadas. Al respecto Edmund Burke (Burke, 2004) comenta:

“Una de las mayores desventajas de muchas metaheurísticas aplicadas a problemas de Scheduling (Timetabling) es que pueden ser muy dependientes a rangos específicos de sus parámetros. La efectividad de los distintos enfoques puede ser muy sensible a la configuración de los parámetros, al igual que determinar una correcta configuración de los mismos puede causar dificultades significativas. Estas dificultades son exacerbadas si los parámetros son configurados por personas inexpertas en metaheurísticas. Por estos motivos, es conveniente investigar enfoques que no sean

tan sensibles a los parámetros, no sólo reduciendo la cantidad de parámetros sino que los incluidos deben ser de un fácil análisis y comprensibles por usuarios inexpertos. Por ejemplo, un planificador docente estaría en serios aprietos si tuviera que conocer el esquema de enfriamiento del Recocido Simulado o la probabilidad de mutación de un algoritmo genético."

Siguiendo las instrucciones de Burke (Burke, y otros, 2004) se decide el uso del algoritmo metaheurístico Gran Diluvio *(Great Deluge)*, inspirado en la búsqueda local.

El Algoritmo del Gran Diluvio, es un algoritmo de búsqueda local que es aplicado a la solución de problemas de optimización propuesto por (Dueck, 1993). Comparte características similares con los algoritmos escaladores de colina y el Recocido Simulado. Su nombre se debe a la analogía con la situación de una persona en medio de un gran diluvio, la cual trata de moverse en cualquier dirección donde el nivel del agua que aumenta no lo sumerja. El algoritmo queda descrito a continuación.

Generar una solución inicial ho

Calcular en nivel inicial L = FO(h) (Evaluación de la función objetivo (FO) para la solución inicial)

Calcular la razón de asenso r (r = (FO (ho) – FO*) / m* donde FO* es el valor deseable la FO y m* es el número de movimientos deseables (Iteraciones))

Mientras que sea posible mejorar FO

Definir la vecindad N(h)

Seleccionar aleatoriamente de N(h) un candidato a solución h'

Calcular FO(h')

Si FO(h') < FO(h) Aceptar h'

Entonces si FO(h') <= L aceptar h'

Bajar el nivel L = L - r

1.7. Técnicas de solución

Las técnicas empleadas para darle solución a este tipo de problemas se clasifican en dos grandes grupos (Araya, 2007):

- Métodos Exactos o Métodos Completos: recorren todo el espacio de soluciones.
- Métodos Heurísticos o Métodos Incompletos: no encuentran todas las soluciones posibles a un problema. Son utilizadas principalmente en el campo de la Inteligencia Artificial para

problemas de alta complejidad, siendo una de las más conocidas las metaheurísticas, que consisten en procesos iterativos que guían y modifican las operaciones de los métodos heurísticos con el fin de desarrollar algoritmos eficientes que sean capaces de entregar buenas soluciones a problemas en donde encontrar el óptimo resulta muy costoso en tiempo de ejecución e incluso en algunos casos imposible.

Los métodos exactos poseen la ventaja de ser más rigurosos, pero con la desventaja de su difícil formulación matemática sumada a la gran cantidad de recursos computacionales que requiere. Las dificultades evidentes en los cálculos de los modelos matemáticos obligan a buscar otros métodos que aunque no garantizan la mejor solución final, buscan acercarse al óptimo. Tales métodos se denominan heurísticos, suelen emplearse con dos fines: en el contexto de un algoritmo de optimización exacto, con el fin de aumentar la velocidad del proceso y en segundo lugar para obtener una solución al problema aunque no óptima, la que puede ser muy difícil encontrar. Además representan una facilidad para la puesta en marcha en el computador y en la confección algorítmica, su principal desventaja es el poco énfasis que le da al modelo matemático aún cuando éste haya sido planteado desde el inicio del problema, por lo cual impide el desarrollo de una aplicación general (Baquero, y otros, 2008) (Bellini M., 2004).

1.7.1. Ejemplos de utilización de los Métodos Exactos

- **Programación Lineal Entera Binaria**

Esta es una técnica tradicional, y corresponde a una extensión de la Programación Lineal Entera (PLE), donde cualquier variable entera puede expresarse, equivalentemente, en términos de un cierto número de variables cero-uno (binarias). Esto sucede, porque a veces la formulación de un modelo no se puede manejar como uno de PLE pura o mixta.

En muchas aplicaciones, las variables binarias pueden ser de gran utilidad en problemas que incluyen decisiones del tipo sí o no interrelacionadas. El uso de esta técnica para la obtención de resultados óptimos requiere de un alto tiempo computacional, existiendo problemas de gran tamaño en los que es posible que no se obtengan resultados en un tiempo computacional razonable.

1.7.2. Ejemplos de utilización de los Métodos Heurísticos

La heurística es una serie de procedimientos o estrategias de las que se suponen que llevan a un destino deseado; pero esto no tiene porque ser siempre cierto, y se definen según Zankis y Evans, en *"Heuristics optimization: why, when, and how to use it"*, como: "procedimientos simples, a

menudo basados en el sentido común, que se supone ofrecerán una buena solución (aunque no necesariamente la óptima) a problemas difíciles, de un modo fácil y rápido" (Burke, y otros, 2003).

Uno de los mayores inconvenientes con los que se enfrentan estos métodos es la existencia de óptimos locales que no sean absolutos, así como el conseguir hacerse independientes de la solución inicial de la que se parta. La tendencia actual es desarrollar métodos generales para resolver clases o categorías de problemas, conocidos como procedimientos metaheurísticos. Estos algoritmos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria, en los que los heurísticos clásicos no son ni efectivos ni eficientes. Los metaheurísticos proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, evolución biológica y mecanismos estadísticos (García Márquez, 2003).

Las metaheurísticas de búsqueda local han sido uno de los enfoques más exitosos para resolver los problemas de optimización combinatoria en los últimos años. La búsqueda local es el nombre común para el conjunto de métodos que (en general) de forma iterativa sustituye la solución actual *s* por una nueva *s'*, hasta satisfacer alguna condición de parada. La calidad de la solución se caracteriza por una función de coste *f(s)*, el objetivo del proceso de búsqueda es reducir al máximo la función. Las variantes de este enfoque difieren en sus mecanismos de aceptar o rechazar la solución candidata de la vecindad, las definiciones de los barrios y las condiciones de detención (Burke, y otros, 2003).

De acuerdo a lo planteado se considera que el enfoque de solución más conveniente a emplear para el desarrollo de esta investigación es el uso de metaheurísticas de búsqueda local. A continuación se presentan algunos métodos metaheurísticos:

- **Hill-Climbing**

El algoritmo más simple de búsqueda local es Hill-Climbing, este método fue utilizado para el PAH en 1960. Una solución candidata es aceptada si es mejor o equivalente a la actual. Hill-Climbing no requiere de la definición de parámetros y su comportamiento es bastante estable. Su objetivo es converger muy rápido, pero a menudo tiene una solución definitiva de calidad relativamente baja, ya que tiende a quedar atrapados en óptimos locales (Burke, y otros, 2003).

- **Recocido Simulado**

Una de las metaheurísticas de búsqueda local más estudiadas es Recocido Simulado *(Simulated Annealing)*, se propuso como una técnica de optimización general en 1983 por S. Kirkpatrick y ha sido reiteradamente aplicada para resolver una amplia gama de problemas.

Se basa en los conceptos descritos originalmente por la mecánica estadística que describe el proceso físico sufrido por un sólido al ser sometido a un baño térmico. Los estados del sistema se

corresponden con las soluciones del problema, la energía de los estados con los criterios de evaluación de la calidad de la solución, el estado fundamental con la solución óptima del problema, los estados metaestables serán los equivalentes a los óptimos locales, y la temperatura correspondería a una variable de control (García Márquez, y otros, 2003).

El algoritmo de Recocido Simulado es un algoritmo heurístico estocástico en el cual las soluciones se buscan utilizando un proceso de enfriamiento con movimientos aleatorios. Según la literatura este algoritmo tiende a caer en soluciones locales durante un largo tiempo, razón por la que no es una heurística muy querida (Alondra, 2009).

- **Umbral de Aceptación**

Una variante determinista del Recocido Simulado es conocida como el método del Umbral de Aceptación *(Threshold Acceptance)*, acepta las peores soluciones, cuando δ no exceda un cierto umbral T. Fue introducido por Dueck y Scheuer, en 1989. Originalmente, los autores sugieren que el umbral debe reducirse cuando el algoritmo no mejora la solución para un largo tiempo, sin embargo no está claro cuándo hacerlo ni cuánto disminuir. El sistema de refrigeración adaptable se introdujo por el método del umbral de aceptación, pero como en el caso anterior, no rindió beneficios prácticos y sensatos por lo que esta técnica no se suele aplicar (Burke, y otros, 2003).

- **Algoritmos Voraces**

Los Algoritmos Voraces, *Greedy Randomized Adaptive Search Procedures (GRASP)*, también conocidos como Algoritmos Ávidos, son una técnica de diseño de algoritmos, que se basan principalmente en usar la mejor opción para un momento dado, sin importar consecuencias futuras, para obtener la mejor solución general. Fueron desarrollados al final de los 80 con el objetivo inicial de resolver problemas de cubrimiento de conjuntos.

Este tipo de algoritmos son rápidos, suelen ser lineales o de segundo grado y que requieren de memoria extra (Halim, 2009). Sin embargo, la calidad de los algoritmos voraces está en relación con las características de las instancias que pretenden resolver: puede arrojar muy buenos resultados para determinadas instancias del problema pero para otras no. Por otra parte, se estancan en óptimos locales de las funciones que pretenden optimizar y quizá no analizan vecindades más allá del criterio goloso por lo que pueden estar dejando de considerar al óptimo global (Tupia & Mauricio, 2004).

- **Algoritmos Evolutivos y Algoritmos Genéticos**

Los Algoritmos Genéticos son una técnica de representación que se basa en los mecanismos de selección que utiliza la naturaleza, donde los individuos más aptos de una población sobreviven, ya que pueden adaptarse más fácilmente a los cambios que se producen en su entorno. Son métodos sistemáticos para la resolución de problemas de búsqueda y selección.

Los Algoritmos Evolutivos, *Evolutionary Algorithms*, son un esquema de representación que aplica una técnica de búsqueda de soluciones enfocada a problemas de optimización, inspirada en la teoría de la evolución de Charles Darwin. Se basa en el algoritmo de selección propio de la naturaleza, con la esperanza de que así se consigan éxitos similares, en relación a la capacidad de adaptación a un amplio número de ambientes diferentes.

En la Inteligencia Artificial, se simula el comportamiento de los individuos sobre la teoría de la evolución en la metaheurística de Algoritmos Genéticos y Algoritmos Evolutivos. Ambos utilizan la teoría de la evolución con factores tales como población, individuos, operadores genéticos, entre otras. La diferencia de ambos radica, principalmente, en la representación del individuo en la población. Estos pueden ser representados en forma binaria para el caso de algoritmos genéticos y representación no binaria para algoritmos evolutivos.

- **Búsqueda Tabú**

La Búsqueda Tabú, *Tabú Search*, es un algoritmo metaheurístico que puede utilizarse para resolver problemas de optimización combinatoria, realiza la búsqueda por entornos, que permite moverse a una solución del entorno aunque no sea tan buena como la actual, de este modo se puede escapar de óptimos locales y continuar la búsqueda de soluciones aun mejores. La forma de evitar viejos óptimos locales es clasificando un determinado número de los más recientes movimientos como “movimientos tabú”, los cuales no son posibles repetir durante un determinado horizonte temporal. Por lo tanto, en este caso, el escape de los óptimos locales se produce de forma sistemática y no aleatoria.

La Búsqueda Tabú, al contrario que el Recocido Simulado, enfatiza en procedimientos determinísticos en lugar de aleatorios. Por lo que es importante considerar que estos requieren de la exploración de un gran número de soluciones en poco tiempo, por ello es crítico el reducir al mínimo el esfuerzo computacional (Araya, 2007). El objetivo es lograr el óptimo global, evitando con ello que el algoritmo se estanque en un óptimo local (García Márquez, y otros, 2003).

- **Algoritmo Gran Diluvio**

El algoritmo de Gran Diluvio fue presentado por G. Dueck, *"New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel"*, en 1993, desafortunadamente no fue muy útil en los años sucesivos. Esta metaheurísticas de búsqueda local es diferente a sus predecesores (*Hill-Climbing, Simulated Annealing,* etc.). Este algoritmo acepta todas las soluciones, para los valores absolutos de la función de costo menor o igual a la valores en la frontera actual, llamado "nivel". La búsqueda local se inicia con el valor inicial de "nivel" igual a una función de coste inicial y durante la búsqueda este valor es reducido. Una disminución de la reducción (definido por el usuario) aparece como un único parámetro del algoritmo

Algunas de sus propiedades son:

- El perfil del proceso es explícito. La búsqueda sigue rígidamente la degradación del nivel.
- El punto de convergencia es muy reconocible. Cuando una solución actual alcanza el valor, donde cualquier mejora es imposible, la búsqueda converge rápidamente. Este momento puede ser fácilmente detectado, a fin de resolver el procedimiento de búsqueda.

En el 2003 los autores ampliaron el algoritmo de Gran Diluvio, al aceptar todos los movimientos cuesta abajo (hibridación con Hill-Escalada). Esta variante se aplicó con éxito al problema de asignación de exámenes para universidades y su desempeño fue bien investigado. En los años posteriores se realizaron nuevas modificaciones a este algoritmo. Estas investigaciones revelan varias ventajas, en particular, permite avanzar en la definición de las características de un proceso de búsqueda (como un tiempo de procesamiento y una región con una solución final esperada) con más precisión que otros métodos. La utilización adecuada de estas propiedades aumenta significativamente el rendimiento de una búsqueda local. Cuando se comparó experimentalmente con otras técnicas existentes, el algoritmo de Gran Diluvio produjo resultados con mayor calidad en la mayoría de las instancias del problema (Burke, y otros, 2008).

Una de las mayores desventajas de muchas metaheurísticas aplicadas a problemas de Scheduling (Timetabling) es que pueden ser muy dependientes a rangos específicos de sus parámetros. La efectividad de los distintos enfoques puede ser muy sensible a la configuración de los parámetros, al igual que determinar una correcta configuración de los mismos puede causar dificultades significativas. Estas dificultades son exacerbadas si los parámetros son configurados por personas inexpertas en metaheurísticas. Por estos motivos, es conveniente investigar enfoques que no sean tan sensibles a los parámetros, no sólo reduciendo la cantidad de parámetros sino que los incluidos deben ser de un fácil análisis y comprensibles por usuarios inexpertos.

