Universidad de Valparaíso
Facultad de Ingeniería
Escuela de Ingeniería Industrial

"Desarrollo de un Modelo para la creación de la oferta académica y asignación de salas en la Escuela de Ingeniería Industrial de la Universidad de Valparaíso".

por

Yerko Daniel Rojas Rojas

Trabajo de Titulo para optar al Grado de Licenciado en Ciencias de la Ingeniería y título de Ingeniero Civil Industrial.

Prof. Guía Luis Seccatore Gómez

Junio, 2016

Dedicatoria.

A mi Madre por su amor, comprensión y enseñanzas. Gracias por haber creído en mí y por estar siempre a mi lado apoyándome para salir adelante.

A mi hermana Andrea y tío José por su valioso apoyo desde el inicio de mis estudios.

A mis amigos y familiares que tuvieron siempre una palabra de apoyo.

Agradecimientos.

A mi madre, hermana y mi tío José, quienes han sido parte importante en esta etapa gracias a su comprensión, motivación y su apoyo incondicional.

A Benito y Raquel, por sus enseñanzas y apoyo entregado durante estos años.

Al profesor Luis Seccatore Gomez, por la confianza puesta en mí, la dedicación y el apoyo en el desarrollo de este Trabajo de Titulo.

A los profesores en general que compartieron sus conocimientos durante mi paso por la Universidad.

Índice

Lista de Abreviaturas

DIVACAD = División Académica de la Universidad de Valparaíso.

UCTP = *University Course Timetabling Problem.*

TTP = *Timetabling Problem* – Problema de Horario.

SP = *Scheduling Problem* – Problema de Programación.

GCP = *Graph Colouring Problem* - Problema de Coloreo de Grafo

SA = *Simulated Annealing* – Recocido Simulado.

ACO = *Ant Colony Optimization* – Optimización por Colonia de Hormigas.

TS = *Tabu Search* – Búsqueda Tabú.

GA = *Genetic Algorithm* – Algoritmo Genético.

DM = *Descent Methods* - Método de la Pendiente.

NN = *Neural Nets* - Redes Neuronales.

ES = *Expert Systems* - Sistemas Expertos.

Lista de Figuras

Lista de Tablas

Introducción

A lo largo del tiempo las instituciones educativas han presentado problemas relacionados con la calidad de los servicios que ofrecen a docentes y estudiantes, generándose dos puntos clave en este contexto, la programación de las actividades académicas correspondiente al plan de estudios vigente y la planificación de la capacidad, es decir, el dimensionamiento de los recursos disponibles para poder operar en el tiempo.

Este tipo de problemas de optimización son difíciles de resolver de forma exacta debido a lo complejo de su solución lo cual se ha convertido en un reto intelectual que ha sido estudiado desde la década de los 90. Muchos procedimientos se utilizan para la construcción de horarios viables y atractivos. Estos enfoques se pueden agrupar en tres categorías: Inteligencia Artificial, interacción Hombre-Máquina y la Investigación de Operaciones. Existen dos áreas dentro de la Investigación de Operaciones que se ocupan en problemas de este tipo, la primera se conoce como *Scheduling* que se encarga de la asignación de recursos y la segunda se conoce como *Timetabling* que asigna ciertos eventos a los distintos bloques de horarios y lugares físicos condicionados a las restricciones y limites propuestos por el sistema.

Los problemas de horario en instituciones se dividen en dos: horarios de universidades y horarios de colegios. Los horarios para colegios resultan más compactos, lo que significa que no debe haber periodos libres entre clases. Todos los cursos tienen la misma hora de inicio, termino y de colación, lo cual no es el caso de los horarios universitarios. Existe una cantidad mayor de programas, clases, docentes y diferentes grupos de estudiantes lo cual hace el problema más complejo.

En particular, el problema de horarios dentro de una universidad está clasificado como un problema NP "*non-polynomial*" (no-polinomial). Esto significa que la cantidad de tiempo y los esfuerzos requeridos para resolver este tipo de problemas se incrementa exponencialmente con el tamaño del problema los cuales se hacen más complejos conforme aumenta el tiempo. Por lo tanto no pueden ser resueltos en tiempo polinómico. Las técnicas de optimización que se utilizan para resolverlos producen soluciones factibles cercanas al óptimo en lugar de soluciones exactas. Los Algoritmos Genéticos son considerados un buen enfoque usado en la actualidad para resolver este tipo de problemas.

Para poder determinar de mejor forma cual será la relación docente-bloque-sala para Instituciones con este tipo de problemas se hace necesario identificar el flujo y dirección de los datos que intervienen en el proceso a modo de establecer también aquellos parámetros que más adelante serán considerados críticos que pueden generar un cambio en la metodología actual.

Este trabajo fue motivado para resolver el problema de horarios y producir una solución totalmente aceptable para todos los usuarios en vista a los conflictos observados en la institución objeto de nuestro estudio. El alcance del mismo consiste en el desarrollo de un modelo que contemple las variables y restricciones requeridas por la dirección de manera de optimizar la gestión universitaria en la creación de la oferta académica y asignación de las salas de clases en la Escuela de Ingeniería Industrial de la Universidad de Valparaíso siendo eficientes en el uso de los recursos.

Este Trabajo de Título está organizada de la siguiente manera. El capítulo 2 se centra en el caso en particular, considerando los antecedentes, las restricciones del problema y los objetivos que busca este trabajo. El capítulo 2 correspondiente al marco teórico, hace una revisión de la literatura relevante, así también como los diversos enfoques para resolver el problema de horarios en las universidades y la justificación de la utilización del algoritmo genético como parte de la solución del problema. Dentro de este mismo capítulo, en el apartado 2.6 se proporciona una visión general del mismo. Aquí se da a conocer los fundamentos de un algoritmo genético y sus aspectos teóricos. Por otra parte se complementa lo anterior con un simple ejemplo que demuestra cómo es el funcionamiento básico del algoritmo mostrado en los anexos de nuestro trabajo. El capítulo 3 describe la metodología de investigación, en donde se hace un detalle de los datos o recursos que se utilizarán, además del diseño del modelo contemplando solamente aquellos parámetros, conjunto de datos, variables y restricciones más significativas del problema. El Capítulo 4 describe la implementación del algoritmo genético a los datos procedentes de la Escuela de Ingeniería Industrial. El Capítulo 5 finalmente llega a la conclusión de la tesis.

1. Problema de Investigación.

1.1. Antecedentes

El proceso de creación de horario actual en la Escuela de Ingeniería Industrial de la Universidad de Valparaíso presenta problemas en la planificación, asignación y gestión de los recursos disponibles. La creación de la oferta académica para un semestre dado debe cumplir ciertas restricciones que surgen al tener en consideración la organización de la Escuela, la infraestructura del edificio y la cantidad de alumnos inscritos que se prevé para las diferentes asignaturas y sus secciones a dictar según el actual programa de estudios.

El proceso actual consiste en asignar en un total de 45 bloques horarios (5 días de 9 bloques cada uno) las respectivas asignaturas junto a su docente en una sala de clases específica. Esto se hace de forma manual en *M.S Excel* y se realiza principalmente a partir de *datos históricos* del semestre par o impar pasado. Uno de los puntos críticos que se da en la creación de la oferta académica es respecto a las asignaturas dictadas de manera excepcional, puesto que al utilizarse datos de la oferta anterior no se sabe si la decisión de la apertura de una nueva sección fue o no la acertada. Los casos que se dan para la incorporación de una nueva sección al horario son:

a) Asignaturas con alta tasa de reprobación: las que en la actualidad se están dictando de manera semestral y que recién una vez que los estudiantes realizan su respectiva inscripción[1] se sabe si fue acertado o no la apertura de solo una sección o si en verdad era necesario considerar una nueva sección de dicho curso, teniendo dos en total. Esta situación genera conflictos con el estudiante al tener que ajustar nuevamente su carga académica, al docente, para reasignar su horario y a la Escuela en la distribución del limitado espacio físico.
b) Levantamiento de Prerrequisitos[2] y avance académico: La aprobación de un grupo solicitudes de levantamiento han generado que exista una mayor demanda por alguno de los diferentes cursos ofrecidos en el plan curricular

[1] La inscripción de asignaturas se realiza una vez conformada y revisada la oferta académica.
[2] Se realizan las solicitudes de levantamiento una vez que comienza el proceso de inscripción.

produciéndose así la apertura de nuevas secciones una vez ya conformada la oferta académica lo que lleva a realizar una reestructuración tardía de la misma. También al haber estudiantes con ramos de diferentes semestres en su carga académica, se van generando grupos numerosos que más adelante realizan solicitudes de cambio de horario que de igual forma hace necesaria una reestructuración de la oferta del semestre.

En algunas situaciones tampoco se respeta el límite máximo de estudiantes por sección al momento de realizar la inscripción, que más adelante implica modificaciones en la actual oferta académica.

