PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ

FACULTAD DE CIENCIAS E INGENIERÍA

ELABORACIÓN DE UNA SOLUCIÓN METAHEURÍSTICA USANDO UN ALGORITMO GENÉTICO QUE PERMITA ELABORAR LA DISTRIBUCIÓN DE LOS HORARIOS ACADÉMICOS

Tesis para optar el Título de Ingeniera Informática, que presenta el bachiller:

Ana Nataly Angeles Diaz

ASESOR: Ing. Miguel Ángel Guanira Erazo

Lima, mayo del 2015

Resumen

El presente documento describe un proyecto de fin de carrera en Ciencias de la Computación. Este proyecto intenta dar solución al problema de generación de horarios académicos en instituciones de nivel superior.

La solución se construye con el uso de un algoritmo genético a partir de una población inicial generada por un algoritmo Grasp fase construcción. Se ha tomado como caso de estudio a la facultad de Ciencia e Ingeniera de la Pontificia Universidad Católica del Perú, en la cual se contó con el apoyo del encargado de realizar el horario de la especialidad de ingeniería informática para el respectivo levantamiento de información, con lo cual se consiguió la adaptación de un algoritmo que cumpla con sus restricciones y requerimientos. Para facilitar la búsqueda de esta solución se aplicarán los operadores de selección, casamiento, mutación y etilismo.

La calidad de las soluciones, generadas por el algoritmo, se medirá en base a la cantidad de restricciones cumplidas. Para determinar los valores de los parámetros de los algoritmos se realizaron varias ejecuciones con diferentes combinaciones de valores y se optó por la que optimizaba la función objetivo de la solución. Se estima que la duración del proyecto será de un año.

FACULTAD DE
CIENCIAS E INGENIERÍA
ESPECIALIDAD DE
INGENIERÍA INFORMÁTICA

PONTIFICIA
UNIVERSIDAD CATÓLICA
DEL PERÚ

TEMA DE TESIS PARA OPTAR EL TÍTULO DE INGENIERO INFORMÁTICO

TÍTULO: Elaboración de una solución metaheurística usando un algoritmo genético que permita elaborar la distribución de los horarios académicos.

ÁREA: Ciencia de la Computación

PROPONENTE: Ing. Miguel Guanira Erazo

ASESOR: Ing. Miguel Guanira Erazo

ALUMNO: Ana Nataly Angeles Diaz

CÓDIGO: 20092042

TEMA N°: 567

FECHA: San Miguel , 14 diciembre de 2014

PONTIFICIA UNIVERSIDAD CATÓLICA DEL PERÚ
Facultad de Ciencias e Ingeniería
08 ABR. 2015
MIGUEL MEJIA PUENTE
DECANO

DESCRIPCIÓN

El objetivo del presente proyecto de fin de carrera es implementar una solución metaheurística usando un algoritmo genético que permita la generación automática de horarios de clases, laboratorios y prácticas de la PUCP en la Facultad de Ciencias e Ingeniería, cuyo objetivo principal es reducir el tiempo de elaboración, maximizar el uso de los recursos y optimizar la distribución de los cursos al momento de la creación de horarios para cada periodo académico.

Para que la solución propuesta pueda generar una distribución óptima que cumpla con las características antes mencionadas se llevará a cabo la elección y adaptación de un algoritmo genético, el cual es considerado metaheurístico y se basa en una técnica de selección natural, que sea capaz de manejar todas las restricciones fuertes de la problemática y la mayoría de las restricciones débiles.

OBJETIVO GENERAL

Implementar una solución metaheurística usando un algoritmo genético que permita elaborar la distribución de los horarios académicos semestrales para la facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú.

OBJETIVOS ESPECÍFICOS

- Modelar el proceso de elaboración del horario académico de la facultad de Ciencias e Ingeniería en la PUCP, usando la metodología BPM.
- Definir la estructura del cromosoma del algoritmo genético, el cual debe cumplir con las especificaciones del problema.

Av. Universitaria 1801
San Miguel, Lima – Perú

Apartado Postal 1761
Lima 100 – Perú

Teléfono:
(511) 626 2000 Anexo 4801

- Establecer la función fitness a desarrollar que cumplan con las consideraciones y restricciones del problema para la PUCP.
- Establecer los operadores genéticos que serán utilizados en el desarrollo del algoritmo genético.
- Implementar un algoritmo genético que me permita resolver el problema de UTCP con las restricciones respectivas.
- Implementar un prototipo funcional del sistema para que soporte las necesidades de los datos de entrada y salida que requiere el problema.

ALCANCE

Con relación al problema de la generación de horarios académicos en la PUCP, se debe tener en cuenta que solo se abarcará la facultad de Ciencias e Ingeniería, no se tomará en cuenta todas las secuencias existentes en la universidad y las materias a asignar en la distribución de horarios solo abarcará las clases, laboratorios y prácticas. Por otro lado con relación a la solución se implementará un software que sea capaz de elaborar la asignación de horarios académicos para un semestre usando un algoritmo genético, el cual necesitara de un algoritmo Grasp para producir su población inicial. Al algoritmo Grasp solo desarrollará su fase de construcción, el alpha y numero de iteraciones serán tomadas de tablas obtenidas en la bibliografía encontrada.

Además es importante mencionar que el proyecto se enfocara más en el desarrollo del algoritmo que en la parte gráfica del software. Estas solo deben cumplir con las especificaciones necesarias para que el algoritmo sea probado.

Máximo: 100 páginas

Tabla de contenido

Índice de Figuras

Índice de Tablas

CAPÍTULO 1

1 Problemática

En toda institución educativa a nivel superior antes del inicio de cada periodo académico se presenta la necesidad de elaborar una programación de horario, la cual contiene la distribución de los cursos a dictar en ese periodo (especificando las secuencias, horas de dictado y los profesores asignados). La elaboración de horarios no es un trabajo fácil de realizar, debido a la gran cantidad de posibles combinaciones que se pueden establecer y por las diversas limitaciones que posee. Por ejemplo, la cantidad de cursos máximos que se puede asignar a un docente, las capacidades de las aulas, la cantidad de horas laborales que posee cada tipo de docente, entre otras. Todas estas limitaciones deben ser respetadas de acuerdo a las políticas de cada universidad. (Jat.S.S y Yang.S, 2008). Por lo tanto, los problemas de horarios educativos o timetabling son considerados problemas NP y han atraído a varios investigadores en el mundo en los últimos 25 años.

Los problemas de timetabling son un caso particular de los problemas de scheduling. El timetabling básicamente consiste en la programación de un cierto número de eventos en un número limitado de intervalos de tiempo (Burke.E et al., 2006). Además, esta asignación debe cumplir con un conjunto de restricciones de diferentes tipos (restricciones fuertes y restricciones débiles). El problema de timetabling abarca diferentes disciplinas como instituciones educativas, aplicaciones deportivas, planificación del transporte, planificación de proyecto, entre otras.

Los problemas de la generación de horarios para las universidades son conocidos en la optimización combinatoria como University Course Timetabling Problem (UCTP), el cual proviene de una caso particular de los problemas de timetabling aplicados a instituciones educativas, la complejidad que este tipo de problema tiene es NP-hard, debido a que el tiempo que necesita para su solución aumenta exponencialmente respecto al tamaño del problema. Los UCTP consisten básicamente en un conjunto de *n* eventos (clases, laboratorios, exámenes, etc.) los cuales deben ser programados en un conjunto de *t* intervalos de tiempo y se tiene en cuenta un conjunto de habitaciones (en los cuales los eventos toman lugar). Además, se tiene un conjunto de estudiantes *e* que asisten a los eventos y cada habitación tiene un conjunto de características para que el evento se pueda desarrollar. Se debe cumplir que un horario es factible si todos los eventos son asignados a un intervalo de tiempo y a una habitación, de manera que las restricciones fuertes expuestas en el problema sean satisfechas y que la violación a las restricciones débiles sean mínimas. (Burke, Edmund y Trick, Michael, 2004).

Una mala gestión al momento de elaborar la distribución de horarios puede significar una gran pérdida de tiempo por parte del personal involucrado en el proceso, lo cual se puede traducir en dinero perdido por la institución respectiva. Por otro lado, si no se escoge una distribución óptima, probablemente se estén desaprovechando varios recursos de la institución (aulas sin utilizar) y se puede estar sobrecargando a los docentes (pueden tener muchas horas seguidas de dictado). Los alumnos también se pueden ver afectados por esta distribución, ya que se puede programar muchas clases o evaluaciones consecutivas, lo cual afecta su rendimiento académico.

Actualmente el proceso de elaboración de horarios en la Pontificia Universidad Católica de Perú (PUCP) no es ajeno a este problema (Bejerano.G et al., 2009). Esta tarea es realizada de manera semi-manual y está compuesta por dos etapas: la primera es la elaboración manual de la distribución de los cursos y la segunda la verificación de esta distribución, estas actividades son ejecutas cada ciclo. A continuación se explicará cada etapa, según lo entrevistado al encargado de la ejecución de horarios en la Especialidad de Ingeniería Informática:

La primera etapa es el proceso de elaboración de la distribución de cursos en la facultad, esta inicia cuando el coordinador de horarios de cada especialidad entrega una lista de los cursos a dictar en el presente periodo para que sea aprobada por la Facultad de Ciencias e Ingeniería. Una vez aceptados los cursos a dictar, a cada coordinador se le entrega un conjunto de aulas disponibles y ellos deben distribuir todos los cursos a dictar tomando en cuenta algunas consideraciones como:

- La disponibilidad de tiempo los docentes, la cual depende del tipo de docente.
- La disponibilidad, capacidad y restricciones de las aulas. Estas restricciones se refieren a las cualidades (equipos y otros materiales de trabajo) que debe tener el aula para que se pueda realizar una clase en ella. Ya que en ocasiones se requiere algún tipo de equipo especial.
- La cantidad de cursos a dictar.
- La cantidad de alumnos por periodo.
- La secuencias que cada tipo de horario debe cumplir.
- Ningún curso del mismo ciclo curricular pueden cruzarse
- Los cursos de ciclos curriculares adyacentes deben tratar de no cruzarse en la medida de lo posible (ya que se le da la oportunidad a un alumno de 6to clico para que pueda llevar cursos de 7mo ciclo)
- Los cursos de los ciclos curriculares menores se dicten en turnos mañana-tarde y los ciclos mayores se dicten en turnos tarde-noche.

Una vez elaborada la distribución inicia la segunda etapa, la cual es la de verificación de la validez de la distribución escogida. La PUCP cuenta con un software que le permite verificar si existe algún tipo de cruce con las aulas, cursos (tanto en clases, laboratorios y exámenes) y asignación del docente (ya que un docente no puede ser seleccionado al mismo tiempo para dos cursos). Finalmente, una vez que se verifique, el horario queda grabado en el sistema (en caso contrario, se realizan los cambios necesarios, de manera manual, para que la distribución sea válida).

Debido a la gran complejidad de este problema, lo que se hace para disminuir el tiempo de construcción en la primera etapa es tomar como base el horario académico del periodo anterior (para no tener que realizar toda la distribución nuevamente) y únicamente se realizan pequeñas modificaciones. El inconveniente de tomar como base el horario del periodo anterior es que, cuando hay que realizar modificaciones estas se hacen de manera manual y no se puede garantizar que se obtenga una distribución óptima. Además se emplea demasiado tiempo para la coordinación de las modificaciones con las otras especialidades de la facultad y el cuadre de estas nuevas modificaciones en el horario ya establecido. En la Figura 1 se puede apreciar el diagrama del proceso de la elaboración del horario académico para una sección de Ciencias e Ingeniería.

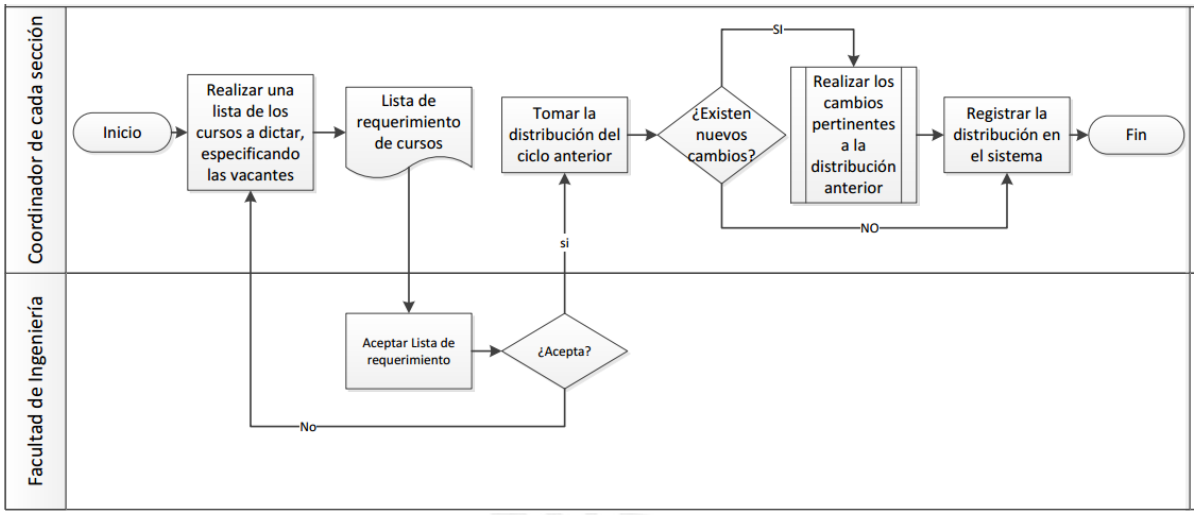


Figura 1.1.- Proceso actual para cada sección de ingeniería.

Vista esta problemática, el objetivo del presente proyecto de fin de carrera es abarcar estos aspectos mediante una solución metaheurística que permita la generación automática de horarios de clases, laboratorios y prácticas de la PUCP en la Facultad de Ciencias e Ingeniería, cuyo objetivo principal es reducir el tiempo de elaboración, maximizar el uso de los recursos y optimizar la distribución de los cursos (logrando una distribución con una carga académica balanceada tanto para los docentes como para los alumnos) al momento de la creación de horarios para cada periodo académico.

Para que la solución propuesta pueda generar una distribución óptima que cumpla con las características antes mencionadas se llevará a cabo la elección y adaptación de un algoritmo genético, el cual se basa en una técnica de selección natural, que sea capaz de manejar todas las restricciones fuertes de la problemática y la mayoría de las restricciones débiles.

Además, se plantea la construcción de un prototipo funcional del sistema de generación de horarios que permita al usuario gestionar los inputs y outputs de las ejecuciones del algoritmo.

1.1 Objetivo general

Implementar una solución metaheurística usando un algoritmo genético que permita elaborar la distribución de los horarios académicos semestrales para la facultad de Ciencias e Ingeniería de la Pontificia Universidad Católica del Perú.

1.2 Objetivos específicos

1. Modelar el proceso de elaboración del horario académico de la facultad de Ciencias e Ingeniería en la PUCP, usando la metodología BPM.

2. Definir la estructura del cromosoma del algoritmo genético y las demás estructuras, las cual deben cumplir con las especificaciones del problema.

3. Establecer la función fitness a desarrollar que cumplan con las consideraciones y restricciones del problema para la PUCP.

4. Establecer los operadores genéticos que serán utilizados en el desarrollo del algoritmo genético.

5. Implementar un algoritmo genético que me permita resolver el problema de UTCP con las restricciones respectivas.

6. Implementar un prototipo funcional del sistema para que soporte las necesidades de los datos de entrada y salida que requiere el problema.

1.3 Resultados esperados

1. Resultado 1 para el objetivo 1: Descripción del proceso del caso de estudio siguiendo notación BPMN.

2. Resultado 1 para el objetivo 2: Descripción y justificación de la estructura del cromosoma y de los demás elementos necesarios para la representación del problema.

3. Resultado 1 para el objetivo 3: Definición y justificación de la función de fitness, la cual toma en cuenta las consideraciones y restricciones del problema.

4. Resultado 1 para el objetivo 4: Definición, descripción y justificación de los operadores genéticos a utilizar en el algoritmo genético.

5. Resultado 1 para el objetivo 5: Diseño del algoritmo genético en pseudocódigo que maneje las restricciones del problema para el caso de estudio.

6. Resultado 1 para el objetivo 6: Prototipo funcional de los módulos del software.

2 Herramientas, métodos, metodologías y procedimientos

2.1 Introducción

A continuación se mencionarán las herramientas, métodos y metodologías que se aplicaran en el presente proyecto de fin de carrera para obtener con éxito los resultados esperados.

2.2 Herramientas

En la siguiente tabla se realiza el listado de las herramientas que se utilizarán en el proyecto para cubrir cada uno de los resultados esperados.

Resultados esperado	Herramientas a usarse
RE1: Modelo del proceso de la elaboración de horarios académicos para la Pontificia Universidad Católica del Perú.	BPMN es una notación gráfica estandarizada que permite el modelado de procesos de negocio, en un formato de flujo de trabajo
RE1: Modelo del proceso de la elaboración de horarios académicos para la Pontificia Universidad Católica del Perú	Bizagi Process Modeler, herramienta para modelar procesos de negocio en notación BPMN.
RE5: Diseño del algoritmo genético en pseudocódigo que maneje las restricciones del problema para el caso de estudio.	Documentos de estudios de investigación (Papers), estudiados en la revisión del marco teórico.
RE6: Prototipo funcional de los módulos del software.	NetBeans IDE 7.2 es un entorno de desarrollo integrado libre.

- BPMN (Business Process Model Notation)
  Es una notación grafica común para modelar el proceso de negocio de forma más clara, estandarizada y completa. De esta manera el diagrama del flujo de proceso puede ser entendido por diferentes áreas del negocio (Bizagi, 2013). En el presente proyecto de fin de carrera esta notación se utilizará para desarrollar el modelo del proceso de generación de horarios de la PUCP, de esta manera lograr la obtención del RE1.

- Bizagi Process Modeler
  Es una herramienta líder en el mercado que permite modelar y documentar los procesos de negocio en notación BPMN. Posee un funcionamiento intuitivo y una interfaz gráfica amigable lo cual permite modelar los procesos de forma rápida y fácil (Bizagi, 2013). Con esta herramienta se desarrollará el modelado del proceso de la elaboración de horarios académicos de la PUCP con el cual se cumplirá el RE1.

- Documentos de estudio de investigación
  Los documentos investigación estudiados servirán de apoyo para la obtención de RE5, ya que se tomaran datos teóricos para las constantes de la población inicial que el algoritmo genético requiere. Entre otros datos que sean necesarios para construcción del algoritmo genético.

- Netbeans IDE
  Es una herramienta de desarrollo pensado para programadores, la cual se encuentra escrita en el lenguaje java. Permite la rápida y fácil implementación de aplicaciones web, escritorios y móviles. Además cuenta con un conjunto de herramientas para desarrolladores de PHP y C/C++ (Netbeans, 2013). En el presente proyecto se utilizará esta herramienta para la implementación del prototipo funcional del software de generación de horarios.

2.3 Métodos y Procedimientos

En la siguiente tabla se realiza el mapeo de los métodos y procedimientos utilizados para cubrir los resultados esperados.

Resultados esperado	Métodos y procedimientos a seguir
RE2: Descripción y justificación de la estructura del cromosoma, el cual representará la solución del problema. RE3: Definición y justificación de la función de fitness, la cual toma en cuenta las consideraciones y restricciones del problema RE4: Definición, descripción y justificación de los operadores genéticos a utilizar en el algoritmo genético. RE5: Diseño del algoritmo genético en pseudocódigo que maneje las restricciones del problema para el caso de estudio.	Algoritmos bio-inspirados, el cual forma parte del método basado en poblaciones.

- Algoritmo bio-inspirado
  Este tipo de algoritmo se inspira en la capacidad de la naturaleza que tienen los seres vivos al evolucionar para adaptarse a su entorno. Este tipo de comportamiento es simulado computacionalmente con iteraciones las cuales representan a las generaciones, en las cuales cada individuo de la población actual es afectado por número de operadores para convertirlos en la siguiente generación. Así mismo en base a criterios de aptitud se selecciona a los individuos que permanecerán en la siguiente generación. (Blum y Roli, 2003)

  Usando este método de desarrollo se pretende obtener el RE2 al definir al individuo del algoritmo, el cual será capaz de manejar la una solución de la asignación de un curso-profesor-horario-aula. También ayudará a definir el resultado RE3 y RE4, en el cual se definirá como se aplicará el uso de los operadores a cada individuo de la población actual y el criterio de aptitud (función fitness) que se le aplicara a cada individuo para evaluar su permanencia en la siguiente generación. Por último, este método también ayuda a la obtención del resultado esperado RE5 al seguir los procedimientos para la construcción del pseudocódigo del algoritmo genético, el cual pertenece a la categoría de algoritmos bio-inspirados.

3 Delimitación

3.1 Introducción

A continuación se mencionarán el alcance y los obstáculos del proyecto de fin de carrera. Así como también los riesgos que este puede presentar a lo largo de su desarrollo.

3.2 Alcance

Con relación al problema de la generación de horarios académicos en la PUCP, se debe tener en cuenta que solo se abarcará la facultad de Ciencias e Ingeniería, no se tomará en cuenta todas las secuencias existentes en la universidad y las materias a asignar en la distribución de horarios solo abarcará las clases, laboratorios y prácticas. Por otro lado con relación a la solución se implementará un software que sea capaz de elaborar la asignación de horarios académicos para un semestre usando un algoritmo genético, el cual necesitara de un algoritmo Grasp para producir su población inicial. El algoritmo Grasp solo desarrollará su fase de construcción, el alpha y número de iteraciones serán tomadas de tablas obtenidas en la bibliografía encontrada.

Además es importante mencionar que el proyecto se enfocara más en el desarrollo del algoritmo que en la parte gráfica del software. Estas solo deben cumplir con las especificaciones necesarias para que el algoritmo sea probado.

3.3 Obstáculos

En relación a la problemática se encuentra una dificultad al momento de modelar el proceso elaboración de horarios ya que las diferentes secciones de ingeniería cuentan con procesos distintos y se tienen que uniformizar en un solo proceso. Por otro lado, en relación a la solución una de las grandes dificultades que presenta el proyecto de fin de carrera es el tiempo asignado. Por lo tanto, se ha definido que el algoritmo no contará con una experimentación numérica debido a que el proyecto se enfocará en el desarrollo del software que permitirá la generación de horarios académicos.

A continuación se muestran los riesgos identificados en el proyecto de fin de carrera, su impacto y la medida correctiva para mitigarlo.

Riesgo identificado	**Impacto en el proyecto**	**Medidas correctivas para mitigar**
Pérdida de la información.	Alto	Generar diversos backups de los entregables (en la nube, en físico, entre otros)
No contar con el apoyo de la oficina de la PUCP para que brinden la respectiva información del proceso.	Alto	Obtener la información a partir de los encargados de armar el horario de cada facultad y de esa manera generalizarlo.
Contar con información poco consistente debido a las diferentes versiones de los entrevistados.	Alto	Crear cuestionarios con preguntas precisas y claras para que los entrevistados puedan brindar una mejor información.
Alto grado complejidad de la adaptación del algoritmo genético a la problemática mencionada.	Alto	Tener la opción de un posible cambio de algoritmo por el de búsqueda tabú.

4 Justificativa y viabilidad del proyecto

4.1 Justificativa

Debido a al problema actual en la generación de horarios académicos de la Facultad de Ciencias e Ingeniera en la PUCP, como se mencionó en la problemática, en el presente proyecto se desarrollará un software que permita la construcción de horarios académicos, el cual tiene como objetivo abarcar de manera eficiente el proceso que actualmente se ejecuta. Cabe resaltar que este tipo de problemas es de complejidad NP, por ello se ha escogido un algoritmo bio-inspirado para resolverlo.

El software implementado tiene como finalidad brindarle un gran beneficio a la PUCP, debido a que se conseguirá disminuir el tiempo de ejecución del proceso respecto al que actualmente se desarrolla, de esta manera se puede aprovechar al personal designado a este proceso para que cumplan otras funciones. También, permitirá una óptima distribución de horarios académicos, con lo cual se generará un uso adecuado de los recursos tanto profesores como aulas, con lo cual se disminuirá los tiempos muertos en los cuales los recursos pueden ser utilizados y se realizará una carga balanceada de las horas de dictado de los docentes.

Otro punto importante a resaltar es que con una óptima distribución de horarios académicos los alumnos de la PUCP tendrán el beneficio de que los horarios de clases se encuentren distribuidos con el objetivo de que un alumno de un mismo ciclo académico pueda llevar todos los cursos especificados en la currícula en un solo turno, de esta manera se disminuyen los tiempos entre clase y clase. Además, se contará con una mejor distribución de exámenes por ciclo para que la carga académica disminuya.

4.2 Viabilidad

En cuanto a la viabilidad del proyecto, para el conocimiento requerido sobre el tema se verifica que según la revisión de estado del arte existen varios problemas relacionados a la asignación de horarios de clases y exámenes en las universidades, a partir de ello se posee una mayor base para el análisis de la problemática. Por otro lado, en cuanto a la solución de problema se han encontrado menores referencias respecto al aplicar un algoritmo genético, sin embargo, estos si son capaces de ser adaptados al problema de timetabling.