Por lo antes expuesto se propone el uso del algoritmo Gran Diluvio como técnica de solución a emplear a partir del horario generado por medio de la programación basada en restricciones.

1.8. Metodologías de Desarrollo de Software

Una metodología puede definirse, en un sentido amplio, como un conjunto de métodos y técnicas que ayudan en el desarrollo de un producto software, tal como lo señala Rumbaugh: “Una metodología de ingeniería de software es un proceso para la producción organizada del software, empleando para ello una colección de técnicas predefinidas y convencionales en las notaciones. Una metodología se presenta normalmente como una serie de pasos, con técnicas y notaciones asociadas a cada paso. Los pasos de la producción del software se organizan normalmente en un ciclo de vida consistente en varias fases de desarrollo.”

Las metodologías pueden ser clasificadas en dos grandes grupos (Letelier, y otros, 2006):

- Metodologías Tradicionales: se centran especialmente en el control del proceso, estableciendo rigurosamente las actividades involucradas, los artefactos que se deben producir, y las herramientas y notaciones que se usarán (RUP, MSF).
- Metodologías Ágiles: dan mayor valor al individuo, a la colaboración con el cliente y al desarrollo incremental del software con iteraciones muy cortas. Este enfoque está mostrando su efectividad en proyectos con requisitos muy cambiantes y cuando se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad (XP *(Extreme Programming)*, Scrum, Iconix, Cristal Methods, AUP entre otras).

Cada una tiene ventajas y desventajas, y se utilizan en aquellas situaciones donde una metodología resulta más apropiada que otra (Nuevo León, 2009). A continuación se presentan algunas de estas metodologías.

- **Proceso Unificado de Desarrollo**

Proceso Unificado de Desarrollo (RUP) es un proceso formal, provee un acercamiento disciplinado para asignar tareas y responsabilidades dentro de una organización de desarrollo. Su objetivo es asegurar la producción de software de alta calidad que satisfaga los requerimientos de los usuarios finales (respetando cronograma y presupuesto). Guiado por casos de uso, centrado en la arquitectura e iterativo e incremental.

Al ser una metodología de desarrollo de software tradicional centra su atención en llevar una documentación exhaustiva de todo el proyecto, así como la realización de muchos artefactos. Está diseñada para proyectos de desarrollo grandes y requiere de un gran equipo de desarrolladores. Centra su atención en cumplir con un plan de proyecto, involucra los altos costos al implementar un cambio y no ofrece una buena solución para proyectos donde el entorno es volátil (G. Figueroa, y otros, 2008).

- **Scrum**

Esta es una de las metodologías de desarrollo ágil utilizado en la producción de diferentes productos de software. Martin en su libro sobre la esencia de la metodología expresa: “Está especialmente indicada para proyectos con un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: El desarrollo de software se realiza mediante iteraciones, denominadas *sprints*, con una duración de 30 días. El resultado de cada sprint es un incremento ejecutable que se muestra al cliente. La segunda característica importante son las reuniones a lo largo del proyecto, entre ellas se destaca la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración”.

La metodología propicia el desarrollo iterativo e incremental, conduce a que el cliente no tiene que esperar hasta que el sistema se entregue completo para usarlo, se pueden aclarar los requisitos, disminuye el riesgo de fracaso de todo el proyecto, puede aplicarse a cualquier proyecto sin importar la complejidad. Tiene como limitante la definición de los requisitos para cada Sprint. Al igual que XP, como se verá más adelante, no centra su desarrollo en la arquitectura (Nuevo León, 2009).

- **Programación Extrema**

La Programación Extrema (XP) es la metodología ágil más difundida, que puede ser usada por equipos pequeños y medianos, para la construcción de aplicaciones con elevada calidad, con un mínimo gasto fijo, dentro de un presupuesto y calendario previsible. Prioriza los resultados de trabajo inmediatos, haciendo el proceso de desarrollo más sencillo y aplicando el sentido común. Aumenta la productividad en el desarrollo, potenciando al máximo el trabajo en equipo (Pérez Díaz, y otros, 2009).

Las características fundamentales del método son (G. Figueroa, y otros, 2008):

- Desarrollo iterativo e incremental: pequeñas mejoras, unas tras otras.
- Pruebas unitarias continuas, frecuentemente repetidas y automatizadas, incluyendo pruebas de regresión. Se aconseja escribir el código de la prueba antes de la codificación.
- Programación por parejas: se recomienda que las tareas de desarrollo se lleven a cabo por dos personas en un mismo puesto. Se supone que la mayor calidad del código escrito de esta manera (el código es revisado y discutido mientras se escribe) es más importante que la posible pérdida de productividad inmediata.
- Frecuente interacción del equipo de programación con el cliente o usuario. Se recomienda que un representante del cliente trabaje junto al equipo de desarrollo.
- Corrección de todos los errores antes de añadir nueva funcionalidad. Hacer entregas frecuentes.
- Refactorización del código, es decir, reescribir ciertas partes del código para aumentar su legibilidad y mantenibilidad pero sin modificar su comportamiento. Las pruebas han de garantizar que en la refactorización no se ha introducido ningún fallo.
- Propiedad del código compartida: en vez de dividir la responsabilidad en el desarrollo de cada módulo en grupos de trabajo distintos, este método promueve el que todo el personal pueda corregir y extender cualquier parte del proyecto. Las frecuentes pruebas de regresión garantizan que los posibles errores serán detectados.

- Simplicidad en el código: es la mejor manera de que las cosas funcionen, cuando todo funcione se podrá añadir funcionalidad si es necesario. La programación extrema apuesta que es más sencillo hacer algo simple y tener un poco de trabajo extra para cambiarlo si se requiere, que realizar algo complicado y quizás nunca utilizarlo.

Se define el uso de la metodología XP debido a que se corresponde en gran medida tanto a las necesidades del tipo de investigación a desarrollar como a las condiciones de trabajo, ya que el equipo de trabajo está conformado por dos programadores y tres clientes y se dispone de un período de desarrollo de cinco meses.

1.9. Lenguajes de Programación

Los lenguajes de programación pueden usarse para crear programas que controlen el comportamiento físico y lógico de una máquina, para expresar algoritmos con precisión, o como modo de comunicación humana. Está formado de un conjunto de símbolos y reglas sintácticas y semánticas que definen su estructura y el significado de sus elementos y expresiones (Rodríguez Ríos, 2010).

- **Java**

Es un lenguaje de programación orientado a objetos, desarrollado por Sun Microsystems a principios de los años 90. Entre noviembre de 2006 y mayo de 2007, Sun liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de tal forma que prácticamente todo el Java de Sun es ahora software libre.

Es independiente de la plataforma puesto que programas escritos en este lenguaje pueden ejecutarse sobre cualquier tipo de hardware, programas de este tipo pueden ejecutarse sobre cualquier tipo de dispositivos. Presenta también un recolector automático de basura, este proceso es casi invisible a los ojos del programador ya que éste no tiene conciencia de cuando se hará la recolección de basura. Fue diseñado para crear un software altamente fiable, para esto lo que hace es que proporciona numerosas comprobaciones en compilación y en tiempo de ejecución (Vallejo Martínez & Garcia Giniebra, 2009).

Java corre sobre máquinas virtuales, la principal desventaja de los lenguajes basados en máquina virtual es que son más lentos que los lenguajes completamente compilados, debido a la sobrecarga que genera tener una capa de software intermedia entre la aplicación y el hardware de la computadora (Menchaca Méndez & García Carballeira, 2000), razón por la cual no es recomendable su uso.

- **Preprocessed Hypertext Pages**

PHP es el acrónimo de *Hipertext Preprocesor*, es un lenguaje de programación del lado del servidor gratuito e independiente de plataforma. Está preparado para realizar muchos tipos de aplicaciones web, gracias a la extensa librería de funciones con la que está dotado (Alvarez, 2003).

Este lenguaje de programación corre sobre un servidor Apache que impone serias restricciones en cuanto a la cota máxima del tiempo de ejecución de los scripts creados para ser ejecutados por el servidor Web. En el caso de los algoritmos que se incluyen en la librería es prácticamente imposible garantizar que tendrán un tiempo de ejecución tan extremadamente reducido y, lo que podría ser incluso más perjudicial, una vez vencido este límite, el hilo de ejecución que Apache asigna al procesamiento de fichero PHP es descartado llevando al sistema a puntos de no retorno lo que convertiría al Ghordo en un sistema no tolerante a fallos.

- **Prolog**

Prolog es un lenguaje de programación diseñado para representar y utilizar el conocimiento que se tiene sobre un determinado dominio. Los programas en Prolog responden preguntas sobre el tema del cual tienen conocimiento. Es un lenguaje declarativo e interpretado, esto quiere decir que el lenguaje se usa para representar conocimientos sobre un determinado dominio y las relaciones entre objetos de ese dominio. La ejecución de Prolog consiste en una búsqueda en profundidad de un árbol conteniendo todas las posibles soluciones, para cada una de ellas se evaluará su validez.

Entre los inconvenientes relacionados con su utilización se puede mencionar que la resolución automática no siempre es eficiente, por lo que eventualmente se podría dar una respuesta incorrecta a una consulta, ciertos problemas están ligados a la representación del conocimiento, que Prolog no posee, los motores de inferencia poseen algunos límites. Prolog algunas veces es incapaz de reconocer que un problema es (para su propio conocimiento) inaplicable o insuficiente. Si el programa no contiene suficiente información para contestar una consulta, es incapaz de reconocerlo y responde "*No"* (Jones Pérez, y otros, 2007). Por lo antes expuesto, esta opción es desechada debido a lo complejo que para este lenguaje es la integración con lenguajes de otra plataforma, requisito indispensable de los productos de soporte como expresa el problema del presente documento.

- **C#**

C# es un lenguaje de propósito general orientado a objetos, es uno de los lenguajes de programación más populares en la informática, permite a los programadores abordar el desarrollo de aplicaciones complejas con facilidad y rapidez. Con C# no sólo se pueden escribir programas para la Web, sino que también permite desarrollar aplicaciones de propósito general (Ceballos Sierra, 2010). Se basa en las lecciones aprendidas de los lenguajes C, C++, Java y Visual Basic,

por ello se trata de un lenguaje que combina todas las cualidades que se pueden esperar de un lenguaje moderno (orientación a objetos, gestión automática de memoria, etc.) y a la vez proporciona un gran rendimiento (González Seco, 2010).

Se trata de un lenguaje que está sujeto a restricciones comerciales por lo que lleva implícito una inversión adicional por concepto de patentes, por lo cual se descarta la posibilidad de su uso debido a que la UCI se inclina hacia el uso de soluciones que no impongan restricciones en cuanto a la propiedad intelectual de las herramientas utilizadas para el desarrollo y del producto en sí.

- **Python 2.5**

Es un lenguaje de programación creado por Guido van Rossum, se desarrolla como un proyecto libre y de código abierto, administrado por la Python Software Foundation. Es considerado por muchos un lenguaje de programación de muy alto nivel, es un lenguaje de programación de propósito general. Es orientado a objetos, pero también permite trabajar en forma procedimental.

El entorno de ejecución detecta muchos de los errores de programación que escapan al control de los compiladores y proporciona información muy rica para detectarlos y corregirlos. Puede usarse como lenguaje imperativo procedimental o como lenguaje orientado a objeto. Posee un rico juego de estructuras de datos que se pueden manipular de modo sencillo. Una ventaja fundamental es la gratuidad de su intérprete, el cual tiene versiones para prácticamente cualquier plataforma en uso: sistemas PC bajo Linux, sistemas PC bajo Microsoft Windows, sistemas Macintosh de Apple, etc. (Marzal & Gracia, 2003).

Ventajas del uso de Python (Alvarez, 2003):

- Propósito general: Se pueden crear todo tipo de programas.
- Multiplataforma: Hay versiones disponibles de Python en muchos sistemas informáticos distintos. Originalmente se desarrolló para Unix, aunque cualquier sistema es compatible con el lenguaje siempre y cuando exista un intérprete programado para él.
- Interpretado: No se debe compilar el código antes de su ejecución. En realidad sí que se realiza una compilación, pero esta se realiza de manera transparente para el programador.
- Interactivo: Dispone de un intérprete por línea de comandos en el que se pueden introducir sentencias. Cada sentencia se ejecuta y produce un resultado visible, que puede ayudarnos a entender mejor el lenguaje y probar los resultados de la ejecución de porciones de código rápidamente.
- Orientado a Objetos: La programación orientada a objetos está soportada en Python y ofrece en muchos casos una manera sencilla de crear programas con componentes

reutilizables. Además, Python también permite la programación imperativa, programación funcional y programación orientada a aspectos.

- Funciones y librerías: Dispone de muchas funciones incorporadas en el propio lenguaje, para el tratamiento de strings, números, archivos, etc. Además, existen muchas librerías que se pueden importar en los programas para tratar temas específicos.
- Sintaxis clara: Tiene una sintaxis muy visual, gracias a una notación identada (con márgenes) de obligado cumplimiento. Esto ayuda a que todos los programadores adopten unas mismas notaciones y que los programas de cualquier persona tengan un aspecto muy similar.
- Mixto: Se puede integrar de manera "fácil" con otros lenguajes de programación.
- Gratuito: Una ventaja fundamental de Python es la gratuidad de su intérprete, se puede descargar desde la página web: http://www.python.org.

Después del estudio de los lenguajes de programación se decide el uso de Python 2.5.

1.10. Herramientas

Para el tema de programación en Python existen varias opciones, se seleccionaron dos de ellas que fueron consideradas las más prometedoras, por una parte está WingIDE que corre sobre los dos sistemas operativos es decir Windows y todas las versiones de Linux pero tiene el inconveniente de estar sujeto a restricciones comerciales.

Por otra parte está PyDEV, es un plugin para Eclipse que permite utilizar este IDE multiplataforma para programar en Python. Cuenta con autocompletado de código (con información sobre cada elemento), resaltado de sintaxis, un depurador gráfico, resaltado de errores, explorador de clases, formateo del código, refactorización, etc. Aunque necesita de una cantidad importante de memoria y no es del todo estable (González Duque, 2009).