Lo anterior influye directamente en la asignación de infraestructura disponible, puesto que la Escuela al no poseer una planificación de su capacidad no aprovecha al máximo los espacios de las salas de clases o bien se produce que exista una sobrepoblación de las mismas. El problema de que exista una reestructuración tardía de la oferta académica produce que no se pueda distribuir de manera eficiente las salas de clases a aquellos nuevos cursos dictados de manera excepcional.

Las salas de clases son un recurso común que comparte la carrera de Ingeniería Civil Industrial con Ingeniería Civil Oceánica. Si bien la demanda de esta última es mucho menor no se producen conflictos de topes entre las diferentes carreras, sino más bien se generan problemas en la asignación como ya fue mencionado anteriormente.

Otro punto en conflicto corresponde a la asignación de los docentes a los diferentes bloques de horario, ya que se considera la disponibilidad que tienen los mismos para dictar alguna asignatura de su interés y sumado a que la dotación de personal es escasa la oferta académica es la que termina por adaptarse a los docentes. Esto conlleva a que la asignación se dé en periodos no deseados para los estudiantes y en algunos casos para los mismos docentes porque no existe un balance de la carga de trabajo diaria producido por una incorrecta planificación de los docentes.

El proceso de creación de la oferta académica se resume mediante el siguiente diagrama de flujos

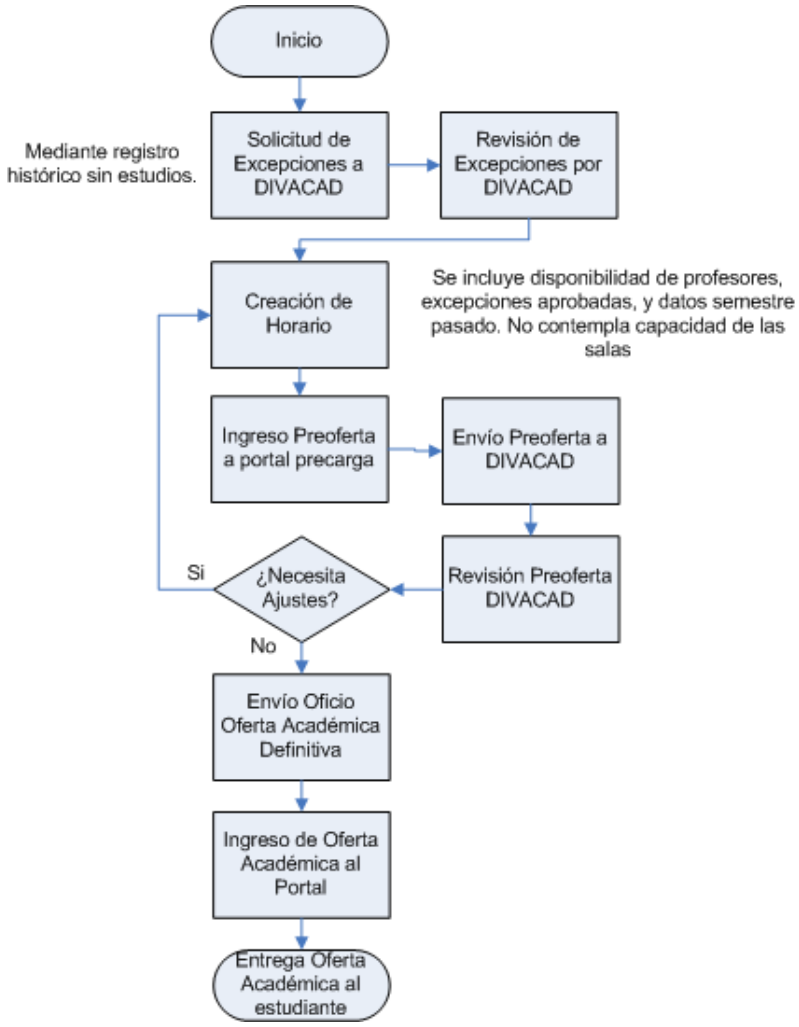

Figura 1. Diagrama de Flujos del proceso de creación de oferta académica

1.2. Restricciones del problema.

El modelo que se empleará en el presente trabajo debe resolver este tipo de problemas de asignación conocido como *UCTP* y para ello se establece dos conjuntos de restricciones: obligatorias y flexibles.

Una correcta asignación horaria es factible si satisface el total de restricciones obligatorias. En general las restricciones obligatorias son aquellos que tiene que ver con las restricciones de los recursos materiales disponibles, de infraestructura y todo lo referente a los docentes y alumnos. Las que se han denominado flexibles son aquellas restricciones deseables que han sido impuestos por la Escuela que no tienen un carácter obligatorio.

Restricciones Obligatorias:

a) Una sección no puede ser asignada a una sala de clases con menor capacidad a la de alumnos inscritos.
b) Dos asignaturas no pueden ocupar la misma sala de clases en el mismo bloque horario.
c) Se deben evitar los topes de ramos correspondientes a un mismo semestre.
d) Se debe respetar la disponibilidad de los académicos y su preferencia.
e) Un docente no puede dictar dos asignaturas en el mismo bloque horario.
f) Cada Asignatura debe cumplir con una cantidad de horas semanales requeridas.
g) Se deben considerar los tipos de salas para ciertas asignaturas (taller, catedra)

Restricciones Flexibles:

a) Asignaturas de repetición deben ser dictadas en las tardes.
b) Priorizar un horario más compacto para aquellos ramos que corresponden al semestre en cuestión.
c) Minimizar en lo posible la existencia de ventanas o asignación en periodos académicos no deseados.
d) Considerar la distribución en los diferentes días para aquellas asignaturas consideradas “densas”, de tal forma que no sean asignadas consecutivamente en los respectivos bloques horarios.

1.3. Objetivos.

1.3.1. Objetivo General.

Proponer un modelo de programación que optimice el proceso de creación de la oferta académica en la Escuela de Ingeniería Industrial para una mejor planificación, asignación, y utilización de los recursos disponibles.

1.3.2. Objetivos Específicos.

a. Identificar los procesos, variables y flujo de datos asociados al actual plan de creación de la oferta académica.
b. Diseñar un modelo de programación para la creación de la oferta académica de cada semestre tomando en consideración las restricciones de infraestructura y disponibilidad de los académicos.
c. Analizar y valorar el desempeño del modelo de programación bajo distintos escenarios para determinar los parámetros críticos del mismo.

2. Marco Teórico.

2.1. Conceptos de Timetabling y Scheduling.

Una primera definición del Problema de Horarios (*Timetabling Problem,* TTP) es la tarea de asignar un número de eventos, tales como exámenes, conferencias, reuniones entre otros, a un intervalo de tiempo de acuerdo a un conjunto de restricciones. En el caso de los Problemas de Programación (*Scheduling Problem, SP*) se busca asignar a ciertos eventos diferentes recursos o bloques de horario.

Algunos pueden considerar que *Timetabling* y *Scheduling* son actividades separadas, pero en este trabajo se utilizará *Scheduling* como un término genérico para diferentes tipos de problemas como es el caso de los horarios.

Anthony Wren (Wren, 1996) define que *Scheduling* es la asignación, sujeto a restricciones, de recursos y tiempo al conjunto de actividades que se colocan en el espacio-tiempo, de tal manera que se minimice el costo total de un conjunto de los recursos utilizados teniendo en cuenta la capacidad limitada de los recursos compartidos. Ejemplos comunes de Programación son la programación de transporte o ruteo de vehículos los cuales buscan reducir al mínimo el número de vehículos o conductores para minimizar el costo total. Otro ejemplo que el autor señala es la programación en un taller de trabajo para tratar de minimizar el número de periodos de tiempo o recurso físico utilizado.

Así mismo (Wren, 1996) expone que *Timetabling* es una forma particular de los SP y lo define como la asignación, sujeto a restricciones, de recursos dados a los objetos que se colocan en el espacio-tiempo, de tal manera que se satisfagan en lo posible un conjunto de objetivos deseables, siendo más importante el cuándo van a ocurrir los eventos que en dónde. Ejemplos de esto son los horarios de clases, horarios para exámenes, asignaciones de personal (ej. dotación de herramientas sujetas a un determinado número de personal).

Ambos conceptos definen entonces una clase de problemas de optimización difíciles de resolver con restricciones de naturaleza combinatoria. Tales problemas se clasifican principalmente como problemas de satisfacción de restricciones (Brailsford, Potts, & Smith,

1999), donde el objetivo principal es satisfacer todas las restricciones del problema, en lugar de la optimización de un número de objetivos.

La automatización de horarios es una tarea de gran importancia, ya que puede ahorrar una cantidad considerable de horas-hombre de trabajo, a instituciones y empresas, y además proporcionar soluciones cuasi óptimas con satisfacción de restricciones en poco tiempo, que puede aumentar la productividad, la calidad de la educación, la calidad de los servicios y finalmente la calidad de vida. Sin embargo, los horarios de gran escala, tales como horarios de la universidad, es necesario un gran esfuerzo y la utilización de bastantes horas de trabajo, ya sea por persona cualificada o por medio un equipo, con el fin de producir los horarios de alta calidad con óptima satisfacción de restricciones y optimización del calendario de objetivos al mismo tiempo.