Con respecto a la recopilación de datos y la respectiva especificación de los requisitos se llevará a cabo mediante entrevistas a los encargados de cada especialidad de la Facultad de Ciencias e Ingeniería, así como a al departamento encargado de la distribución de horarios de la facultad. De esta manera se pretende poder obtener un conocimiento claro del proceso de elaboración de horarios, para luego iniciar con el análisis de la solución. Es viable realizar el levantamiento de dicha información ya que se encuentra en la misma universidad y no son datos confidenciales.

Para finalizar, el presente proyecto no contará con una disponibilidad financiera ya que se trata de un proyecto de fin de carrera. Las herramientas necesarias para su desarrollo serán software libre. Finalmente, el proyecto será ejecutado en un periodo de tiempo de cuatro meses y estará a cargo de la tesista.

CAPÍTULO 2

1 Marco teórico

1.1 Introducción

A continuación se realizará una breve descripción de los conceptos que se tratarán a lo largo del proyecto.

1.2 Objetivo del marco conceptual

La finalidad de la descripción de los conceptos que se presentarán a continuación es tener una mayor comprensión del documento al momento de leerlo, ya que estos serán utilizados a lo largo del mismo (algunos conceptos tienen varias o distintas definiciones, en el proyecto se utilizarán las que se presentan en esta sección). Para lograr este objetivo se dividirán los conceptos en dos sub-secciones: los relacionados a la problemática entorno a la PUCP, donde se presentarán los conceptos relacionados al proceso de elaboración de horarios, y los relacionados a la solución.

1.2.1 Conceptos relacionados a la problemática

1.2.1.1 Problemas NP

Son aquellos problemas para los cuales existe un algoritmo determinístico los cuales pueden verificar si una solución es correcta en un tiempo polinomial. La resolución de un problema no determinístico consta de dos fases: Fase de suposición y fase de verificación (Alsuwaiyel, 1998).

La fase de suposición, es aquella en la cual se genera una respuesta arbitraria encontrada mediante un algoritmo no determinístico en un número de pasos polinomiales. Esta respuesta puede ser o no una solución del problema.

La fase de verificación, es en la cual se verifica mediante un algoritmo determinístico si la respuesta se encuentra en un formato apropiado y si la respuesta es una solución para la instancia de ese problema.

1.2.1.2 Problemas de Optimización

Son aquellos problemas que consisten en la búsqueda de la mejor solución en base a un conjunto de variables para llegar a cumplir las metas deseadas. En este tipo de problemas se pueden encontrar dos categorías: Soluciones que son codificadas en variables reales y soluciones en codificadas en variables discretas (Blum y Roli, 2003).

1.2.1.3 Problema de programación de horarios (Scheduling)

Este tipo de problema es muy común en el área industria de manufactura y servicios, la cual comprende diferentes tipos de tareas y máquinas para su realización. Este problema se encarga de distribuir los diferentes recursos para cumplir ciertas tareas en intervalos de tiempo determinado con el fin de optimizar el uso de diferentes tipos de recursos que involucre la tarea (Pinedo, 2012).

Como ya se mencionó según el tipo de organización se utiliza diferentes tipos de tareas y recursos de acuerdo al rubro que se encuentren. Alguna de estas tareas son: procesos de producción, planeación de despegue de aviones, etapas de un proyecto de construcción, la ejecución de programas en un ordenador, entre otros. Además algunos de los recursos utilizados son: máquinas de producción, equipos de construcción, pistas despegue de los aeropuertos, unidades de procesamiento en un entorno de computación, entre otros. Por otro lado cada problema también tiene diferentes objetivos como minimizar el tiempo o makespan de realización de cada tarea, minimizar el número de tareas, minimizar la cantidad de recursos utilizados, entre otros.

En la literatura este tipo de problema es considerado NP lo cual quiere decir que no se puede encontrar una solución óptima en tiempo polinomial. El problema de scheduling es descrito por tres variables $\alpha|\beta|\gamma$. La variable α describe el escenario de máquinas y contiene una sola entrada; la variable β consiste en los detalles de procedimiento y las restricciones que se encuentran involucradas y puede contener una o varias especificaciones; y por último la variable γ describe el valor a ser optimizado que generalmente contiene una sola entrada. Generalmente estos problemas están dirigidos con el proceso de programación de tareas industriales cada una con una función objetiva específica. A continuación se mencionará los casos más conocidos para el escenario de α (Pinedo, 2012):

- Single machine: Este caso es uno de los más simples de todos los posibles escenarios de scheduling. Como su nombre consta de una máquina sola como recurso.

- Identical Machine in parallel (Pm): En este escenario se identifica m maquinas idénticas en paralelo y una tarea j que requiere una operación simple y puede ser procesada en cualquier tipo de m maquinas.

- Flow Shop (Fm): Consta de m maquinas en serie y un grupo de tareas que deben ser procesadas o asignadas a cada una de las m maquinas. Todas las tareas son secuenciales, esto quiere decir que la tarea 1 debe pasar por la maquina 1, luego la maquina 2 y así sucesivamente. Además conforme las tareas van llegando a una maquina están deben estar en una cola para poder esperar su turno a ser procedas. Para este caso se utilizan colas bajo la condición de FIFO (primero en entrar y primero en salir).

- Job shop (Jm): Consta de m máquinas y cada tarea tiene una distribución de las maquinas que debe visitar. Se hace una distinción en la visita de las tareas, algunas tareas pueden visitar solo una vez una máquina y otras en los cuales la misma tarea puede pasar por una misma maquina más de una vez.

- Open Shop (Om): Consta de m máquinas y cada tarea debe ser procesada de nuevo en cada una de las maquinas m, sin embargo algunos de estos tiempos pueden ser cero. Además no existe restricciones en cuanto a la ruta de trabajo, esto quiere decir que se puede determinar una ruta para cada puesto de trabajo y los diferentes puestos de trabajo pueden tener diferentes rutas.

1.2.1.4 Timetabling

Este tipo de problemas consiste básicamente en asignar cierta cantidad de eventos o recursos en periodos de tiempo. Estos eventos están sujetos a restricciones fuertes y débiles. Basta tener una solución que cumpla con las todas las restricciones fuertes y algunas débiles para que se considere como una solución óptima. Dependiendo de la cantidad de restricciones débiles que se cumplan se mide la calidad de la solución. (Burke, Edmund y Trick, Michael, 2004).

Las restricciones Fuertes: son aquellas que se deben cumplir de todas maneras. Por ejemplo, cuando se realiza un horario de clases un mismo profesor no puede estar asignado a dos cursos en el mismo horario.

Las restricciones Débiles: son aquellas que no necesariamente deben ser cumplidas, pero aportan calidad a la solución. Por ejemplo, para los horarios de la universidad que los cursos de los ciclos académicos menores no sean dictados en las noches.

En la literatura se puede encontrar varias variantes a este problema. A continuación se menciona algunas:

- Sports timetabling: En este tipo de problema se desea elaborar horarios para un evento deportivo, distribuyendo los horarios de los partidos (según las reglas propuestas). Además, cabe mencionar que este problema solo tiene una dimensión (el tiempo).

- University timetabling: En este tipo de problema se desea elaborar la asignación de cursos en universidades, tomando en cuenta diversos factores que hacen que el problema tenga una complejidad alta. El objetivo de esta variante es encontrar una distribución óptima que tome en cuenta la disponibilidad de los profesores, las restricciones de cada curso, la disponibilidad y capacidad de las aulas, entre otros. Este problema está sujeto a restricciones fuertes las cuales deben ser cumplidas para construir una distribución de horarios útiles y restricciones suaves que no necesariamente deben ser cumplidas pero aportara calidad a la solución.

- Transport timetabling: En este tipo de problema se desea elaborar una programación de horas para los arribos y llegada de los transporte (ya sea de aéreas o terrestres). Además, se busca encontrar una distribución de asignación de conductores para la operación de transporte con el objetivo de reducir al mínimo el costo del programa operativo.

- Employee timetabling: En este tipo de problema se desea elaborar horarios de turno de trabajo para diversas áreas de profesionales. El objetivo de esta variante es poder cumplir las exigencias que requiere cada institución como las horas de trabajo de cada empleado. Además se debe tener en cuenta las restricciones legales que el ámbito laboral requiera.

1.2.2 Conceptos relacionados a la solución

1.2.2.1 Optimización Combinatoria (CO)

Es una rama de las matemáticas y ciencias de la computación que se encarga de encontrar una solución óptima en un conjunto finito de soluciones. Este tipo de problemas suelen ser complejos debido a que pueden poseer una gran cantidad de variables, lo cual hace que el conjunto de soluciones posibles sea inmenso, y en muchos casos, es computacionalmente imposible encontrar la solución óptima, en estos casos se utilizan métodos heurísticos y metaheurísticos para hallar una solución bastante aproximada a la óptima. (Papadimitriou, 1982)

Algunos ejemplos de los problemas de optimización combinatoria son: el problema travelling Salesman (TSP), Quadratic Assigment (QAP), timetabling y problemas de scheduling.

1.2.2.2 Algoritmos Metaheurísticos

Son métodos aproximados que se define formalmente como una estrategia de muy alto nivel para explorar espacios de búsqueda. Este proceso consiste en una generación iterativa, la cual tiene como objetivo encontrar de manera eficiente soluciones óptimas (Blum y Roli, 2003). Es importante mencionar que la exploración del espacio de búsqueda debe ser balanceada entre la diversificación y la intensificación.

Los algoritmos metaheurísticos se pueden clasificar según las características que este presenta. A continuación se mencionará los tipos de clasificación más importantes (Blum y Roli, 2003):

- Inspirados en la naturaleza vs lo no inspirados.
- Basado en poblaciones vs un solo punto de búsqueda.
- Función objetivo dinámica vs la estática.
- Una estructura de vecinos vs varias.
- Uso de mayor memoria vs a un uso menor.

1.2.2.3 Métodos basados en poblaciones

Estos métodos se basan en trabajar con un conjunto de soluciones en cada iteración del algoritmo en vez de trabajar con una sola solución. Proporcionan una forma natural para la exploración del espacio de búsqueda. Los métodos más estudiados en base a poblaciones son la computación evolutiva (CE) y optimización de colonia de hormigas (Blum y Roli, 2003).

El método de CE está inspirado en la capacidad de la naturaleza que tienen los seres vivos para adaptarse a su entorno. Estos pueden ser comparados con el proceso evolutivo. Estos consisten en la modificación de un individuo mediante operadores de mutación o cruzamiento para generar la siguiente generación de la población. La fuerza motriz de los algoritmos evolutivos es la selección de los individuos dependiendo de sus características en cuanto a la mayor aptitud tengan, tienen mayor probabilidad a ser elegidos como miembros en la próxima generación. En ejemplo claro de este tipo de métodos es el algoritmo genético.

1.2.2.4 Algoritmos Genéticos

Son algoritmos de optimización numérica, basados en los genes y la selección genética natural. Básicamente consiste en la selección de una población, extraída de las conjeturas de la solución a un problema, en la cual se van generando buenas y malas soluciones individuales bajo ciertos criterios. Luego, mediante operaciones se provoca la mezcla de los fragmentos de las mejores soluciones para formar un nuevo promedio de mejores soluciones (Coley.D , 1999).

- Cromosomas:
  Es la representación en binario de una solución o individuo, en donde se conoce a cada elemento como gen. Cuya representación se puede observar en la Figura 2.1 (a).

- Población:
  Es un conjunto finito de cromosomas. Cuya representación se puede observar en la Figura 2.1 (b).

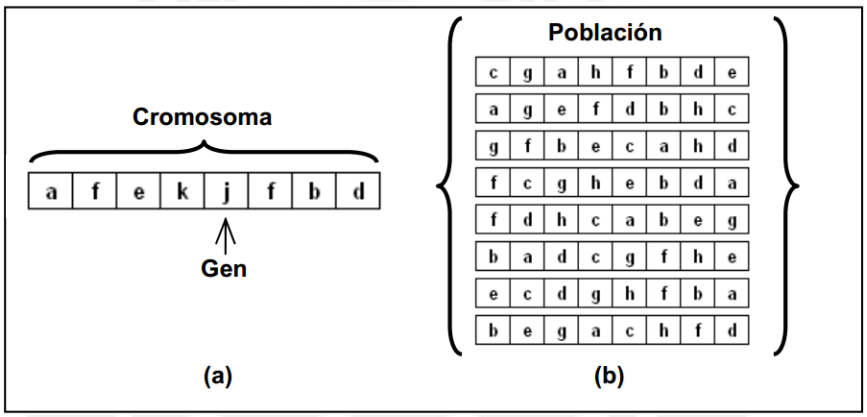


Figura 2.1.- Representación de la población en un algoritmo genético. Imagen recuperada de (Vélez.M y Montoya.J, 2007).

- Aptitud:
  Se refiere al criterio que se utiliza para condicionar la calidad de un cromosoma. A mayor aptitud se observa una mejor solución y posee una mayor probabilidad para sobrevivir y transmita sus características a sus generaciones.

- Cruce:
  Es una operación que se utilizan para producir un nuevo descendiente a la población a partir de dos cromosomas padre seleccionados al azar. En la Figura 2.2 (a) se muestra un ejemplo de este operador.

- Mutación:
  Es una operación que se encarga de seleccionar y cambiar al azar uno o más genes de un cromosoma, produciendo una nueva especie. Esta operación ocurre con probabilidades muy bajas. En la Figura 2.2 (b) se muestra un ejemplo de este operador.

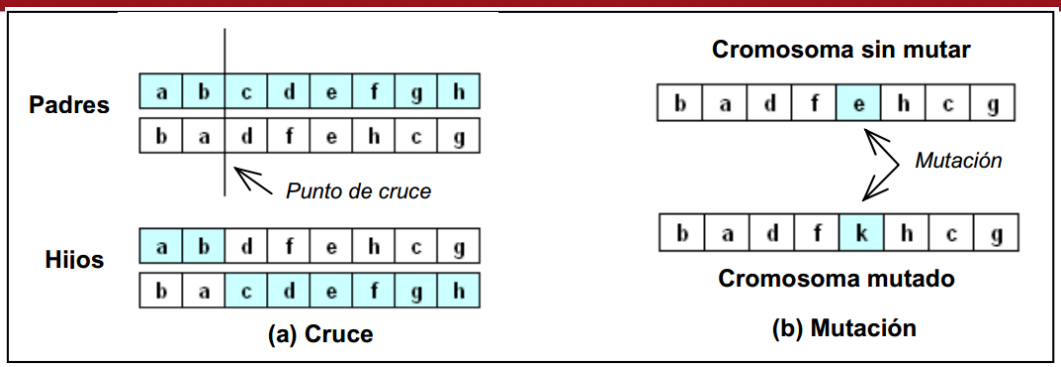


Figura 2.2.- Operaciones básicas en algoritmos genéticos. Imagen recuperada de (Vélez.M y Montoya.J, 2007).

1.2.2.5 Algoritmo Grasp (Greedy Randomized Adaptive Search Procedures)

Es un algoritmo metaheurístico que consta de una técnica de muestreo aleatorio en el cual por cada iteración proporciona una solución al problema. Existe dos fases por cada iteración del GRASP: fase de construcción y fase de mejora. En la primera se construye una solución inicial a partir de una función voraz adaptiva aleatoria y la segunda aplica un procedimiento de búsqueda local sobre la solución inicial con la esperanza de encontrar una solución mejor (Feo y Resende, 1994).

Este algoritmo consta de dos parámetros claves para su solución el α que es conocido como el grado de relajación, lo cual permite tener una solución más aleatoria o más voraz y el número de iteraciones lo cual nos ayuda a determinar una solución con mejor calidad.

En el presente proyecto de fin de carrera se implementara el algoritmo grasp para generar la población inicial que el algoritmo genético requiere.

2 Estado del arte

2.1 Introducción

A continuación se realizará una descripción de las soluciones que se han presentado para la problemática UCTP. Revisando la literatura se han encontrado diversos algoritmos que pueden resolver el problema, así como también sistemas ya implementados. De los cuales se describirán sus funcionalidades específicas y objetivos en forma breve.

2.2 Objetivos de la revisión del estado del arte

La finalidad es que el lector pueda obtener un amplio conocimiento de los métodos y software existentes que se utilizaron para resolver el problema UCTP. Con ello se busca realizar un análisis de comparación entre las soluciones y poder definir si alguna de ellas cubre la problemática propuesta.

2.3 Técnicas existentes para la resolución del problema

A lo largo de los años se han propuesto diversas formas de resolver el problema de UCTP. Estas diversas técnicas poseen diferentes métodos para encontrar la solución, los cuales se pueden clasificar en: método de clúster, métodos secuenciales, búsquedas generalizadas y técnicas basadas en la construcción. A continuación se explicarán alguno de los siguientes métodos ver Figura 2.3.

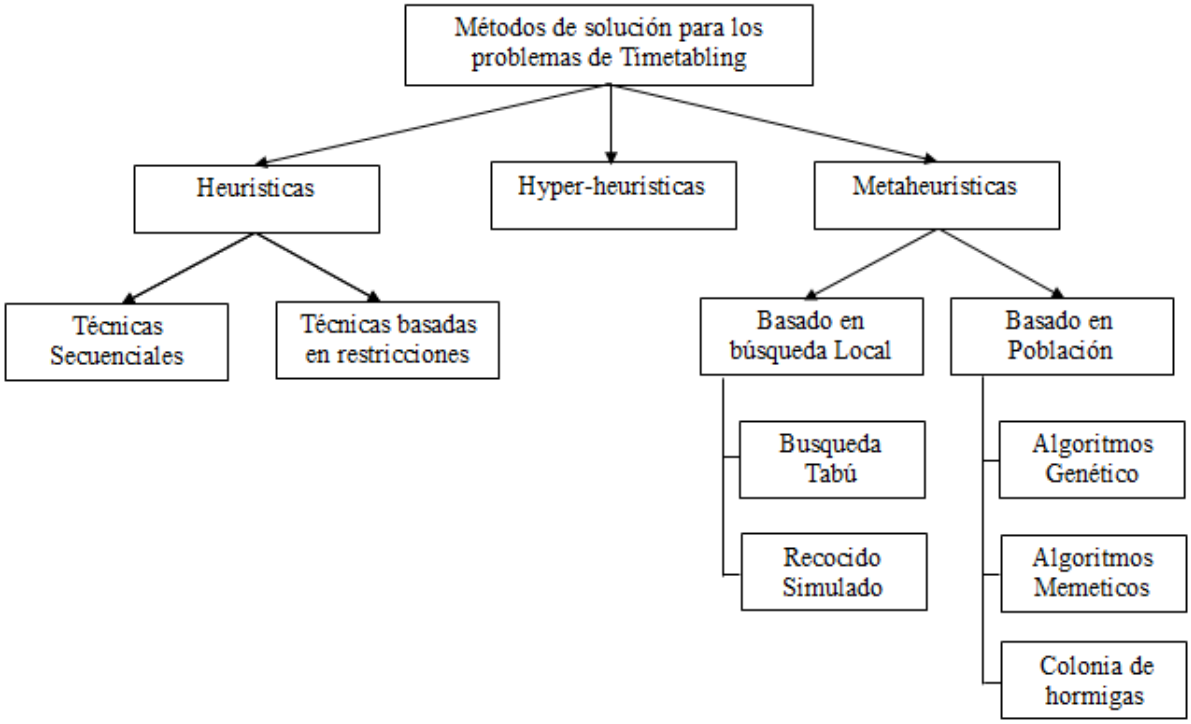


Figura 2.3.- Algunas Técnicas existentes para la resolución del problema de timetabling.

2.3.1 Técnicas secuenciales basadas en grafos

Los problemas de timetabling pueden ser modelados como los problemas de coloración de grafos. Este problema consiste en representar a los eventos como los vértices y las restricciones fuertes como las aristas en grafo no dirigido. El objetivo del problema de los grafos es poder colorear todos los vértices pero con la restricción que los vértices adyacentes no tengan el mismo color, esto se puede hacer de manera análoga en el problema del timetabling, con los vértices como eventos y los colores que se deben pintar en cada vértice reflejan un intervalo de tiempo particular, y las aristas entre los vértices reflejan el problema de la asignación de diferentes intervalos de tiempo que se le asigna a los eventos. Como al modelar el problema no se toman en cuenta las restricciones suaves es necesario que estas sean consideradas aparte, ya que esas restricciones determinan la calidad de la solución (Qu.R, Burke.E,.. ,2006).

El método de coloreado de grafos aplicado en los problemas de timetabling es un método de construcción que ordena los eventos y se van asignando uno por uno, bajo el criterio de la dificultad que tengan al momento de realizar un horario. Debido a que la fase de ordenamiento es muy importante, a lo largo de la literatura existen diversos investigadores que han tratado de buscar mejores estrategias de ordenamiento. En el 2004, Burke y Newall investigaron una nueva manera de realizar el proceso de ordenamiento dinámicamente (por cada iteración al hallar la solución del problema el ordenamiento se iba adaptando). La actualización de cada iteración se lograba gracias a la experiencia obtenida en el proceso anterior dependiendo de la dificultad de asignación, de esta manera iba aprendiendo iterativamente. Además cabe recalcar que de esta manera ya no es necesario hallar un ordenamiento inicial.

Se puedo observar otras aplicaciones de este método en (Asumini, 2004) donde además aplican funciones de lógica difusa para evaluar apropiadamente ordenamiento encontrado.

2.3.2 Técnicas basadas en restricciones

Estas soluciones están basadas en las técnicas de programación de lógica de restricciones. Para el caso de los problemas de timetabling se modela a un evento como una variable con dominios finitos, los valores de los intervalos de tiempo y salas se asignan secuencialmente para ir creando la solución. Además se utiliza el backtraking, volver a través de los pasos, ya que cuando la asignación va avanzando y llega a un punto donde es imposible darle un valor a una variable se necesita volver al último estado donde se tenía otra posibilidad de elección, de esta manera va a ir formando otra solución posible. Este tipo de métodos generalmente tienen un costo computacionalmente elevado ya que el tiempo de ejecución aumenta exponencialmente con respecto al número de variables. Esta técnica por lo general no arroja buenas soluciones comparado con las otras soluciones existentes cuando el problema de optimización es complejo.

En el 2003 Merlo, utilizo la programación basadas en restricciones para producir soluciones iniciales. Luego, esta sirvió como entrada para mejorar las soluciones en otros métodos como los algoritmos de recocido simulado y escalando la montaña. Se aplicó para generar un horario de exámenes ordenándolos primero según el tamaño de intervalo de tiempo que tenía y luego se iba asignando uno por uno para formar una solución. Este híbrido fue probado en la universidad de Melbourne.

Algo muy parecido fue utilizado por Duong y Lam en el 2004, quien empleó la programación basada en restricciones para crear una solución inicial y luego lo aplico en su algoritmo de recocido simulado. Además empleó el backtraking para disminuir el esfuerzo de búsqueda. Este fue utilizado para solucionar el problema en la universidad tecnológica HoChiMinh.

2.3.3 Técnicas basadas en búsquedas locales

Las búsquedas locales son aquellas que resuelven un problema realizando una búsqueda en su propio vecindario. El vecindario es creado a partir de la modificación mediante diversos operadores de una solución inicial, de esta manera se obtiene un espacio de búsqueda. Esta búsqueda posee una función objetivo para determinar qué solución es mejor para la generación de horarios. Dependiendo de los parámetros y las características del espacio de búsqueda se puede determinar el rendimiento y eficiencia del uso de este tipo de técnica. A continuación se presentara dos técnicas muy nombradas en la literatura:

- **Busqueda tabú**

Este algoritmo consiste en la revisión de un espacio de búsqueda pero tomando en cuenta de que no visita el mismo punto dos veces ya que este algoritmo posee una lista llamada *lista tabú* en la cual se conservan los últimos movimientos realizados. Ellos pueden determinar que se encontró una mejor solución en base a una estrategia de aspiración. Si es que no se encontró una solución revisando todo su optimo local entonces puede saltar la búsqueda a otros vecindarios, si estos vecindarios no otorgan buenas soluciones este puede conservar la solución de su optimo local.

En el 2001 Di Gaspero y Schearf aplicaron este algoritmo con la particularidad de que su vecindario de soluciones estaba conformado por soluciones que violaban algunas restricciones fuertes y débiles del problema, por lo tanto se dedicaron a estudiar una estrategia de selección exhaustiva. Además, utilizaron una lista tabú dinámica y adaptaban la función de coste mientras que se realizaba la búsqueda. Este enfoque se aplicó en otros problemas y tuvo un resultado similar comparado con la utilización de otros algoritmos como memeticos (apud Qu.R, Burke.E,.. ,2006). Más tarde se mejoró el enfoque utilizando múltiples vecindarios con características peculiares (Di Gastparo, 2002).

Existen otras investigaciones donde se aplicó el algoritmo tabú en los problemas de timetabling con diferentes enfoques, como darle diferentes prioridades a las restricciones (Paquete y Stuzle, 2002), tomar restricciones lo más semejantes al mundo real para hacer una comparación con otros resultados (Nguyen.k, Tran.N,.., 2010), combinar el tabú con Grasp (soluciones hibridas) para mejor la solución (Bejerano.G, Alva.F,.., 2009), entre otros.