Otra opción gratuita a considerar PyScripter, un Entorno de Desarrollo Integrado (IDE) para Python. Está construido en Delphi usando P4D y creado con la ambición de crear un entorno de programación Python que sea competitivo con Windows comerciales basadas en IDEs disponibles para otros idiomas. Es bastante ágil en comparación con otros IDE de Python y proporciona una amplia mezcla de características que lo convierten en un entorno de desarrollo productivo, algunas de ellas son (Vlahos, 2006):

- Resaltado de sintaxis.
- Utilidades código fuente de Python.

- Completamiento de código y consejos de llamada.
- Código y depurador de sugerencias.
- Comprobación de sintaxis a medida que escribe.
- Ayuda sensitiva del contexto de las palabras claves de Python.
- Plantillas de parámetros de código.
- Notificación de modificación del archivo.
- Ventana de relojes.
- Ventana de puntos de ruptura.
- Editor de vistas.
- Documentación en formato HTML (|pydoc|).
- Explorador de código.
- Explorador de archivos.
- Acceso a los manuales de Python por el menú Ayuda.
- Integración con herramientas de Python.

Por lo anteriormente expuesto se propone el uso del IDE PyScripter 1.7.2.0 como herramienta a utilizar.

1.11. Conclusiones Parciales

Las publicaciones relacionadas con el PAHC se corresponden a instancias de una institución en particular, sin embargo todas coinciden en que no existe un modelo general que pueda ser aplicado en todas las realidades.

A partir de la investigación realizada se evidencia la necesidad de implementar una Biblioteca de Algoritmos que permita la generación automática del horario docente teniendo en cuenta las características específicas de la Facultad, debido a que los sistemas existentes a nivel internacional no se ajustan a las normas de la educación superior de nuestro país, utilizándose la metodología, lenguaje de programación y herramienta seleccionada para la implementación de la solución.

CAPÍTULO 2: SOLUCIÓN PROPUESTA

2.1. Introducción

Una vez analizado el Problema de Asignación de Horarios de Clase (PAHC) se describen los aspectos fundamentales de la solución propuesta, reflejados en los artefactos obtenidos con el uso de la metodología seleccionada, Programación Extrema (XP), así como en la presentación de restricciones a tener en cuenta y la estructurada la Biblioteca de Algoritmos.

2.2. Biblioteca de Algoritmos

La Biblioteca de Algoritmos constituye una nueva funcionalidad de la aplicación web Gestor de Horario Docente (Ghordo) que recibe los datos a través de varios ficheros con la información referente a los grupos docentes, profesores, locales, la planificación de las asignaturas (P1) y el tipo de planificación a realizar, que puede ser de una a varias semanas. Una vez cargados los datos, se ejecutan los algoritmos implementados, que dan como resultado el conjunto de encuentros que serán utilizados por el Ghordo para publicar el horario docente. La Biblioteca de Algoritmos no requiere de la existencia de otros sistemas para devolver el horario en el formato estándar.

Para la implementación de la biblioteca se tiene en cuenta un conjunto de restricciones específicas de la universidad que serán abordas en próximos epígrafes, se definieron dos tipos de planificaciones a realizar: para una semana determinada o para un rango de semanas, de igual forma brinda la posibilidad de que el responsable de la planificación decida el orden de los años a planificar y el tipo de algoritmo que desea utilizar. Se tiene en cuenta la cantidad de frecuencias de las asignaturas, la cantidad de turnos dobles (dos turnos consecutivos), la cantidad de grupos de un profesor, los profesores externos, las afectaciones de los grupos docentes, profesores y locales, entre otros aspectos. A partir de ello se realiza la distribución de los locales existentes, teniendo en cuenta el tipo de local que requiere cada actividad. Finalmente se obtiene una versión factible del horario docente en función de los grupos, asignaturas y profesores.

2.3. Descripción de las restricciones

A partir de la investigación realizada se definen las siguientes restricciones en un lenguaje natural, las cuales se clasifican en dos grandes grupos: fuertes y débiles.

Restricciones fuertes:

- Un profesor no puede tener clases con un grupo en el tiempo que no esté disponible.
- Un profesor, grupo docente o local no debe ser asignado a más de una actividad al mismo tiempo.
- Un grupo docente no pueden tener actividades en un local que no tenga la capacidad requerida.
- Todas las asignaturas a impartir deben ser planificadas.
- La asignatura Educación Física no puede tener actividades que la precedan o que le sucedan.
- No se pueden asignar actividades consecutivas en el tercer y cuarto turno de clases.
- Los profesores, externos no deben tener planificada clases a sexto turno.
- Después de planificarse una actividad con tipo de frecuencia “Conferencia” se debe dejar un día de diferencia con relación a la próxima actividad de esa asignatura.
- No asignar ninguna actividad al grupo, profesor o local en turnos con afectaciones previamente declaradas.
- Los turnos dobles se deben planificar en el mismo local.

Restricciones débiles:

- Los turnos de un mismo grupo deben ser planificados en los tres primeros turnos del día o en los tres últimos.
- Los profesores vinculados a la producción no deben tener actividades planificadas en el segundo o quinto turno del día.
- Se deben planificar las actividades, de ser posible, en los mismos locales.
- No se debe planificar, de ser posible, clases los sábados.
- Un profesor no debe dar clases dos sábados consecutivos.
- Los profesores que imparten una actividad a más de un grupo se les debe planificar los turnos consecutivos el mismo día.
- Evitar planificar actividades en el tercer y sexto turno para las asignaturas de los departamentos Programación y Ciencias Básicas.
- No se debe planificar tres actividades consecutivas de las asignaturas del departamento de Ciencias Básicas.

- Las semanas planificadas deben ser lo más similares posible con relación a los turnos de las asignaturas.

2.4. Función Objetivo

La función objetivo es una relación matemática entre las variables de decisión, parámetros y una magnitud que representa el objetivo o producto del sistema (Bellini M., 2004). El Problema de Asignación de Horarios a Clase (PAHC) formulado como Problema de Optimización Combinatoria queda formulado de la siguiente forma:

Encontrar un horario $h \in H$ con el menor número de violaciones de las restricciones débiles.

$$\min FO = \sum_{i=1}^{4} C_i(h)$$
$$sujeto\ a$$
$$h \in H$$

Donde:

- h : Horario factible (horario que cumple con todas las restricciones fuertes).
- H : Conjunto de todos los horarios factibles.

El indicador de calidad será tomado para de las cuatro restricciones débiles que se priorizan en esta fase de desarrollo de la investigación.

$C_i(h)$: Indicador de calidad del horario h.

- $C_1(h)$: Cantidad de turnos planificados los sábados.
- $C_2(h)$: Cantidad de turnos planificados en el segundo o quinto turno a los profesores vinculados a la producción.
- $C_3(h)$: Cantidad de tercer y sextos turnos planificados para las asignaturas de los departamentos Programación y Ciencias Básicas.
- $C_4(h)$: Cantidad veces que se planifican tres turnos consecutivos de las asignaturas del departamento de Ciencias Básicas.

2.5. Programación Basada en Restricciones

La idea de la Programación Basada en Restricciones (PBR, en lo adelante) es resolver problemas mediante la declaración de restricciones sobre el área del problema y consecuentemente encontrar soluciones que satisfagan todas las restricciones, y en su caso optimicen unos criterios determinados (Barber, y otros, 2003). Se basa en el modelamiento y resolución con restricciones. Dado un problema descrito a través de variables, donde cada variable posee un dominio asociado, compuesto por un conjunto de potenciales valores y un conjunto de restricciones que representan las diferentes relaciones entre las variables que deben ser satisfechas para resolver el problema.

La PBR involucra tanto el modelamiento como la resolución de sistemas basados en restricciones. Entre sus principales campos de aplicación se encuentran los problemas de optimización combinatoria y ordenamiento, análisis financiero, diseño y construcción de circuitos integrados, la biología molecular y la resolución de problemas geométricos, entre otras (Grandón, 2004).

Se implementaron cuatro algoritmos utilizando el paradigma de PBR, el primero basa su funcionamiento en darle mayor prioridad a las asignaturas con mayor cantidad de profesores externos, sin menospreciar las asignaturas que poseen mayor cantidad de frecuencias. El segundo algoritmo, al contrario, prioriza las asignaturas que poseen mayor cantidad de frecuencias, sin dejar de tener en cuenta a las que presentan mayor cantidad de profesores externos. El tercero brinda la posibilidad de asignar prioridad a las asignaturas de acuerdo a la conveniencia del usuario y el cuarto algoritmo utiliza una función matemática para asignar la prioridad a las asignaturas que se define como:

P = (CF + CPE) / 2, donde:

CF: Cantidad de frecuencias de una asignatura.

CPE: Cantidad de profesores externos de una asignatura.

Además fue implementado un quinto algoritmo basado en la metaheurística Gran Diluvio que, a partir de un horario factible generado previamente por alguno de los algoritmos descritos, devuelve una nueva solución, pudiendo mejorar o no la inicial. En el próximo epígrafe será descrito con mayor nivel de detalle.

2.6. Optimización de la solución generada

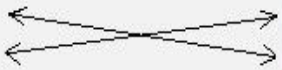
Tomando como punto de partida la solución devuelta por uno de los algoritmos explicados anteriormente fue implementado un quinto algoritmo que analiza cuál de ellos ofrece una mejor solución de acuerdo al resultado de la función objetivo. Teniendo la solución inicial factible, se

realizan permutaciones en el orden de prioridades de la lista de asignaturas, como se evidencia en el siguiente ejemplo:

Para primer año:

Figura 2. Permutaciones entre asignaturas

Asignaturas	Prioridad	Valor FO
MD2	7	40
DN	6	
P1	5	
M2	4	
AL	3	
PP1	2	
PHCCU	1	



Asignaturas	Prioridad	Valor FO
MD2	7	36
DN	6	
M2	5	
P1	4	
AL	3	
PP1	2	
PHCCU	1	

Fuente: Elaboración propia.

Es decir, pequeños cambios en el orden de prioridad de las asignaturas se puede mejorar o no el resultado de la función objetivo.

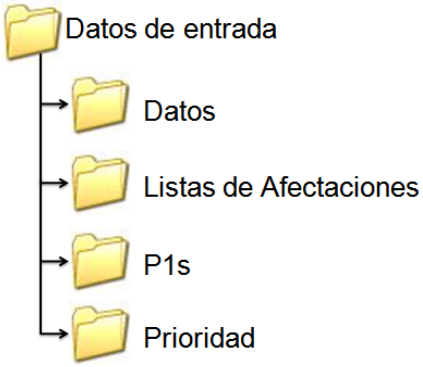
2.7. Entrada y Salida de datos

Para la entrada y salida de datos de la Biblioteca de Algoritmos se utilizan ficheros con extensión ".csv", conocido como "delimitado por comas", donde cada línea del fichero constituye un registro individual o fila de la tabla. Son comúnmente utilizados para transferir datos entre bases de datos y se pueden abrir y editar por la mayoría de los editores de texto y hojas de cálculo.

2.7.1. Entrada de datos

A continuación se muestra el mapa de distribución de las carpetas contenedoras de los ficheros de entrada de datos de la Biblioteca de Algoritmos.

Figura 3. Jerarquía de ficheros de entrada de la Biblioteca de Algoritmos.



Fuente: Elaboración propia.

La carpeta **Datos:** Incluye los ficheros básicos de la Biblioteca de Algoritmos:

- Grupos_Profesores_Asignaturas: Contiene datos de los grupos organizados por años consecutivamente, el semestre a planificar, el identificador de cada profesor, nombre y apellidos del profesor que imparte una asignatura a dicho grupo, el tipo de profesor, el nombre de la asignatura y el nombre del departamento al cual pertenece la misma (Anexo 1).
- Locales: Incluye el número identificador, el tipo y la capacidad del local.
- EF: Incluye el grupo, nombre y apellidos del profesor y los días y turnos de las clases de Educación Física (EF).

La carpeta **Lista de afectaciones:** Recoge las afectaciones a tener en cuenta.

- Listas_Afectaciones: Este fichero contiene las afectaciones pertenecientes a los grupos, locales, departamentos y profesores consecutivamente. Las afectaciones se establecen mediante la combinación día-turno. Los días son: lunes, martes, miércoles, jueves, viernes y sábado y los turnos se corresponden a los valores consecutivos del 1 al 6 (Anexo 2).

La carpeta **P1s:** Recoge la planificación de las asignaturas.

- P1: Se muestran las asignaturas, número de semana a planificar, tipo de frecuencia, cantidad de frecuencias y tipo de local que se requiere para realizar la actividad (Anexo 3).

La carpeta **Prioridad:** Incluye los ficheros referentes a la cantidad de semanas y el orden de los años a planificar.

- Cantidad_Semanas_Planificar: Se puede planificar una semana determinada o desde una semana inicial hasta una semana final (Anexo 4).
- Prioridad: Se especifica el orden de los años a planificar, el semestre, la sesión y el algoritmo a utilizar (Anexo 5).

- Prioridad_Asignatura: En este fichero se define la prioridad de las asignaturas de los años a planificar, será empleado en caso de que se utilice el algoritmo número tres o cinco. Por cada año se asigna un valor a cada asignatura, la que contenga mayor valor numérico será la de mayor prioridad, o sea, la primera que se tendrá en cuenta para la planificación (Anexo 6).

2.7.2. Salida de datos

En la carpeta Base de Datos-Saliente se devuelven los siguientes ficheros:

- Lista_Encuentros_General: Contiene la lista de los encuentros obtenidos en el proceso de generación del horario (Anexo 7).
- Horario: Organiza la información de la Lista_Encuentros_General en función de los grupos docentes, de acuerdo al formato utilizado para la visualización por parte de los estudiantes (Anexo 8).
- Lista_Reportes: En el mismo se exponen los turnos que no se pudieron incluir en el horario (en caso de que ocurra), especificándose el grupo, la asignatura, el tipo de frecuencia y a la semana que pertenece.
- Valores_Funcion_Objetivo: Devuelve el valor del cálculo de la función objetivo aplicado a un año, de acuerdo al algoritmo seleccionado (Anexo 9).

2.8. Definición de abreviaturas

A continuación se muestran las diferentes abreviaturas y sus respectivos significados que serán utilizadas en la entrada y salida de datos de la Biblioteca de Algoritmos. En el caso particular de las abreviaturas de las asignaturas y los nombres de los departamentos no se hace una imposición en cuanto a su uso, ya que pueden ser descritas a decisión del usuario, sólo se proponen con el objetivo de mantener el estándar utilizado hasta el momento.