En el marco de este trabajo nos centramos sobre la programación de horarios y la asignación de las salas de clases en Universidades para el caso particular aunque de igual forma se presentarán otros tipos de problemas de programación horaria, los diferentes métodos y resoluciones y algunos trabajos ya aplicados en esta temática.

2.2. Problemas de Programación Horaria.

Los sistemas desarrollados para problemas de programación horaria son, por lo general, muy específicos y de compleja generalización. Los TTP comparten en su mayoría ciertas características generales pero difieren sus soluciones debido a lo diferente de los problemas. Es por ello que los investigadores se han especializado en cada uno de los diferentes problemas de horarios, de esta forma se puede aprovechar el conocimiento y generar horarios de buena calidad al implementar el método que se ajuste ha dicho problema.

Problemas de Programación de Horarios Deportivos – (*Sports Timetabling Problem*). La forma más común para las competiciones deportivas es el *Round-Robin Tournament* (torneo de todos contra todos) en donde se presentan variaciones de partidos simples o dobles (de local y visitante). En este tipo de problemas existen diferentes tipos de restricciones como las reglas de la liga, las necesidades de los medios de comunicación y las propias necesidades

de los equipos, pero lo que busca finalmente la programación de los horarios es minimizar las distancias entre ambos equipos como señalan (Easton, Nemhauser, & Trick, 2001) y (Werra, 1981), ya que en algunos casos el equipo visitante queda muy retirado del lugar donde se encuentra el equipo local. En nuestro caso particular de programación horaria y asignación de salas no aplica el criterio de las distancias recorridas por los alumnos ya que por lo general todas las clases se imparten en la misma facultad.

Problemas de Programación horaria de Empleados – (*Employee Timetabling Problem*). (Franses & Post, 2003) describen un problema de asignación especial de tareas para los empleados en un laboratorio. El servicio debe ser proporcionado de tal manera que se satisfaga en su totalidad la agenda de la semana, además de programar los lugares de trabajo de los empleados en el laboratorio, farmacia, sala de rayos X y el departamento de operaciones en solo un turno de 8 horas para los días hábiles de trabajo. Este tipo de problemas consiste, por lo general, en asignar a los empleados en sus respectivos lugares de trabajo y con su agenda diaria en el turno correspondiente de manera de cumplir con las restricciones prácticas.

Problemas de Programación de Horarios en el Transporte Público - (*Public Transport Timetabling Problem*). Este tipo de problema es aplicable a diferentes medios de transporte (autobús, metro, avión, etc.). A diferencia de los otros problemas, los recursos son móviles y recorren distancias para completar sus tareas lo cual significa que el desplazamiento es parte importante del problema (Wren, 1981) a diferencia de lo que ocurre con la programación de horarios de clases y salas, en donde el desplazamiento de los estudiantes no forma parte del problema en sí.

Problemas de Programación Horarios en Colegios y Universidades – (*University Course and School Timetabling Problem*). Dado un conjunto de asignaturas, un conjunto de docentes, un conjunto de salas de clases, un conjunto de bloques de horario y finalmente un tiempo de planificación (en la semana) los problemas de horarios en Colegios y Universidades consisten en asignar asignaturas a las salas de clases y en periodos de tiempo, satisfaciendo además las restricciones adicionales que sean impuestas por las instituciones educativas. La literatura contiene un gran número de variantes del problema de horarios. En el caso de (Schmidt & Strohlein, 1980) enumeran muchos *papers* que han aparecido antes de 1980. (Schaerf, 1997) da un estudio de las diversas formulaciones de problemas de horarios y clasifica el problema de horario en tres categorías:

- **Problemas de Programación de Horario en Colegios** (*School Timetabling Problem*): (Tripathy, 1984) La programación semanal de todas las clases de un colegio. Evita que los docentes se reúnan en dos clases al mismo tiempo y evita que una clase tenga dos docentes al mismo tiempo. Además de producir un horario factible es necesario equilibrar las cargas de trabajo para los docentes y clases.
- **Problema de Programación de Horarios para Examenes** (*Examination Timetabling Problem*): (Burke, Bykov, & Petrovic, 2001) La programación de exámenes de las asignaturas en las Universidades. Evita que los exámenes de los estudiantes se superpongan en periodo de examen establecido por cada Institución.
- **Problema de Programación de Horarios en Universidades** (*University Timetabling Problem*): (Stallert, 1997) La programación semanal de todas las clases en una Universidad evitando que una clase se superponga con otra que tienen estudiantes en común.

2.3. Modelos y técnicas de solución.

Como lo describe (Werra, 1985) los problemas de horarios pueden ser clasificados dependiendo si se pretende optimizar o no la función objetivo.

Problema de Optimización, que consisten en encontrar una programación que pueda satisfacer las restricciones obligatorias y que maximice (o minimice) la función objetivo que incluye las restricciones flexibles.

Problema de Búsqueda, que consisten en encontrar una programación de horario que satisfaga todas las restricciones obligatorias.

Una larga lista de diversos métodos ha sido propuesta en la literatura para resolver los problemas de horarios. Estos métodos forman parte de diversas disciplinas tales como la Investigación de operaciones, Inteligencia Artificial y la Inteligencia Computacional.

2.3.1. Métodos de Optimización.

Todo problema de optimización se modela originalmente en fórmulas matemáticas, expresión gráfica o lenguajes informáticos describiendo las variables de decisión, parámetros (es decir, valores constantes), las limitaciones mismas de las variables de decisión y funciones objetivos. A continuación se presentan 4 métodos de optimización, la Programación Matemática y Flujos de Redes (métodos de optimización exactos), Coloreo de Grafos (métodos de optimización secuenciales) y la Satisfacción de las Restricciones (métodos de optimización de agrupamiento).

2.3.1.1. Programación Matemática (*Mathematical Programming*)

Un enfoque tradicional para la solución de TTP es la programación matemática, y hay extensos estudios que utilizan estos modelos de programación tales como los propuestos por (Tripathy, 1984), (Tripathy, 1992) y (Daskalaki, Birbas, & Housos, 2004).

La programación matemática al tratar los problemas de horario genera un número elevado de variables y restricciones, que hacen que para instancias de tamaño real el problema no pueda tratarse de manera práctica (Carter, 1986)

2.3.1.2. Flujo en Redes (*Network Flow*)

Para el problema de programación horaria (Ostermann & Werra, 1983) lo redujeron a una secuencia de problemas de flujo de red. Crearon una red para cada periodo de modo que el flujo en la red identifica las clases dictadas en ese periodo. Posteriormente (Werra, 1985) propone un método similar creando una red para cada clase. La solución entregada es siempre un número entero debido a la propiedad unimodular de la matriz (Papadimitriou & Steiglitz, 1982) y da una programación horaria para todas las clases de una asignatura. La construcción de la red se repite para todas las asignaturas y, finalmente, si se encuentra una solución para todas las redes, se completa la asignación horaria para todas las asignaturas. No hay retroceso

en las clases ya programadas puesto que no existe ninguna garantía de que la solución encontrada exista.

2.3.1.3. Coloreo de Grafos (*Graph Coloring Problem*).

El método de coloreo de grafos fue la primera técnica para encontrar solución a los TTP. En este método se define un grafo en donde cada uno de sus nodos representa las asignaturas a programar y las aristas (nodos sin agrupar) representar los conflictos entre los diferentes cursos. Lo que se pretende es que todos los nodos estén pintados y que los nodos que se encuentren conectados no sean pintados del mismo color.

Este tipo de heurística ordenan los eventos estimando la dificultad de asignarlos a un horario (Carter & Laporte, 1996) . Por lo general las heurísticas usadas en este método son:

- Grado más grande primero (*Largest Degree*): Esta heurística programa primero aquellos eventos (o nodos) con mayor número de conflictos.
- Grado más grande ponderado (*Largest Weighted Degree*): Esta heurística es similar a la anterior excepto que los bordes están ponderados por el número de estudiantes que están en conflicto.
- Grado de Color (*Color Degree*): Esta heurística da prioridad con aquellos eventos que tengan un número alto de conflictos con los eventos que ya han sido programados en el horario.
- Grado de Saturación (*Saturation Degree*): En esta heurística se elige primero el evento que tenga el menor número de periodos disponibles en el horario que se puede seleccionar sin violar las restricciones obligatorias.

2.3.1.4. Satisfacción de Restricciones (*Constraint satisfaction*)

En la última década, los Problemas de Satisfacción de Restricciones (*Constraint Satisfaction Problem*, CSP) se han convertido en el método de elección para el modelado de muchos tipos de problemas de optimización: en particular, aquellos que implican restricciones heterogéneas y búsquedas combinatorias. En muchos campos, la rápida respuesta se ha

convertido en uno de los factores más importantes para el éxito, especialmente en el mundo comercial de hoy. En los campos que utilizan las técnicas de CSP, el tiempo de ejecución es un factor crítico. Esto es especialmente cierto cuando se desee utilizar técnicas de CSP para buscar alguna solución óptima.