- **Recocido simulado**

Este método es basado en la analogía del simulado de recocido de los metales. El término de recocido se refiere a un proceso físico en el que un sólido es calentado mediante temperaturas altas para que este pase a una fase liquida y luego sea enfriado lentamente mediante la disminución de la temperatura. De esta manera se dice que las partículas enfriadas poseen menos energía (Van Laarhoven. P y

Aarts.E, 1992). En los problemas de optimización combinatoria esa fluctuación de energía aleatoria en el sistema se utiliza para escapar del mínimo local hacia el mínimo global. Si se hace analogía con el problema de timetabling, los estados del sistema vendrían a ser las soluciones factibles, la energía seria la función de coste, la temperatura vendrían a ser el parámetro de control y el estado congelado la solución heurística del problema.

Existen diversas investigaciones usando este algoritmo sobre la problemática de timetabling a lo largo de la literatura, como Duong and Lam (2004), quienes presentan un método de solución al problema de la generación de horarios de exámenes usando dos fases: la primera en encontrar una solución inicial y utilizar el algoritmo de recocido simulado para encontrar una buena vecindad y la segunda fase presenta mecanismos de refinamiento para la solución. Así como otras.

2.3.4 Técnicas basadas en algoritmos de población

Son algoritmos metaheurísticos que se basan en el comportamiento natural. Son comúnmente usados para ayudar a resolver los problemas de optimización combinatorio como el problema de timetabling. A continuación se presentará alguno de estos:

- **Algoritmo genético**

Esta técnica consiste básicamente en tener una solución inicial representada en un cromosoma e ir evaluando este cromosoma mediante los operadores de mutación y cruce para que en la evolución se mejore la solución. Este método es común mente usado en la resolución de muchos problemas de timetabling y existen diversas investigaciones en este tema (Qu.R, Burke.E,.. ,2006). En el 2002 Wong, Cote y Gely utilizaron este método para resolver un problema de generación de horarios de exámenes, este fue modelado para satisfacer las restricciones del problema. Además se realizó un torneo de selección para escoger a los padres y las estrategias de mutación para mejorar la producción de mejores candidatos.

- **Algoritmo memeticos**

Este tipo de métodos combina la habilidad de la exploración de búsqueda de los algoritmos evolutivos con los métodos de búsquedas locales (apup Sadaf.N y Yang.S, 2008). Esta técnica presenta buenas soluciones pero toma un alto tiempo computacional.

Los algoritmos Memeticos han sido aplicados a diferentes problemas de timetabling. En el 2008 se publicó "A Memetic Algorithm for the University Course Timetabling Problem" por Sadaf.N y Yang.S, en el cual se presenta una la solución al problema UCTP. Para ello se mezclaron dos técnicas de búsqueda local dentro de un algoritmo genético. Existen otros estudios en el cual se puede observar el uso de este tipo de algoritmos (Abdullah.S, Burke.E y McColumm, 2007).

- **Algoritmo colonia de hormigas**

Esta técnica simula la búsqueda de los caminos más cortos que usan las hormigas cuando van a buscar su comida. Estas se guían dependiendo de la intensidad del olor de las feromonas que dejan en los caminos, en el más corto se intensifica el olor. De

forma similar se realiza una analogía con los problemas de optimización combinatoria, donde cada hormiga se utiliza para construir una solución y las feromonas representan la información obtenida durante la búsqueda, la cual ayuda a generar soluciones en la siguiente etapa.

Este tipo de algoritmos no son comúnmente usado en los problemas de timetabling pero en el 2004 Naji Azimi utiliza este algoritmo para resolver el problema de generación de horarios de exámenes.

2.4 Sistemas Existentes

En la actualidad existen algunos sistemas para la generación de horario de clases, de exámenes u otro tipo de evento. A continuación mencionaremos algunos de ellos para poder concluir si alguno de estos sistemas satisface la necesidad de nuestra problemática.

- **UTTS (University timetabling scheduler)**

En el 2000 Lim desarrollo un sistema para la generación de horarios de clase y de exámenes para todas las facultades de la universidad nacional de Singapur (UTTS). Este desarrollo básicamente se dividió en dos porciones uno que era la generación de clase y la otra de examen (UTTSExam).

El sistema se encuentra desarrollado en lenguaje java, usando Visual Age para java y Microsoft Access para la base de datos. El sistema está diseñado en tres niveles que es muy común cuando se desarrolla una aplicación cliente/ servidor. Estas tres capas son la vista donde se pueda apreciar la interfaz del software con todos sus componentes, la aplicación donde se encuentra la lógica y la persistencia.

Para la elaboración eficiente de horarios el sistema utiliza un algoritmo genético combinado con búsqueda tabú para mejorar la solución. En la Figura 2.4 se puede observar una vista del software para la generación de horarios.

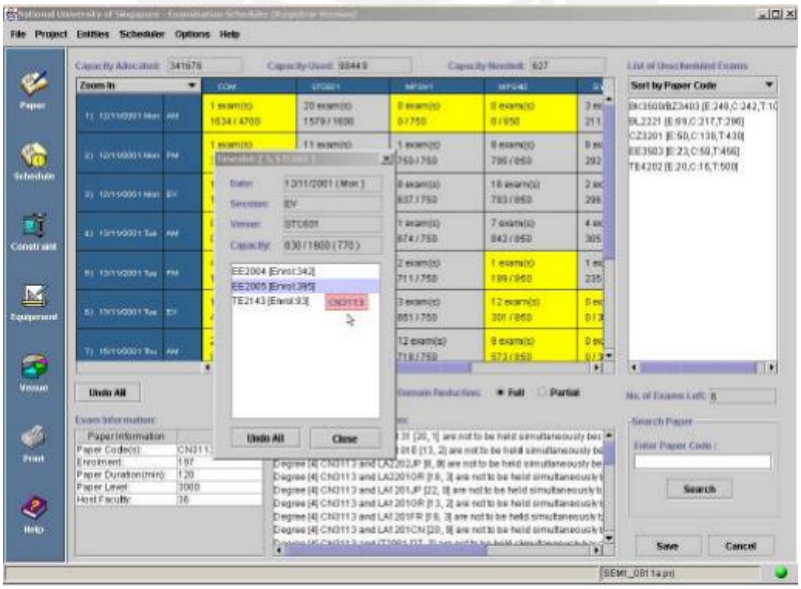


Figura 2.4.- Pantalla del generador de horarios. Imagen recuperada de (Lim.A, Ang.J, Ho.W y Oon.W ,2002).

- **Sistema generador de horarios de clase para la universidad Economía y negocio de Atenas**

  En el 2001 Dimopoulou and Miliotis desarrollo un sistema para los problemas de generación de horario de clase y examen. Este sistema ha sido implementado como Microsoft Access para la base de datos. El método de programación entera se ha desarrollado en los paquetes MPCODE y XPRESS/MP para windows.

  Este sistema cuenta con cinco módulos: la data, control del sistema, Optimización, generación de reportes y evaluación.

  Módulo de data: Contiene toda la distribución de las tablas de data. Todo lo relacionado al problema cursos, profesores, horas, entre otros.

  Módulo de control: Incluye las interface de usuario y el motor para hallar la una distribución optima de los horarios.

  Módulo de Optimización: En el cual se puede realizar modificaciones manuales a la solución ya entregada por el modulo anterior.

  Módulo de reportes: Genera diversos reportes como los horarios de los profesores, la disponibilidad de los salones de clase, los cursos por cada departamento, entre otros.

  Módulo de evaluación: Evalúa la calidad de la distribución final. Este es un proceso semi manual ya que se necesita de datos de entrada.

  El sistema utiliza el modelo de programación entera para asignar los cursos a un intervalo respectivo para un salón dado. A continuación se muestra la distribución de todo el sistema.

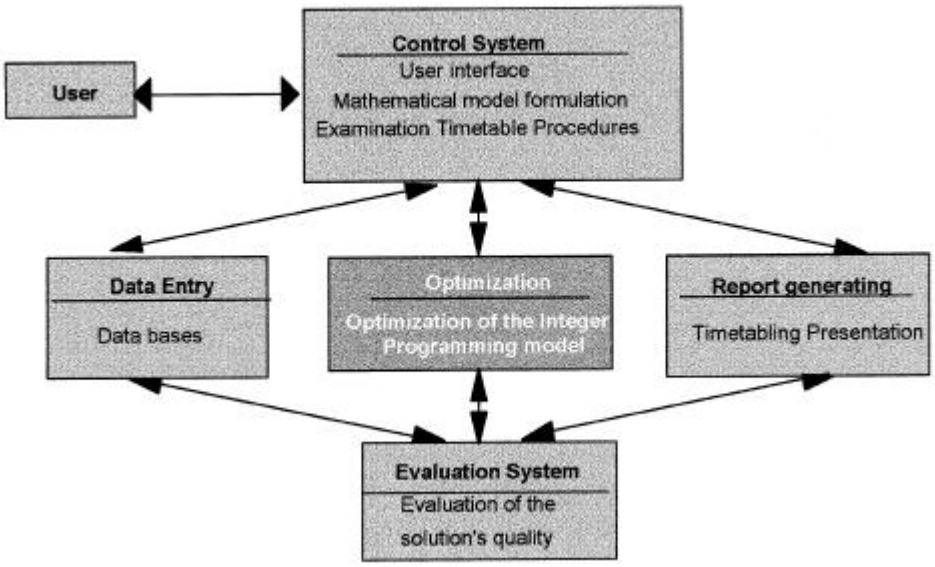


Figura 2.5.- Sistema Completo. Imagen recuperada de (Dimopoulou.M, Miliotis.P, 2001).

- **SiPUCP**
  Este sistema consta de diversas funcionalidades que se encuentran divididas en módulos. Uno de estos módulos apoya la elaboración de horarios de clases, examen y laboratorios en la PUCP.

  Este módulo básicamente se encarga de verificar que no existen cruces de horarios de clase, de salones, de profesores, entre otros. Además, el sistema brinda apoyo al realizar las búsquedas de disponibilidad de horas y salones. La desventaja de este sistema es que para poder realizar la verificación se necesita ingresar la distribución de horario, lo que implica tener que elaborar los horarios manualmente.

2.5 Conclusiones sobre el estado del arte

Según lo presentado se observa que existen diferentes métodos que han sido aplicados para resolver el problema de timetabling, sin embargo, cada uno de ellos poseen una calidad de solución diferente. Lo cual depende de diversos factores como la complejidad de elaboración del algoritmo, el tiempo computacional de la técnica, la adaptación y flexibilidad al problema, entre otros. Adicionalmente se observó que los métodos híbridos arrojan buenas soluciones ya que poseen una solución inicial bastante buena mediante un algoritmo eficiente y luego esta se vuelve a optimizar con otro método, lo cual presenta una solución sumamente buena.

Por otro lado, en cuanto a los sistemas existentes se observa que son muy eficientes para el escenario planteado, pero como se mencionó al inicio el problema de UCTP puede ser muy variable ya que cada institución posee diferentes reglas y restricciones. Por lo tanto, como las soluciones expuestas se realizaron para un caso específico sería complicado poder adecuar ese sistema para nuestra problemática. Además, cabe mencionar que son sistemas privados y no se encuentran en el mercado. Con respecto al sistema actual de la PUCP se puede observar que no cubre toda la problemática, solo te ayuda a simplificar algunos aspectos (en este caso la verificación de cruce).

Debido a ello en el siguiente trabajo se busca elaborar un sistema para la generación de horarios mediante el uso de un algoritmo hibrido, para mostrar una distribución óptima y mejorar la utilización de recursos.

CAPÍTULO 3

1 Resultado Esperado 1

Descripción del proceso del caso de estudio siguiendo notación BPMN.

1.1 Introducción

A continuación se describirá detalladamente el modelo del proceso de la generación de horarios que utiliza actualmente la PUCP. Además, se mencionará los conceptos claves para el mejor entendimiento del proceso.

1.2 Conceptos

1.2.1 Clasificación de docente según su dedicación

Existen tres tipos distintos de docentes según su dedicación, los cuales se diferencian en la cantidad de horas de trabajo (esta cantidad está estipulada en el contrato de acuerdo a su condición):

- Profesores de tiempo completo (TC): Son aquellos que trabajan 40 horas semanales en la universidad (entre las cuales no todas son dictado de clase). Cabe recalcar que este tipo de docente posee una mayor flexibilidad para ser ubicado en un horario determinado.

- Profesores parciales convencionales (TPC): Son aquellos que trabajan 20 o 30 horas semanales en la universidad, de igual manera que los profesores de tiempo completo no todas las horas son de dictado de clase. Debido a esta condición se les puede considerar con una restricción de tiempo moderada.

- Profesores parciales por asignaturas (TPA): Son aquellos cuya actividad académica se centra en la instrucción de a uno o más asignaturas. La cantidad máxima de horas que se le puede asignar es de 19 horas semanales. Cabe mencionar que este tipo de docente es aquel que posee más restricciones de tiempo.

1.2.2 Tipo de Horario

Cada curso que se dicta en la PUCP tiene asociado un conjunto de tipos de horarios. La universidad maneja los siguientes (no todos los cursos contienen todos, pero si por lo menos uno):

- Clase (T): Consta de sesiones teóricas, las cuales deben ser dictadas con un periodo semanal. Además, deben cumplir con el mínimo de horas que se le impone según el curso.

- Laboratorio (L): Es un medio de evaluación las cuales son desarrolladas en laboratorios o talleres. Son llamadas también prácticas de tipo b.

- Práctica (P): Es un medio de evaluación el cual tiene como objetivo el desarrollo de la capacidad crítica y análisis del estudiante. Son llamadas como prácticas de tipo a, las cuales son desarrolladas en las aulas.

- Examen (E): Es una evaluación que se toma dos veces por semestre, en el cual se evalúa todo lo aprendido hasta el momento.

1.2.3 Grupo de Horario

Algunos curso de la universidad posee más de un grupo de horario. Esto se debe a que la cantidad de alumnos supera la capacidad de las aulas y por ello es necesario abrir un nuevo grupo de horario. Por ejemplo: El curso de algoritmo del quinto ciclo posee dos grupos de horario el H101 y H102.

1.2.4 Turnos de estudio

En la PUCP se tiene tres tipos de turnos de estudio, los cuales son un factor muy importante en la distribución de horarios de cursos ya que generalmente los horarios de un ciclo académico se encuentran en un solo turno. La universidad maneja los siguientes:

- Mañana: Considerado a partir de 7:00 am hasta las 12:00 pm.
- Tarde: Considerado a partir de 12:00 pm hasta las 6:00 pm.
- Noche: Considerado a partir de 6:00 am hasta las 10:00 pm.

1.2.5 Secuencias

Las secuencias nos indican las fechas en las que se va a dictar un tipo de horario. Para algunos tipos de horario todos los cursos manejan la misma secuencia, pero hay algunos casos en los que no. En la Figura 3.1 se puede apreciar el calendario de PUCP indicando las secuencias respectivas. A continuación se define a detalle las más importante secuencias que maneja la PUCP:

- Clases
  Secuencia C: Es aquella en la cual se dictan todas las semanas durante el periodo académico. Con la excepción de la semana de exámenes.

- Practicas
  Secuencia A: Es aquella que se realizan quincenalmente y empiezan la primera semana de inicio de prácticas.

  Secuencia B: Es aquella que se realizan quincenalmente y empiezan la segunda semana de inicio de prácticas.

  Secuencia S: Son aquellas que se realizan semanalmente.

- Laboratorios
  Secuencia J: Es aquella que se realizan quincenalmente y empiezan la primera semana de inicio de prácticas.

  Secuencia K: Es aquella que se realizan quincenalmente y empiezan la segunda semana de inicio de prácticas.

  Secuencia W: Son aquellos que se realizan semanalmente.

- Exámenes
  Sección E: Se realizan dos veces en el semestre por cada curso.

FACULTAD DE CIENCIAS E INGENIERÍA

Calendario Académico del semestre 2013-2

Aprobado por C.F. de FCI (21/06/2013)

PUCP

Semana	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SABADO
1	V X 19-ago	V X 20-ago	V X 21-ago	V X 22-ago	V X 23-ago	V X 24-ago
2	V X 26-ago	V X 27-ago	V X 28-ago	V X 29-ago	**FERIADO** 30-ago	V X 31-ago
3	V X W K1 02-sep	V X W K1 03-sep	V X W K1 04-sep	V X W K1 05-sep	V X W K1 06-sep	***O1*** V X W I1 K1 07-sep
4	***Y1*** S X W A1 H1 J1 09-sep	***Y1*** X W A1 H1 J1 10-sep	X W A1 H1 J1 11-sep	***Y1*** X W A1 H1 J1 12-sep	X W A1 H1 J1 13-sep	***Y1*** X W A1 H1 J1 14-sep
5	***Z1*** S V W B1 L1 K2 16-sep	V W L1 K2 17-sep	V W B1 L1 K2 18-sep	V W B1 L1 K2 19-sep	V W B1 L1 K2 20-sep	V W B1 L1 K2 21-sep
6	S X W A2 M1 J2 23-sep	X W A2 M1 J2 24-sep	X W A2 M1 J2 25-sep	X W A2 M1 J2 26-sep	X W A2 M1 J2 27-sep	X W A2 M1 J2 28-sep
7	***O2*** S V W B2 N1 K3 30-sep	V W N1 K3 01-oct	V W B2 N1 K3 02-oct	V W B2 N1 K3 03-oct	V W B2 N1 K3 04-oct	V W B2 I2 N1 K3 05-oct
8	**FERIADO** 07-oct	**FERIADO** 08-oct	V X W J3 09-oct	***Y2*** V X W J3 10-oct	V X W J3 11-oct	***Y2*** V X W J3 12-oct
Exámenes Regulares	***Z2*** **E1** 14-oct	**E1** 15-oct	**E1** 16-oct	**E1** 17-oct	**E1** 18-oct	**E1** 19-oct

Figura 3.1.- Calendario académico hasta exámenes parciales observando las secciones existentes. Imagen recuperada de página web de la Universidad Católica del Perú.

1.2.6 Preferencias de los docentes

Si bien en principio los docentes pueden dictar sus cursos en cualquier aula (siempre y cuando esta cumpla las condiciones necesarias para el dictado), algunos poseen ciertas preferencias debido a diversos factores, los cuales en algunos casos deben ser tomados en cuenta (casos de lesión que impidan al docente subir escaleras, que una docente se encuentre en gestación, etc.).

1.2.7 Restricciones de aulas

Existen ciertos cursos que necesariamente deben ser dictados en un tipo de aula específica (o tienen un tipo de horario que requiere una), debido a las necesidades del mismo. Por ejemplo:

- Algoritmia: Si bien las clases de este curso pueden ser dictadas en un aula convencional, las sesiones de laboratorio deben llevarse a cabo en un aula informática que posea las herramientas requeridas.

- Redes: Al igual que el curso de algoritmia, las clases y algunas evaluaciones como prácticas y exámenes pueden ser dictadas en salones convencionales. Sin embargo, las sesiones de laboratorio requieren material específico para su desarrollo.

1.2 Proceso de elaboración de Horario

Actualmente la PUCP cuenta con un proceso semi-manual. El cual consta básicamente de dos etapas: la primera parte es la elaboración de la distribución del horario para el semestre planteado (manual) y la segunda parte consta de la verificación de este (automática). En el Anexo 1 se puede observar el modelo de todo el proceso completo.

A continuación, se explicará el flujo para que se pueda observar los procedimientos o actividades que se podrían mejorar con la ayuda de la solución planteada.

En este proceso se encuentran involucrados principalmente 3 subdivisiones las cuales son la Secretaria de la Facultad de Ciencia e Ingeniera (Secretaria CI), la Secretaria de cada Especialidad (Secretaria EI) y el encargado de cada especialidad para realizar el horario. El flujo comienza en la Secretaria CI, la cual es encargada de realizar las actualizaciones de las secuencias existentes, luego de esta actualización la Secretaria EI se encarga de registrar la lista de cursos disponibles para el semestre, los cuales deben estar previamente aprobadas por la facultad de Ciencia e Ingeniería. En este paso se debe registrar la cantidad de vacantes por cada curso, los tipos de horario que tiene cada curso, el ciclo al que pertenecen, la cantidad de horarios por semestre, la cantidad de horas necesarias de dictado, entre otros. A continuación, se registrar la lista de profesores disponibles para el ciclo, para ello es necesario que se registre la actualización de la disponibilidad de tiempo, los cursos que puede dictar, el tipo de profesor ya que este puede variar según contrato de un ciclo a otro e impacta con la cantidad de horas máximas dictadas, entre otras actualización de información personal. Para finalizar las actualizaciones de los insumos necesarios para la elaboración de horarios es necesario registrar las aulas disponibles para ese semestre, así como la capacidad de cada aula y las características de cada aula.

Luego de culminar con todas las actualizaciones de los insumos necesarios en el sistema existente, se ingresa el horario generado por el semestre anterior. La razón de cargar este horario es reducir el tiempo de preparación ya que si no tendría que armar una nueva combinación con la cantidad de cursos, profesores, aulas; lo cual tomaría varias semanas para su elaboración. En la Figura 3.2 se puede aprecia el flujo de la secuencia explicada.

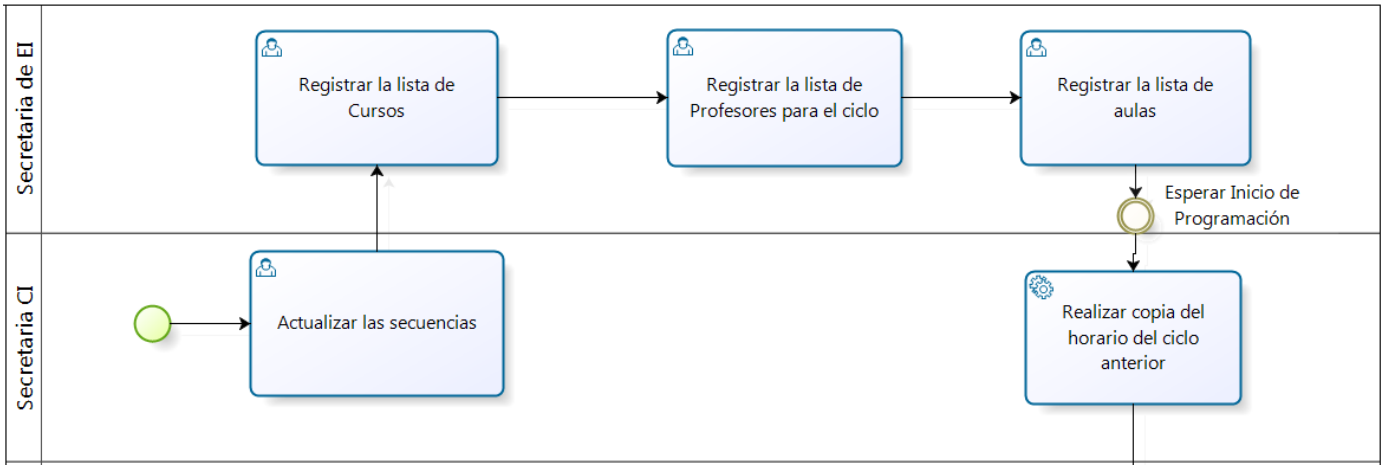


Figura 3.2- Parte inicial del proceso de elaboración de Horario.

Luego de culminar la primera etapa, el proceso tiene dos opciones para continuar dependiendo si la condición de *¿Existe algún cambio?* se cumple o no. Para esta parte del proceso la universidad cuenta con un Sistema llamado SIPUCP el cual cuenta con un módulo que ayuda a la modificación y verificación de los horarios. Hay que tener en cuenta que el Sistema existente no fue diseñado con el objetivo de realizar la creación de horarios, sino que fue adaptado para que uno de sus módulos pueda servir de ayuda a la elaboración de este.

Continuando con el flujo, si no existe algún cambio con respecto al horario anterior (lo cual tiene una probabilidad de ocurrencia muy baja) se procede con la finalización del proceso. Por otro lado, si esto no sucede existen tres tipos de modificaciones que se pueden realizar: crear un nuevo Horario, modificar el Horario (en el cual se puede variar una de sus 4 componentes: el profesor, el aula, el curso o el tiempo) y eliminar un horario. En la Figura 3.3 se puede aprecia el flujo de las tres posibles modificaciones.