- De acuerdo a las asignaturas de cada año.

Tabla 1. Abreviatura de las Asignaturas de 1er año

1er Año			
Semestre 1		Semestre 2	
Idioma Extranjero I	IE1	Idioma Extranjero II	IE2
Introducción a la Programación	IP	Programación I	P1
Matemática I	M1	Matemática II	M2
Matemática Discreta I	MD1	Matemática Discreta II	MD2
Algebra Lineal I	AL1	Algebra Lineal 2	AL2
Defensa Nacional	DN	Defensa Nacional	DN
		Panorama Histórico Cultural de Cuba	PHCCU

Fuente: Elaboración propia.

Tabla 2. Abreviatura de las Asignaturas de 2do año.

2do Año			
Semestre 1		Semestre 2	
Economía Política	EP	Economía Política	EP
Física I	F1	Física II	F2
Idioma Extranjero III	IE3	Idioma Extranjero IV	IE4
Máquina Computadora I	MC1	Máquina Computadora II	MC2
Matemática III	M3	Matemática IV	M4
Programación II	P2	Base de Datos	BD

Fuente: Elaboración propia.

Tabla 3. Abreviatura de las Asignaturas de 3er año.

3er Año			
Semestre 1		Semestre 2	
Problemas Sociales de la Ciencia y la Tecnología	PSCT	Administración de Empresas	AE
Teleinformática I	TL1	Teleinformática II	TL2
Ingeniería de Software I	ISW1	Ingeniería de Software II	ISW2
Contabilidad y Finanzas	CF	Gráfico por Computadoras	GXC
Programación III	P3	Programación IV	P4
Sistemas Operativos	SO	Probabilidades y Estadísticas	PE

Fuente: Elaboración propia.

Tabla 4. Abreviatura de las Asignaturas de 4to año.

4to Año			
Semestre 1		Semestre 2	
Comercio Electrónico	CE	Ingles V	IE5
Gestión de Software	GSW	Inteligencia Artificial	IA
Investigación de Operaciones	IO	Metodología de la Investigación Científica	MIC
		Seguridad Informática	SI

Fuente: Elaboración propia.

Tabla 5. Abreviatura de las Asignaturas de 5to año.

5to Año	
Semestre 1	
Formación Pedagógica	FP
Ética Informática	EI
Historia de la Informática	HI

Fuente: Elaboración propia.

Tabla 6. Otras abreviaturas utilizadas

Departamentos de las asignaturas	Abreviatura
Humanidades	H
Ciencias Básicas	CB
Sistemas Digitales	SD
Programación	P
Ingeniería de Software	ISW
Tipo de profesor	
Externo	E
Externo - vinculado a la producción	EVP
Interno	I
Interno - vinculado a la producción	IVP
Adjunto	Ad
Prestación de Servicio	PS
Sesión	
Tarde	T
Mañana	M
Sesión Completa	X
Tipo de Frecuencia	
Conferencia	C
Clase Práctica	CP

Clase Teórico-Práctico	CTP
Seminario	S
Taller	T
Examen	E
Días de la semana	
Lunes	L
Martes	M
Miércoles	Mi
Jueves	J
Viernes	V
Sábado	S
Tipo de local	
Aula	A
Salón de Conferencia	SC
Laboratorio	L

Fuente: Elaboración propia.

2.9. Fases de desarrollo

La metodología XP posee algunas similitudes con el Proceso Unificado de Desarrollo (RUP) y en algún modo puede ser considerado una versión abreviada del RUP con modificaciones, sin embargo XP no utiliza representaciones gráficas (diagramas) para el análisis y diseño de los procesos. Basado en los métodos que utiliza XP, se espera que el código producido sea de alta calidad. El ciclo de vida ideal consiste de seis, ellas son: Exploración, Planificación de la Entrega, Iteraciones, Producción, Mantenimiento y Muerte del Proyecto (Campos, 2004).

2.9.1. Fase: Exploración

En esta fase, los clientes plantean a grandes rasgos las Historias de Usuario (HU) que son de interés para la primera entrega del producto. Al mismo tiempo el equipo de desarrollo se familiariza con las herramientas, tecnologías y prácticas que se utilizarán en el proyecto. Se prueba la tecnología y se exploran las posibilidades de la arquitectura del sistema construyendo un prototipo. La fase de exploración toma de pocas semanas a pocos meses, dependiendo del tamaño y familiaridad que tengan los programadores con la tecnología (Letelier, y otros, 2004).

2.9.1.1. Historias de Usuario

Las Historias de Usuario (HU) son la técnica utilizada en XP, representan una breve descripción del comportamiento del sistema, emplea terminología del cliente sin lenguaje técnico, se realiza una por cada característica principal del sistema, se emplean para hacer estimaciones de tiempo y para el plan de lanzamientos, son la base para las pruebas de unidad y presiden la creación de las pruebas de aceptación.

Se trata de tarjetas en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. Las HU son descompuestas en tareas de programación y asignadas a los programadores para ser implementadas durante una iteración. Respecto a la información contenida en la historia de usuario, existen varias plantillas sugeridas pero no existe un consenso al respecto. (Letelier, y otros, 2004). De acuerdo a la investigación realizada se propone el uso de la siguiente estructura para realizar la historia de usuario.

Tabla 7. Plantilla de la historia de usuarios.

Historia de Usuario	
Nombre Historia de Usuario:	
Número:	**Usuario:**
Dependiente:	**Iteración Asignada:**
Prioridad:	**Puntos Estimados:**
Descripción:	
Observaciones:	

- Número: Identificador de la HU.
- Usuario: Creador de la HU.
- Puntos Estimados: Las estimaciones de esfuerzo asociado a la implementación de las historias la establecen los programadores utilizando como medida el punto. Un punto, equivale a una semana ideal de programación. Las historias generalmente valen de 1 a 3 puntos.
- Prioridad: Prioridad en la implementación de la HU respecto al resto de las HU, puede ser de tres tipos: alta, media o baja.
- Dependiente: Una HU no debería ser dependiente de otra historia, pero a veces es inevitable.
- Tipo de actividad: Puede ser de tres tipos: nueva, corrección o mejora.
- Descripción: Descripción sintetizada de la HU.
- Observaciones: Información de interés.

Durante este proceso se identifican cinco HU, las cuales se detallan a continuación.

Tabla 8. HU: Recibir parámetros de entrada.

Historia de Usuario	
Nombre Historia de Usuario: Recibir parámetros de entrada.	
Número: 1	**Usuario:** Cliente
Dependiente: -	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 1
Descripción: El usuario debe introducir los datos a la aplicación, que incluye los grupos, los profesores, los locales, el semestre y la sesión en que se van a planificar las actividades, es decir llenar los ficheros que se definieron en la carpeta “Bases de Datos-Entrante”.	
Observaciones: Para recibir esta información la Biblioteca de Algoritmos debe estar integrada a la aplicación web Ghordo y deben existir varias interfaces destinadas a estas operaciones.	

Tabla 9. HU: Gestionar afectaciones.

Historia de Usuario	
Nombre Historia de Usuario: Gestionar afectaciones.	
Número: 2	**Usuario:** Cliente
Dependiente: -	**Iteración Asignada:** 1
Prioridad: Alta	**Puntos Estimados:** 1
Descripción: Se coleccionan y atienden las afectaciones de los profesores, grupos y locales durante las semanas a planificar, las cuales impiden asignar actividades en un día y en un turno determinado.	
Observaciones: Para recibir esta información la Biblioteca de Algoritmos debe estar integrada a la aplicación web Ghordo y deben existir varias interfaces destinadas a estas operaciones.	

Tabla 10. HU: Generar encuentros.

Historia de Usuario	
Nombre Historia de Usuario: Generar encuentros.	
Número: 3	**Usuario:** Cliente
Dependiente: 1, 2.	**Iteración Asignada:** 2
Prioridad: Alta	**Puntos Estimados:** 5
Descripción: Crea el horario docente que cumple con las restricciones fuertes e intenta dar cumplimiento a las restricciones débiles, estableciendo los encuentros entre grupos, profesores, asignaturas y locales.	
Observaciones: El orden de planificación de cada año se indica en el fichero “Prioridad”. Todos los años no tienen que comenzar a planificarse desde la semana inicial del curso, o sea, puede darse el caso de que primer y segundo año comiencen las actividades docentes desde la primera semana del curso y que el resto de los años comiencen una o varias semanas después.	

Tabla 11. HU: Devolver horario general.

Historia de Usuario	
Nombre Historia de Usuario: Devolver horario general	
Número: 4	**Usuario:** Cliente
Dependiente: 3	**Iteración Asignada:** 3
Prioridad: Alta	**Puntos Estimados:** 3
Descripción: Se organizan todos los encuentros generados por año y se devuelve el fichero "Horario" y el fichero "Lista_Encuentros_General".	
Observaciones: El fichero generado devuelve la información de cada uno de los encuentros generados de una forma estructurada, o sea, en el formato utilizado para la publicación del horario para los estudiantes, para cada una de las semanas a planificar.	

Tabla 12. HU: Generar reporte de restricciones débiles.

Historia de Usuario	
Nombre Historia de Usuario: Devolver valores de Función Objetivo.	
Número: 5	**Usuario:** Cliente
Dependiente: 4	**Iteración Asignada:** 3
Prioridad: Alta	**Puntos Estimados:** 2
Descripción: Se crea el fichero "Valores_Funcion_Objetivo" dando una idea de la calidad del horario generado para cada unos de los años.	
Observaciones: En este fichero se devuelve el número de violaciones de cada una de las restricciones débiles empleadas en la función objetivo, así como el valor general correspondiente al total de violaciones presentes.	

2.9.2. Fase: Planificación de la Entrega

En esta fase el cliente establece la prioridad de cada historia de usuario, y correspondientemente, los programadores realizan una estimación del esfuerzo necesario para implementar cada una de ellas. Esto se expresa utilizando como medida el punto, el cual se considera como una semana ideal de trabajo. Por otra parte, el equipo de desarrollo mantiene un registro de la "velocidad" de desarrollo, establecida en puntos por iteración, basándose principalmente en la suma de puntos correspondientes a las HU que fueron terminadas en la última iteración.

La planificación se puede realizar basándose en el tiempo o el alcance. La velocidad del proyecto es utilizada para establecer cuántas historias se pueden implementar antes de una fecha determinada o cuánto tiempo tomará implementar un conjunto de historias, además permiten determinar si una iteración está sobrecargada. Al planificar por tiempo, se multiplica el número de iteraciones por la velocidad del proyecto, determinándose cuántos puntos se pueden completar. Al planificar según alcance del sistema, se divide la suma de puntos de las HU seleccionadas entre la

velocidad del proyecto, obteniendo el número de iteraciones necesarias para su implementación (Campos, 2004) (Letelier, y otros, 2004).

2.9.2.1. Estimación de esfuerzo por Historias de Usuario

A continuación se presenta la estimación para cada una de las HU identificadas:

Tabla 13. Estimación de esfuerzo por Historias de Usuario.

No.	Historias de Usuarios	Puntos de Estimación
1	Recibir parámetros de entrada.	1
2	Gestionar afectaciones.	1
3	Generar encuentros.	5
4	Devolver horario general.	3
5	Devolver valores de Función Objetivo.	2

2.9.3. Fase: Iteraciones

Esta fase incluye varias iteraciones sobre el sistema antes de ser entregado, cada iteración no debe ser más extensa que tres semanas. Este plan define cuáles HU serán implementadas para cada iteración del sistema y las posibles fechas para estas liberaciones. Al final de la última iteración el sistema estará listo para entrar en producción (Campos, 2004).

2.9.3.1. Plan de iteraciones

El plan de iteraciones especifica las HU que serán implementadas en cada iteración del sistema y las posibles fechas para las liberaciones. Se definieron tres iteraciones para la realización del sistema, las cuales se mencionan a continuación:

- Iteración 1: Esta iteración tiene como objetivo la implementación de la HU de número 1 y 2, las cuales garantizan la entrada de los parámetros necesarios para comenzar el trabajo de las próximas iteraciones, así como las primeras restricciones fuertes que se deben cumplir.
- Iteración 2: El objetivo de esta iteración es la implementación de las HU número 3, que constituyen el uno de los pilares fundamentales de la Biblioteca de Algoritmos, debido a que permite la creación del horario en función de los profesores y asignaturas asociados a los grupos de un año específico.
- Iteración 3: En esta última iteración se implementarán las HU número 4 y 5, que devuelve el horario general formado por la combinación de los grupos docentes de los diferentes años,

profesores, asignaturas y locales, así como los valores de la FO para cada uno de los años planificados.

2.9.3.2. Plan de duración de las iteraciones

Este plan permite al equipo de desarrollo definir la duración de cada una de las iteraciones, además de mostrar el orden en que serán implementadas las HU.

Tabla 14. Plan de duración de las iteraciones.

Iteraciones	Orden de las Historias de Usuario a implementar	Duración total de las Iteraciones
1	Recibir parámetros de entrada.	1 semana
1	Gestionar afectaciones.	1 semana
2	Generar encuentros.	5 semanas
3	Devolver horario general.	3 semanas
3	Devolver valores de Función Objetivo.	2 semanas

2.9.3.3. Plan de Entregas

A través del plan de entregas clientes y desarrolladores determinan la fecha de entrega de las versiones del producto en las iteraciones estimadas, definiendo el orden en que las HU serán implementadas.

Tabla 15. Plan de Entregas.

Historia de Usuario	Final 1ra Iteración	Final 2da Iteración	Final 3ra Iteración
Recibir parámetros de entrada.	X		
Gestionar afectaciones.	X		
Generar encuentros.		X	
Devolver horario general.			X
Devolver valores de Función Objetivo.			X

2.9.3.4. Plan de Tareas

El Plan de Tareas permite descomponer cada HU en varias tareas que serán implementadas para dar cumplimiento a la misma, facilitando el trabajo de implementación de los desarrolladores.

Tabla 16. Plan de Tareas.

Historias de Usuario	Tareas
Recibir parámetros de entrada.	1. Recibir años a planificar organizados consecutivamente. 2. Recibir las afectaciones. 3. Recibir información de grupos, profesores y asignaturas. 4. Recibir información de locales existentes. 5. Recibir información de la planificación de las asignaturas. 6. Recibir información del tipo de planificación a realizar.
Gestionar afectaciones.	7. Llenar listado de turnos válidos.
Generar encuentros.	8. Saber disponibilidad de locales. 9. Crear encuentros. 10. Añadir encuentros creados a la lista general de encuentros.
Devolver horario general.	11. Devolver fichero “Lista_Encuentros_General”. 12. Devolver fichero “Horario”.
Devolver valores de FO.	13. Devolver fichero “Valores_Funcion_Objetivo”.