Según (Tsang, 1993) un CSP se puede expresar como "Un problema compuesto por un conjunto finito de variables, cada una de los cuales está asociada con un dominio finito de valores, y un conjunto de restricciones que restringe los valores de las variables pueden tomar de forma simultánea. La tarea consiste en asignar un valor a cada variable que satisface todas las restricciones".

2.3.2. Métodos Metaheurísticos (*Metaheuristics Method*)

Este tipo de métodos han resultado muy convenientes a la hora de resolver TTP. Las metaheurísticas son algoritmos los cuales no aseguran entregar el óptimo global pero si buenas soluciones. Las soluciones entregadas parten de una solución inicial y con el avance de la ejecución del algoritmo va mejorando la solución obtenida. Entre las características más importantes de estos métodos se encuentra su independencia del problema, la capacidad de escapar de los óptimos locales y así buscar soluciones en otras áreas del espacio de soluciones.

A continuación se describen brevemente los principios básicos de cada metaheurística aplicable al caso de programación horaria y asignación de salas en Universidades.

2.3.2.1. Recocido Simulado (*Simulated Annealing*)

El autor (Abramson, 1991) aplica el recocido simulado a un problema de horarios en colegios. Considera, como una extensión, la posibilidad de que dos diferentes clases puedan tener estudiantes en común. Con esta extensión, su estructura cae en la categoría de problemas de horario. La solución es descrita por una lista de un conjunto de clases, una lista para cada periodo. Dada una solución, se realiza la elección de una solución cercana

seleccionando al azar un periodo y una clase en el periodo seleccionado moviendo la clase a un periodo diferente elegido al azar. La función objetivo (a ser minimizada) es la suma ponderada del número de conflictos de clases y los conflictos de los docentes. Aunque experimentó también con otros valores (Abramson, 1991) elige una velocidad de enfriado de 0,9. El número de iteraciones realizadas es del orden de tres millones.

Como es una variante de la búsqueda local, puede quedar atrapado prematuramente en un óptimo local. Con cada iteración se genera una nueva vecindad, en el caso del TTP un nuevo horario factible se modifica ligeramente de forma aleatoria para crear uno nuevo que también sea factible. Esta vecindad será aceptada como la solución actual solo si tiene una baja penalidad, de esta forma se convierte en parte del horario actual. Las soluciones que entrega este método dependen de las soluciones inicialmente encontradas, es decir, las soluciones encontradas al principio de la ejecución (exploración) son mucho peores que las encontradas al final (explotación).

2.3.2.2. Colonia de Hormigas (*Ant Colony*)

Este método metaheurístico propuesto por (Dorigo, Birattari, & Stützle, 2006) es uno de los más empleados y recientes para enfrentar los TTP. El principal componente en un ACO es el uso de un mecanismo de construcción de solución probabilístico, basado en estigmergia[3]. La inspiración de ACO es el comportamiento de forrajeo de las hormigas reales. Cada hormiga en la colonia realiza inicialmente en la búsqueda de su alimento trayectorias aleatorias, al hallarlo estudia según sus propios mecanismos la cantidad y la calidad y regresa a la colonia depositando una feromona, la cual permite a otras hormigas seguir el rastro y de esta manera ellas van reforzando la intensidad del rastro y así evitar su evaporación. En el caso de los TTP el concepto de evaporación de la feromona es utilizado para que el algoritmo converja a un óptimo local. En el caso de que las feromonas no desaparezcan, se abriría un amplio mundo de posibilidades de exploración de trayectoria, es decir, una amplia exploración de soluciones.

Lo que pretende esta metodología es obtener una alta organización y distribución de las denominadas hormigas artificiales para utilizarlas en la administración de la población de

[3] Estigmergia: Colaboración entre distintos agentes.

agentes artificiales, obteniendo de esta forma mejoras en la solución de problemas de optimización combinatoria.

2.3.2.3. Búsqueda Tabú (*Tabu Search*)

El método de Búsqueda Tabú es considerado una metaheurística de búsqueda local que utiliza conceptos de memoria adaptativa y exploración sensible para evitar el atrapamiento en óptimos locales y lograr un efectivo equilibrio de intensificación y diversificación. TS ha demostrado ser muy potente en la búsqueda de soluciones de alta calidad para problemas de difícil combinatoria como lo señala (Glover & Laguna, 1998).

TS permite la búsqueda para ir explorando soluciones que no disminuyan el valor de la función objetivo, pero sólo en los casos en que estas soluciones no son prohibidas (una solución está prohibida si se obtiene aplicando un movimiento tabú a la solución actual.). La búsqueda Tabú permite moverse a una solución de su entorno o vecindad que no sea tan buena como la actual, de tal manera de poder salir de los óptimos locales y continuar de manera estratégica en la búsqueda de soluciones aún mejores. Esto se obtiene normalmente mediante el seguimiento de las últimas soluciones en términos del movimiento utilizado para transformar una solución a la siguiente. Estos movimientos se denominan “movimientos Tabú”. Cuando se realiza un movimiento, el movimiento inverso es considerado tabú para las próximas iteraciones. Por lo tanto el escape de los óptimos locales se produce de forma sistemática y no aleatoria. Finalmente la filosofía de esta técnica es la creencia de que la elección de una mala estrategia sistemática de búsqueda es mejor que una buena elegida al azar.

2.3.2.4. Método de la Pendiente (*Descent Methods*)

Entre las técnicas de búsqueda local tenemos la de Método de la Pendiente o Método del Descenso (Ross & Corne, 1995). Este analiza todos los movimientos posibles en un vecindario y elige el que da la mejor solución, acepta al candidato moverse solo si este mejora el valor de la función, es decir si ningún movimiento desde dentro del mismo vecindario puede

mejorar la solución actual se detiene allí alcanzando prontamente un mínimo local. Un movimiento vecino puede ser elegido de dos maneras: el movimiento de mayor pendiente o seleccionado por azar. La búsqueda del movimiento de mayor pendiente permite al método recorrer la vecindad entera calculando y eligiendo aquel movimiento que otorga una mayor calidad a la solución entregada.

2.3.2.5. Algoritmos Genéticos (*Genetic Algorithms*).

Los Algoritmos Genéticos han sido utilizados para resolver los TTP inicialmente por los autores (Colorni, Dorigo, & Maniezzo, 1990).

Los algoritmos genéticos son algoritmos de búsqueda basados en la idea de la selección y la genética natural. La eficiencia de los algoritmos genéticos depende fuertemente en sus parámetros. Estos parámetros se establecen, por lo general, de acuerdo a recomendaciones de expertos vagamente formuladas o por el llamado algoritmo genético de dos niveles, donde en el algoritmo genético de primer nivel se optimiza los parámetros del segundo. La auto-adaptación parece ser una forma prometedora de los algoritmos genéticos, donde los parámetros se optimizan durante el mismo ciclo de evolución como el problema mismo.

Inspirado en la teoría de la evolución de Darwin de 1859, donde los individuos con más aptitudes para sobrevivir y dejar un mayor número de descendientes, son los más favorecidos (aptos) y transmiten a sus hijos los caracteres favorables de manera hereditaria, un algoritmo genético hace que una población de soluciones individuales evolucionen bajo el control de los dos operadores: "*Mutation*" y "*Crossover*". *Mutation* opera en uno o posiblemente muchos genes (atributos) de una solución mediante la alteración de sus valores. *Crossover* consiste en el apareamiento de los genes de los padres por parejas, produciendo descendencia con nuevas propiedades. Sólo los individuos de mejor ajuste es probable que sobrevivan de generación en generación.

2.3.2.6. Métodos Hiperheurísticos (*Hyper-Heuristics Method*)

Los métodos Hiperheurísticos se pueden definir como "una heurística que elige heurísticas " o como " algoritmos para elegir el algoritmo adecuado para la situación correcta" según (Burke, y otros, 2003). La razón principal para usar el término hiperheurística en lugar de metaheurística es que las hiperheurísticas representan un método con una variedad de diferentes heurísticas (que puede incluir metaheurísticas).

La principal motivación detrás de hiperheurística es elevar el nivel de generalidad en el que las metodologías de búsqueda pueden funcionar. Se pueden distinguir de otros algoritmos de búsqueda heurística, ya que operan en una búsqueda de espacio heurístico (o componentes heurísticos) en lugar de directamente en el espacio de búsqueda de soluciones para el problema de fondo.

2.3.2.7. Redes Neuronales (Neural Nets)

Una red neuronal (NN) se puede definir como un sistema compuesto de unidades de procesamiento distribuido paralelo que calcula ciertas funciones matemáticas sencillas (por lo general no lineales). Esta estructura está inspirada en el formato del cerebro humano, que consiste en una vasta red de células llamadas neuronas según (Arbib, 2003).