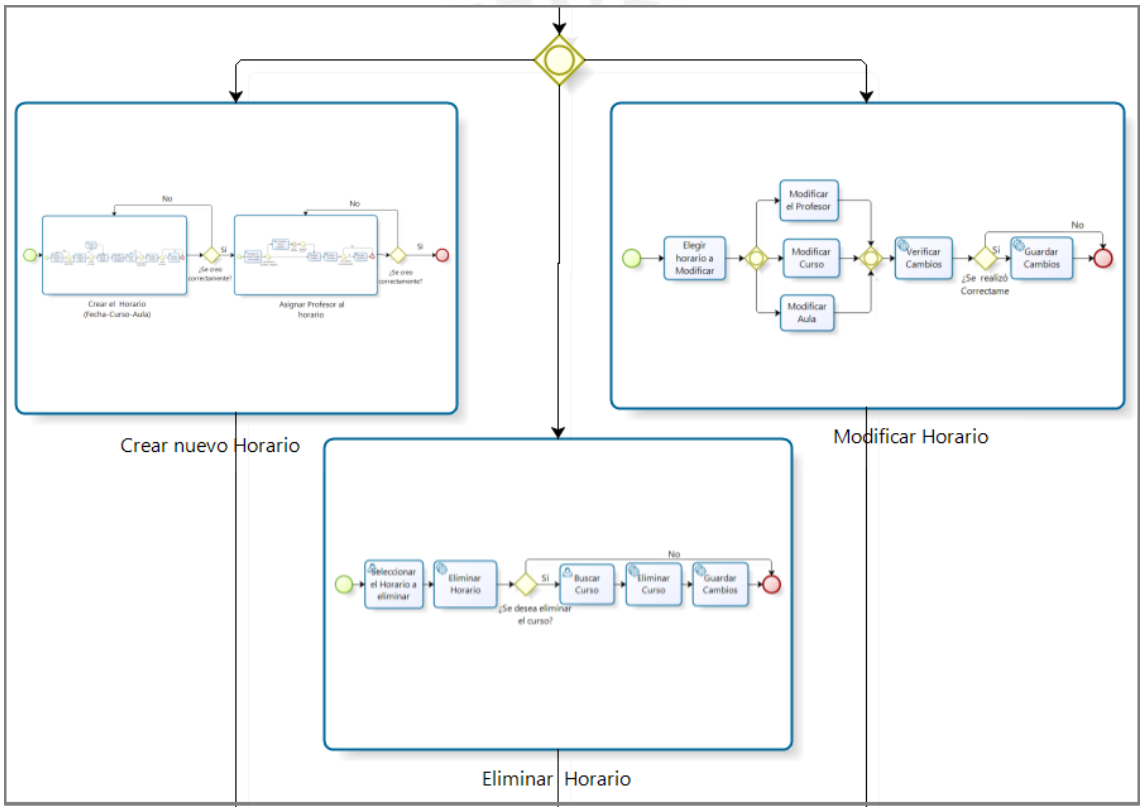


Figura 3.3.- Posibles modificaciones que se pueden realizar al horario anterior.

A continuación se explicara a detalle los tres tipos de modificaciones:

- Creación de un nuevo horario
  Se llama horario a la relación de asignación de un curso, un aula y un profesor en un determinado tiempo. Esta actividad se realiza cuando se desea generar una nueva relación nunca antes asignada (todos los miembros de la asignación van hacer cambiados). Este subproceso tiene principalmente dos etapas: Crear el horario (Fecha-Curso-Aula) y Asignar al profesor a dictar.

  La primera etapa se inicia cuando se escoge el slot de tiempo donde se desea crear la nueva asignación, luego se elige el curso si este no existe (en la

aplicación del módulo de generación de horario) se crea en caso contrario se selecciona y se procede a escoger el tipo de horario. A continuación se procede con la elección de numero de comisión, esto quiere decir que si se quiere tener dos grupos de horarios con las mismas características en cuestión de relación Curso-Profesor. En este paso solo se selecciona la cantidad, luego se procede a crear el horario para la siguiente comisión ejecutando el proceso nuevamente. A continuación se procede con la búsqueda de aulas disponibles esta actividad termina cuando se halla encontrado una. Luego el sistema anuncia los cambios que se están realizando y los guarda.

La segunda Etapa consiste en asignar la relación de un horario (Fecha-Curso-Aula) a un profesor. Si todavía no se tiene determinado que profesor se puede asignar, se procede a realizar conversaciones con las posibles opciones para verificar que se tenga disponibilidad en el horario indicado. En el anexo 1.1 se puede observar el detalle del subproceso modelado.

- Modificación de un horario
  Esta actividad se realiza cuando se desea modificar uno o más elementos (Profesor, Curso y Aula) de un horario antes asignado. Este subproceso comienza con la selección del horario a modificar, luego se realiza los cambios a los elementos necesarios, el sistema anuncia los cambios y los guarda.

- Eliminación de un nuevo horario
  Esta actividad se realiza cuando se desea eliminar la asignación ya existente (Tiempo-Profesor-Curso-Aula). Esto se puede deber a que se haya eliminado algunos de sus elementos o que el slot asignado ya no sea factible. Este subproceso comienza con la selección del horario a eliminar, luego se realiza la eliminación. Por último, el sistema anuncia los cambios y los guarda.

Para finalizar el proceso es necesario preguntar si es que no existe algún otro cambio. Si existe se procede de nuevo con los subprocesos antes mencionados. Si no el sistema verifica los cambios, si existen cruces el sistema te alerta sobre los cruces, no obstante si lo aceptas el sistema permitirá registrar un horario con cruces.

Actualmente el sistema es un apoyo el cual permite realizar las modificaciones antes mencionadas, pero los usuarios de este sistema han encontrado ciertas deficiencias en su uso. Por ejemplo: Cuando se realiza la búsqueda de aulas disponibles se tiene que indicar un slot de tiempo preciso y no se puede realizar por bloque de horas, de esta manera, si se busca registrar un curso de 3 horas de dictado se tiene que encontrar manualmente las horas indicadas. O cuando se desea eliminar un curso, esta eliminación es tediosa ya que se tiene que eliminar cada tipo de curso que este tiene asociado. Estos son algunos ejemplos que se pueden resaltar entre otros.

CAPÍTULO 4

1 Resultado Esperado 2

Descripción y justificación de la estructura del cromosoma y de los demás elementos necesarios para la representación del problema.

1.1 Introducción

A continuación se describirá la estructura del cromosoma que se utilizará para resolver el problema planteado. Además, se detallarán las otras estructuras necesarias para la resolución.

1.2 Detalle de la estructura del cromosoma

La estructura definida para la solución al problema consta de una matriz de par de tuplas de dos dimensiones. La cual representa la asignación de los horarios. Esta matriz está conformada por los ejes (evento por slot de tiempo).

Para mayor entendimiento se debe definir algunas estructuras:

- Un evento representa el curso que se quiere asignar, el tipo de horario al que pertenece el curso, el número de comisión que tiene y el grupo de horario al que pertenece.

  Por Ejemplo:

  Evento 1 =

Algoritmia	Clase	3	H102

- Un tiempo de slot se encuentra definido por unidad de tiempo en horas. Por lo tanto, se toma la cantidad de horas semanales y cada una de estas horas representa un slot de tiempo. Es por eso que la estructura de Ti será:

  $$T= \{t1, t2, t3….t\}$$

  Donde ti representa el slot de tiempo en el orden i = 1, 2,3… 86.

  Por Ejemplo:

  Time Slot 1 = Lunes 08:00 – 09:00 am.
  Time Slot 2 = Lunes 09:00 – 10:00 am.

- Una tupla (Pt, At) representa a la asignación de un profesor-aula a un específico evento y en un determinado tiempo de slot. La primera componente de la tupla hace referencia al profesor escogido y la segunda componente al aula.

  Por Ejemplo:

  (Pt, At) = (5,6), lo cual representa al profesor de código 5 en el aula 6.
  (Pt, At) = (5,7), lo cual representa al profesor de código 5 en el aula 7.

En donde Pt hace referencia al código del profesor y At representa el código del aula. Esto códigos sirven para identificar todas las características que poseen ambas componentes.

Por ejemplo cuando se obtenga el código del profesor con un valor de 5. Se presentan las siguientes características:

Profesor 5 =

Luis Bello	40	19	Disponibilidad	Cursos

Donde 40 es la máxima cantidad de horas de dictado, 19 es la mínima cantidad de horas (estos valores dependen del tipo de dedicación que tenga el profesor), la disponibilidad es descrita por cada tiempo de slots y los cursos son aquellos que el profesor puede dictar.

Por otro lado, en el caso del código de aula con valor de 6. Se presenta las siguientes características:

Aula 6 =

V202	30	Laboratorio	Disponibilidad

Donde V202 es la descripción del aula, 30 es la capacidad máxima del salón, el laboratorio es el tipo de aula y por último se cuenta con la disponibilidad de uso.

Considerando todas las estructuras antes planteadas se ha elaborado la estructura del cromosoma, la cual será representará por una matriz de dos dimensiones (eventos x slot) que se muestra a continuación:

	Tiempo Slot 1	Tiempo Slot 2	Tiempo Slot 3	…
Evento 1	(P1,A1)			(Pt,At)
Evento 2	.			
Evento 3	.		(P3,A3)	
Evento 4	.			
Evento 5	.	(P2,A2)		
Evento 6	.			
Evento 7	.			
…	(Pk,Ak)			

Figura 4.1.- Estructura del Cromosoma

Para una mayor comprensión de la interpretación del cromosoma planteado se muestra un pequeño ejemplo de asignación, el cual cuenta con la siguiente información: 5 eventos, 7 slot de tiempos y un conjunto de 6 profesores y 7 aulas disponibles a asignar.

H = {

(1,2)	(1,2)	(1,2)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)
(2,3)	(2,3)	(2,3)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)
(-1,-1)	(-1,-1)	(-1,-1)	(3,4)	(3,4)	(3,4)	(-1,-1)
(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)
(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)	(-1,-1)	(-2,5)	(-2,5)

}

Para iniciar con la interpretación del cromosoma se debe tener claro lo siguiente: cada fila de la matriz de tuplas representa a un evento, de esta manera, la primera fila está relacionada a la asignación del evento 1, la segunda al evento 2 y así sucesivamente. Para el caso de las columnas de la matriz cada una de estas identifica un tiempo de slot, de esta manera, se entiende que la primera columna le pertenece al slot 1, la segunda al slot 2 y así sucesivamente.

En cuanto a la interpretación del valor que se presenta en cada tupla se observar que los slots que no han sido asignados serán identificados cuando contengan un valor (-1,-1), esto ayudará a identificar que slot están asignados y cuáles no. Además, con esta asignación de -1 se puede identificar que eventos no cuentan con una asignación a lo largo de todas las posibilidades de tiempo, por lo tanto, según el ejemplo el evento 4 no posee ninguna asignación. Por otro lado, se observar que existen tuplas que contienen un valor positivo en ambas componentes y otras que solo contiene una valor positivo en la segunda componente y la primera componente tiene una asignación de -2, esto se debe básicamente al tipo de evento que se está asignando. Los eventos que son de tipo clase se ven en la necesidad de tener asignado a un profesor y un aula fijo, debido a ello es que ambas componentes de la tupla tienen un valor positivo, según se muestra en el ejemplo el evento 1 se le ha asignado en el slot 1, slot 2 y slot 3 la tupla de (1,2), lo cual indica que se le ha asignado el profesor con código 1 y el aula con código 2. A diferencia de los eventos clases existen los eventos tipo evaluación (prácticas y laboratorios), los cuales no poseen una asignación fija de profesor ya que estos eventos son desarrollados de diferente manera y cuentan con uno o más jefes de evaluación que no necesariamente son categorizados como docentes en la PUCP. Por este motivo se está trabajando con un valor de -2 en la primera componente de los eventos tipo evaluación como se muestra en el ejemplo, el evento 5.

Cabe mencionar que el cromosoma planteado tiene la capacidad de soportar asignaciones de dos o más eventos en el mismo tiempo de slot, como se aprecia en la asignación del evento 1 y el evento 2 del ejemplo, esta asignación será de manera correcta siempre y cuando los eventos pertenezcan a diferentes grupos de horario, diferentes grupos de secuencias (laboratorios y prácticas pertenecen a diferentes secuencias por ello si pueden ser asignados en el mismo tiempo) y diferentes grupos de comisiones si es que pertenecen a un mismo grupo de horario.

Finalmente, se considera que la estructura del cromosoma planteado cumple con las necesidades que requiere el problema. Además, por la cantidad de posibles combinaciones que se tiene es mejor la aplicación de tuplas para que se pueda acceder a las asignaciones directamente, ya que un matriz de cuatro dimensiones ocuparía demasiado espacio de memoria y el recorrido de la asignación aumentaría el tiempo de ejecución del algoritmo.

1.3 Detalle de las demás estructuras

A continuación se detallará el conjunto de todos los datos que serán necesarios para el planteamiento del problema. En el próximo capítulo se utilizarán estas definiciones para diseñar las restricciones del problema.

- Sea D = {d1, d2, d3...} el vector de docentes que encuentran disponibles para dictar en el semestre evaluado.

  Ejemplo:

Alfonso M.	Manuel T.	Miguel G.	...

- Sea A = {a1, a2, a3...} el vector de aulas que se encuentras disponibles para el semestre evaluado.

  Ejemplo:

V201	V202	V203	...

- Sea C = {c1, c2, c3…} el vector de cursos académicos que se dictarán en el semestre evaluado.

  Ejemplo:

Algoritmo	LP1	Ética	...

- Sea S = {s1, s2, s3,…} el vector de tipo de horario. Dos distintos tipos de horario pueden tener asignado un mismo curso, en cuyo caso las distintas sesiones no pueden ser programados en un mismo periodo de tiempo si es que pertenecen a una misma secuencia.

  Ejemplo:

Clase	Práctica	Laboratorio	...

- Sea G = {g1, g2, g3,…} el vector de grupo de horario. Cada grupo de horario tiene asignado un conjunto de cursos que pertenecen a un mismo ciclo curricular. Además, los cursos asignados a un grupo de horario no pueden ser programados en un mismo periodo de tiempo.

  Ejemplo:

H101	H102	H103	...

- Sea L = {I1, I2…, I6} el vector de días de la semana. En el cual se describe los días que están permitidos las asignaciones de clases.

  Ejemplo:

Lunes	Martes	...	Sábado

- Sea T = {1, 2 ,3….t} el vector de tiempo. Cada uno representa un slot de tiempo, este vector tendrá una longitud de 86 elementos.

  Ejemplo:

Lunes 08:00	Lunes 09:00	...	Sábado 06:00

- Sea TR= {tr1, tr2,...} el vector de turnos. Los cuales según lo explicado son 3.

Ejemplo:

Mañana	Tarde	Noche	...

- Sea N= {n1, n2….} sea el vector de ciclos de la curricular. Cada ciclo contiene un grupo de cursos, los cuales en lo deseable deben ser asignados a un turno en particular.

Ejemplo:

Quinto C.	Cuarto C.	...	Decimo C.

- Sea DC la matriz de los cursos que dictan de los profesores en un semestre determinado. Donde los cursos ∈ C y los profesores ∈ D.

Ejemplo:

	Miguel G.	Manuel T.	...	DC n
LP1	0	1	...	1
LP2	0	0	...	0
Ética	1	1	...	1
...	0	0	...	0

- Sea DD la matriz de la disponibilidad de los profesores. Las cuales están dentro de los slots permitidos. Donde los slots ∈ T y los profesores ∈ D.

Ejemplo:

	Lunes 08:00	Lunes 09:00	....	DD p
Miguel G.	0	0	....	0
Manuel T.	1	1	....	1
...	0	0	....	0

- Sea CA el vector de la capacidad máxima de las aulas. Donde las aulas ∈ A.

Ejemplo:

V201	V203	...	CA q
50	30	...	20

- Sea CH el vector de cantidad de horas por cursos semanalmente según la currícula. Donde los cursos ∈ C.

Ejemplo:

Ética	Lp1	...	CH m
6	6	...	5

- Sea CHM el vector de cantidad de horas máximo por cursos por día en un ciclo determinado. Donde los cursos ∈ C.

Ejemplo:

Ética	Lp1	…	CH m
3	3	…	5

- Sea DH el vector de horas máximas que un docente puede dictar semanalmente. Donde los profesores ∈ D.

Ejemplo:

Miguel G.	Manuel T.	…	DH n
10	12	…	7

- Sea DHM el vector de horas mínimas que un docente puede dictar semanalmente. Donde los profesores ∈ D.

Ejemplo:

Miguel G.	Manuel T.	…	DH n
7	5	…	7

- Sea DA la matriz de la descripción de las aulas. La cual indica que tipo de horario puede ser dictado en un aula especifica según las características de cada aula.

Ejemplo:

	V201	V202	…	DA q
Clase	0	0	…	0
Práctica	1	1	…	1
…	1	1	…	1

- Sea TC la matriz de descripción de la composición de los cursos. En la cual se indica si un curso posee más de un tipo de horario.

Ejemplo:

	LP1	LP2	…	TC m
Clase	1	1	…	1
Práctica	1	1	…	0
…	0	0	…	0

- Sea R el vector que contiene restricciones especiales para formar el horario académico. Lo cual contendrá slots prohibidos.

Ejemplo:

30	45.	…	60

- Sea NAC la matriz de la descripción de cursos al nivel académico. Es la relación que se tiene en el plan curricular de cada especialidad.

Ejemplo:

	LP1	LP2	…	NAC m
Quinto C	1	0	…	0
Cuarto C	0	0	…	0
…	0	0	…	0

- Sea PTN la matriz de las preferencias de dictado de nivel.

Ejemplo:

	Quinto C	Cuarto C	…	PTN l
Mañana	0	1	…	0
Tarde	1	1	…	1
Noche	0	0	…	0

- Sea PD la matriz de la preferencia de dictados de los profesores.

Ejemplo:

	Lunes 08:00	Lunes 09:00	….	PD t
Miguel G.	0	1	…	0
Manuel T.	1	1	…	1
…	0	0	…	0

CAPÍTULO 5

1 Resultado Esperado 3

Definición y justificación de la función de fitness, la cual toma en cuenta las consideraciones y restricciones del problema.

1.1 Introducción

En este capítulo se describirá detalladamente las restricciones que posee el problema y en base a estas se definirá la función de fitness (función de adaptación) que cumpla con las necesidades de la problemática.

1.2 Definición de las restricciones del problema

Como ya se mencionó anteriormente existen dos tipos de restricciones en los problemas de timetabling: Las fuertes y las débiles. A continuación, se explicarán las restricciones que se tomarán en cuenta en este caso de estudio las cuales se caracterizan en los tipos antes mencionados.

Para ello se tiene que tomar en cuenta:

$X_{tcda}=1$; Si el curso c es asignado al docente d en el aula a en el periodo de tiempo t

$X_{tcda}=0$; Si es que no el evento no se encuentra asignado

1.2.1 Restricciones Fuertes

RF1: Un curso C a lo más será asignado a un docente D en un periodo de tiempo T y en un aula específica A.

$$\sum_{d=1}^{n}\sum_{a=1}^{q} X_{tcda} \leq 1$$

$$\forall c \in C, \forall t \in T$$

RF2: Un docente D a lo más deberá ser asignado a un curso C y un aula A en un mismo periodo T.

$$\sum_{c=1}^{m}\sum_{a=1}^{q} X_{tcda} \leq 1$$

$$\forall d \in D, \forall t \in T$$

RF3: Un aula A lo más debe ser asignada a un curso C y docente D en un mismo periodo de tiempo T.

$$\sum_{c=1}^{m}\sum_{d=1}^{n} X_{tcda} \leq 1$$

$$\forall a \in A, \forall t \in T$$

RF4: Un docente D debe ser asignado respetando sus horas de disponibilidad, las cuales depende del tipo de profesor según su dedicación.

$$X_{tcda} = \mathrm{DD}_{dt}$$
$$\forall Xtcda = 1, \forall c \in C, \forall t \in T, \forall d \in D, \ \forall a \in A$$

RF5: Un docente D debe ser asignado a un curso C que este calificado para enseñar. Para ello se necesita tener la relación de cursos que el profesor está permitido dictar.

$$X_{tcda} = \mathrm{DC}_{dc}$$
$$\forall Xtcda = 1, \ \forall c \in C, \forall t \in T, \forall d \in D, \ \forall a \in A$$

RF6: Cada curso dictado debe cumplir con la cantidad de horas estipuladas en el plan de estudios.

$$\sum_{t=1}^{p}\sum_{d=1}^{n}\sum_{a=1}^{q} X_{tcda} = CHc$$
$$\forall c \in C$$

RF7: Los bloques de horas de un evento que pertenecen a un mismo curso se deben dictar de manera consecutivas.

$$\sum_{d=1}^{n}\sum_{a=1}^{q}\sum_{t=r}^{CH_c} X_{tcda} = CHMc$$
$$\forall c \in C$$

Donde: r es el inicio de periodo del curso c

RF8: Un curso C a lo más debe de tener tres horas consecutivas de clase por día.

$$\sum_{d=1}^{n}\sum_{a=1}^{q}\sum_{t=r}^{CH_c} X_{tcda} \leq 3$$
$$\forall c \in C$$

Donde: r es el inicio de periodo del curso c

RF9: Un bloque de horas de curso a los más se debe asignar uno por día.

$$\sum_{d=1}^{n}\sum_{a=1}^{q}\sum_{t=r}^{f} X_{tcda} = 1$$
$$\forall c \in C$$

Donde: r es el inicio de periodo del curso c y f fin del bloque de tiempo.

RF10: Los cursos de tipo laboratorio deben ser asignados en aulas con la cualidad de laboratorios.

$$DA_{ai} = TC_{ci}$$
$$\forall Xtcda = 1, \forall c \in C, \forall t \in T, \forall d \in D, \ \forall a \in A$$

Donde: i es de tipo laboratorio

RF11: Los cursos de tipo clase no deben asignarse a aulas tipo laboratorio.

$$DA_{ak} = TC_{ck}$$
$$\forall Xtcda = 1\ ,\ \forall c \in C, \forall t \in T, \forall d \in D,\ \forall a \in A$$

Donde: k es de tipo teoría

RF12: No se podrá asignar cursos en fechas indicadas (jueves cultural u otras restricciones).

$$R_{tcda} \neq X_{tcda}$$
$$\forall Xtcda = 1\ , \forall c \in C, \forall t \in T, \forall d \in D,\ \forall a \in A$$

RF13: No deben coincidir los cursos que corresponden a un mismo nivel académico.

$$\sum_{t=1}^{p} \sum_{n=1}^{l} \left(\sum_{c=1}^{m} X_{tcda} = 1\right)$$
$$\forall a \in A, \forall d \in D$$

RF14: Los cursos son clasificados en turnos dependiendo al nivel académico al que pertenece.

$$PTN_{tcda} = X_{tcda}$$
$$\forall Xtcda = 1\ , \forall c \in C, \forall t \in T, \forall d \in D,\ \forall a \in A$$

1.2.2 Restricciones Débiles

RD1: Las horas de dictado de clase deben distribuirse de la mejor manera; es decir, que no sean asignados en días consecutivos.

$$X_{tcda} \neq X_{(t+1)cda}$$
$$\forall Xtcda = 1\ , \forall c \in C, \forall t \in L, \forall d \in D,\ \forall a \in A$$

RD2: Se debe evitar que una evaluación (práctica o laboratorio) sea asignada consecutivamente a una clase.

$$DA_{ak} = DA_{aj}$$
$$\forall Xtcda = 1\ , \forall c \in C, \forall t \in T, \forall d \in D,\ \forall a \in A$$

Donde: k es de tipo teoría y j es laboratorio.

RD3: Se va a considerar las horas de preferencia de dictado de los profesores.

$$PD_{tcda} = X_{tcda}$$
$$\forall Xtcda = 1, \forall c \in C, \forall t \in T, \forall d \in D, \forall a \in A$$

RD4: Todo profesor tendrá una mínima cantidad de cursos a dictar en la semana.

$$\sum_{c=1}^{m}\sum_{a=1}^{q}\sum_{t=1}^{p} X_{tcda} \geq DHM_d$$
$$\forall d \in D$$

RD5: Cada profesor asignado no deberá pasar las horas máximas semanalmente por ciclo.

$$\sum_{c=1}^{m}\sum_{a=1}^{q}\sum_{t=1}^{p} X_{tcda} \leq DH_d$$
$$\forall d \in D$$

1.3 Definición de la función fitness

A partir de las restricciones descritas anteriormente se ha definido la siguiente función objetivo:

$$\mathcal{M}im \quad \mathcal{F} = \mathcal{W}1 * \mathcal{F}rf(x) + \mathcal{W}2 * \mathcal{F}rd(x)$$

Donde:

$$Frf(x) = w1 * RF4 + w2 * RF5 + w3 * RF7 + w4 * RF8 + w5 * RF11 + w6 * RF14$$

$$Frd(x) = w1 * RD1 + w2 * RD2 + w3 * RD3 + w4 * RD4 + w5 * RD5$$

$\mathcal{W}1, \mathcal{W}2$ = Peso por el incumpliendo de las restricciones. $\mathcal{W}1 \geq \mathcal{W}2$

La función fitness busca minimizar el valor de las restricciones que no han sido cumplidas, dando un mayor valor de penalización al incumplimiento de las restricciones fuertes que a las débiles. También se debe tener en cuenta que no todas las restricciones fuertes han sido consideradas en la función fitness ya que algunas son fundamentales, es decir, que si estas no son cumplidas la solución se convierte en inválida. Para este caso, las restricciones que no se han consideradas en la función objetivo son las siguientes: RF1, RF2, RF3, RF6, RF9, RF10, RF12 y RF13 ya que son consideradas fundamentales, si se incumplen estas restricciones la solución sería errónea debido a que estarían arrojando como solución situaciones irreales, como por ejemplo que un profesor sea asignado en el mismo slot de tiempo a dos distintos eventos o que no sea asignado todas las horas de un curso según lo estipulado la currícula.