A continuación se especifican las tareas para una mejor comprensión de las mismas.

Tabla 17. Tarea #1 Recibir años a planificar organizados consecutivamente.

Tarea	
No. Tarea: 1	**No. Historia de Usuario:** Recibir parámetros de entrada.
Nombre de la tarea: Recibir años a planificar organizados consecutivamente.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: A través del fichero “Prioridad” el usuario debe especificar los años que serán planificados, el semestre, la sesión así como el algoritmo que desee utilizar para planificar.	

Tabla 18. Tarea #2 Recibir las afectaciones.

Tarea	
No. Tarea: 2	**No. Historia de Usuario:** Recibir parámetros de entrada.
Nombre de la tarea: Recibir las afectaciones	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: A través del fichero “Listas_Afectaciones” el usuario debe especificar las afectaciones de los grupos, locales, departamentos y profesores, las cuales pueden ser para uno o más turnos, en un día o más días. No se puede asignar ninguna actividad durante el o los turnos afectados.	

Tabla 19. Tarea #5 Recibir información de grupos, profesores y asignaturas.

Tarea	
No. Tarea: 3	**No. Historia de Usuario:** Recibir parámetros de entrada.
Nombre de la tarea: Recibir información de grupos, profesores y asignaturas.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: A través del fichero "Grupos_Profesores_Asignaturas" se recibe la información detallada de todos los grupos, profesores y asignaturas.	

Tabla 20. Tarea #7 Recibir información de locales existentes.

Tarea	
No. Tarea: 4	**No. Historia de Usuario:** Recibir parámetros de entrada.
Nombre de la tarea: Recibir información de locales existentes.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: A través del fichero "Locales" se recibe la información más detallada de cada uno de los locales con los que se cuenta para realizar la planificación.	

Tabla 21. Tarea #8 Recibir información de la planificación de las asignaturas.

Tarea	
No. Tarea: 5	**No. Historia de Usuario:** Recibir parámetros de entrada.
Nombre de la tarea: Recibir información de la planificación de las asignaturas.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: A través del fichero "P1s" se recibe la información perteneciente a la planificación de de cada una de las asignaturas (P1) de los años que se deseen planificar.	

Tabla 22. Tarea #9 Recibir información del tipo de planificación a realizar.

Tarea	
No. Tarea: 6	**No. Historia de Usuario:** Recibir parámetros de entrada.
Nombre de la tarea: Recibir información del tipo de planificación a realizar.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: A través del fichero "Cantidad_Semanas_Planificar" se describe que cantidad de semanas el usuario desea planificar, ya sea una en específica o un rango de semanas.	

Tabla 23. Tarea #10 Llenar listado de turnos válidos.

Tarea	
No. Tarea: 7	**No. Historia de Usuario:** Gestionar afectaciones.
Nombre de la tarea: Llenar listado de turnos válidos.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: Una vez cargadas todas las afectaciones se pasa a llenar los listados de turnos validos de los grupos, locales y fundamentalmente de los profesores que son casi siempre los que mayor cantidad de afectaciones presentan por diversas razones.	

Tabla 24. Tarea #11 Saber disponibilidad de locales.

Tarea	
No. Tarea: 8	**No. Historia de Usuario:** Generar encuentros.
Nombre de la tarea: Saber disponibilidad de locales.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: Es de suma importancia a la hora que crear un encuentro, ya que sin un local disponible sería imposible de realizarse el mismo, esta tarea se realizara antes de crear cada uno de los encuentros del horario.	

Tabla 25. Tarea #12 Crear encuentros.

Tarea	
No. Tarea: 9	**No. Historia de Usuario:** Generar encuentros.
Nombre de la tarea: Crear encuentros.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: Luego de cumplir con una serie de restricciones tanto débiles como todas las fuertes y tener un local libre es creado el encuentro, que no es más que la combinación de un profesor que imparte una asignatura a un grupo en un local determinado.	

Tabla 26. Tarea #13 Añadir encuentros creados a la lista general de encuentros.

Tarea	
No. Tarea: 10	**No. Historia de Usuario:** Generar encuentros.
Nombre de la tarea: Añadir encuentros creados a la lista general de encuentros.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: Luego de creado el encuentro el mismo pasa es insertado en una lista general que es donde se van guardando todos los encuentros a medida de que son creados para su posterior devolución.	

Tabla 35. Tarea #13 Devolver fichero “Lista_Encuentros_General”.

Tarea	
No. Tarea: 11	**No. Historia de Usuario:** Devolver horario general.
Nombre de la tarea: Devolver fichero “Lista_Encuentros_General”.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: Se devuelve el fichero “Lista_Encuentros_General” que contiene todos los encuentros creados, el cual es el que será utilizado por la aplicación web Ghordo para mostrarles el horario a la facultad.	

Tabla 27. Tarea #14 Devolver fichero "Horario".

Tarea	
No. Tarea: 12	**No. Historia de Usuario:** Devolver horario general.
Nombre de la tarea: Devolver fichero "Horario".	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: Se crea y devuelve el fichero "Horario", a partir de los encuentros guardados en la lista general de encuentros.	

Tabla 28. Tarea #18 Devolver fichero "Valores_Funcion_Objetivo".

Tarea	
No. Tarea: 13	**No. Historia de Usuario:** Devolver valores de Función Objetivo.
Nombre de la tarea: Devolver.	
Tipo de tarea: Desarrollo	**Puntos estimados:**
Fecha inicio:	**Fecha fin:**
Programador responsable: Isabel y Humberto	
Descripción: Se crea y devuelve el fichero "Valores_Funcion_Objetivo", que recoge el año y el valor que arroja el aplicar un determinado algoritmo en dicho año.	

2.9.4. Fase: Producción

La fase de producción requiere de pruebas adicionales y revisiones de rendimiento antes de que el sistema sea trasladado al entorno del cliente. Al mismo tiempo, se deben tomar decisiones sobre la inclusión de nuevas características a la versión actual, debido a cambios durante esta fase. Las ideas que han sido propuestas y las sugerencias son documentadas para su posterior implementación.

2.9.4.1. Tarjetas Clase - Responsabilidad – Colaborador

Tabla 29. Tarjeta CRC: Clase Script de Control.

Clase Script de Control	
Responsabilidades	**Clases relacionadas**
Cargar todos los datos de la base de datos. Inicializar las clases. Llenar las listas que se usan en la ejecución del código. Controlar el orden de ejecución de los métodos. Crear los ficheros resultantes de la generación del horario.	Controladora

Tabla 30. Tarjeta CRC: Clase Controladora.

Clase Controladora	
Responsabilidades	**Clases relacionadas**
Maneja las clases y manipula de forma general los todos los datos. Contiene los principales métodos.	Todas las clases

Tabla 31. Tarjeta CRC: Clase Asignatura.

Clase Asignatura	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a las asignaturas como el nombre, el año y el semestre a que pertenece.	Controladora Script

Tabla 32. Tarjeta CRC: Clase Encuentros.

Clase Encuentro	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a los encuentros, entiéndase por esto como los topes entre los grupos, asignaturas, profesores, y un local dado un día y un turno valido.	Controladora Script

Tabla 33. Tarjeta CRC: Clase Grupo.

Clase Grupo	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a los Grupos como el identificador el año y las listas de los profesores que pertenecen a dicho grupo.	Controladora Script

Tabla 34. Tarjeta CRC: Clase Local.

Clase Local	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a los Locales como el identificador, el tipo de local y la capacidad.	Controladora Script

Tabla 35. Tarjeta CRC: Clase Profesor.

Clase Profesor	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a los profesores: nombre, apellidos, asignatura y tipo de profesor.	Controladora Script

Tabla 36. Tarjeta CRC: Clase P1.

Clase P1	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a los P1, que incluye el nombre de las asignaturas y una lista que contiene las semanas con todas las actividades planificadas.	Controladora Script Semanas

Tabla 37. Tarjeta CRC: Clase Semanas.

Clase Semana	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a las semanas: el número de las semanas y una lista de frecuencias de cada semana.	P1 Frecuencias

Tabla 38. Tarjeta CRC: Clase Frecuencias.

Clase Frecuencia	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a las frecuencias: el tipo de frecuencia, cantidad de frecuencias y el tipo de local en que se imparte la actividad.	Semanas

Tabla 39. Tarjeta CRC: Clase Instancia2.

Clase Instancia2	
Responsabilidades	**Clases relacionadas**
Contiene la información referente a las combinaciones día-turno para el control de los mismos.	Controladora Script

2.9.4.2. Pruebas

La realización de las pruebas es uno de los aspectos esenciales de la metodología XP, permite detectar desde las primeras iteraciones los errores que pueden surgir, contribuyendo a reducirlos y a obtener un programa con mayor calidad que es capaz de aceptar los cambios conforme pasa el tiempo. Las pruebas se dividen en dos categorías: pruebas unitarias, llevadas a cabo por los desarrolladores con el objetivo de verificar la funcionalidad del código, y las pruebas de aceptación, que verifica que la funcionalidad desarrollada. A continuación se muestran los casos de prueba de aceptación para las HU implementadas:

Tabla 40. CPA-01 Recibir parámetros de entrada.

Caso de Prueba de Aceptación	
Código: CPA-01	**No. Historia de Usuario:** 1
Nombre: Recibir parámetros de entrada.	
Descripción: Se obtienen una serie de datos necesarios para el posterior desarrollo del programa, como: los años a planificar, los cuales deben tener un orden de prioridad, la información referente a los profesores, locales, grupos, las listas de profesores de cada grupo, así como las listas de las posibles afectaciones (Ver Anexo 2).	
Condiciones de Ejecución: Todos los datos mencionados anteriormente deben estar previamente almacenados en bases de datos, que serán llenadas por la aplicación web Ghordo.	
Entrada/Pasos de Ejecución: Existe una clase llamada “Script_de_Control.py” encargada de cargar todos los ficheros “.csv” que contienen la información necesaria.	
Resultado esperado: Tener en memoria todos los datos necesarios para la posterior ejecución del programa.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 41. CPA-02 Gestionar afectaciones.

Caso de Prueba de Aceptación	
Código: CPA-02	**No. Historia de Usuario:** 2
Nombre: Gestionar afectaciones.	
Descripción: Se obtienen un conjunto de afectaciones del tipo día-turno, para profesores, grupos docentes y locales (Ver Anexo 3).	
Condiciones de Ejecución: Todos los datos mencionados anteriormente deben estar previamente almacenados en bases de datos, que serán llenadas por la aplicación web Ghordo.	
Entrada/Pasos de Ejecución: Existe una clase llamada “Script_de_Control.py” encargada de cargar todos los ficheros “.csv” que contienen la información necesaria.	
Resultado esperado: Tener en memoria todas las afectaciones, para poder llenar las listas de turnos validos de cada una de las clases que lo necesiten para la posterior ejecución del programa.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 42. CPA-03 Generar horario de un año.

Caso de Prueba de Aceptación	
Código: CPA-03	**No. Historia de Usuario:** 3
Nombre: Generar horario de un año.	
Descripción: Es una funcionalidad vital, ya que trabaja con todos los datos que hasta el momento se tienen en memoria, revisa las restricciones fuertes, de estricto cumplimiento, y otras débiles, para finalmente crear encuentros, que no son más que los topes de grupos con asignaturas y profesores en un día y turno determinado.	
Condiciones de Ejecución: Se deben haber cargado en memoria todos los datos con anterioridad.	
Entrada/Pasos de Ejecución: Se crean los encuentros.	
Resultado esperado: Tener una lista general con todos los encuentros creados que posteriormente serán interpretados y guardados en los ficheros correspondientes.	
Evaluación de la prueba: Todavía se encuentra en fase de prueba**.**	

Tabla 43. CPA-03 Devolver horario general.

Caso de Prueba de Aceptación	
Código: CPA-04	**No. Historia de Usuario:** 4
Nombre: Devolver horario general.	
Descripción: Toma la lista general de encuentros creados y crea los ficheros “Horario” y “Lista_Encuentros_General”.	
Condiciones de Ejecución: Se deben haber creados todos los encuentros de todos los años que se mandaron a planificar.	
Entrada/Pasos de Ejecución: Se crean los ficheros correspondientes.	
Resultado esperado: Tener una lista general con todos los encuentros creados que posteriormente serán interpretados y guardados en los ficheros correspondientes.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 44. CPA-03 Devolver valores de Función Objetivo.

Caso de Prueba de Aceptación	
Código: CPA-05	**No. Historia de Usuario:** 5
Nombre: Devolver valores de Función Objetivo.	
Descripción: Toma la lista de los encuentros generados para cada uno de los años y calcula el valor de la FO, así como la ocurrencia de cada uno de los indicadores de calidad y todos estos valores los guarda en el fichero “Valores_Funcion_Objetivo”.	
Condiciones de Ejecución: Se deben haber creados todos los encuentros de todos los años que se mandaron a planificar.	
Entrada/Pasos de Ejecución: Se crea el fichero “Valores_Funcion_Objetivo”.	
Resultado esperado: Tener los valores de calidad para el horario generado.	
Evaluación de la prueba: Prueba satisfactoria.	

2.9.5. Fase: Mantenimiento

Mientras la primera versión se encuentra en producción, el proyecto XP debe mantener el sistema en funcionamiento al mismo tiempo que desarrolla nuevas iteraciones. Para realizar esto se requiere de tareas de soporte para el cliente. De esta forma, la velocidad de desarrollo puede bajar después de la puesta del sistema en producción. La fase de mantenimiento puede requerir nuevo personal dentro del equipo y cambios en su estructura (Letelier, y otros, 2004).

2.9.6. Fase: Muerte del Proyecto

Es cuando el cliente no tiene más historias para ser incluidas en el sistema, esto requiere que se satisfagan las necesidades del cliente. Se genera la documentación final del sistema y no se realizan más cambios. La muerte del proyecto también ocurre cuando el sistema no genera los beneficios esperados por el cliente o cuando no hay presupuesto para mantenerlo (Letelier, y otros, 2004).

2.10. Conclusiones Parciales

En este capítulo se especifican las características fundamentales de la propuesta de solución, detallando las restricciones fuertes a cumplir por un horario para que pueda ser considerado factible, así como las restricciones débiles, cuyo grado de satisfacción define la calidad del mismo. De igual forma se definieron las historias de usuarios y la estructura de la entrada y salida de datos de la Biblioteca de Algoritmos.