Las NN son técnicas heurísticas alternativas para la resolución de problemas de predicción, clasificación y reconocimiento de patrones que en el caso de TTP no existen muchos resultados de su aplicación en el último tiempo. Se pueden clasificar como una arquitectura de red. Esencialmente, hay tres tipos de arquitectura: las redes acíclicas con una sola capa, redes acíclicas con múltiples capas y redes recurrentes. En las redes acíclicos los datos fluyen a través de la red tomando lugar hacia la entrada de la salida. Pueden ser de una sola capa o múltiples capas. Las redes recurrentes difieren, en cuanto a su arquitectura, mediante la presentación de algunas neuronas, como aporte a la salida de sí mismos o de otros después de ellos.

Por lo general para encontrar soluciones a problemas de optimización combinatoria se utilizan las NN.

2.3.2.8. Sistema de Expertos (Expert System)

Una técnica de solución basada en los sistemas expertos la dan (Meisels, Gudes, & Kuflik, 1991) y (Solotorevsky, Gudes, & Meisels, 1994) , para ello definen un lenguaje basado en reglas, llamado RAPS, para especificar los problemas de asignación de recursos en general, y lo utilizan por supuesto para los TTP. En particular, consideran las clases como actividades y los periodos (bloques horarios) como los recursos que se asignarán a las actividades. RAPS tiene cinco tipos de reglas, reglas de asignación, reglas de restricción, reglas de cambio locales, reglas de contexto, y las reglas de prioridad.

- Reglas de asignación, asignan clases a los plazos, uno a la vez, que son el núcleo del sistema. Estas reglas, son suministrados por el usuario, por lo tanto la heurística utilizada no están predefinida, pero es elegida por el usuario.
- Reglas de restricción, especifican las limitaciones que la solución debe satisfacer. Ello se comprueba cada vez que una nueva clase se asigna a un período (una nueva actividad se asigna a un recurso). Estas reglas se dividen en positivos y negativos, es decir, las restricciones que deben cumplirse y restricciones que pueden fallar. Permiten identificar conflictos tan pronto como se presenten.
- Reglas de cambio locales, especifican la acción a realizar cuando hay una clase que las reglas de asignación no son capaces de afrontar.
- Reglas de contexto, seleccionan el contexto activo. Un sistema permite múltiples contextos: En diferentes contextos las diversas clases y períodos puede tener diferentes prioridades. La prioridad de los objetos determina qué objeto, entre los de un tipo dado, se procesa primero.
- Reglas de prioridad determinan las prioridades de las clases y los períodos, en cada contexto. Las prioridades se calculan cada vez que un contexto es introducido.

El sistema puede funcionar en dos modos posibles: voraz y no voraz. En el modo voraz, cuando las reglas de asignación no pueden hacer una asignación, el sistema selecciona una clase diferente. Por el contrario, en el modo no voraz, cuando una falla se produce el control se pasa las reglas de cambio local.

2.3.2.9. Basados en el Conocimiento (*Knowledge Base*)

Uno de los métodos poco usados en los TTP es el llamado Método Basado en el Conocimiento. Los autores (Kanagasabapathy, Radhakrishnan, & Balasubraimanian, 2000) señalan que el conocimiento no es fácil de medir o auditar, por lo cual las organizaciones deben gestionar el conocimiento para aprovechar al máximo las experiencias y habilidades provenientes de sus sistemas o el conocimiento tácito de sus empleados.

Existe poca información sobre el uso de este método, aunque (Gudes, Kuflik, & Meisels, 1990) señalan un paradigma general para solucionar problemas de programación horaria y asignación de recursos.

2.4. Algunos trabajos relacionados ya aplicados.

Para el método de Coloreo de Grafos (Selim, 1988)emplea esta metodología para el problema de horario en la facultad donde los vértices se dividieron con el fin de reducir el número cromático. El enfoque fue probado con los datos reales de la Facultad de Ciencias de la Universidad Americana de El Cairo.

El Recocido simulado presenta aplicaciones como la de (Elmohamed, Coddington, & Fox, 1998) quienes aplicaron un algoritmo de recocido simulado con diferentes horarios de refrigeración (geométrica, de adaptación y de adaptación con recalentamiento) a un problema programación de horarios. Los algoritmos fueron probados en los datos reales de la Universidad de Syracuse. Los resultados experimentales muestran que el algoritmo adaptativo recocido simulado con el enfriamiento y recalentamiento superó a otros métodos.

La Búsqueda Tabú presenta variedad de trabajos, entre los más importantes tenemos el de (Alvarez-Valdes, Crespo, & Tamarit, 2000) que utilizaron la búsqueda tabú para la asignación de los estudiantes a las secciones del curso en la Facultad de Matemáticas de la Universidad de Valencia. El enfoque se llevó a cabo en dos fases. La primera fase generaba un conjunto de las mejores soluciones para cada estudiante. Estas soluciones se combinaron y se empleó un algoritmo de búsqueda tabú con oscilación estratégica y una lista tabú fijo para

mejorar la calidad de la solución para cada estudiante. Luego (Alvarez-Valdes, Crespo, & Tamarit, 2002) desarrollaron un horario de clases utilizando búsqueda tabú. Construyeron un algoritmo de tres fases en el que se obtiene el horario inicial en la primera fase. El procedimiento para mejorar la calidad del horario se lleva a cabo en la segunda fase en la que el algoritmo de búsqueda tabú juega su papel (que se considera que es la parte más importante de este algoritmo). La fase final se concentra en mejorar la asignación de salas de clases. Los experimentos se llevaron a cabo en una selección diferente de movimientos (es decir, movimiento simple, de intercambio y multi-intercambio) utilizando una lista tabú estática y dinámica con y sin la lista de candidatos.

También (Hertz, 1991) aplicó el algoritmo de Búsqueda Tabú, pero lo hizo en la Facultad de Economía de la Universidad de Génova, de esta forma minimizó los conflictos entre la facultad y los estudiantes. Se utilizaron datos de la misma Universidad con 1729 estudiantes, 143 facultades, 67 salas y 288 asignaturas.

En el caso de los Algoritmos Genéticos estudios como el de (Rahoual & Saad, 2006) codifican los horarios como vectores que asocian 3 genes con cada clase, en referencia al período, la salas y el docente asignado a la clase. En un intento de promover una búsqueda del espacio de soluciones bien extendido, la generación inicial propuesta fue construida al azar mediante la asignación de un intervalo de tiempo al azar junto con un docente para cada clase en una moda "codiciosa". A continuación, definen su función objetivo como la suma ponderada de todas las restricciones violadas, cada restricción se asocia con una pena (un peso) en proporción a la importancia que se le atribuyó a la restricción. Los individuos (horarios) se clasifican en el orden descendente de las condiciones de aptitud (como medida por la función objetivo), el individuo de mejor ajuste se posiciona con un 0. La probabilidad de la sustitución de un individuo se define entonces como la relación de la suma de todas las filas (*ranking*). En otras palabras, cuanto más pobre es la forma física, mayor es la probabilidad de un individuo que está siendo reemplazado.

Los autores (Ueda, Ouchi, Takahashi, & Miyahara, 2001) presentaron un Algoritmo Genético de dos fases para el problema de programación horarios. El problema se deriva del plan de estudios de la Facultad de Ciencias de la Información en la Universidad de la ciudad de Hiroshima en 1997. Dos tipos de población fueron generadas, una población se encargaba de la programación de clases (utilizado en la primera fase) y la otra población de la asignación

de salas (utilizado en la segunda fase). Estas dos poblaciones evolucionaron por separado. El algoritmo fue capaz de encontrar una solución factible.

Para la Colonia de Hormigas, (Socha, Knowles, & Samples, 2002) aplican una variante del método llamado Sistema de Hormigas MIN-MAX para horarios de clases en universidades. El problema de horarios se transforma en un problema de ruta óptima que puede ser abordado mediante la generación de un grafo. La asignación de cursos a los intervalos de tiempo es dependiente del valor feromona dentro de los límites. Comparan su algoritmo de búsqueda local con un reinicio aleatorio (repite este mismo algoritmo de búsqueda local, utilizada por el sistema de hormigas MIN-MAX en el que comienza a partir de una solución al azar y mantiene la mejor solución encontrada). Los resultados muestran que el sistema de hormiga MIN-MAX se comporta mejor que el método de búsqueda local de reinicio aleatorio.

Los autores (Rossi-Doria, y otros, 2003) presentaron una comparación de cinco meta-heurísticos diferentes aplicados a un problema universitario curso de horarios. Uno de los meta-heurísticos utilizados es el de Colonia de Hormigas. En este enfoque, una hormiga construye un horario usando una estrategia secuencial donde se elige un curso de una lista predefinida y lo asigna a un intervalo de tiempo de una manera probabilística. Los resultados experimentales muestran que el rendimiento de la optimización de colonia de hormigas es ligeramente peor que el recocido simulado y búsqueda tabú. Sin embargo, es mejor que un algoritmo genético.