CAPÍTULO 6

1 Resultado Esperado 4

Definición, descripción y justificación de los operadores genéticos a utilizar en el algoritmo genético.

1.1 Introducción

En este capítulo se describirá detalladamente los operadores genéticos que se utilizarán para generar nuevos individuos para la población.

1.2 Operador de selección

Se utiliza para seleccionar a dos individuos de la población, los cuales luego serán afectados por el operador de cruzamiento o casamiento para formar parte de los descendientes de la siguiente generación. Esta selección se realizará con el método de la ruleta para favorecer a los individuos mejor adaptados, se le asignará un valor de probabilidad en función de su adaptación. Por lo tanto, el que posee un menor valor en la función de adaptación (minimizar) tendrá una probabilidad mayor de ser elegido.

Por ejemplo para una población de 4 individuos se tiene lo siguiente:

# Individuo	1	2	3	4
Valor Función Fitness	170.1	197.3	207.7	412.9
Probabilidad de selección	0.32	0.28	0.27	0.13

Tabla 6.1.- Selección de probabilidad y función fitness

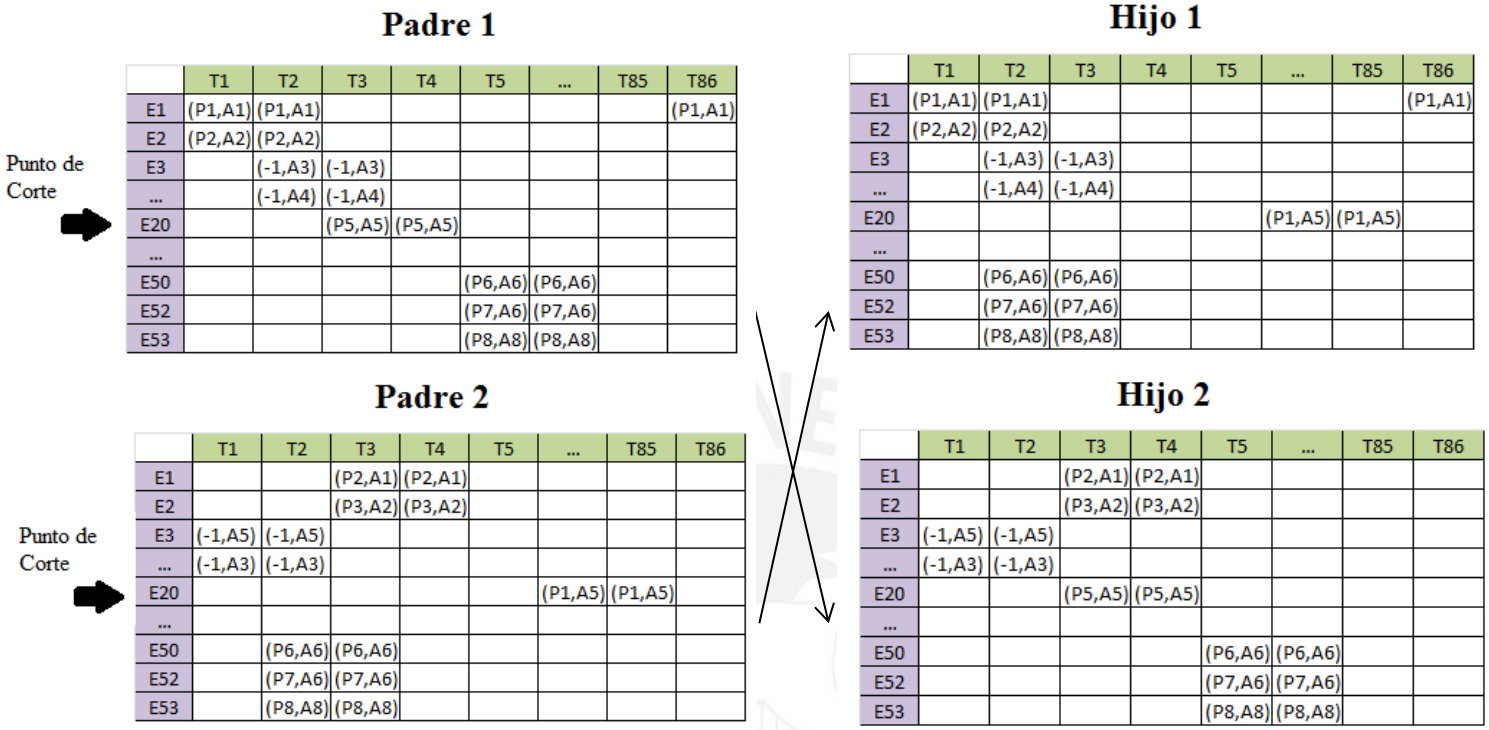
En la tabla anterior se puede observar que el individuo 1, el cual posee un valor de función de adaptación menor, tiene mayor porcentaje de probabilidad de ser elegido con un rango de [0,0.32], por otro lado el individuo 4 el cual posee un valor de función de adaptación muy alto posee una menor probabilidad de ser elegido con una rango de [0.87, 1]. Luego de haber definido los valores de probabilidad de selección de cada individuo, se elige un número random entre [0,1] con el cual se definirá que individuo será seleccionado para el cruzamiento. Por ejemplo: si los números aleatorios son 0.20, 0.54 y 0.62 entonces los individuos seleccionados serian 1, 2 y 3 respectivamente. De esta manera se seleccionaran a los padres para generar la siguiente generación.

1.3 Operador de casamiento

Se utiliza para cruzar a dos individuos de la población con el objetivo de formar una nueva generación de individuos. Para este operador se utilizará una tasa de casamiento, la cual determinará la probabilidad de cruces que puede existir en una población, ya que no todos los pares de individuos que existen son afectados por el operador. Según lecturas el valor de la probabilidad normalmente está comprendido entre 0.5 y 1.0.

El procedimiento se inicia con los dos padres seleccionados anteriormente por el operador de selección. Luego, se determina una posición al azar (Punto de Corte) por la cual se producirá el corte de los dos individuos para poder recombinar sus extremos y formar los nuevos individuos. Esta combinación permita la creación de dos hijos, el primero de ellos está conformado por la parte 1 del individuo 1 y la parte 2 del individuo 2 y el segundo por la parte 1 del individuo 2 y la parte 2 del individuo 1.

Como se muestra a continuación en la Figura 6.1. Estos dos nuevos individuos creados Hijo1 y Hijo2 pasaran luego por un proceso de evaluación para saber si son aptos para la supervivencia a la siguiente generación. Cabe recalcar que el punto de corte se ha elegido de manera al azar tomando como máximo valor la cantidad de los eventos.



Padre 1	T1	T2	T3	T4	T5	...	T85	T86
E1	(P1,A1)	(P1,A1)						(P1,A1)
E2	(P2,A2)	(P2,A2)						
E3		(-1,A3)	(-1,A3)					
...		(-1,A4)	(-1,A4)					
E20			(P5,A5)	(P5,A5)				
...								
E50					(P6,A6)	(P6,A6)		
E52					(P7,A6)	(P7,A6)		
E53					(P8,A8)	(P8,A8)		

Hijo 1	T1	T2	T3	T4	T5	...	T85	T86
E1	(P1,A1)	(P1,A1)						(P1,A1)
E2	(P2,A2)	(P2,A2)						
E3		(-1,A3)	(-1,A3)					
...		(-1,A4)	(-1,A4)					
E20						(P1,A5)	(P1,A5)	
...								
E50		(P6,A6)	(P6,A6)					
E52		(P7,A6)	(P7,A6)					
E53		(P8,A8)	(P8,A8)					

Padre 2	T1	T2	T3	T4	T5	...	T85	T86
E1			(P2,A1)	(P2,A1)				
E2			(P3,A2)	(P3,A2)				
E3	(-1,A5)	(-1,A5)						
...	(-1,A3)	(-1,A3)						
E20						(P1,A5)	(P1,A5)	
...								
E50		(P6,A6)	(P6,A6)					
E52		(P7,A6)	(P7,A6)					
E53		(P8,A8)	(P8,A8)					

Hijo 2	T1	T2	T3	T4	T5	...	T85	T86
E1			(P2,A1)	(P2,A1)				
E2			(P3,A2)	(P3,A2)				
E3	(-1,A5)	(-1,A5)						
...	(-1,A3)	(-1,A3)						
E20			(P5,A5)	(P5,A5)				
...								
E50					(P6,A6)	(P6,A6)		
E52					(P7,A6)	(P7,A6)		
E53					(P8,A8)	(P8,A8)		

Figura 6.1.- Operador de cruzamiento basado en un solo corte.

1.4 Operador de mutación

Se utiliza para mutar un gen de los individuos, con el objetivo de no dejar ningún espacio de búsqueda con probabilidad cero para ser examinado. Para la aplicación de este operador se utilizará una tasa de mutación, la cual determinara una probabilidad de que un individuo mute, ya que no necesariamente todos los individuos deben ser afectados.

La operación se evalúa sobre los dos hijos generados por el operador de casamiento. Primero se genera un número random entre [0,1] si este número es mayor a la tasa de mutación entonces no se procede con la operación, por otro lado, si este es menor el individuo pasa a ser mutado. Para iniciar la mutación se escoge primero al azar un profesor, luego se escoge los cursos que este puede dictar para de esta manera determinar a qué eventos el profesor puede ser asignado. Como se puede apreciar en la Figura 6.2.

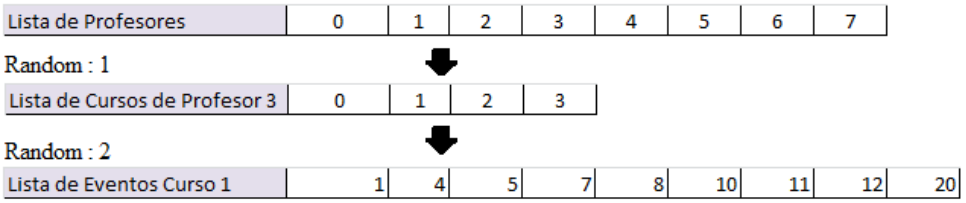
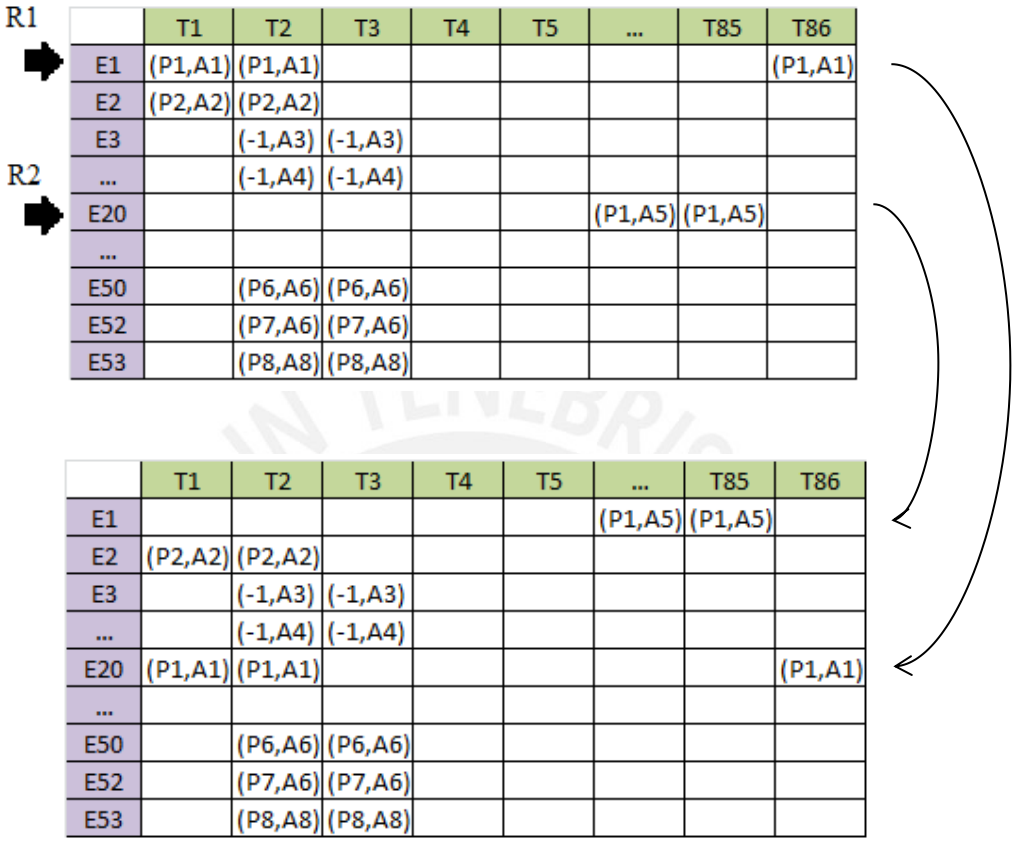


Figura 6.2.- Pasos pasa realizar el operador de mutación para Hijo 1.

Finalmente, se escoge al azar dos eventos del individuo que pertenezcan a la lista de eventos del curso y se procede con el cambio de asignación de slots. Como se muestra en la Figura 6.3.



	T1	T2	T3	T4	T5	...	T85	T86
E1	(P1,A1)	(P1,A1)						(P1,A1)
E2	(P2,A2)	(P2,A2)						
E3		(-1,A3)	(-1,A3)					
...		(-1,A4)	(-1,A4)					
E20						(P1,A5)	(P1,A5)	
...								
E50		(P6,A6)	(P6,A6)					
E52		(P7,A6)	(P7,A6)					
E53		(P8,A8)	(P8,A8)					

	T1	T2	T3	T4	T5	...	T85	T86
E1						(P1,A5)	(P1,A5)	
E2	(P2,A2)	(P2,A2)						
E3		(-1,A3)	(-1,A3)					
...		(-1,A4)	(-1,A4)					
E20	(P1,A1)	(P1,A1)						(P1,A1)
...								
E50		(P6,A6)	(P6,A6)					
E52		(P7,A6)	(P7,A6)					
E53		(P8,A8)	(P8,A8)					

Figura 6.3.- Operador de Mutación al Hijo 1. Cambio de evento E1 y E20.

1.5 Etilismo

Es un método de selección que consiste en copiar al mejor individuo de la población y conservarlo a través de las generaciones hasta que se encuentre un individuo mejor y se actualice el individuo guardado. Este método es utilizado debido a que el uso de los operadores de cruce y mutación tiene una alta probabilidad de perder a los mejores individuos de la población durante las generaciones. Por ello, cabe recalcar que la utilización de este operador puede mejorar el algoritmo genético al conservar la mejor solución.

CAPÍTULO 7

1 Resultado Esperado 5

Diseño del algoritmo genético en pseudocódigo, el cual permita manejar las restricciones del problema para el caso de estudio.

1.1 Introducción

En este capítulo se describirá el pseudocódigo del algoritmo genético implementado en la solución y el algoritmo Grasp fase construcción para generar la población inicial.

1.2 Población Inicial (Algoritmo Grasp- fase Construcción)

Para generar la población inicial se construirá un Grasp fase construcción. El pseudocódigo de la estructura principal se puede observar en la figura 7.1, consta básicamente de encontrar la mejor solución en base a la comparación de las soluciones que arroja la función de fase de construcción. La evaluación de la mejor solución se repetirá tantas veces como se indique en el número de iteraciones (parámetro del algoritmo).

```
1 SubProceso solucionFC <- InicializaPoblacionGrasp (numeroIteraciones, alpha)
2     i <- 0
3     mejorSolucion <- null
4
5     Mientras i < numeroIteracion Hacer
6           solucionFC <- faseConstruccion()
7
8           si(solucionEsMejor)
9                 mejorSolucion <- solucionFC
10          FinSi
11    Fin Mientras
12
13 Fin SubProceso
```

Figura 7.1.- Pseudocódigo de la estructura principal del Grasp

Los valores de los parámetros del Grasp según la literatura revisada (T.Feo y M.Resender), indica un número iteraciones de 10000 para un alpha de 0.5 con el cual se logrará obtener una mejor solución ya que se tiene una obtención mayor de posibles soluciones, además un alpha de 0.5 representa que se encuentra en el punto medio de lo voraz y aleatorio. A partir de esta información se realizó una calibración, explicada en el siguiente capítulo, para finalmente obtener los valores de número de iteraciones de 1000 y un alpha de 0.61.

La función **soluciónEsMejor** tiene dos roles, la primera sirve para evaluar si la solución actual que ha generado la fase de construcción es mejor que la solución que guarda actualmente como la mejor. Esta comparación se base en cual tiene mejor asignación de eventos según el turno al que pertenece el curso. La segunda es la encargada de evaluar si la solución cumple con la asignación de todas las horas de los curso según indica la plan de estudios de la universidad, ya que si no cumple esta restricción básica no sería posible considerarlo como una solución.

Continuando con la explicación del algoritmo en la figura 7.2 se puede observar el pseudocódigo de la fase de construcción, la cual es fundamental ya que cada asignación elaborada cumple con las restricciones fuertes del problema para que no produzca alguna solución inválida.

```
1 SubProceso solucionFC <- faseConstruccion ()
2      solucion <- Inicializo
3      HorasCursosAsignados <-Inicializo
4      profesoresAsignados <-Inicializo
5      aulasAsignadas <-Inicializo
6      clasesAsignadasPorCiclo <-Inicializo
7
8   Para ciclo <-0 hasta TotalCiclo con Paso 1 Hacer
9            cursosCiclo <- EscogeCursosDelCiclo
10           eventosCiclo <- EscogeEventosDelCiclo
11           eventoTipoClase <- EscogerEventosTipo
12           cursosTipoClase <- EscogerCursosTipo
13           aulasTipoClase  <- EscogerAulasTipo
14
15           Para curso <-0 hasta cursoTipoClase con Paso 1 Hacer
16                 eventoCurso <- EscogerEventosPorCurso
17                 profesoresCurso <- EscogerProfesoresPorCurso
18                 conjuntoElemento <- CombinarCandidatoDePosiblesAsignaciones
19
20                 Mientras conjuntoElementoNoVacio o seAsigno
21                       mejorElemento <- EscogerMejorElemento
22                       peorElemento <- EscogerPeorElemento
23
24                       Para Elemento <-0 hasta conjuntoElemento con Paso 1 Hacer
25                             si peorElemento < conjuntoElemento(Elemento).valorFuncionObjetivo y
                                     conjuntoElemento(Elemento).valorFuncionObjetivo <
                                     (peorElemento +(mejorElemento-peorElemento)*alpha))
26                                         Elemento se añade a RCL
27                             FinSi
28
29                             ElementoEscogido <- Se escoge un elemento aleatorio de RCL
30
31                             si elemento es Valido
32                                   Solucion <- Solucion U ElementoEscogido
33                                   Solucion <- llenarBloqueDeHorasRestantes
34                                   Solucion <- llenarHorasCompletaPorEvento
35                                   Solucion <- llenarEventosHorario
36                             FinSi
37
38                             remover ElementoEscogido de la RCL
39
40                       FinPara
41
42                 FinMientras
43
44           FinPara
45
46           Se asigna las horas de Practicas y Laboratorios
47
48     FinPara
49
50 Fin SubProceso
```

Figura 7.2.- Pseudocódigo de la fase de construcción

En la fase de construcción los cursos se asignaran por ciclo, esto quiere decir, que se separaran todos los cursos que pertenezcan a un mismo ciclo y se asignan, luego se escoge el siguiente ciclo y se asignan todos los cursos, así sucesivamente.

Para la asignación de los eventos de un mismo ciclo primero se separará en dos grupos: los eventos de tipo clase y los eventos tipo laboratorio y práctica. Para el primer grupo, se escogen la lista de aulas de tipo clase que pueden ser usadas en la asignación, luego se recorrerá por cursos y se separará los eventos que pertenecen a ese curso (los cuales están integrados por todos los diferentes horarios que posee el curso). Por cada curso se escoge la lista de profesores que pueden dictar y los slots disponibles de ello. Cuando ya se tiene preparado la lista de aulas posibles, los profesores y los slots, se prepara la lista del conjunto de elementos que se pueden asignar. A continuación, se procede a la formación de la RCL según el algoritmo Grasp (escoger el mejor, el peor y seleccionar los que cumplan la condición de la RCL). Una vez que ya se tiene la RCL se escoge el elemento a asignar aleatoriamente, si es válido se asigna el elemento sino se procede a escoger otro elemento del RCL. Luego que se asignó la primera hora del evento se procede a la asignación del bloque de horas del evento y las horas que el evento necesita cubrir en la semana. Para terminar la asignación de los eventos tipo clase se procede asignar todos los eventos que estaban restantes (los demás horarios del mismo curso), tratando de ser asignados en lo mejor posible al mismo slot que se asignó el evento anterior.

Para el segundo grupo se procede con la asignación por tipo de curso (laboratorio y práctica), en este caso se tiene que tener en cuenta que no se asignará a un profesor ya que las prácticas y laboratorios cuentan jefes de practica (lo cual no se tomará en cuenta en el alcance). Otro detalle que se tiene que tomar en es que las prácticas serán asignadas en una secuencia distinta al de los laboratorios, por ello estos dos tipos de evento pueden cruzarse en el tiempo. Cabe recalcar que se está considerando las comisiones que se pueden tener por este tipo de evaluación tanto en prácticas como en laboratorios.

1.3 Algoritmo Genético

Existen tres parámetros importantes en el algoritmo genético el tamaño de la población, la tasa de casamiento y la tasa de mutación. La tasa de casamiento y mutación indica la probabilidad de que los individuos sean afectados por las operaciones de casamiento y mutación respectivamente. Los valores de los parámetros serán seleccionados en base a la literatura y las pruebas ejecutadas por el algoritmo, las cuales son explicadas en el siguiente capítulo. A partir de esta información se definió el tamaño de la población inicial de 80, tasa de casamiento de 0.75 y tasa de mutación de 0.01.

En la figura 7.3 se observa el pseudocódigo principal del algoritmo genético simple, el cual comienza con una población inicial que está conformada por una cantidad determinada de individuos, cada uno de estos representa una solución del problema. Luego esta población inicial pasa por la evaluación de aptitud, conocida como función fitness, para determinar su habilidad de adaptación. Una vez que toda la población ha sido evaluada, se prosigue con la aplicación de los operadores de selección, de casamiento y de mutación. Además, se utilizará etilismo para mantener la mejor solución a través de las iteraciones de las generaciones.

Estas evaluaciones se realizarán tantas veces como lo indique su condición de parada, para la cual se utilizará un número máximo en la cual la población se encuentra sin cambiar a través de las generaciones.

```
1 Proceso AlgoritmoGenetico (tamaño poblacion, tasa casamiento, tasa mutacion)
2       Poblacion <- Lista de soluciones
3       MejorSolucion
4
5       Poblacion <- Inicializar Porblacion inicial
6       Evalua elementos de la población: funcionFitness
7       Mientras  no se cumpla la condicion de parada Hacer
8               P1,P2<- Seleccion_Padres: Operador Seleccion
9               H1,H2 <- Casar Padres Seleccionados : Operador Casamiento
10              H1,H2 <- Mutar nueva generación : Operador Mutacion
11              Poblacion <- evaluar si ingresan a la nueva generación
12              MejorSolucion <- evaluacion de etilismo
13      FinMientras
14 FinProceso
```

Figura 7.3.- Pseudocódigo Principal del genético.

A continuación se presentará el seudocódigo de las principales funciones del algoritmo genético. En la Figura 7.4 se presenta la aplicación del operador de selección, el cual se desarrolla con la lógica antes mencionada, se escoge a un individuo con el método de la ruleta la cual genera mayor probabilidad de selección al que posea mejor valor de función de fitness.

```
1 Proceso Operador Seleccion
2       suma <- 0
3       numeroAleatorio <- Numero random entre [0,1]
4       Para i <- 0 Hasta tamañoPoblacion Con Paso 1 Hacer
5               individuo <- Escoger individuo i
6               suma <- suma + probablididad de individuo
7               si (suma >= numeroAleatorio)
8                       escoger individuo i
9               FinSi
10
11      FinPara
12 FinProceso
```

Figura 7.4.- Operadores del Algoritmo genético (Selección).

En la Figura 7.5 se presenta al operador de casamiento, en el cual tiene como input a los dos padres seleccionados por el operador anterior. Luego se selecciona un punto de corte al azar y se procede a generar los nuevos hijos. Cabe resaltar que para el operador de reproducción o casamiento no se muestra en el pseudocódigo el uso de la tasa de casamiento, ya que esta es aplicada anteriormente para determinar la cantidad de casamiento que va a tener esa generación. (Tamaño de la población * tasa de casamiento).

```
15 Proceso Operador Casamiento (individuo 1, individuo 2)
16     numeroAleatorio <- Numero random entre [0,numeroEventos]
17
18     hijo1 <- Se copia la solucion del individuo 1 del evento [0,numeroAleatorio]
19     hijo1 <- Se copia la solucion del individuo 2 del evento
[numeroAleatorio,numeroEvento]
20     hijo2 <- Se copia la solucion del individuo 2 del evento [0,numeroAleatorio]
21     hijo2 <- Se copia la solucion del individuo 1 del evento
[numeroAleatorio,numeroEvento]
22
23     retorna hijo1 y hijo2
24 FinProceso
```

Figura 7.5.- Operadores del Algoritmo genético (Casamiento).