Para dar solución al problema planteado se implementaron cinco algoritmos, cuatro de ellos priorizan algún criterio en particular, con el objetivo de devolver soluciones iniciales factibles que serán utilizadas por el quito algoritmo que basa su funcionamiento en la metaheurística Gran Diluvio, mediante la cual se realiza un conjunto de transformaciones para intentar mejorar la solución inicial.

La función objetivo descrita se utiliza para evaluar la calidad del horario generado y permite, a partir de la realización de varios casos de prueba, comparar los resultados obtenidos de acuerdo a los algoritmos implementados para el análisis de los resultados, que se detalla en el próximo capítulo.

CAPÍTULO 3: ANÁLISIS DE LOS RESULTADOS

3.1 Introducción

Una vez presentada la propuesta de solución al Problema de la Asignación de Horarios de Clases (PAHC) para la Facultad 15 de la Universidad de las Ciencias Informáticas (UCI), en el presente capítulo se realiza el análisis de los resultados, así como la viabilidad de la investigación desde el punto de vista de costos y beneficios.

3.2 Casos de prueba

La información empleada en las pruebas se corresponde a datos reales de la Facultad 15, para el segundo semestre del curso teniendo en cuenta cuatro años de la carrera, los datos de prueba son los siguientes:

- Se cuenta con 24 aulas, 5 salones de conferencia y 11 laboratorios.
- Se imparten 7 asignaturas a primer año, 6 para segundo, 5 para tercero y 4 para cuarto, resultando un total de 22 asignaturas, sin incluir Educación Física que es para todos los años.
- En primer año existen 16 grupos, en segundo 13, en tercero 11 y en cuarto 14, para un total de 54 grupos.
- Se cuenta con 81 profesores para primer año, 57 para segundo, 31 para tercero y 31 para cuarto, para un total de 200 profesores.

A continuación se muestran los valores de las pruebas realizadas, de acuerdo a la función objetivo, para cada uno de los años por separado y para la semana 1 del curso.

Donde:

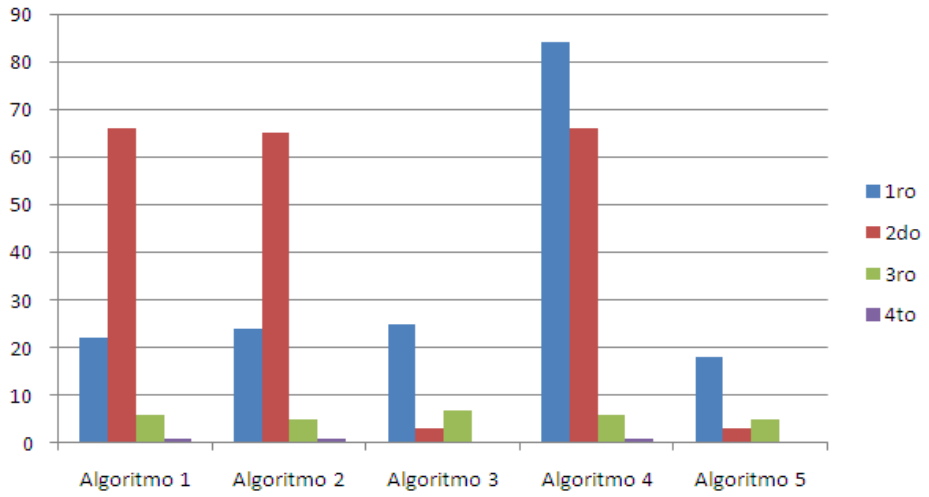
- VFO: Valor del cálculo de la función objetivo.
- DE: Desviación Estándar.
- M: Media.
- Alg: Algoritmo.
- Rp: Significa que generó algún reporte al ejecutarse mediante ese algoritmo.

Tabla 45. Resultados de la función objetivo para una semana

Año	Alg. 1 (VFO)	Alg. 2 (VFO)	Alg. 3 (VFO)	Alg. 4 (VFO)	Alg. 5 (VFO)
1ro	22	24	25	84	**18**
2do	66	65	3	66	**3**
3ro	6	5	7	6	**5**
4to	1	1	0	1	**0**
M	**23,75**	**23,75**	**8,75**	**39,25**	**6,5**
DE	**873,583333**	**856,916667**	**125,5833333**	**1762,25**	**63**

Fuente: Elaboración propia.

Figura 4. Resultados de la función objetivo para una semana.



Fuente: Elaboración propia.

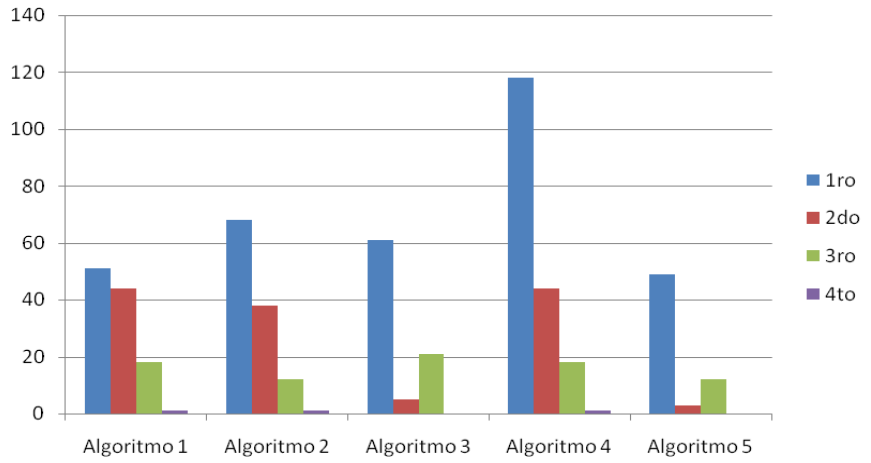
La siguiente tabla contiene los valores de las pruebas realizadas, de acuerdo a la función objetivo, para cada uno de los años por separado y para un rango de semanas, desde la 2 a la 4.

Tabla 46. Resultados de la función objetivo para un rango de semanas de la 2 a la 4.

Año	Alg.. 1 (VFO)	Alg.. 2 (VFO)	Alg. 3 (VFO)	Alg. 4 (VFO)	Algoritmo 5 (VFO)
1ro	51	68	61	118-**RP**	**49**
2do	44	38	5	44	**3**
3ro	18	12	21	18	**12**
4to	1	1	0	1	**0**
M	**28,5**	**29,75**	**21,75**	**45,25**	**16**
DE	**537,666667**	**890,916667**	**764,9166667**	**2664,916667**	**510**

Fuente: Elaboración propia.

Figura 5. Resultados de la función objetivo para un rango de semanas de la 2 a la 4.



Fuente: Elaboración propia.

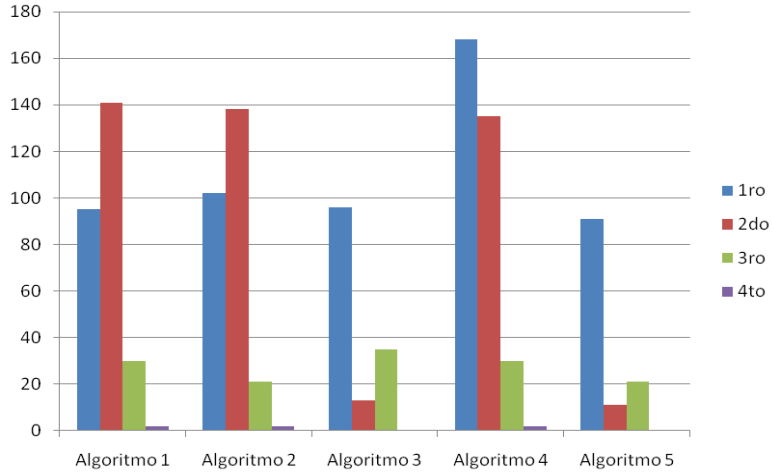
Los resultados del próximo caso de prueba se corresponden a los valores del cálculo de la función objetivo, para los cuatro años juntos y para un rango de semanas, desde la 1 a la 5.

Tabla 47. Resultados de la función objetivo para un rango de semanas de la 1 a la 5.

Año	Alg. 1 (VFO)	Alg. 2 (VFO)	Alg. 3 (VFO)	Alg. 4 (VFO)	Alg. 5 (VFO)
1ro	95	102	96	168-**RP**	**91**
2do	141-**RP**	138-**RP**	13	135-**RP**	**11**
3ro	30	21	35	30	**21**
4to	2	2	0	2	**0**
Total	**268**	**263**	**144**	**335**	**123**
M	**67**	**65,75**	**36**	**83,75**	**30,75**
DE	**3951,33333**	**4200,25**	**1808,666667**	**6432,25**	**1686,916667**

Fuente: Elaboración propia.

Figura 6. Resultados de la función objetivo para un rango de semanas de la 1 a la 5.



Fuente: Elaboración propia

Estos casos de prueba se realizaron con el objetivo de comparar los resultados obtenidos durante la generación del horario a través de los algoritmos implementados y para determinar cuál devuelve la mejor solución. Se evidencia que uno de ellos puede ofrecer mejores resultados para un año con respecto al resto y que para algunos años se pueden generar reportes al crear el horario, debido a que el orden en que se planifican las asignaturas no es el mismo para ninguno de los cuatro algoritmos. Por su parte el Alg. 5 toma como punto de partida los valores del algoritmo que devuelva los mejores resultados de acuerdo a la función objetivo, garantizando que no existan reportes de ningún tipo, ya que parte de una solución factible con el objetivo de mejorarla.

A continuación se muestran algunas tablas que reflejan la cantidad de violaciones de los indicadores de calidad, obtenidas a partir del cálculo de la función objetivo, durante la planificación de los cuatro años juntos para el rango de semanas desde la uno hasta la cinco.

- Valores obtenidos al utilizar el Alg. 1:

Tabla 48. Violaciones obtenidas para el algoritmo 1.

Año	Alg. 1	Clases los Sábados	Profesores VP a 2da y 5ta	Clases CB y P a 3ra o 6ta	Clases CB un mismo día	Valor de la FO
1ro	1	16	0	65	14	**95**
2do	1	35	0	56	0	**141-RP**
3ro	1	0	0	30	0	**30**
4to	1	0	0	2	0	**2**
Total		**51**	**0**	**153**	**14**	**268**

Fuente: Elaboración propia.

En caso de que exista algún reporte se le suma 50 al valor de la función objetivo, para que pierda calidad en comparación con los valores obtenidos por los demás algoritmos. Esta situación se evidencia en la Tabla 53 cuando se planifica 2do año utilizando el algoritmo 1, por lo que para este juego de datos no resulta factible la utilización este algoritmo.

- Valores obtenidos al aplicar el Alg. 2:

Tabla 49. Violaciones obtenidas para el algoritmo 2.

Año	Alg. 2	Clases los Sábados	Profesores VP a 2da y 5ta	Clases CB y P a 3ra o 6ta	Clases CB un mismo día	Valor de la FO
1ro	2	17	0	70	15	**102**
2do	2	38	0	50	0	**138-RP**
3ro	2	0	0	21	0	**21**
4to	2	0	0	2	0	**2**
Total		**55**	**0**	**143**	**15**	**263**

Fuente: Elaboración propia.

Al aplicar el Alg. 2 se puede apreciar de que 2do año continúa presentando problemas, ya que genera un horario con reportes, por lo que tampoco es factible dicho algoritmo para este año. Sin embargo, para 3er año arroja un menor resultado en comparación con el devuelto por el Alg. 1, esto se debe a que para este año es más importante priorizar las asignaturas con mayor cantidad de frecuencias que las asignaturas con mayor cantidad de profesores externos.

- Valores obtenidos al aplicar el Alg. 3:

Tabla 50. Violaciones obtenidas para el algoritmo 3.

Año	Alg. 3	Clases los Sábados	Profesores VP a 2da y 5ta	Clases CB y P a 3ra o 6ta	Clases CB un mismo día	Valor de la FO
1ro	3	15	0	61	20	**96**
2do	3	5	0	8	0	**13**
3ro	3	0	0	35	0	**35**
4to	3	0	0	0	0	**0**
Total		**20**	**0**	**104**	**20**	**144**

Fuente: Elaboración propia.

En esta tabla se evidencia que aplicar el Alg. 3 se obtienen buenos valores para todos los años, sin tener ningún reporte, a pesar de que no siempre puede darse esta situación, ya que el mismo depende del orden de prioridad que se le de a las asignaturas.

- Valores obtenidos al aplicar el Alg. 4:

Tabla 51. Violaciones obtenidas para el algoritmo 4.

Año	Alg. 4	Clases los Sábados	Profesores VP a 2da y 5ta	Clases CB y P a 3ra o 6ta	Clases CB un mismo día	Valor de la FO
1ro	4	21	0	76	21	**168-RP**
2do	4	33	0	52	0	**135-RP**
3ro	4	0	0	30	0	**30**
4to	4	0	0	2	0	**2**
Total		**54**	**0**	**160**	**21**	**335**

Fuente: Elaboración propia.

Los valores mostrados para este caso de prueba no son favorables, demostrando que con este juego de datos el algoritmo 4 no devuelve buenos resultados.

- Valores obtenidos al aplicar el Alg. 5:

Tabla 52. Violaciones obtenidas para el algoritmo 5.

Año	Alg. 5	Clases los Sábados	Profesores VP a 2da y 5ta	Clases CB y P a 3ra o 6ta	Clases CB un mismo día	Valor de la FO
1ro	5	12	0	65	14	**91**
2do	5	3	0	8	0	**11**
3ro	5	0	0	21	0	**21**
4to	5	0	0	0	0	**0**
Total		**15**	**0**	**94**	**14**	**123**

Fuente: Elaboración propia.

Estos valores demuestran, una vez más, que mediante el uso del Alg. 5 se obtienen los mejores valores de la función objetivo. A pesar de esto se debe tener en cuenta que cada algoritmo posee sus características propias y en ocasiones el uso de uno de ellos puede devolver mejores resultados, como es el caso del Alg. 2 para 3er año, que devuelve un valor que no es mejorado por el resto de las soluciones.