2.5. Algoritmo usado en el problema.

Los Algoritmos Genéticos y Búsqueda Tabú son dos métodos de solución que han sido ampliamente utilizados para los TTP en la actualidad. En el siguiente apartado se hará una comparación de ambos métodos, primero en base a una prueba realizada por (Gülcü, Kuzucuoğlu, & Bulkan, 2011) en la Universidad de Marmara respecto a la calidad de la mejor solución reportada por cada algoritmo, el espacio de soluciones que conduce a dicha solución y el costo de encontrar cada solución y segundo por los autores (Naupari & Rosales, 2010) según variados criterios de comparación de diferentes metaheurísticas.

2.5.1. Comparación entre Búsqueda Tabú y Algoritmo Genético para TTP.

Para (Gülcü, Kuzucuoğlu, & Bulkan, 2011) el problema consistía asignar 140 cursos en 15 salas de clases y 50 bloques de horario (5 días, 10 bloques por cada día). El número de los docentes es 25. Cada docente dicta 5 cursos en promedio. El objetivo aquí era asignar todos los cursos a sus respectivas salas de clases y bloque de horario de tal manera que todas las restricciones duras se cumplieran, y las restricciones blandas fueran satisfechas en lo posible.

Ambos métodos fueron ejecutados de manera más sencilla para resolver este problema. Las restricciones que fueron establecidas correspondían a cinco duras y a cuatro blandas (deseables). Los dos algoritmos se compararon según los siguientes criterios:

- Calidad de la mejor solución encontrada.
- El costo computacional de obtener dicha solución.
- La trayectoria de búsqueda desde el inicio hasta el final de la solución.

Los métodos fueron capaces de producir soluciones factibles, por lo tanto, los dos algoritmos se comparan en base a lo óptimo de la mejor solución encontrada. La TS no logró satisfacer una de las restricciones blandas para 41 cursos, en cambio el GA no lo logro para 17 cursos. Adicionalmente la TS violó otra restricción para 3 cursos mientras que el GA no violó ninguna.

La cantidad de tiempo empleado y el número de iteraciones para encontrar cada una de las soluciones son señaladas a continuación.

Tabla 1. Violación de restricciones blandas caso Universidad Marmara.

	Restricción Blanda 1	Restricción Blanda 2	Restricción Blanda 3	Restricción Blanda 4	# de iteraciones	Tiempo CPU (min)
GA	0	17	0	0	34	38
TS	0	41	3	0	106	131

Según la tabla anterior y con las observaciones dadas, ambos algoritmos no son superiores entre sí en términos de la solución encontrada. Los dos algoritmos fueron capaces de dar soluciones factibles, pero el GA fue ligeramente mejor que TS en cuanto a la

satisfacción de algunas restricciones blandas. El GA alcanzó su mejor solución después de aproximadamente 34 iteraciones, mientras que la TS tomó 106 iteraciones para llegar a su mejor solución. La cantidad de tiempo de CPU requerida para el GA para llegar a la mejor solución fue de 38 minutos en cambio la TS llegó a la mejor solución en 131 minutos.

Los autores concluyen que el Algoritmo Genético resulta una mejor elección por sobre la Búsqueda Tabú aunque señalan que la implementación del mismo requiere mucho más esfuerzo que una TS.

El otro estudio fue realizado por (Naupari & Rosales, 2010) en donde calificaron cuatro de las más importantes metaheurísticas aplicadas al TTP en base a variados criterios según se muestra a continuación.

Tabla 2. Comparación de Metaheurísticas aplicadas a TTP

Características	Algoritmo Memético	Algoritmo Genético	Búsqueda Tabú	Recocido Simulado
Simplicidad	0.5	1	0.75	0.75
Independencia	1	1	1	1
Coherencia	0.75	1	0.75	1
Efectividad	1	1	1	0.75
Eficacia	0.75	0.75	0.75	0.5
Eficiencia	1	1	0.75	0.5
Generalidad	0.5	0.75	1	1
Adaptabilidad	1	1	1	0.75
Robustez	0.75	0.75	0.75	0.75
Interactividad	0.75	0.75	0.75	0.75
Diversidad	1	1	0.5	0.5
Autonomía	1	1	1	1
Puntaje Final	10	11	10	9.25

En la Tabla 2 a cada característica se le asignó un puntaje entre 0 y 1, siendo 0 el puntaje más bajo y 1 el puntaje más alto. A partir de esta comparación se dedujo que el uso de Algoritmo Genético es el que mejor se adapta a la situación en particular para resolver el problema de programación horaria y asignación de salas de clases.

Finalmente teniendo en consideración ambos estudios desarrollados por los autores mencionados anteriormente y por las características que presenta nuestro problema de asignación de horarios en particular es que se utilizará el método de **Algoritmos Genéticos** para la resolución del problema.

2.6. Algoritmo Genético.

Los Algoritmos Genéticos son métodos adaptativos, usados en problemas de búsqueda y optimización de parámetros, que se basan en la reproducción sexual y en el principio de supervivencia del más apto (Gestal, Rivero, Rabuñal, Dorado, & Pazos, 2010).

Según (Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, 1989) “los Algoritmos Genéticos son algoritmos de búsqueda basados en la mecánica de selección natural y de la genética natural. Combinan la supervivencia del más apto entre estructuras de secuencias con un intercambio de información estructurado, aunque aleatorizado, para constituir así un algoritmo de búsqueda que tenga algo de las genialidades de las búsquedas humanas”.

Para alcanzar la solución a un problema se parte de un conjunto inicial de individuos, llamado población, la cual es generada de manera aleatoria. Cada uno de estos individuos representa una posible solución al problema. Estos individuos evolucionarán tomando como base los esquemas propuestos por Darwin sobre la selección natural, y se adaptarán en mayor medida tras el paso de cada generación a la solución requerida (Darwin, 2007).

2.6.1. Origen del Algoritmo Genético.

Durante millones de años las diferentes especies se han adaptado para poder sobrevivir en un medio cambiante. De la misma manera se podría tener una población de potenciales soluciones a un problema, de las que se irían seleccionando las mejores hasta que se adaptasen perfectamente al medio, en este caso el problema a resolver (Michalewicz & Fogel, 2000) (Bäck, 1996) (Whitley, 1994). En términos muy generales se podría definir la computación evolutiva como una familia de modelos computacionales inspirados en la evolución. Más formalmente, el término de computación evolutiva se refiere al estudio de los fundamentos y aplicaciones de ciertas técnicas heurísticas basadas en los principios de la evolución natural (Tomassini, 1995). Estas técnicas heurísticas podrían clasificarse en 3 grandes categorías o grupos, tal como se muestra a continuación:

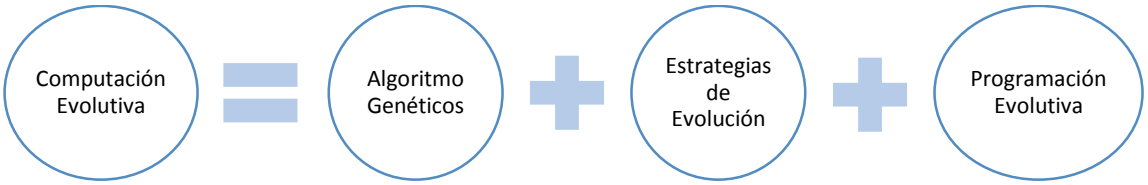

Figura 1 Computación Evolutiva

El desarrollo de los Algoritmos Genéticos se debe en gran medida a (Holland, 1975), investigador de la Universidad de Michigan quien desarrolló una técnica que imitaba en su funcionamiento a la selección natural, la que más adelante se conocería con el nombre de Algoritmos Genéticos. A grandes rasgos un Algoritmo Genético consiste en una población de soluciones codificadas de forma similar a cromosomas. Cada uno de estos cromosomas tendrá asociado un ajuste, valor de bondad o *fitness*, que cuantifica su validez como solución al problema. En función de este valor se le darán más o menos oportunidades de reproducción. Además, con cierta probabilidad se realizarán mutaciones de estos cromosomas (Goldberg, 2002).

La Programación Evolutiva surge principalmente a raíz del trabajo de los autores (Fogel, Owens, & Walsh, 1966). En este caso los individuos, conocidos aquí como organismos, son máquinas de estado finito. Los organismos que mejor resuelven alguna de las funciones objetivo obtienen la oportunidad de reproducirse. Antes de producirse los cruces para generar la descendencia se realiza una mutación sobre los padres.

La computación evolutiva designa a un amplio conjunto de técnicas heurísticas de resolución de problemas complejos que basan su funcionamiento en un mecanismo análogo a los procesos de la evolución natural. Puede verse como uno de los campos de investigación de lo que se ha dado en llamar *Soft Computing* como se puede apreciar en la Figura 2.