Por ultimo en la figura 7.6 se presenta el operador de mutación, el cual recibe como input el hijo que desea mutar y la tasa de mutación. Esta tasa indica una probabilidad si este individuo va ser afectado por el operador o no. Para realizar la alteración del cromosoma se optó por escoger dos eventos que son dictados por un mismo profesor e intercambiarlos, finalmente se retorna al hijo mutado o no.

```
27 Proceso Operador Mutacion (hijo,tasaMutacion)
28     probabilidadMutacion <- Numero random entre [0,1]
29
30     si (probabilidadMutacion < tasaMutacion)
31             profesor <- escoger un procesor al azar
32             eventos  <- escoger eventos que puede dictar el profesor
33
34             evento1 <- se escoge al azar un evento de eventos
35             evento2 <- se escoge al azar un evento de eventos
36
37             se cambia de slot de tiempo a los dos eventos elegidos (evento1,evento2)
38     FinSi
39
40 FinProceso
```

Figura 7.6.- Operadores del Algoritmo genético (Mutación).

Continuando con la explicación de las funciones básicas del algoritmo es importante mencionar donde se está aplicando la función de fitness. En la figura 7.3 se mostró que esta función se aplica por primera vez para evaluar a la población inicial, luego esta misma se usará para evaluar si el nuevo individuo creado es apto para ingresar a la población. Esta evaluación consiste básicamente en dos pasos: el primero evalúa con la función fitness el valor de adaptación de este nuevo individuo y el segundo mide la capacidad máxima de la población si esta excede al máximo tamaño, incluyendo al nuevo individuo, entonces el algoritmo procederá a tomar por el método de la ruleta a los que pertenecerán a la nueva población que se generará, sino solo lo ingresará. Este procedimiento se muestra en la Figura 7.7.

```
1 Proceso Evaluacion de Población (PoblacionInicial, individuoNuevo)
2       evalua individuoNuevo : funcion fitness
3       Si ( TamañoPoblacionInicial <= tamanoMaxPoblacion)
4                               numSelecciones = TamañoPoblacionInicial
5                       Sino
6                               numSelecciones = tamanoMaxPoblacion;
7       FinSi
8
9       Individuo seleccionado;
10      Para  k <- 0 Hasta  numSelecciones Con Paso 1 Hacer
11              seleccionado = PoblacionInicial.RemoverEscogerMetodoRuleta
12              Agregar individuo Poblacion (seleccionado)
13      FinPara
14 FinProceso
```

Figura 7.7.- Evaluación del nuevo individuo a la población.

Por último, en la Figura 7.8 se describe la función de control de aberraciones, la cual es fundamental para el algoritmo ya que esta se encargará de controlar a las soluciones que no cumplan con las restricciones fuertes del problema, las cuales convierten a la solución en inválida. Por lo tanto, se encargará de eliminarlas para que no aparezcan en la siguiente generación y no continúen reproduciéndose.

Esta función controla diversos casos, los cuales se explicarán a continuación:

1. Todos los eventos que sean asignados deben cumplir con la validación de que en sus bloques de tiempo consecutivos de un mismo día se les asignen un mismo profesor y aula. Por ejemplo, el evento de clase de algoritmia asignado en el slot 1, slot2 y slot3 (ya que cuenta con tres horas de clase), tenga la asignación del mismo profesor y aula para los tres slots de tiempo.

2. Los bloques de horas asignados de un evento deben pertenecer al mismo día. Por ejemplo, si el evento E asignado es asignado a los slot1, slot2 y slot3. Estos slots deben a un mismo día (lunes, martes, miércoles, etc.)

3. Dos bloques de horas que pertenecen a un mismo evento no pueden ser asignados el mismo día. Por ejemplo, la clase de algoritmia no puede asignarse el lunes por la mañana y luego por la tarde.

4. Ningún evento debe estar asignado en los tiempos prohibidos especificados en las restricciones.

5. Se debe asegurar que los cursos de una misma especialidad, de un mismo ciclo y un mismo horario de clase no se crucen.

6. Es obligatorio que las sesiones tipo laboratorios sean asignados en aulas tipo laboratorio.

7. Un profesor no puede ser asignado en un mismo slot de tiempo a dos eventos distintos. Lo mismo sucede para las aulas.

8. Todos los eventos asignados deben cumplir con las horas estipuladas en la currícula.

Todas estas consideraciones son necesarias para validar que el algoritmo no arroje una solución inválida.

```
1 Proceso Control de Aberraciones (individuo)
2       Para i<-0 Hasta numeroEventos Con Paso 1 Hacer
3               si evento es asignado a diferente aula o profesor por bloque
4                       errorAsignacionEvento <- Falso
5               FinSi
6               si bloque de horas asignados pertenecen a diferentes dias
7                       errorAsignacionBloqueSlot <- Falso
8               FinSi
9               si mas bloques de horas son asignados el mismo dia
10                      errorAsignacionSlot <- Falso
11              FinSi
12              si el bloque se asigno en un horario de restriccion
13                      errorAsignacion <- Falso
14              FinSi
15              si existen cruces de eventos del mismo nivel
16                      errorAsignacionEventoCiclo <- Falso
17              FinSi
18              si evento es laboratorio y aula asignada no es laboratorio
19                      errorAsignacionLaboratorio <- Falso
20              FinSi
21      FinPara
22
23      Para i<-0 Hasta numeroProfesor Con Paso 1 Hacer
24              si Profesor es asignado a mas de un evento en el mismo slot
25                      errorAsignacionProfesor <- Falso
26              FinSi
27      FinPara
28
29      Para i<-0 Hasta numeroAula Con Paso 1 Hacer
30              si Aula es asignado a mas de un evento en el mismo slot
31                      errorAsignacionAula <- Falso
32              FinSi
33      FinPara
34
35      Para i<-0 Hasta numeroCurso Con Paso 1 Hacer
36              si Curso no es asignado a todas las horas de curricula
37                      errorAsignacionCurso <- Falso
38              FinSi
39      FinPara
40      esAberracion <- errorAsignacionEvento y errorAsignacionSlot y errorAsignacionBloqueSlot
         y errorAsignacionEventoCiclo y errorAsignacionLaboratorio y errorAsignacionProfesor
         y errorAsignacionAula y errorAsignacionCurso
41 FinProceso
```

Figura 7.8.- Función de control de aberraciones.

CAPÍTULO 8

1 Pruebas y Resultados Experimentales

1.1 Introducción

En este capítulo se mostrará los resultados que ha arrojado el algoritmo explicado en el capítulo anterior. Además, se presentarán las calibraciones de los parámetros necesarios para la ejecución.

1.2 Calibraciones de los parámetros del algoritmo Grasp

Los valores de los parámetros del Grasp tienen gran influencia en la calidad de la solución, debido a que dependiendo del valor que tome alpha la solución será más voraz o más aleatoria, por lo tanto se debe establecer un término medio en la cual el algoritmo arroje soluciones con buena bondad. En el capítulo anterior se definió un número iteraciones de 1000 y un alpha de 0.61, con lo cual se espera obtener una buena calidad de solución. A continuación se explicará cómo se establecieron estos valores:

Cabe recalcar que la calibración tiene como base los parámetros recomendados en la literatura. Primero se determinará el valor del alpha, la cual partirá con el valor de 0.5, se plantea variar este valor en un rango de -0.2 hasta +0.2 con el objetivo de evaluar los valores de la bondad de solución. Además, se ha fijado el número de iteración con un valor de 1000. El siguiente cuadro se muestra seis resultados de la ejecución del algoritmo, en las cuales se especifica el valor de la solución dependiendo del valor del alpha otorgado. Además, se ha adicionado al final una columna con el promedio de todas las ejecuciones.

Alpha	1era Corrida	2da Corrida	3ra Corrida	4ta Corrida	5ta Corrida	6ta Corrida	Promedio
0.7	99.1	101.5	104.7	99.7	99.8	94.6	99.9
0.65	98.6	91.9	91.9	77.7	97.5	106.9	94.1
0.6	82.3	72.6	97.5	99.4	82.2	112.9	91.2
0.55	85.8	99.8	95.4	99.2	101.3	86.4	94.7
0.5	88.4	110.5	96.2	93.2	83.6	107.6	96.6
0.45	107	76.5	114.9	109.1	88.8	89.4	97.6

Tabla 8.1.- Resultados de calibración del valor del alpha (0.05).

Se ha iniciado con un incremento del alpha del 0.05 para determinar un rango en el cual se presente mejor bondad de solución. Se observó que en el rango de alpha de [0.55, 0.65] la bondad de solución promedio se mantiene menor, lo cual conviene ya que el objetivo es minimizar. Por lo tanto, se ha decidido realizar otra experimentación pero ahora con un incremento menor de +0.02 y -0.02 aplicado en el rango antes seleccionado. En la tabla siguiente se observa los resultados de la experimentación.

Alpha	1era Corrida	2da Corrida	3ra Corrida	4ta Corrida	Promedio
0.65	100.2	99.3	97	102.3	99.7
0.63	85.3	87.4	97.6	93.2	90.875
0.61	95.6	80.2	89.7	84.6	87.525
0.59	97.6	97.3	85.5	86.4	91.7
0.57	91.5	92.4	89.9	85.2	89.75

Tabla 8.2.- Resultados de calibración del valor del alpha (0.02).

Según lo observado en la tabla anterior, se decidió utilizar un alpha de 0.61 ya que presenta una bondad de solución promedio mejor, cabe recalcar que ese valor obtenido convierte a la solución en un poco más voraz que aleatoria.

Para el segundo paso de calibración se decidió establecer el número de iteración con un valor de 10000 como lo indica en la literatura, para luego aplicarle un valor de variación. La cual consiste en ir disminuyendo el valor máximo en saltos de -1000 hasta llegar a un punto en la cual se pueda observar que los valores de bondad de solución lleguen a una meseta. En el cuadro siguiente se pueden apreciar los resultados de la experimentación, tener en cuenta que tanto el valor de la solución y su tiempo de ejecución son promedios.

# Iteraciones	Bondad de Solución	Tiempo de Ejecucion (s)
1,000	84.17	10.00
2000	80.13	15.33
3000	82.93	20.00
4000	81.47	30.00
5000	84.43	35.00
6000	83.60	45.00

Tabla 8.3.- Resultados de calibración del número de iteraciones.

Como se puede apreciar en el cuadro anterior se observa que no existe mucha variación en cuanto a la bondad de solución promedio pero si en el tiempo de ejecución, es por ello que se ha decido tomar el valor de 1000 iteraciones debido a que posee menor tiempo de ejecución.

1.3 Calibraciones de los parámetros del algoritmo genético

Los valores de los parámetros del algoritmo genético que se definieron en el capítulo anterior son: tamaño de la población inicial de 80, tasa de casamiento de 0.75 y tasa de mutación de 0.01 con lo cual se obtienen un mejor valor de la solución. A continuación se explicará cómo se determinaron estos valores:

Se comenzará con fijar el tamaño de la población inicial, según revisado en la literatura, se recomienda no obtener un valor muy grande ya que esta va incrementado su tamaño por cada generación que pasa. Debido a ello se decidió hacer un calibre con un incremento de +10 hasta un valor de 100 como máximo. En el siguiente cuadro se muestran los resultados de la experimentación.

Tamaño Población	Bondad de Solución	Tiempo de Ejecución (s)
50.00	47.45	16.00
60.00	44.85	15.00
70.00	43.85	16.40
80.00	43.85	16.30
90.00	39.95	17.00
100.00	39.95	18.00

Tabla 8.4.- Resultados de calibración del tamaño de población inicial.

Como se puede observar el tamaño de la población de 80 muestra un valor de bondad de solución menor y con un tiempo de ejecución muy cercano al promedio de los tiempos de ejecución, es por ello que se decidió mantener en ese valor para el tamaño de la población.

Una vez obtenido el valor fijo del tamaño de la población inicial se procede a la calibración de la tasa de casamiento y de mutación. Para ambas se sigue el mismo procedimiento, se toma como referencia el valor mencionado en la literatura y se le aplica un factor de variación. A continuación se muestran los resultados de la experimentación.

Tasa de Casamiento	Bondad de Solución
0.60	89.7
0.65	89.6
0.70	86.3
0.75	84.20
0.80	84.2
0.85	84.3
0.90	85.4

Tasa de Mutación	Bondad de Solución
0.00	82.0
0.01	81.72
0.02	83.0
0.03	83.4
0.04	83.2
0.05	83.6
0.06	81.5

Tabla 8.5.- Resultados de calibración tasa de casamiento y mutación.

Como se puede observar en la tabla 8.5 la tasa de casamiento de 0.75 posee un valor mejor que las demás en cuanto a bondad de la solución. De la misma manera sucede con la tasa de mutación con un valor de 0.01 lo cual es bastante bajo y conveniente ya que a menor factor de mutación menos probabilidad de que se aplique el operador y existan más aberraciones. Además, el valor de ambas tasas se encuentra dentro del rango que recomiendan en la literatura.

1.4 Resultados

Estos resultados tienen el objetivo de mostrar la funcionalidad de los algoritmos implementados, comprobando que arrojen resultados que sean reales. Esto quiere decir, que cumplan con las restricciones fuertes, las cuales no pueden ser incumplidas, expuesta en los capitulos anteriores.

Ademas, se realizará una comparación con el horario academico que actualmente esta utilizando la PUCP para el semetres 2014-II. Esta comparación basicamente consistirá en evaluar dos criterios:

- La distribución de los cursos : En el cual se evalua que el mismo curso no sea asignado dos veces en el mismo día o que no se encuentren en dias consecutivos. Lo ideal seria que se encuentren distribuidos a lo largo de la semana.

- La distribucion de los profesores : En la cual se evaluará la cantidad de horas asignadas por dia de un profesor, asi como tambien las horas de dictado semanal.

Para realizar la ejecución del algoritmo se tomará como input las clases del primer ciclo de la facultad de Ciencias e Ingenieria (quinto ciclo) en la especialidad de Informatica. Acontinuacion se detalla la información mencionada:

	Lunes	Martes	Miercoles	Jueves	Viernes
07:00 - 08:00		IEE229-530			INF220-530/531
08:00 - 09:00	INF263-530/531	IEE229-530			INF220-530/531
09:00 - 10:00	INF263-530/531	IEE229-530		INF220-530/531	
10:00 - 11:00	INF263-530/531			INF220-530/531	
11:00 - 12:00					
12:00 - 13:00	INF246-530				
13:00 - 14:00	INF246-530				
14:00 - 15:00	INF246-530				
15:00 - 16:00					
16:00 - 17:00					
17:00 - 18:00					
18:00 - 19:00		IND251-0531			IND251-0532
19:00 - 20:00	IND251-0535/0536	IND251-0531	IND251-0530	IND251-0533/0534	IND251-0532
20:00 - 21:00	IND251-0535/0536	IND251-0531	IND251-0530	IND251-0533/0534	IND251-0532
21:00 - 22:00	IND251-0535/0536		IND251-0530	IND251-0533/0534	

Figura 8.1.- Horario de clases quinto ciclo Especialidad de Informática. Generado por el algoritmo genético.

	Lunes	Martes	Miercoles	Jueves	Viernes
07:00-08:00		IND251-0535/0536			
08:00-09:00		IND251-0534/0535/0536	INF220-581/582		INF220-581/582
09:00-10:00		IND251-0534/0535/0536	INF220-581/582		INF220-581/582
10:00-11:00		IND251-0534	INF246-581		IEE229-0581
11:00-12:00			INF246-581		IEE229-0581
12:00-13:00			INF246-581		IEE229-0581
13:00-14:00					
14:00-15:00					
15:00-16:00			INF263-581/582		
16:00-17:00			INF263-581/582		
17:00-18:00			INF263-581/582		
18:00-19:00					IND251- 0537
19:00-20:00		IND251- 0531		IND251- 0532/0533	IND251- 0537
20:00-21:00		IND251- 0531		IND251- 0532/0533	IND251- 0537
21:00-22:00		IND251- 0531		IND251- 0532/0533	

Figura 8.2.- Horario de clases quinto ciclo Especialidad de Informática. Actualmente utilizado por la facultad de Ciencia e Ingeniería

La Figura 8.1 muestra la solución arrojada por el algoritmo. La cual demuestra que se están cumpliendo todas las restricciones fuertes del problema (no existen cruces entre horarios del mismo ciclo). Además, se puede observar que se ha tratado de diversificar la asignación a lo largo de los días de la semana. A comparación con el horario

académico que existe actualmente en la PUCP, se puede decir que la solución generada por el algoritmo es bastante buena, ya que distribuye los cursos de una manera adecuada y no genera carga de clases en un solo día. Por ejemplo: Los días lunes en el horario actual no ha sido asignado ningún horario de clase, en consecuencia esto provoca que otros días de la semana como los miércoles o viernes estén más cargado de horas de clase consecutivas, a diferencia de lo generado que en el día lunes muestra clases asignadas y estas no se encuentra de manera conjunta.

Cabe resaltar que el resultado del horario generado respeta la restricción de turnos, la cual menciona que todos los cursos de ciclos menores que pertenecen a la especialidad se han asignados en turno más temprano. Se tiene que tener en cuenta que el curso IND251 no cumple con esta restricción debido a que no pertenece a la especialidad de informática y este ha sido asignado con más frecuencias en el turno de noche debido a que este curso es dictado por profesores tipo TPA (parciales por asignatura), lo cuales poseen mayor tiempo de disponibilidad por el turno de la noche.

A continuación se muestra la distribución de horas asignadas para los profesores considerando solo los cursos dictados en el primer ciclo de la especialidad de informática. Se puede observar que no existe sobrecarga de horas de dictado, por el contrario, se encuentran balanceadas a lo largo de todos los profesores disponibles para dictar la asignatura.

Curso	Horario	Profesor	Horas Asignadas
IEE229	530	Ramirez	3
IND251	530	Leidinger	3
	531/534	Luna	6
	532	Vargas	3
	533	Nakama	3
	535/536	Vega	6
INF263	530	Alva	3
	531	Melgar	3
INF246	530	Aguilera	3
INF220	530	Klevniko	4
	531	Bello	4

Tabla 8.6.- Distribución de horas de dictado semanales de los profesores (solo para quinto ciclo curricular).

Se ha realizado una comparación de las asignaciones semanales de los profesores del horario actual y el horario generado. Ambos muestran una distribución adecuada comparando solo el quinto ciclo, lo adecuado sería poder estimar las horas de dictado de los profesores total cuando se hayan corrido el algoritmo para toda la facultad de Ciencia e Ingeniería, ya que un profesores puede dictar cursos en más de una especialidad.

CAPÍTULO 9

1 Resultado Esperado 6

Prototipo funcional de los módulos del software.

1.1 Introducción

En este capítulo se describirá el modelo de datos implementado en el prototipo del software, así como también las descripciones de los distintos módulos del software.

1.2 Modelo Entidad Relación

La implementación de una base de datos para el presente proyecto se vio como una necesidad ya que el software actualmente persiste toda información de los inputs y outputs del algoritmo. Además, esta información puede ser actualizada mediante las vistas graficas que presenta el software implementado.

La base de datos total se puede apreciar en el anexo 2, para visualizar el esquema de datos se ha decidido dividir el modelo en vistas que contiene el software:

1.2.1 Vista de Mantenimiento de Profesores

En la Figura 9.1 se puede apreciar cuatro tablas las cuales conforman la vista de mantenimiento de profesores. A continuación se realizará una breve descripción de las tablas que conforman la vista:

La tabla Teacher es un maestro que básicamente contiene información personal de los docentes de la PUCP, así como también la disponibilidad de sus tiempos, la preferencia, tipo de dedicación y la unidad académica a la que pertenece.

La tabla Teacher_has_Course muestra la relación los profesores con los cursos que este alguna vez ha dictado. Además, indica si existe alguna preferencia por parte del profesor hacia el curso para ser dictado.

La tabla Academy_unit y Dedication_type son maestros que indican todas las unidades académicas y los tipos de dedicación que existen, respectivamente.

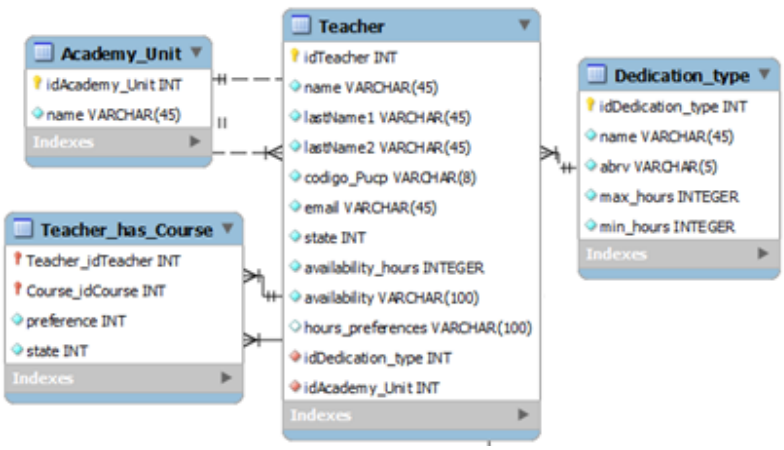


Figura 9.1.- Vista de Profesores del modelo de BD

1.2.2 Vista de Mantenimiento de Salones

En la Figura 9.2 se puede apreciar tres tablas las cuales conforman la vista de mantenimiento de salones. Esta relación permite el registro y modificación de las aulas. A continuación se realizará una breve descripción de las tablas que conforman la vista:

La tabla Room es un maestro que básicamente contiene información de las aulas de la PUCP indicando su capacidad para sesiones de clases, capacidad para sesiones de evaluación, su tipo de salón, pabellón al que pertenece y la disponibilidad que presenta cada salón.

La tabla Room_type y Pabellon son maestros que indican todos los tipos de aulas y los distintos pabellones que existen, respectivamente.

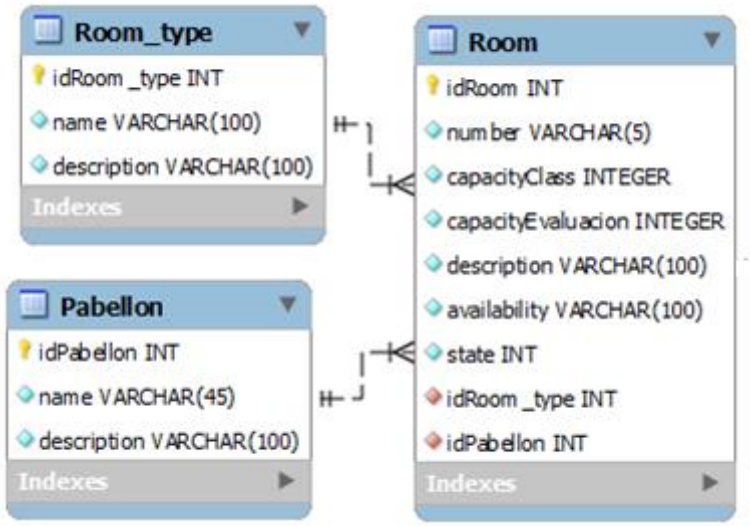


Figura 9.2.- Vista de Salones del modelo de BD

1.2.3 Vista de Mantenimiento de Cursos

En la Figura 9.3 se puede apreciar seis tablas las cuales conforman la vista de mantenimiento de cursos. Esta relación permite el registro y modificación de los cursos. A continuación se realizará una breve descripción de las tablas que conforman la vista:

La tabla Course_code es un maestro que básicamente contiene el código de los cursos indicando su unidad académica y a la especialidad que pertenece. Cabe resaltar que en esta tabla se almacenará la lista de cursos que son aprobados por la sección correspondiente, en el caso del proyecto sección de Ciencia e Ingeniería.

La tabla Course es un maestro donde se almacena la información de cada curso. A diferencia, de la tabla anterior los cursos que estén registrados aquí serán tomados por el algoritmo ya que son los cursos creados en el sistema, en cambio la tabla de Code es la lista cursos aprobada por la unidad académica. Cabe mencionar que esta tabla contiene un atributo llamado Split, el cual indica si ese curso es compartido a nivel curricular por más de una especialidad, esto permitirá decidir a qué ciclo curricular pertenece el curso.

La tabla Course_has_CourseType guarda la relación de tipo de sesiones que posee cada curso. Además, contiene información de la cantidad de vacante por tipo de curso, el número de horarios, horas máximas, horas curriculares, el tipo de aula que necesita (este campo es importante cuando se trata de sesiones tipo laboratorio) y la cantidad de comisiones. Cabe mencionar que los horarios tipo clase no poseen comisiones, esto aplica solamente para sesiones tipo evaluación.

La tabla Course_Type y Speciality son maestros que indican todos los tipos de cursos y las distintas especialidades que existen, respectivamente.