3.3 Análisis de costos y beneficios

El proceso de desarrollo de software lleva implícito un costo asociado, por lo que se hace necesario realizar un análisis del costo-beneficio que implica su implementación y utilización. Los beneficios pueden ser económicos y de orden social, estos últimos tienen tanta importancia como los primeros. La Biblioteca de Algoritmos está dirigida al sector educacional, particularmente a la Facultad 15 de la Universidad de las Ciencias Informáticas (UCI), para ser integrada a la aplicación web Gestor de Horario Docente (Ghordo) que se encargará de la publicación del horario. Al agregar esta nueva funcionalidad el Ghordo podrá ser utilizado en las facultades de la universidad interesadas en automatizar este proceso, debido a que el mismo se ajusta a las restricciones y condiciones particulares de la UCI. A raíz de lo expresado se puede concluir que su beneficio principal es de orden social.

No es un producto comercial, debido a que su principal objetivo es solventar los problemas relacionados al proceso de planificación docente existentes en la Facultad 15 de la UCI, lo cual contribuye a ganar en eficiencia y perfeccionar el complejo proceso de planificación, reduciendo el tiempo y los recursos invertidos en la realización del horario docente por parte del personal encargado que lo ejecuta de forma manual. Otro resultado importante es su característica funcional, debido a que no requiere de su integración a la aplicación web Ghordo para devolver el horario docente generado en el formato estipulado para su publicación. La tecnología utilizada en el desarrollo es libre, o sea, no está sujeta a restricciones comerciales. Por lo que se puede concluir

que la implementación de la Biblioteca de Algoritmos es factible debido a los numerosos beneficios que reporta.

3.4 Funcionalidades y consideraciones finales

Entre las funcionalidades que posee la Biblioteca de Algoritmos se destacan las siguientes:

- Se puede generar el horario docente para los cinco años de la carrera.
- Administra todo tipo de afectaciones.
- Permite planificar una semana determinada o un rango de semanas.
- Permite la planificación de actividades en bloque.
- Permite planificar los exámenes.
- Permite devolver diferentes variantes del horario docente.
- Funcional: No requiere de ningún otro sistema para devolver el horario.
- Rápida generación del horario.
- Se ajusta a las condiciones de la facultad 15, al igual que al resto de las facultades de la UCI.

Para el correcto funcionamiento de la Biblioteca de Algoritmos es necesario tener en cuenta los siguientes aspectos:

- No se debe cambiar la estructura de entrada de datos.
- Todos los campos deben se llenados, excepto el caso particular de las afectaciones.
- Se debe escoger uno de los dos tipos de planificación a realizar.

3.5 Conclusiones Parciales

En este capítulo se mostraron los resultados de algunos de los casos de prueba realizados a la Biblioteca de Algoritmos, para ello se emplearon datos reales correspondientes al segundo semestre del curso 2009-2010. La información referente a los profesores, grupos docentes y locales se ajustan a la realidad de la Facultad 15 de la Universidad de las Ciencias Informáticas, que actualmente es la que posee mayor cantidad de recursos humanos.

Para la realización de los casos de prueba se utilizaron diferentes juegos de datos, con el objetivo de verificar la validez del horario generado y realizar la comparación entre los algoritmos implementados de acuerdo al número de violaciones reflejadas en los indicadores de calidad, así como los valores totales del cálculo de la función objetivo. Evidenciándose finalmente que el quinto algoritmo es el que presenta la menor cantidad de violaciones de restricciones débiles, sin embargo se debe tener en cuenta que el resto de los algoritmos devuelven horarios factibles priorizando diferentes criterios. Finalmente se confirmó la factibilidad de la Biblioteca de Algoritmos debido a los beneficios significativos que reporta.

CONCLUSIONES GENERALES

La realización de este trabajo responde a la necesidad de de buscar una solución al problema planteado en la Facultad 15 de la Universidad de las Ciencias Informáticas. Se considera validada la hipótesis de la cual se partió y cumplido el objetivo general planteado, evidenciándose en las siguientes conclusiones generales:

- El estudio del estado del arte permitió definir las técnicas de solución más adecuadas a utilizar.
- La modelación del Problema de Asignación de Horarios de Clases fue esencial en el desarrollo de la Biblioteca de Algoritmos, ya que posibilitó la comprensión del problema y sirvió como punto de partida para el diseño y desarrollo de la solución.
- Mediante el diseño y aplicación los casos de prueba se logró validar los resultados y verificar que la Biblioteca de Algoritmos cumple con las restricciones definidas, evidenciándose la factibilidad de su implementación.
- La Biblioteca de Algoritmos implementada contribuirá a facilitar y humanizar el proceso de planificación docente que actualmente se realiza de forma manual, garantizando soluciones en un tiempo razonable, reduciendo los recursos invertidos y permitiendo gestionar los posibles cambios.

RECOMENDACIONES

Se recomienda:

- Integrar la Biblioteca de Algoritmos a la aplicación web Gestor de Horario Docente (Ghordo).
- Una vez integrada la Biblioteca de Algoritmos, la aplicación debe permitir la modificación de forma manual del horario generado por parte del personal encargado de realizar la planificación docente.
- El Ghordo debe realizar la validación de los datos de entrada a la Biblioteca de Algoritmos.
- Continuar el estudio de las metaheurísticas, con el objetivo de crear nuevos algoritmos que devuelvan mejores resultados de acuerdo al cálculo de la función objetivo.
- Mejorar en el Algoritmo 5 el proceso de diversificación y de intensificación de búsqueda de soluciones, de forma tal que se explore de forma más exhaustiva el espacio de búsqueda.
- El número de profesores externos de cada grupo docente debe ser distribuido uniformemente, con el objetivo de evitar la acumulación este tipo de profesores en uno u otros grupos.
- Estudiar sistemáticamente las restricciones para determinar cuándo dejan de ser restricciones en sí o pasan de ser restricciones fuertes a débiles.
- En futuras versiones de la Biblioteca de Algoritmos se debe posibilitar la personalización de los indicadores de calidad del horario.

BIBLIOGRAFÍA

Araya, Juan Enrique Molina. 2007. Algoritmos Evolutivos para la resolución de un problema de tipo Timetabling. [En línea] 2007. [Citado el: 30 de Diciembre de 2009.] http://www.biomedica.uv.cl/profesores/rsalas/Tesis/2006-JMolina-Memoria.pdf.

Real Academia Española. 2010. *Real Academia Española.* [En línea] 2010. [Citado el: 4 de Enero de 2010.] http://buscon.rae.es/draeI/. Vigésima segunda edición.

Velásquez, D. (17 de Abril de 2009). *Curriculum y evaluación*. Recuperado el 30 de Enero de 2010, de http://curriculumyevaluacion1.blogspot.com/2009/04/que-es-la-planificacion-educativa.html

BERAZA, M. A. (Octubre de 2004). Recuperado el 31 de Enero de 2010, de Guía para la planificación didáctica de la docencia universitaria en el marco EEES: http://www.unavarra.es/conocer/calidad/pdf/guiaplan.PDF

Tamayo, S. C., Pérez Campaña, M., & Rodríguez Expósito, F. (Diciembre de 2007). *Alternativa para el proceso de planificación de horarios docentes de una Universidad.* Recuperado el 31 de Enero de 2010, de Holguín Ciencias: http://www.ciencias.holguin.cu/2007/Diciembre/articulos/ARTI6.htm

Alondra, Y. (Junio de 2009). Recuperado el 3 de Febrero de 2010, de Recocido simulado aplicado al problema de la mochila: http://it.ciidit.uanl.mx/~elisa/teaching/prog/pc/2009/proyecto_2009_alondra.pdf

(2009). Recuperado el 3 de Febrero de 2010, de Lenguajes de Programación: http://www.lenguajes-de-programacion.com/programacion-en-c.shtml

Ambite, Á. A., & de Pinto Hernández, L. (2008). Recuperado el 3 de Febrero de 2010, de El Problema del Viajante de Comercio: análisis teórico y estrategias de resolución: http://www.it.uc3m.es/jvillena/irc/practicas/07-08/ProblemaDelViajante.pdf

Ceballos Sierra, J. (2010). Recuperado el 3 de Febrero de 2010, de Agapea.com: http://www.agapea.com/libros/El-lenguaje-de-programaci--isbn-8478975004-i.htm

González Seco, J. A. (2010). Recuperado el 4 de Febrero de 2010, de Programación en castellano: http://www.programacion.com/tutorial/csharp/3/

Marzal, A., & Gracia, I. (2003). *Introducción a la programación con Python.* Recuperado el 4 de Febrero de 2010, de http://biblioteca.uci.cu/sbd/biuci/index.html

Tupia, M., & Mauricio, D. (2004). Recuperado el 10 de Febrero de 2010, de UN ALGORITMO VORAZ PARA RESOLVER EL PROBLEMA DE LA PROGRAMACIÓN DE TAREAS DEPENDIENTES EN MÁQUINAS DIFERENTES: http://sisbib.unmsm.edu.pe/BibVirtualData/publicaciones/risi/N1_2004/a03.pdf

Halim, S. (5 de Junio de 2009). *Introducción a los Algoritmos voraces*. Recuperado el 10 de Febrero de 2010, de National University of Singapore: http://www.comp.nus.edu.sg/~stevenha/myteaching/notes/7_greedy.html

Astrid Pacheco, Astrid Pacheco. 2004. Desarrollo de un modelo para la asignación de horarios de clase a través de algoritmos genéticos. *Universidad Católica Andrés Bello.* [En línea] 10 de Junio de 2004. [Citado el: 2010 de Febrero de 10.] http://www.ucab.edu.ve/tesis2004.

Gil Tallavó, Marcos y Martínez, Amadís Antonio. 2006. *Algoritmo basado en Tabú Search para el problema de asignación de horarios de clases.* [En línea] Julio de 2006. [Citado el: 11 de Febrero de 2010.] http://servicio.cid.uc.edu.ve/facyt/.

B. Cooper, T. y H. Kingston, J. 1995. Lecture Notes In Computer Science. *The Complexity of Timetable Construction Problems.* s.l.: Lecture Notes In Computer Science, 1995. Vols. Vol. 1153, pages 283–295.

Gunawan, A., Ming, K. y Leng Poh, K. 1995. A MATHEMATICAL PROGRAMMING MODEL FOR A TIMETABLING PROBLEM (2006). *Scheduling, Timetabling and Rostering - a special relationship.* Vol. 1153, pages 46-75: Lecture Notes In Computer Science, 1995.

Burke, E., y otros. 1997. The Computer Journal. *Automated University Timetabling: The State of the Art.* 1997. Vols. Vol. 40 pp 565-571, [Citado el: 11 de Febrero de 2010.] http://servicio.cid.uc.edu.ve/facyt/.

Menchaca Méndez, C. R., & García Carballeira, F. (1 de Octubre de 2000). *Revista Digital Universitaria.* Recuperado el 12 de Febrero de 2010, de Arquitectura de la Máquina Virtual Java: http://www.revista.unam.mx/vol.1/num2/art4/index.html

Schaerf, A. 1999. A survey of automated timetabling. s.l.: Universidad de Roma "La Sapienza", 1999. Vol. Vol.13, No. 2. Pages 87–127.

Burke, Edmund, y otros 2003. CiteSeer. *A TIME-PREDEFINED APPROACH TO COURSE TIMETABLING.* [En línea] Yugoslav Journal of Operations Research, 2003. [Citado el: 2010 de Febrero de 22.] http://citeseerx.ist.psu.edu/search;jsessionid=434D4BA25E0132051A21931F4588296C?q=a+time-predefined+approach+to+course+timetabling&submit=Search&sort=rel#.

G. Figueroa, Roberth, J. Solís, Camilo y A. Cabrera, Armando. 2008. *Adonisnet's Weblog. METODOLOGÍAS TRADICIONALES VS. METODOLOGÍAS ÁGILES.* [En línea] 18 de Junio de 2008. [Citado el: 22 de Febrero de 2010.] http://adonisnet.wordpress.com/?s=metodolog%C3%ADas+%C3%A1giles.

Hernández, Rodrigo, Miranda P., Jaime y A. Rey, Pablo. 2008. Revista Ingeniería de Sistemas. *Programación de Horarios de Clases y Asignación de Salas para la Facultad de Ingeniería de la Universidad Diego Portales Mediante un Enfoque de Programación Entera.* [En línea] 2008. [Citado el: 30 de Diciembre de 2009.] http://www.dii.uchile.cl/~ris/RISXXII/horariosUDP_RISVersion%20FINAL.pdf.

García Márquez, Fausto y Laguna, Manuel. 2003. Terra. *Optimización: Conceptos Fundamentales y Tendencias Actuales.* [En línea] 2003. [Citado el: 24 de Febrero de 2010.] http://buscador.terra.es/Results.aspx?ca=s&source=Search&query=Optimizaci%u00c3%u00b3n.

Burke, Edmund, Bykov, Yuri y Hirst, Jonathan. 2008. Scientific Commons. *Great Deluge Algorithm for Protein Structure Prediction.* [En línea] 2008. [Citado el: 25 de Febrero de 2010.] http://webhost.ua.ac.be/eume/workshops/reallife/burke2.pdf.

Astaiza A., Luis Gerardo. 2005. Redalyc. *Programación de exámenes: un enfoque práctico.* [En línea] 2005. [Citado el: 25 de Febrero de 2010.] http://redalyc.uaemex.mx/redalyc/pdf/643/64325311.pdf. 003.

K. Burke, Edmund y Petrovic, Sanja. 2002. KFUPM ePrints. *Recent Research Directions in Automated Timetabling.* [En línea] 2002. [Citado el: 26 de Febrero de 2010.] http://eprints.kfupm.edu.sa/61273/1/61273.pdf.

Alvarez, Miguel Angel. 2003. DesarrolloWeb.com. *Qué es Python.* [En línea] 9 de Noviembre de 2003. [Citado el: 4 de Marzo de 2010.] http://www.desarrolloweb.com/articulos/1325.php.

Jones Pérez, Kathryn M. y Yong Morales, Gustavo A. 2007. Di-mare.com. *Introducción al lenguaje de programación lógica Prolog.* [En línea] 2007. [Citado el: 2010 de Marzo de 4.] http://www.di-mare.com/adolfo/cursos/2007-2/pp-Prolog.pdf.

Yoteca. 2010. *Yoteca.* [En línea] 16 de Enero de 2010. [Citado el: 4 de Marzo de 2010.] http://www.yoteca.com/pg/glosario-de-internet.asp.