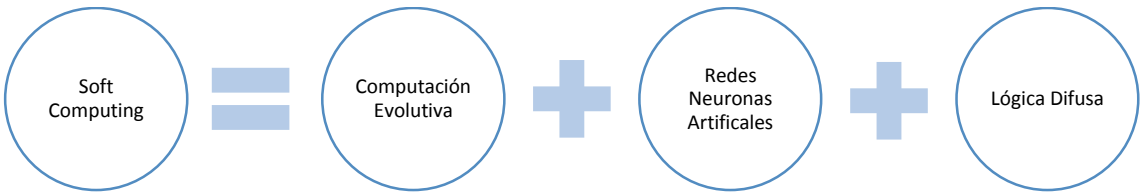

Figura 2 Composición del Soft Computing

2.6.2. Bases Biológicas.

Algoritmo Genético (GA) es un método de optimización que se basa principalmente en la teoría de la evolución de Darwin y el conocimiento actual de la genética. Como en la naturaleza, los algoritmos genéticos incluyen conceptos tales como cromosomas, genes, el apareamiento, cruce, mutación y evolución, entre otros.

Todos los organismos vivos consisten en células que incluyen los cromosomas. Un **cromosoma** está formado por millones de genes cada uno de los cuales codifica una característica específica del organismo, tales como el tipo de sangre o color de ojos en los seres humanos. Estas características se denominan **fenotipos**, por ejemplo, 0, A, B, AB para el tipo de sangre o verde, azul, café para el color de los ojos. Los ajustes particulares de genes se llaman **genotipo** y los valores de los genes son llamados alelos. El fenotipo de un organismo está formado por su genotipo. Los Alelos en término biológico son considerados el valor de la solución en GA.

En la naturaleza, la mayoría de las especies son diploides, es decir, que sus cromosomas están dispuestos en pares.

Los seres humanos también son diploides y tienen 23 pares de cromosomas en cada cuerpo celular. Durante el **cruce**, los genes de cada padre se intercambian entre cada par de cromosomas para formar un único cromosoma, es decir, un **gameto**. Entonces, un gameto de la madre y uno del padre se emparejan para crear un conjunto completo de cromosomas diploides para un hijo.

Esto explica cómo ciertos genotipos se heredan de una persona a su hijo. De vez en cuando, algunos valores de los genes cambian debido a efectos externos. Este tipo de cambio en el valor de un gen se denomina **mutación**. Las personas que están más adaptadas a las condiciones ambientales tienen mayor probabilidad de supervivencia. Esto también significa que los individuos y sus genes que están mejor adaptados lo más probable es que tiendan a permanecer mientras que aquellos que no están adaptados tiendan a extinguirse conforme pase el tiempo. Este proceso natural de la supervivencia del más fuerte se conoce como **evolución darwiniana**.

Durante las dos últimas décadas, los algoritmos genéticos se han utilizado ampliamente como métodos de búsqueda y optimización en diversos dominios de problemas pertenecientes a las ciencias, ingeniería y gestión. A pesar de que no garantizan encontrar soluciones óptimas, los GA son exitosos en la búsqueda de soluciones con alta una aceptación. Su amplia aplicabilidad y facilidad es la razón por la cual los investigadores eligen los algoritmos genéticos. El término cromosoma en algoritmo genético se refiere a un candidato de la solución para el problema dado. El genotipo se conoce como la solución codificada mientras que la solución que este representa se conoce como el fenotipo, es decir, la solución decodificada. El cruce consiste en el intercambio de material genético entre dos cromosomas individuales de los padres. La mutación consiste en reemplazar un gen elegido aleatoriamente con un nuevo gen también elegido aleatoriamente.

2.6.3. Codificación.

Desde los primeros trabajos de Holland la codificación suele hacerse mediante valores binarios. Holland señala en (Holland, 1975) la capacidad que tienen simples representaciones (en cadenas de bits) para codificar estructuras complicadas y mejorarlas. Holland demostró que con la estructura de control adecuada, podrían producirse mejoras rápidas de cadenas de bits. Una población de cadenas de bits "evoluciona", como lo hacen las poblaciones de animales. Para comenzar se asigna un determinado número de bits a cada parámetro y se realiza una discretización de la variable representada por cada gen. El número de bits asignados dependerá del grado de ajuste que se desee alcanzar. Evidentemente no todos los parámetros tienen por qué estar codificados con el mismo número de bits. Cada uno de los bits pertenecientes a un gen suele recibir el nombre de alelo.

2.6.4. Algoritmo Principal.

Los Algoritmos Genéticos trabajan sobre una población de individuos. Cada uno de ellos representa una posible solución al problema que se desea resolver.

Los pasos básicos de un algoritmo genético son:

1. Evaluar la puntuación de cada uno de los cromosomas generados.
2. Permitir la reproducción de los cromosomas siendo los más aptos los que tengan más probabilidad de reproducirse.
3. Con cierta probabilidad de mutación, mutar un gen del nuevo individuo generado.
4. Organizar la nueva población.

Estos pasos se repetirán hasta que se dé una condición de terminación. Se puede fijar un número máximo de iteraciones antes de finalizar el algoritmo genético o detenerlo cuando no se produzcan más cambios en la población (convergencia del algoritmo). Esta última opción suele ser la más habitual.

El funcionamiento de un Algoritmo Genético puede apreciarse en el diagrama:

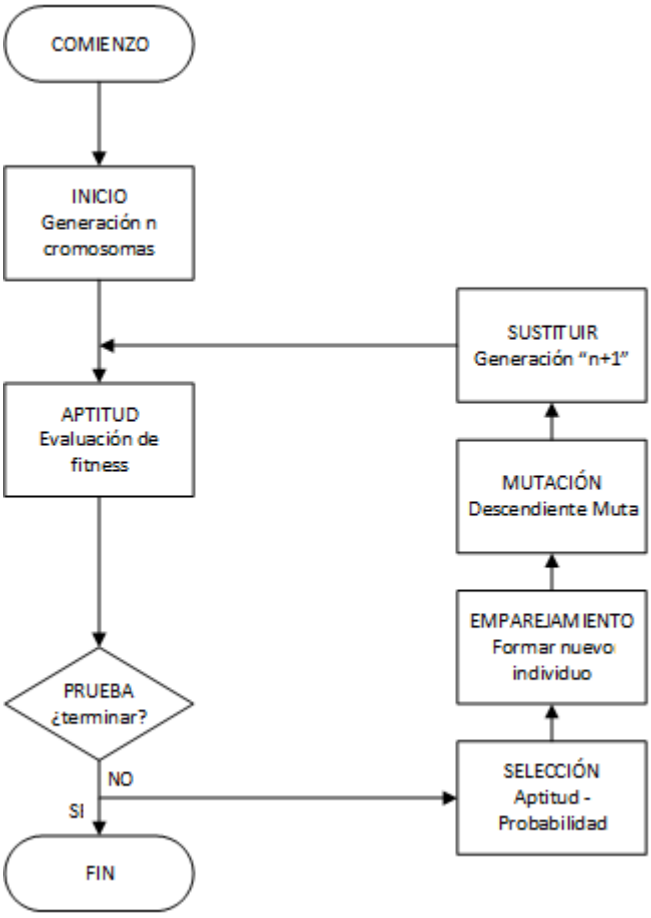

Figura 3 Funcionamiento de un Algoritmo Genético

2.6.5. Operadores de un Algoritmo Genético.

Un algoritmo genético debe tener los siguientes componentes para resolver un problema:

1. Una representación cromosomática de la solución al problema,
2. Una forma de crear una población inicial de soluciones,
3. Una función de evaluación que califique la calidad de solución según su "aptitud",
4. Los operadores genéticos que alteran la estructura de los "cromosomas hijos" durante la reproducción,
5. Los valores de los parámetros que utiliza el algoritmo genético (tamaño de la población, las probabilidades de aplicar operadores genéticos entre otros).

Para el paso de una generación a la siguiente se aplican una serie de operadores genéticos. Los más empleados son los operadores de selección, cruce, copia y mutación.

2.6.5.1. Cromosoma.

Un cromosoma es un conjunto de parámetros que definen una propuesta de solución al problema que el algoritmo genético está tratando de resolver. El cromosoma es a menudo representado como una cadena simple. La aptitud de un cromosoma depende de lo bien que el cromosoma resuelve el problema en cuestión.

2.6.5.2. Población Inicial

El primer paso en el funcionamiento de un GA es la generación de una población inicial. Cada miembro de esta población codifica una posible solución al problema. Después de crear la población inicial, cada individuo se evalúa y se le asigna un valor llamado *fitness* de acuerdo con la función de aptitud. Se sabe que si la población inicial para el GA es buena, entonces el algoritmo tiene mayor posibilidad de encontrar una buena solución y además si bloques iniciales construidos no son lo suficientemente grandes o lo suficientemente buenos entonces será difícil para el algoritmo encontrar una buena solución.