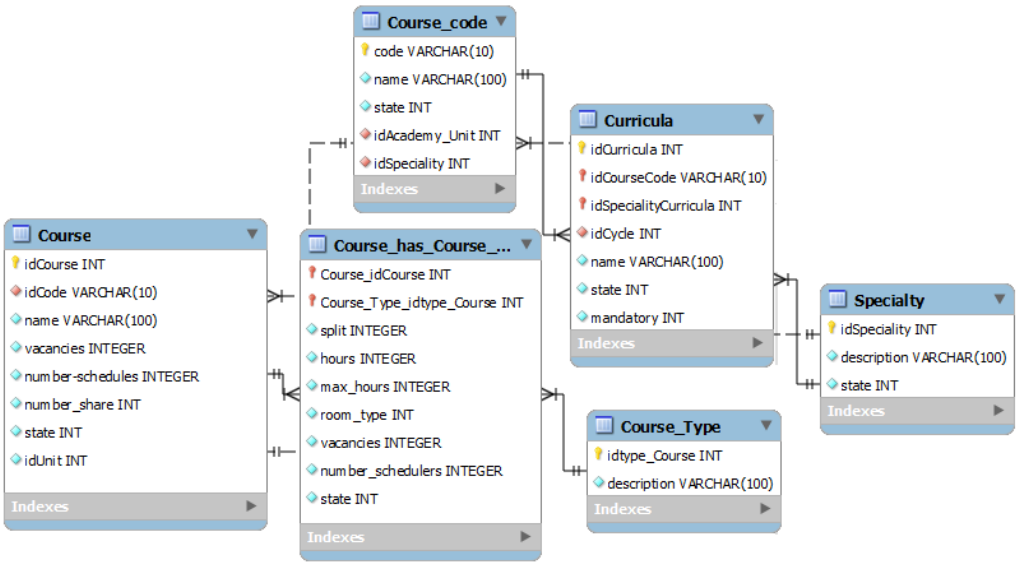


Figura 9.3.- Vista de Cursos del modelo de BD

1.2.4 Vista de Manteamiento de Generación de Horarios

En la Figura 9.4 se puede apreciar siete tablas las cuales conforman la vista de mantenimiento de horarios, esta relación permite el registro de nuevas generaciones de horarios. A continuación se realizará una breve descripción de las tablas que conforman la vista:

La tabla Schedule es donde finalmente se almacenará la respuesta de la ejecución del algoritmo genético. Esta tabla tiene relaciones de diferentes entidades como la de curso, el tipo de curso, los horarios, los salones, los slots y los profesores. Toda esta información es necesaria para armar el horario académico que se ha arrojado como respuesta. Además, esta tabla posee dos campos que ayudan a identificar las distintas ejecuciones del algoritmo, estos campos son idUniqueSchedule y creation_date, los cuales indica el número de horario generado y la fecha de creación de la posible solución. Cabe recalcar que no todos los horarios académicos que se guardan en esta tabla son los que han sido usados para una asignación real, debido a ello se tiene un indicador que ayuda a identificar si este horario ha sido asignado alguna vez y en que semestre.

La tabla Class_Schedule y Slot son maestros que indican todas las diferentes secuencias de horarios por ciclo que existen y las unidades de tiempo (especificando su día y turno), respectivamente.

Las demás tablas que conforman esta vista han sido explicadas con anterioridad en las otras vistas de mantenimientos.

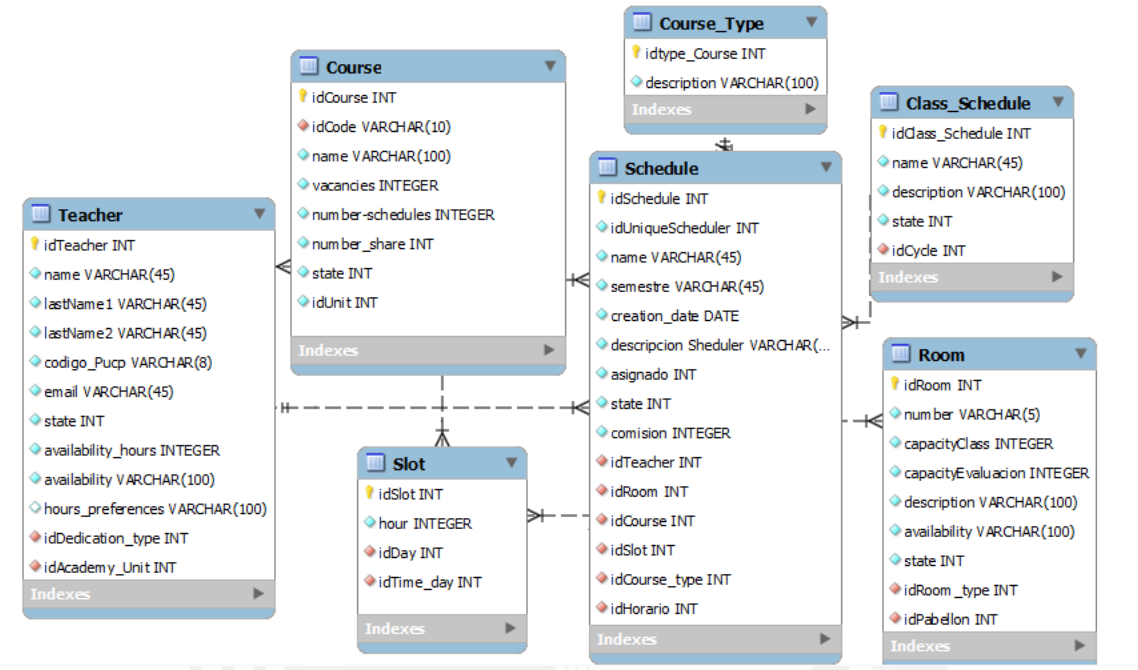
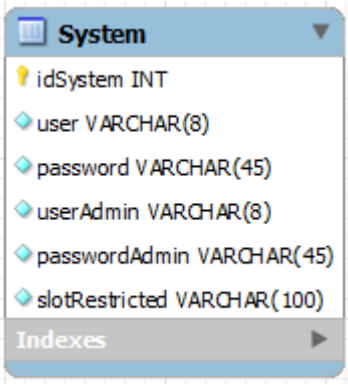


Figura 9.4.- Vista de Horario del modelo de BD

1.2.5 Vista de Parámetros

Esta vista está conformada por una sola tabla la cual contiene información básica del sistema como: el usuario y contraseña de sistemas, las especificaciones del administrador y por último contiene un cadena de char que representa los slots de restricciones, el cual me indica en que tiempo de la semana está prohibido realizar unas asignación.



dFigura 9.5.- Vista de Parámetros

1.3 Diseño de la Interfaz Gráfica

A continuación se mostrará el diseño de las interfaces utilizadas en la implementación del prototipo del sistema que permitirá generar horarios académicos.

1.3.1 Módulo de Mantenimiento de Profesores

La vista principal de este módulo es la pantalla de búsqueda. Inicialmente se muestra el listado de todos los profesores en la PUCP, indicando su código de la PUCP, los nombres, apellidos y la unidad Académica a la que pertenece. Además, el sistema permite realizar búsquedas especializadas ya sea por nombre, apellido, email, unidad académica, tipo de dedicación y estado (Activos o Inactivos). Como se puede apreciar en la Figura 9.6.

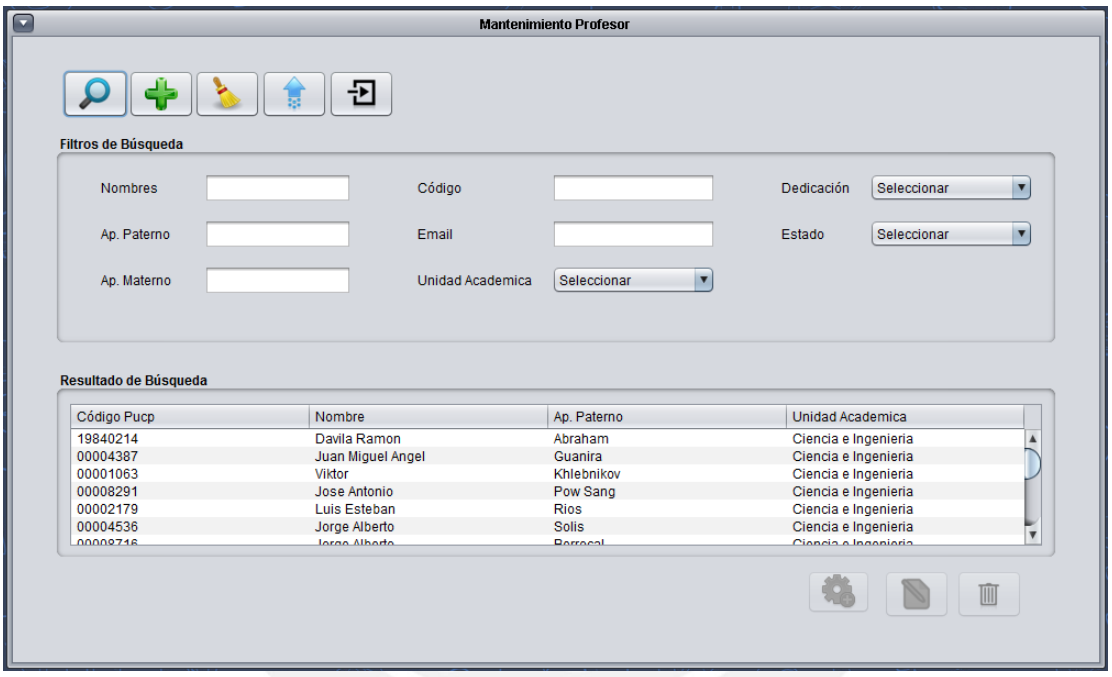


Figura 9.6.- Mantenimiento de Profesores - Busqueda

A partir de esta vista se puede navegar a las otras actividades necesarias para realizar el mantenimiento del maestro de profesores. Si se desea agregar un nuevo profesor se hace click en el botón de +. Si se desea editar la información de un profesor que ya existe en los registros se selecciona la fila y a continuación se selecciona el botón que contiene la imagen de un lápiz para poder editar. Por último, si se desea eliminar al profesor registrado se selecciona la fila requerida y luego el botón que contiene un tacho como imagen.

En la figura 9.7 se visualiza la pantalla para crear un nuevo profesor. La cual cuenta con tres secciones Datos personales, Datos laborales y Clases. En la primera sección se registrar la información personal del profesor (nombre, apellido, código, etc.). En la segunda sección se indica la unidad académica a la que pertenece, el tipo de dedicación, la disponibilidad de horas y los cursos disponibles a dictar. Por último, en la última sección se considera las preferencias de dictado de clases tanto de las horas como de los cursos.

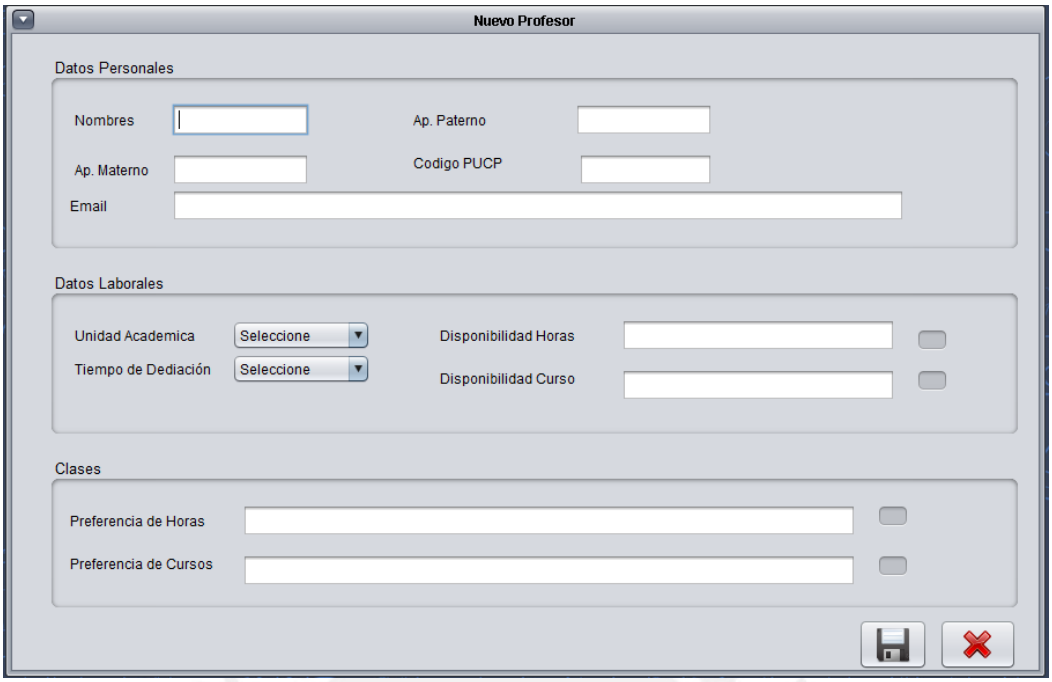


Figura 9.7.- Mantenimiento de Profesores – Nuevo Profesor.

La Figura 9.8 muestra la pantalla de detalle de profesor. Esta pantalla se encuentra dividida en tres pestañas: información personal, Horas disponibles y Cursos Disponibles. En estas dos últimas se observan las disponibilidades y preferencias de los profesores tanto en tiempo como en cursos.

Figura 9.8.- Mantenimiento de Profesores – Detalle de Profesor.

1.3.2 Módulo de Mantenimiento de Salones

La vista principal de este módulo es la pantalla de búsqueda. Inicialmente se muestra el listado de todos los salones en la PUCP, indicando su número de aula, el pabellón al que pertenece, descripción de las características del aula y la capacidad física. Además, el sistema permite realizar búsquedas especializadas ya sea por pabellón, número de aula, descripción, tipo de aula, rangos de capacidad y estado (Activos o Inactivos). Como se puede apreciar en la Figura 9.9.

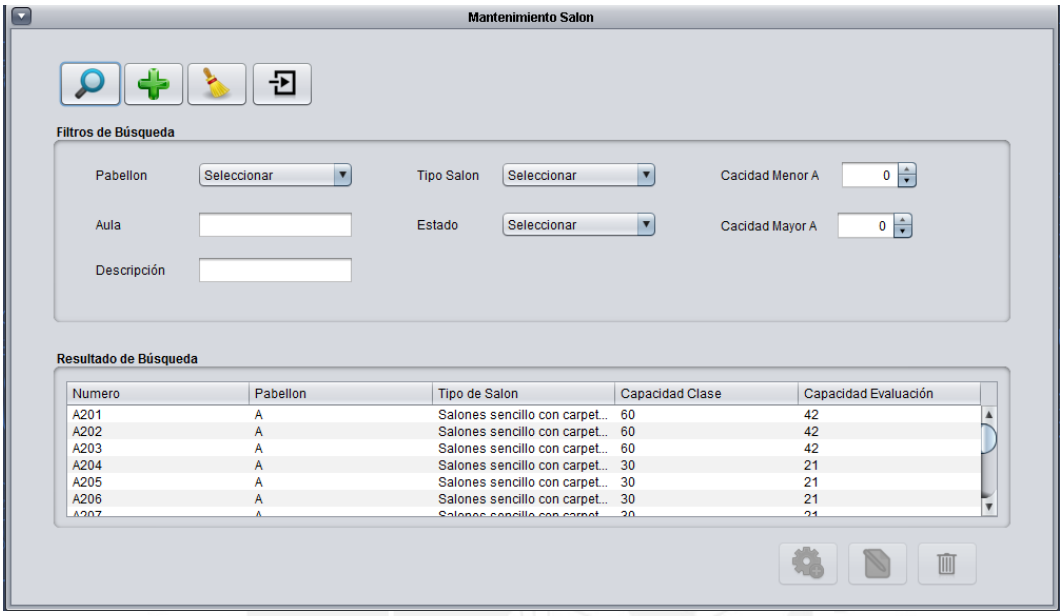


Figura 9.9.- Mantenimiento de Aulas – Búsqueda.

En la Figura 9.10 se muestra la pantalla del registro de un nuevo salón, para ello es necesario indicarle el número de aula, la descripción, su capacidad, el tipo, la disponibilidad y el pabellón al que pertenece.

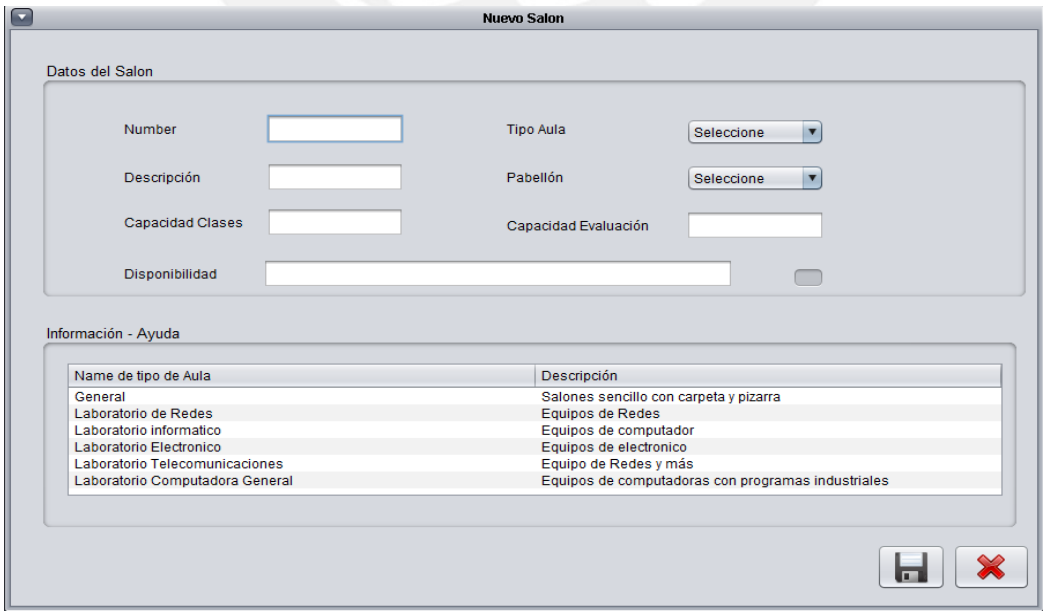


Figura 9.10.- Mantenimiento de Aulas – Nueva Aula.

Por último, se tiene la pantalla del detalle de las aulas registradas en el sistema. En esta se visualiza la información básica del aula (número, descripción, capacidad, pabellón, etc.) y del tipo de salón que pertenece el aula. Como se aprecia en la Figura 9.11.

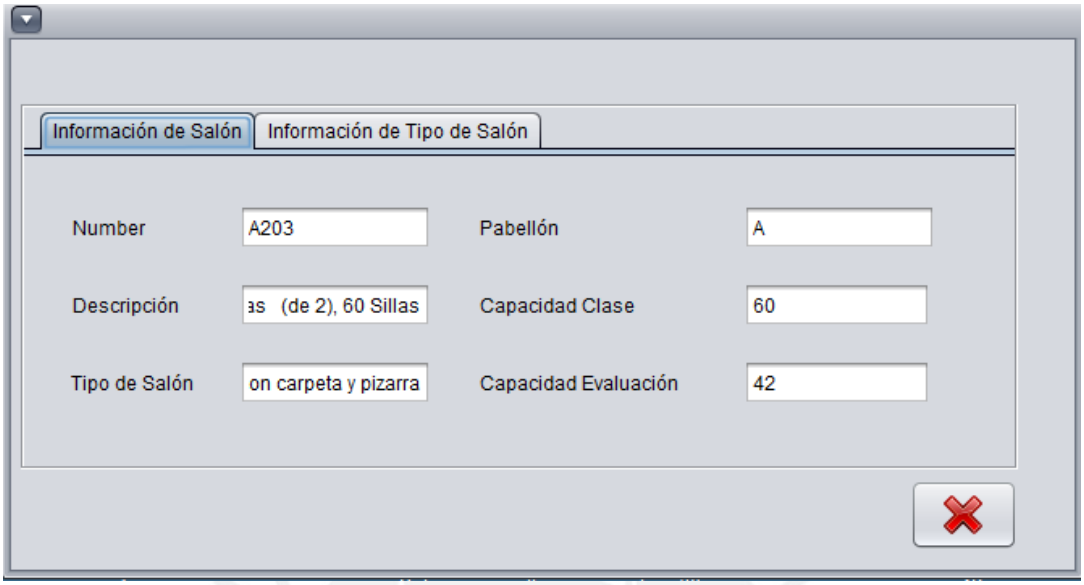


Figura 9.11.- Mantenimiento de Aulas – Detalle de Aula.

1.3.3 Módulo de Mantenimiento de Cursos

La vista principal de este módulo es la pantalla de búsqueda. Inicialmente se muestra el listado de todos los cursos dictados en la PUCP, indicando la clave del curso, su descripción y la unidad académica que pertenece. Además, el sistema permite realizar búsquedas especializadas ya sea por clave de curso, nombre, unidad académica, especialidad y estado (Activos o Inactivos). Como se puede apreciar en la Figura 9.12.

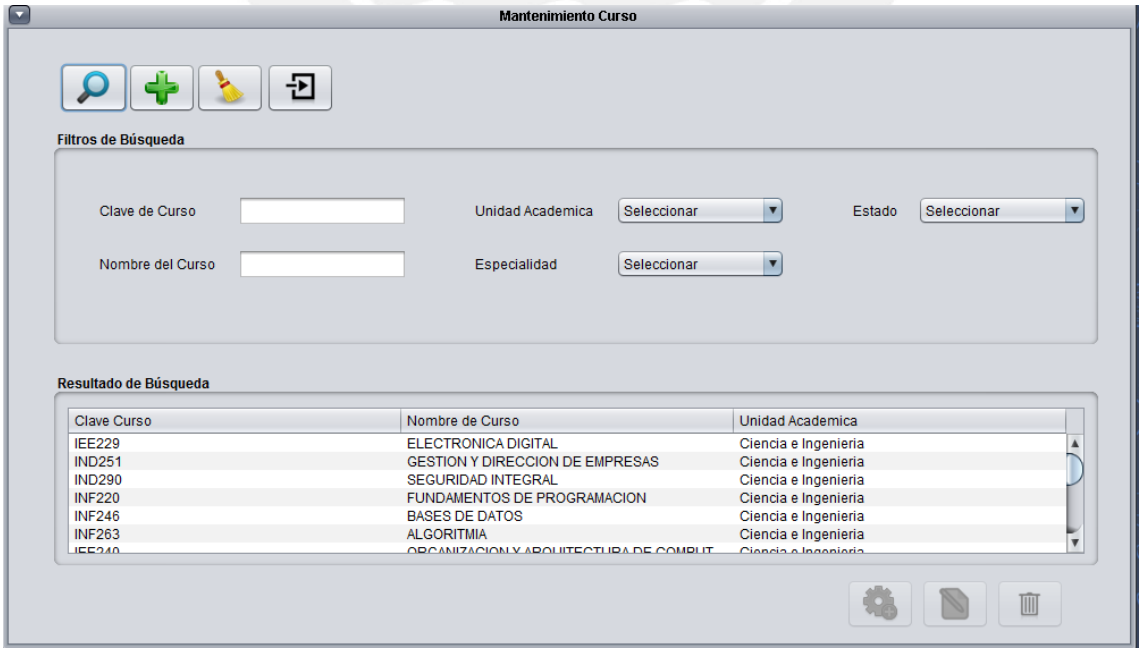


Figura 9.12.- Mantenimiento de Cursos – Búsqueda

En la Figura 9.13 se muestra la pantalla del registro de un nuevo curso. La cual consiste básicamente en dos secciones: La primera contiene información básica del curso como especialidad, clave y nombre. Por otro lado, la segunda muestra la composición de cada curso, en la cual se indica, que tipo de sesiones posee el curso (clase-práctica-laboratorio) e indica información adicional de cada relación como la cantidad de horas semanales, cantidad de horas máximas, número de horarios, cantidad de vacantes, número de comisiones y el tipo de salón necesario para su evaluación.