Rodríguez Ríos, Edison. 2010. EDISON RODRIGUEZ RIOS. *ENSAYO LENGUAJES DE PROGRAMACIÓN.* [En línea] 9 de Febrero de 2010. [Citado el: 4 de Marzo de 2010.] http://erodriguezri.blogspot.com/2010/02/ensayo-lenguajes-de-programacion.html.

Petrovic, Sanja y Burke, Edmund. 2004. Cite Seerx. *University Timetabling.* [En línea] 2004. [Citado el: 4 de Marzo de 2010.] http://citeseerx.ist.psu.edu/search;jsessionid=5494BC25B95548181EC6498A248E34F7?q=University+Timetabling&submit=Search&sort=rel.

González Duque, Raúl. 2009. Mundo Geek. *Python para todos.* [En línea] 16 de Septiembre de 2009. [Citado el: 5 de Marzo de 2010.] http://mundogeek.net/tutorial-python/.

Vlahos, Kiriakos. 2006. MMM-experts. [En línea] 26 de Octubre de 2006. [Citado el: 5 de Marzro de 2010.] http://mmm-experts.com/Products.aspx?ProductId=4.

Letelier, Patricio y Penadés, Mª Carmen. 2004. Laboratorio de Sistemas de Información. *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* [En línea] 2004. [Citado el: 17 de Marzo de 2010.] http://www.dsic.upv.es/asignaturas/facultad/lsi/doc/MetodologiasAgilesyExtremeProgramming.doc.

Campos, Mauricio. 2004. *Extreme Programming.* [En línea] Julio de 2004. [Citado el: 19 de Marzo de 2010.] http://apit.wdfiles.com/local--files/start/02_apit_xp_conceptos.pdf.

Casas, Sandra y Reinaga, Héctor. 2009. *Identificación y Modelado de Aspectos Tempranos dirigido por Tarjetas de Responsabilidades y Colaboraciones.* [En línea] 5 de Mayo de 2009. [Citado el: 25 de Marzo de 2010.] http://espanol.oocities.com/profeprog2/INVPAPER25.pdf.

Barber, Federico y Salido, Miguel A. 2003. *Introducción a la Programación de Restricciones.* [En línea] 2003. [Citado el: 23 de Abril de 2010.] http://users.dsic.upv.es/~msalido/publications.htm.

Grandón Espinoza, Carlos. 2004. *Programación con restricciones para el tratamiento de incertidumbre en CSP numéricos.* [En línea] Enero de 2004. [Citado el: 28 de Abril de 2010.] http://www-sop.inria.fr/coprin/cgrandon/Publica/grandon2004.pdf.

BIBLIOGRAFÍA

Franco Baquero, John Fredy, Toro Ocampo, Eliana Mirledy y Gallego Rendón, Ramón Alfonso. 2008. Scielo. *Problema de asignación óptima de salones resuelto con Búsqueda Tabú.* [En línea] 2008. [Citado el: 7 de Mayo de 2010.] http://www.scielo.org.co/scielo.php?pid=S0122-34612008000200011&script=sci_arttext.

Murray, Keith, Muller, Tomás y Rudová, Hana. 2007. Unitime. *Modeling and Solution of a Complex University Course Timetabling Problem.* [En línea] 2007. [Citado el: 11 de Mayo de 2010.] http://www.unitime.org/papers/patat07.pdf.

Wren, *A.* Scheduling, Timetabling and Rostering - a special relationship. 1996. pages 46-75, s.l.: Springer Berlin / Heidelberg, 1996, Vol. Volume 1153.

Lewis, Rhydian. 2007. A Survey of Metaheuristic-based Techniques for University. Timetabling Problems. [En línea] 29 de Mayo de 2007. [Citado el: 17 de Mayo de 2010.] http://www.cardiff.ac.uk/carbs/quant/rhyd/TTSurvey.pdf.

Daskalaki, S., Birbas, T. y Housos, E. 2004. *An integer programming formulation for a case study in university timetabling.* [En línea] 2004. [Citado el: 17 de Mayo de 2010.] http://www.ceet.niu.edu/faculty/ghrayeb/IENG576s04/papers/IP/An%20integer%20programming%20formulation%20for%20a%20case%20study.pdf.

Gunawan, Aldy, Ming, Kien y Leng, Kim. 2006. An Improvement Heuristic for the Timetabling Problem. [En línea] Diciembre de 2006. [Citado el: 17 de Mayo de 2010.] http://www.knu.edu.tw/lecture/%E6%95%99%E5%AD%B8%E8%B3%87%E6%96%99/APIEMS-2006/Paper/M3E4.pdf.

Dueck, Gunter. 1993. New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel. [En línea] 1993. [Citado el: 18 de Mayo de 2010.] http://logistics.iem.yzu.edu.tw/Teachers/Ycliang/Heuristic%20Optimization%20922/class%20note/New%20optimization%20heuristics.pdf.

Echevarría Cartaya, Yuviny y Miranda Pérez, Ridelio. 2010. SISTEMA INFORMÁTICO PARA LA CONFECCIÓN DE HORARIOS DOCENTES. [En línea] 2010. [Citado el: 2010 de Mayo de 24.] http://uciencia.uci.cu/files/V%20Taller%20de%20Matematica%20y%20Fisica%20Computacionales/FMC.08-1431603200/FMC.08.pdf.

Bellini M., Franco. 2004. INVESTIGACIÓN DE OPERACIONES. *LA INVESTIGACIÓN DE OPERACIONES Y EL USO DE MODELOS.* [En línea] 2004. [Citado el: 4 de Junio de 2010.] http://www.investigacion-operaciones.com/Formulacion%20Problemas.htm.

GLOSARIO DE TÉRMINOS

Alg. 1: Algoritmo número uno.

Alg. 2: Algoritmo número dos.

Alg. 3: Algoritmo número tres.

Alg. 4: Algoritmo número cuatro.

Alg. 5: Algoritmo número cinco.

Biblioteca de Algoritmos: Conjunto de rutinas de programa.

EF: Educación Física (asignatura).

Ghordo: Gestor de Horario Docente, aplicación web desarrollado en la Facultad 15 de la UCI.

HU: Historia de Usuario.

IA: Se denomina Inteligencia Artificial a la ciencia que intenta la creación de programas para máquinas que imiten el comportamiento y la comprensión humana.

NP-Completo: Es el subconjunto de los problemas de decisión en NP tal que todo problema en NP se puede reducir en cada uno de los problemas de NP-completo.

P1: Plantilla que recoge la información relacionada con la planificación de las asignaturas, en la cual se incluye la cantidad y el tipo de frecuencias, así como el tipo de local que se requiere para realizar una actividad en función de las semanas del semestre al que pertenece una asignatura.

PAHC: Problema de Asignación de Horarios de Clases.

UCI: Universidad de las Ciencias Informáticas.

ANEXOS

Anexo 1: Ejemplo de juego de datos entrante: Grupos_Profesores_Asignaturas

	A	B	C	D	E	F	G
1	Grupo	Semestre	ID Profesor	Nombre y Apellidos	Tipo de Profesor	Asignatura	Departamento
2	15101	2	1	Annielis Rodriguez Sotolongo	I	P1	P
3	15101	2	2	Silvia Maria Llarch Leyva	E	AL	CB
4	15101	2	3	Moises Alain Mayet	I	MD2	CB

Anexo 2: Ejemplo de juego de datos entrante: Listas_Afectaciones

	A	B	C	D	E	F	G
1		Listado de Afectaciones					
2	Grupos						
3	15101	L	4	L	5	L	6
4	15102	Mi	1				
5	Locales						
6	101	J	5				
7	102						
8	Departamentos						
9	H	J	1	J	2	J	3
10	CB	Mi	1	Mi	2	Mi	3
11	Profesores						
12	1	Mi	6	J	1		
13	2	M	1	S	2		
14	3	V	4				

Anexo 3: Ejemplo de juego de datos entrante: P1

	A	B	C	D	E
1	Asignatura	Numero de Semana	Tipo de Frecuencia	Cantidad de Frecuencia	Tipo de local
2	P1	1	C	1	A
3	P1	1	CP	1	A
4	P1	1	L	1	L
5	P1	2	CP	1	A

Anexo 4: Ejemplo de juego de datos entrante: Cantidad_Semanas_Planificar

	A	B
1	Sem. Determinada	
2	5	
3	Sem. Inicio	Sem. Fin
4		

	A	B
1	Sem. Determinada	
2		
3	Sem. Inicio	Sem. Fin
4	2	6

Anexo 5: Ejemplo de juego de datos entrante: Prioridad

	A	B	C	D
1	Anno	Semestre	Seccion	Algoritmo
2	1	2	T	3
3	2	2	T	5
4	3	2	M	1
5	4	2	X	2

Anexo 6: Ejemplo de juego de datos entrante: Prioridad_Asignatura

	A	B
1	Asignatura	Prioridad
2	DN	7
3	P1	6
4	M2	5
5	MD2	4
6	AL	3
7	PP1	2
8	PHCCU	1

Anexo 7: Ejemplo de juego de datos saliente: Lista_Encuentros_General

	A	B	C	D	E	F	G	H
1	15101	1	L	4	DN	C	Luis Braulio Navarrete	A-101
2	15101	1	L	5	P1	C	Annielis Rodriguez Sotolongo	A-101
3	15101	1	L	6	M2	C	Aray Perez Degue	A-101
4	15101	1	M	2	EF	-	Humberto Rodriguez	A.Dpt
5	15101	1	M	4	MD2	C	Moises Alain Mayet	A-101
6	15101	1	M	5	AL	C	Silvia Maria Llarch Leyva	A-101
7	15101	1	M	6	PP1	CTP	Dailin Benvides Jorge	A-101
8	15101	1	Mi	4	P1	CP	Annielis Rodriguez Sotolongo	A-101
9	15101	1	Mi	5	M2	CP	Aray Perez Degue	A-101
10	15101	1	Mi	6	PHCCU	C	Daykel Cardonell	A-101

Anexo 8: Ejemplo de juego de datos saliente: Horario

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	15101														
2	Semana 1	L	M	Mi	J	V	S		Semana 2	L	M	Mi	J	V	S
3	1								1						AL(CP-A-101)
4	2		EF(--A.Dpt)						2		EF(--A.Dpt)				
5	3								3						
6	4	MD2(C-A-101)	M2(C-A-101)	MD2(CP-A-101)	P1(L-L-1101)	M2(C-L-1101)			4	MD2(CP-A-101)	MD2(C-A-101)	DN(CP-A-101)	M2(CP-A-101)	M2(C-A-101)	
7	5	DN(C-A-101)	AL(C-A-101)	P1(CP-A-101)	M2(CP-A-101)	AL(CP-L-1101)			5	DN(C-A-101)	P1(L-L-1101)	M2(CP-A-101)	AL(C-A-101)		
8	6	P1(C-A-101)	PP1(CTP-A-101)	PHCCU(C-A-101)					6	P1(CP-A-101)	P1(L-L-1101)	PP1(CTP-A-101	PHCCU(T-L-1101)		
9															
10	15102														
11	Semana 1	L	M	Mi	J	V	S		Semana 2	L	M	Mi	J	V	S
12	1								1						
13	2		EF(--A.Dpt)						2		EF(--A.Dpt)				
14	3								3						
15	4	MD2(C-A-102)	M2(C-A-102)	MD2(CP-A-102)	P1(L-L-1102)	M2(C-L-1102)			4	MD2(CP-A-102)	MD2(C-A-102)	DN(CP-A-102)	M2(CP-A-102)	M2(C-A-102)	
16	5	DN(C-A-102)	AL(C-A-102)	P1(CP-A-102)	M2(CP-A-102)				5	DN(C-A-102)	P1(L-L-1102)	M2(CP-A-102)	PP1(CTP-A-102)	AL(CP-A-102)	
17	6	P1(C-A-102)	PP1(CTP-A-102)	PHCCU(C-A-102)	AL(CP-L-1103)				6	P1(CP-A-102)	P1(L-L-1102)	AL(C-A-102)	PHCCU(T-L-1102)		
18															
19	15103														
20	Semana 1	L	M	Mi	J	V	S		Semana 2	L	M	Mi	J	V	S
21	1	EF(--A.Dpt)							1	EF(--A.Dpt)					
22	2								2						
23	3								3						
24	4	DN(C-A-103)	M2(C-A-103)	P1(CP-A-103)	P1(L-L-1103)	M2(C-L-1103)			4	DN(C-A-103)	M2(CP-A-103)	M2(CP-A-103)	P1(L-L-1103)	AL(CP-A-103)	
25	5	MD2(C-A-103)	AL(C-A-103)	MD2(CP-A-103)	M2(CP-A-103)	PP1(CTP-A-103)			5	MD2(CP-A-103)	MD2(C-A-103)	DN(CP-A-103)	P1(L-L-1103)	PP1(CTP-A-103)	
26	6	P1(C-A-103)	PHCCU(C-A-103)		AL(CP-L-1104)				6	P1(CP-A-103)	AL(C-A-103)	PHCCU(T-L-110	M2(C-A-103)		
27															

	A	B	C	D	E	F	G
1	15101						
2	Semana 1	L	M	Mi	J	V	S
3	1						
4	2		EF(--A.Dpt)				
5	3						
6	4	MD2(C-A-101)	M2(C-A-101)	MD2(CP-A-101)	P1(L-L-1101)	M2(C-L-1101)	
7	5	DN(C-A-101)	AL(C-A-101)	P1(CP-A-101)	M2(CP-A-101)	AL(CP-L-1101)	
8	6	P1(C-A-101)	PP1(CTP-A-101)	PHCCU(C-A-101)			
9							
10	15102						
11	Semana 1	L	M	Mi	J	V	S
12	1						
13	2		EF(--A.Dpt)				
14	3						
15	4	MD2(C-A-102)	M2(C-A-102)	MD2(CP-A-102)	P1(L-L-1102)	M2(C-L-1102)	
16	5	DN(C-A-102)	AL(C-A-102)	P1(CP-A-102)	M2(CP-A-102)		
17	6	P1(C-A-102)	PP1(CTP-A-102)	PHCCU(C-A-102)	AL(CP-L-1103)		
18							

Anexo 9: Ejemplo de juego de datos saliente: Valores_Funcion_Objetivo

	A	B	C	D	E	F	G
1	Anno	Algortimo	Clases los sabados	Profesores VP a 2da y 5ta	Clases CB y P a 3ra o 6ta	clases CB mismo dia	Valor de la FO
2	1	5	12	0	65	14	91
3	2	5	0	0	0	0	0
4	3	5	0	0	0	0	0
5	4	5	0	0	0	0	0