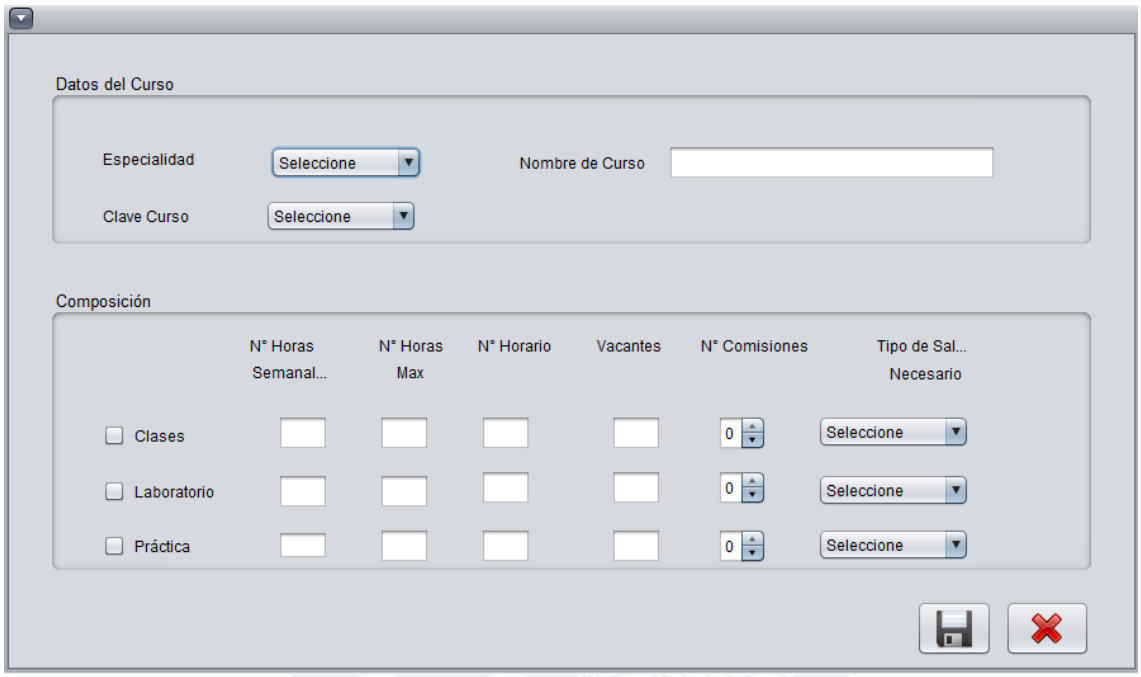


Figura 9.13.- Mantenimiento de Cursos – Nuevo Curso.

Por último se tiene la pantalla del detalle de los cursos registrados en el sistema. En esta se visualiza la información básica del curso, la composición de sus sesiones y el plan curricular del curso (indicando la especialidad y el ciclo al que pertenece). Como se observa en la Figura 9.14.

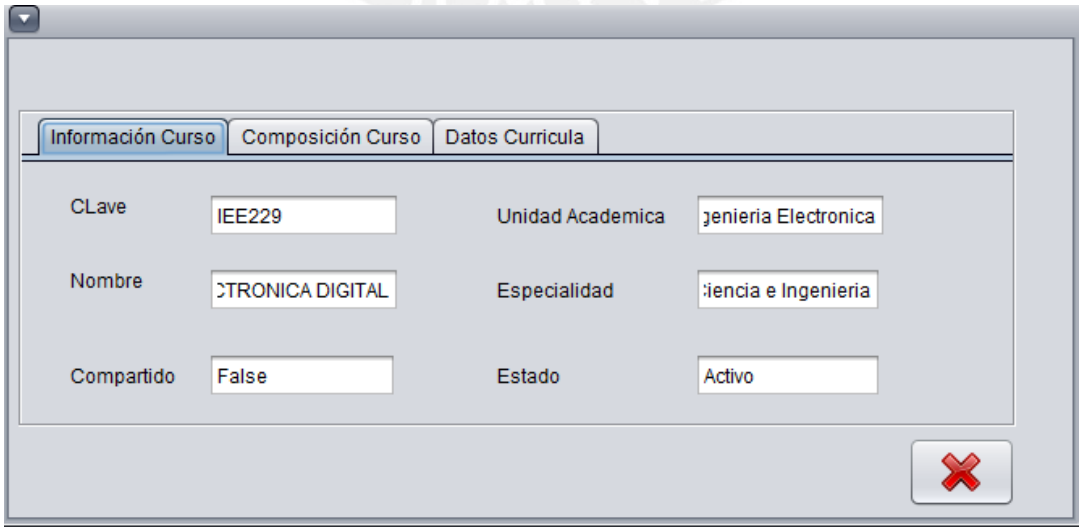


Figura 9.14.- Mantenimiento de Cursos – Detalle de Curso.

1.3.4 Módulo de Mantenimiento de Horarios

La vista principal de este módulo es la pantalla de búsqueda. Inicialmente se muestra el listado de todos los horarios académicos retornados por el algoritmo, indicando el identificador de grupo de horario, descripción, fecha de creación, asignación y estado (Activos o Inactivos). Como se puede apreciar en la Figura 9.15.

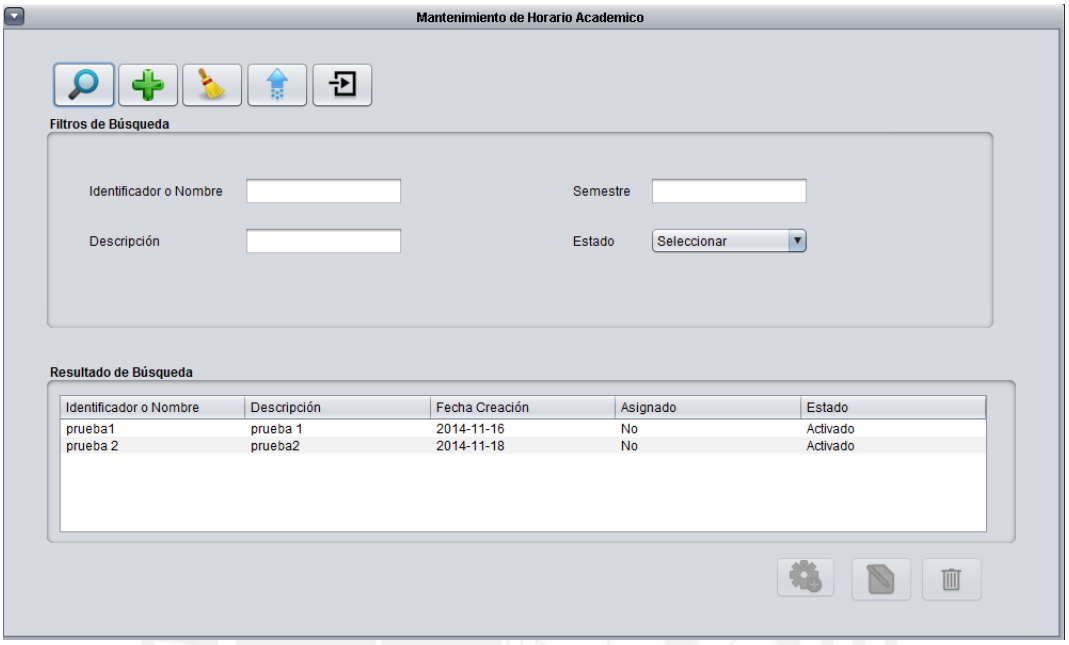


Figura 9.15.- Mantenimiento de Horario – Búsqueda.

En la Figura 9.16 se muestra la pantalla del registro de un nuevo horario. La cual permitirá elegir que especialidades se desean asignar y los valores para los parámetros de los algoritmos. Si no se desea modificar ningún valor, el algoritmo se ejecutará con los parámetros determinados en los capítulos anteriores.

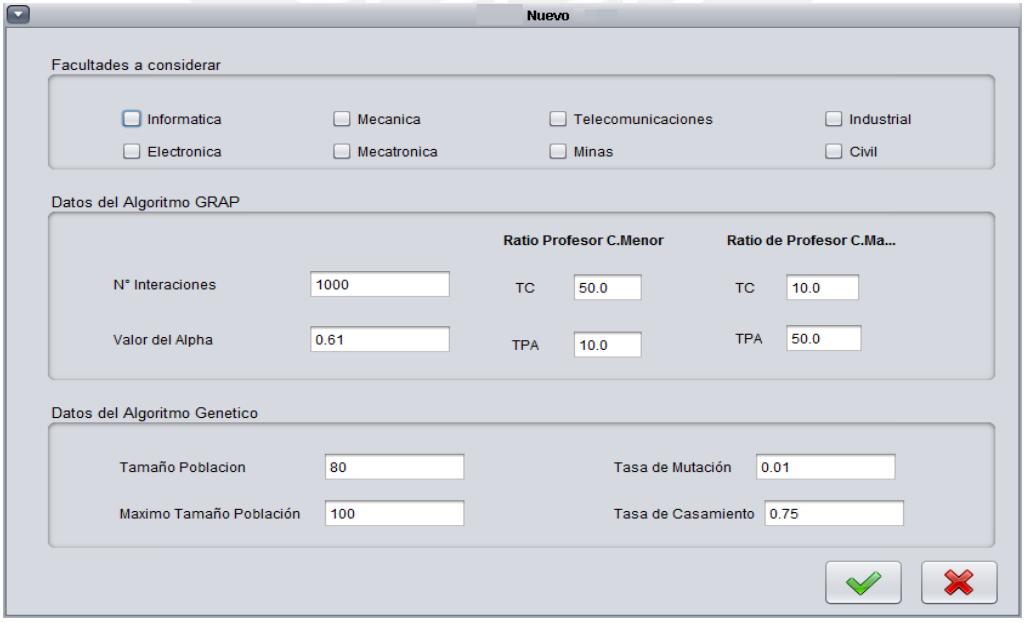


Figura 9.16.- Mantenimiento de Horario – Nuevo Horario.

Finalmente, se tiene la pantalla para ver los resultados del algoritmo. Esta permite visualizar las diferentes ejecuciones registradas en el sistema. En la pantalla de la Figura 9.17 se observa en la parte superior información básica para identificar a las ejecuciones. La cual contiene el nombre del horario generado, el semestre al cual se ha aplicado la ejecución y una descripción.

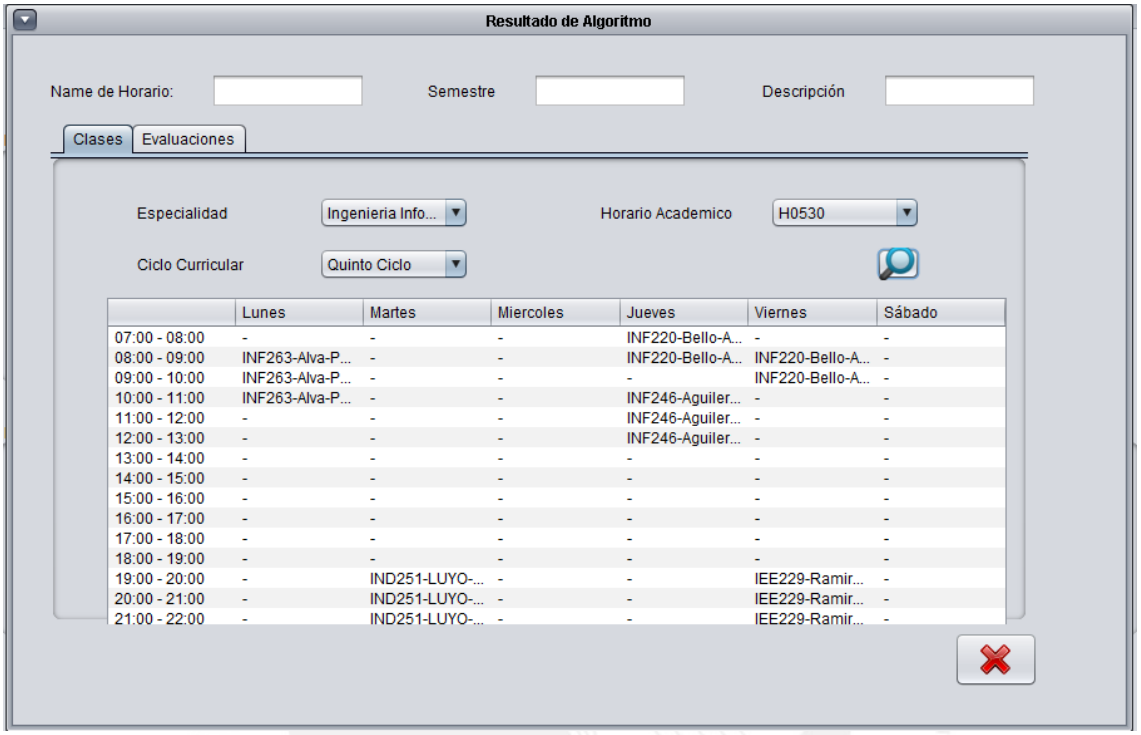


Figura 9.17.- Mantenimiento de Horario – Visualización de horario

Para visualizar la estructura del horario es necesario que se seleccione la especialidad, el ciclo en el cual deseas ver la asignación y el horario al que pertenece la asignación. Ya que esta vista no permitirá ver más de una asignación por espacio de tiempo. Además, se debe tener en cuenta que los resultados se encuentran divididos en dos pestañas, la de asignación de clases y la de evaluaciones. Esto es debido a que los laboratorios y prácticas tienen una probabilidad alta de que contengan asignaciones en el mismo espacio de tiempo, lo cual estaría correcto ya que pertenecen a una secuencia distinta, pero según la estructura definida para la visualización no se podrían mostrar juntos. Es por ello que la pestaña de Evaluaciones contiene un combo para seleccionar que tipo de evaluación se desea mostrar.

CAPÍTULO 10

1 Conclusiones

A partir de la investigación realizada en el presente proyecto de fin de carrera se concluye que el problema de asignación de horarios académicos es un trabajo complejo, debido a la gran cantidad de restricciones y consideraciones que este presenta. Para este proyecto se ha tratado de abarcar la mayoría de condiciones que existen actualmente en las instituciones académicas de nivel superior, pero no son todas. Debido a que observó en el proceso de levantamiento de información que existen algunas particularidades, lo cual convierte en una tarea complicada la construcción de una solución totalmente automática. Por lo tanto, se ha concluido que estas particularidades deben ser tratadas como una excepción y ser aplicadas una vez que ya se tiene una solución inicial generada como base, lo cual se logrará con la ayuda del sistema.

Con respecto al planteamiento de la solución, la construcción del cromosoma no ha sido una tarea fácil ya que existen diversas relaciones que se tienen que tomar en cuenta para armar complemente un horario y todas estas deben estar expresadas en el cromosoma. Inicialmente se había pensado en implementar un cromosoma binario, pero debido a la gran cantidad de relaciones la manipulación de este hubiera sido engorrosa ya que sería una matriz de más de 4 dimensiones. Es por ello, que se decidió optar por un cromosoma que posea una codificación directa y de fácil acceso para manipular. Esto quiere decir que una vez encontrado una unidad de asignación, en el eje x y eje y de la matriz, se puede decodificar el cromosoma mediante los códigos que presenta la tupla. En cuanto a al uso de los operadores genéticos se ha observado que realizan una mayor exploración al espacio de búsqueda, lo cual se demuestra en la mejora del valor de la solución generada por el algoritmo Grasp.

Por otro lado, en cuanto a la implementación del algoritmo genético, se ha conseguido obtener soluciones considerablemente aceptables dentro del rango de restricciones que se han establecido, debido a ello, se puede decir que el objetivo general se ha cumplido a cabalidad. En este punto es importante recalcar que ha sido muy importante la implementación de la solución inicial por el algoritmo Grasp, ya que esta arroja una solución inicial con un valor de bondad lo bastante bueno para luego ser mejorada por el genético, si se hubiera contado al inicio con una solución aleatoria no se hubiera llegado a la calidad de solución que actualmente se tiene, por lo tanto se concluye que una buena población inicial es fundamental para mejorar la calidad del algoritmo genético. Además, cabe resaltar que la construcción del algoritmo se ha enfocado más encontrar una adecuada solución al problema que la eficiencia con respecto a los tiempos de ejecución.

Para finalizar se concluye que la automatización de esta tarea ahorraría mucho tiempo en la ejecución manual, ya que inicialmente te arroja un resultado lo bastante bueno que cumple con todas las restricciones expuestas por la institución. Además, con la ayuda de la implementación del software las particularidades que se presentan podrán ser gestionadas.

2 Recomendaciones

A lo largo de la ejecución del presente proyecto se ha identificado aspectos claves que se considera importante mencionarlos ya que pueden ser de utilidad para mejorar la solución presentada o problemas similares en trabajos futuros. Con respecto a la construcción del algoritmo considero que esta implementación ha sido más enfocada a encontrar una solución al problema que a construir un algoritmo que posea un tiempo de ejecución mínimo, es por ello que se recomienda mejorar la ejecución del algoritmo para poder contar con tiempos menores a los que actualmente se tiene, esto se puede lograr cambiando algunas estructuras básicas de objetos en otras estructuras más simples.

Por otro lado, con respecto a la construcción del software, se ha implementado un prototipo bastante completo que permite ingresar y visualizar los inputs y outputs del algoritmo. Debido a ello, se recomienda que se prosiga con la construcción del prototipo para agregarle una cantidad mayor de funcionalidades, las cuales tiene como objetivo gestionar de una mejor manera la asignación de horarios, otorgando al usuario mayor facilidad de uso. De esta manera la integración del sistema - algoritmo obtendrá un valor mayor.

En el presente proyecto se ha considerado todas las características en su totalidad de las asignaciones de los horarios tipo clase, pero los horarios tipo evaluación no han sido abarcados en su totalidad. Como se mencionó en los capítulos anteriores los horarios tipo evaluación tienen ciertas características que no han sido contempladas en la solución presentada, debido al alcance de este. Por lo tanto, se recomienda que para trabajos futuros se pueda considerar la asignación de los docentes y jefes de prácticas a los horarios tipo evaluación, para ello es necesario tener en cuenta que en un horario de evaluación se pueden asignar a más de un docente o jefe de práctica, los cuales no necesariamente poseen las características mencionadas en el capítulo 3 de este documento.

Otro aspecto que se podría adicionar como trabajo futuro es la asignación de los eventos tipo examen de cada curso. Esta agregación daría un valor más completo a la solución presentada y no abarcaría muchos cambios al algoritmo ya planteado, ya que pueden ser tratados como horarios tipo evaluación.

Finalmente, como una segunda etapa de este proyecto se recomienda agregar a la implementación del algoritmo todas las secuencias que tiene actualmente la PUCP, esto implicaría considerar un mayor número de restricciones con lo cual la solución se volvería más compleja pero a su vez se convertirá en una solución más acorde a las necesidades de la universidad.

Referencias bibliográficas

ALSUWAIYEL, M.
1998 "Algorithms: Design Techniques and Analysis". Lecture Notes Series on Computing. Londres, Vol 7.

ASUMI, Hishammuddin, BURKE, Edmund, GARIBALDI, J y MCCOLLUM.B
2004 "Fuzzy multiple ordering criteria for examination timetabling". The 5th International Conference on the Practice and Theory of Automated Timetabling. Springer Lecture Notes in Computer Science, vol. 3616.

AZIMI, Naji
2004 "Comparison of metaheuristic algorithms for examination timetabling problem". Applied Mathematics and Computation, Vol.16.

BLUM, Christian y ROLI, Andrea
2003 "Metaheuristics in Combinatorial Optimization: Overview and Conceptual Comparison". ACM Computing Surveys. New York. Vol 35 Issue 3.

BURKE, Edmund y TRICK, Michael
2004 "Practice and Theory of Automated Timetabling V". 5th International Conference, PATAT 2004, Pittsburgh, PA, USA

BURKE, Edmund, MCCoLLUM, Barry y ABDULLAH,Salwani
2007 "A hybrid Evolutionary Approach to the university Course Timetabling Problem". Evolutionary Computation. CEC 2007. IEEE Congress on. Singapore.

BURKE, Edmund, PETROVIC, Sanja y RONG, Qu
2006 "Case-based heuristic selection for timetabling problems". Journal of Scheduling. UK. Volume 9, Issue 2

BURKE, Edmund y NEWALL, Jim
2004 "Solving examination timetabling problems through adaptation of heuristic orderings". Annals of Operational Research.Netherlands, vol 129.

COLEY, David
1999 "An introduction to genetic algorithms for scientists and engineers". Singapore: World Scientific publishing Co.

DI GASPERO, Luca
2002 "Recolour, shake and kick: A recipe for the examination timetabling problem". Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling. KaHo St.-Lieven, Gent, Belgium.

DI GASPERO, Luca y SCHAERF, Andrea
2001 "Tabu search techniques for examination timetabling". Practice and Theory of Automated Timetabling. the 3rd International Conference. Springer Lecture Notes in Computer Science, vol. 2079.

DIMOPOULOU, M y MILIOTIS, P
2001 "Implementation of a university course and examination timetabling system". European Journal of Operational Research. Vol. 130.

DUONG, T y LAM, K
2004 "Combining constraint programming and simulated annealing on university exam timetabling". Proceedings of the 2nd International Conference in Computer Sciences. Hanoi, Vietnam.

GALLART, Joseph, ALVA, Fernando, ALAMA, Anthony, BEJERANO, Gissella
2009. "Generación Inteligente de Horarios empleando heurísticas GRASP con Búsqueda Tabú para la Pontifica Universidad Católica del Perú" . VI Congreso Español sobre Metaheurísticas, Algoritmos Evolutivos y Bioinspirados.

GOFFE, William, FERRIER, Gary y ROGERS, John
1994 "Global optimization of statistical functions with simulated annealing". North-Holland. Journal of Econometrics.Vol 60.

JAT, S y YANG, Shengxiang
2008 "A Memetic Algorithm for the University Course Timetabling Problem". IEEE computer society. UK. vol 1.

MERLOT, Liam, BOLAND, Natasha, HUGHES, Barry y STUCKEY, Peter
2003 "A hybrid algorithm for the examination timetabling problem". Practice and Theory of Automated Timetabling: the 4th International Conference. Springer Lecture Notes in Computer Science, vol. 2740

NGUYEN, k, TRAN, N y TRIEU, K
2010 "Automating a Real-Word University Timetabling Problem with tabu search algorithm". Computing and Communication Technologies Innovation, and Vision for the Future (RIVF), 2010 IEEE RIVF International Conference on. Hanoi

LIM, Andrew, ANG, Juay-Chin, HO, Wee-Kit y OON, Wee-Chong
2002 "UTTSExam: A Campus-Wide University Exam-Timetabling System". American Association for Artificial Intelligence. Singapore.

OSMAN, Ibrahim y LAPORTE, Gilbert
1996 "Metaheuristics: A bibliography". Annals of Operations Research. Vol 6. Isuee 5.

PAPADIMITRIOU, Christos y STEIGLITZ, Kenneth.
1982 Combinatorial optimization, Algorithms and complexity. Prentice-Hall, USA.

PAQUETE, Luis y STUTZLE, Thomas
2002 "Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem". Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling. Belgica.

PINEDO, Michael
2012 Scheduling: theory, algorithms and systems. Fourth edicion. Prentice-Hall,USA.

QU, Rong, BURKE, Edmund, MCCOLLUM, Barry, LIAM Merlot y LEE, Sau
2006 "A survey of search methodologies and automated Approaches for examination timetabling". Computer Science Technical Report NOTTCS-TR-2006-4.UK, vol 12, Isuue 1

REIS, Luis y OLIVEIRA, Eugenio
2001 A Language for Specifying Complete Timetabling Problems. Third International Conference, PATAT 2000 Konstanz, Germany. Vol 2079.

TAVARES, R y GODINHO, M
2013 "Literature review regarding Ant Colony Optimization applied to scheduling problems: Guidelines for implementation and directions for future research". Engineering Applications of Artificial Intelligence. Bazil. Volume 26, Issue 1

TORRES, Fernando
2009 "Integración del PMBOK al RUP para proyectos de Desarrollo de Software". UNMSM. Perú.

VELEZ, M y MONTOYA, J
2007 "Metaheurísticos: una alternativa para la solución de problemas combinatorios en administración de operaciones". Revista EIA. Colombia, ISSN 1794-1237 Numero 8.

WONG, T, COTE, P y GELY, P
2002 "Final exam timetabling: a practical approach". IEEE Canadian Conference on Electrical and Computer Engineering. Vol. 2.

PMBOK
2004 "Fundamentos de la dirección de proyectos". Norma Nacional Americana ANSI/PMI 99-001-2004 .Tercera ediccion.

Tesis

BEJARANO, Gissella
2009 Planificación de horarios del personal de cirugía de un hospital del estado aplicando algoritmos genéticos (Timetabling problem). Pregrado. Lima: Universidad Católica del Perú, Ingeniería informática.

CORTEZ, A, ROSALES, G, QUIROZ, R y HUERTA, H
2010 Sistema de apoyo a la generación de horarios basado en algoritmos genéticos. Lima: Universidad Nacional de San Marco y Universidad Ricardo Palma, Ingeniería informática.

Páginas WEBS

Universidad Católica del Perú
2013 "Estatuto de los docentes de la universidad". Consulta 01 de setiembre del 2013.
<http://www.pucp.edu.pe/documento/pucp/estatuto.pdf>

Universidad Católica del Perú
2013 "Calendario académico de la Facultad de Ciencias e Ingeniería". Consulta 01 de setiembre del 2013
<http://facultad.pucp.edu.pe/ingenieria/index.php?option=com_content&task=view&id=61&Itemid=63>

NeatBeans
2013 "".Consulta 07 de noviembre 2013.
< https://netbeans.org/features/index.html>

Bizagi
2013 "Documentación del producto". Consulta 07 de noviembre 2013.
<http://www.bizagi.com/index.php/es/productos/bizagi-process-modeler>

Bizagi
2013 "BPMN (Business Process Model and Notation)".Consulta 07 de noviembre 2013.
<http://wiki.bizagi.com/es/index.php?title=BPMN#BPMN_.28Business_Process_Model_and_Notation.29>

IBM
2013 "RUP". Consulta 07 de noviembre 2013.
<http://www-01.ibm.com/software/rational/rup/>