2.6.5.3. Evaluación

Consiste en asignar un valor de adecuación (fitness) a cada individuo en la población. Este valor evalúa qué tan bien resuelve cada individuo el problema en cuestión, y es utilizado para guiar el mecanismo genético.

2.6.5.4. Selección.

Es el proceso que determina candidatos adecuados, de acuerdo a sus valores de *fitness*, para la aplicación de los operadores evolutivos con el objetivo de engendrar la siguiente generación de individuos. Los tres procesos de selección más usados son:

a) **Selección por ruleta:** Los padres se seleccionan de acuerdo a su fitness. Los individuos mejores (con mayor fitness) son los que tienen mayores posibilidades de ser elegidos.
b) **Selección por torneo:** Se escogen de forma aleatoria un número de individuos de la población, y el que tiene puntuación mayor se reproduce, sustituyendo su descendencia al que tiene menor puntuación.
c) Selección por rango: En este método a cada cromosoma se le asigna un rango numérico basado en su aptitud y la selección se realiza en base a este ranking

2.6.5.5. Cruce.

En los GA, el cruce es un operador genético utilizado para variar la programación de un cromosoma o cromosomas de una generación a la siguiente. Es análogo a la reproducción y cruzamiento biológico, sobre el cual se basan los algoritmos genéticos. El cruce es un proceso que consiste en tomar más de una solución de los padres para producir una solución niño a partir de ellos (Mitchell, 1999). Entre las técnicas de cruce básicas tenemos:

a) Cruce de 1 punto

Es una de las formas clásicas de cruce. Los dos cromosomas padres se cortan por un punto. Luego se copia la información genética de uno de los padres desde el inicio hasta el punto de cruce y el resto se copia del otro progenitor. En la figura 5 se ilustra esta forma de cruce.

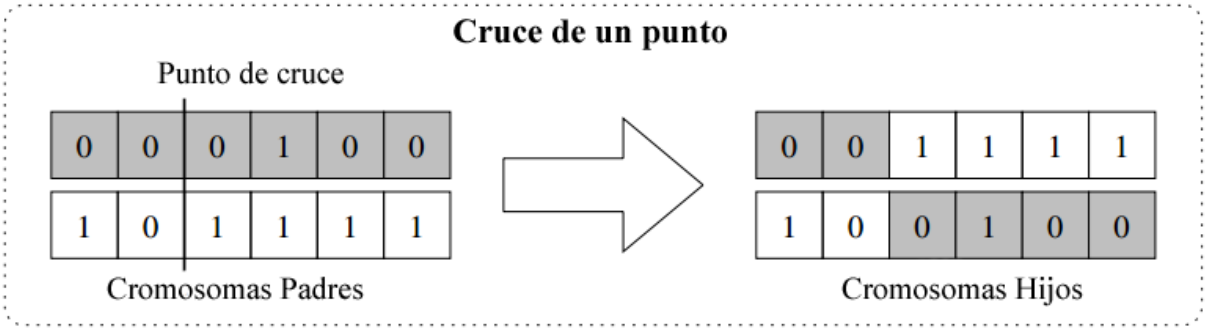

Figura 4 Cruce de un punto

b) Cruce de 2 puntos

Se trata de la misma filosofía que en el caso anterior pero en este caso los padres se cortan por dos puntos. Se copiará al descendiente los genes de un cromosoma progenitor desde el principio hasta el primer punto de cruce, los genes del otro progenitor desde el primer punto de cruce hasta el segundo y del segundo punto de cruce hasta el final se copiará del primer progenitor. Esto se puede apreciar en la Figura 6.

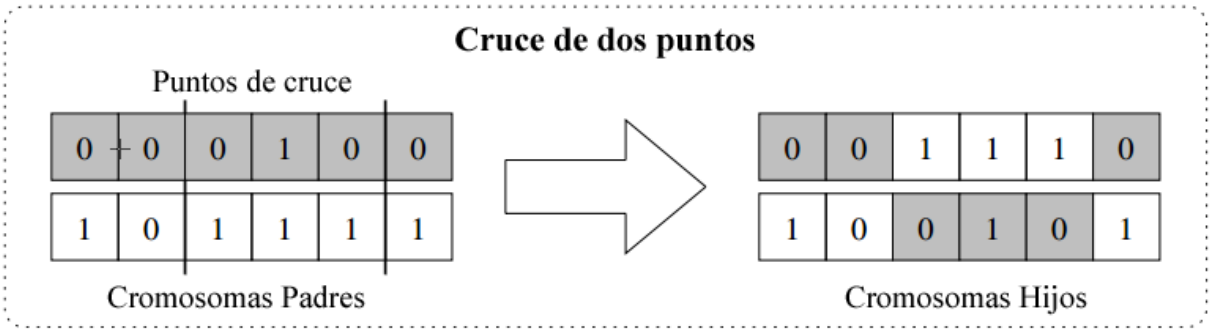

Figura 5 Cruce de dos puntos

c) Cruce uniforme

Cada gen del descendiente se obtiene de cualquiera de los padres de forma aleatoria. Una opción es generar un número aleatorio. Si este número supera un cierto umbral se elegirá un padre determinado y si no lo supera se elige al otro.

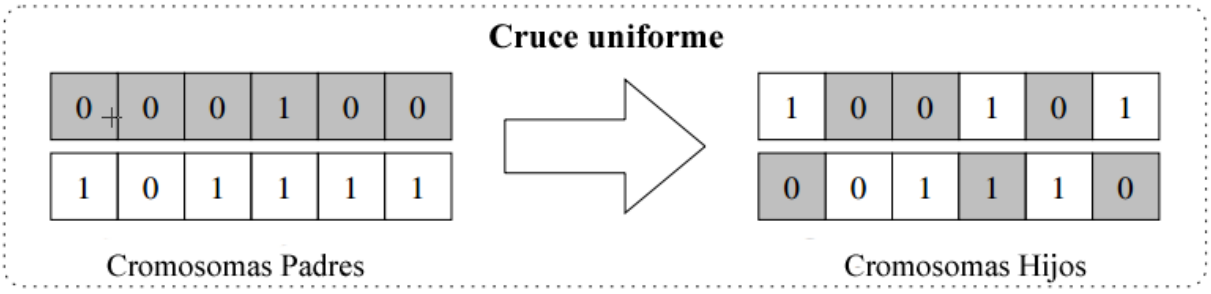

Figura 6 Cruce uniforme

2.6.5.6. Copia.

La copia es la otra estrategia reproductiva para la obtención de una nueva generación a partir de la anterior. A diferencia del cruce, se trata de una estrategia de reproducción asexual. Consiste simplemente en la copia de un individuo en la nueva generación.

El porcentaje de copias de una generación a la siguiente es relativamente reducido, pues en caso contrario se corre el riesgo de una convergencia prematura de la población hacia ese individuo. De esta manera el tamaño efectivo de la población se reduciría notablemente y la búsqueda en el espacio del problema se focalizaría en el entorno de ese individuo. Lo que generalmente se suele hacer es seleccionar dos individuos para el cruce y, si éste finalmente no tiene lugar, se insertan en la siguiente generación los individuos seleccionados.

2.6.5.7. Elitismo.

El elitismo es un caso particular del operador de copia consistente en copiar siempre al mejor, o en su caso los mejores, individuos de una generación en la generación siguiente. De esta manera se garantiza que el proceso de búsqueda nunca dará un paso atrás en cuanto

a la calidad de la mejor solución obtenida, sino que un cambio en ésta siempre implicará una mejora. Una variación de este proceso consiste en copiar al mejor o mejores individuos de una generación en la siguiente solamente cuando tras el paso de una generación no se haya mejorado con los operadores de cruce o mutación la mejor solución de la generación actual.

2.6.5.8. Mutación.

La mutación es un operador agenético que se utiliza para mantener la diversidad genética del GA de una generación de una población de cromosomas para la próxima. Es análogo a la mutación biológica. La mutación altera uno o más valores de genes en un cromosoma de su estado inicial. En mutación, la solución actual puede cambiar por completo de la solución anterior. De ahí que el GA puede llegar a una mejor solución mediante el uso de mutación. Este operador voltea al azar alguno de los bits en un cromosoma. Además de la mutación básica existen diferentes alternativas como la mutación estándar, la mutación sobre genes, la mutación no estacionaria, mutación no uniforme, etc., la actuación de este operador plantea los mismos problemas que el operador de cruce en relación a los descendientes obtenidos (Mitchell, 1999).

2.6.5.9. Función de Aptitud (*Fitness*)

La función de aptitud no es otra cosa que la función objetivo del GA y es la que mide la calidad de las soluciones generadas. La función de aptitud es siempre dependiente de un problema en particular, en los campos de la programación genética y algoritmos genéticos. Cada diseño de la solución se representa comúnmente como una serie de números que se refiere a un cromosoma. Después de cada ronda de pruebas, o simulación, la idea es eliminar el peor diseño (castigar) de las "*n"* soluciones y añadir otras "n" desde el mejor diseño de las soluciones. Cada solución de diseño, por lo tanto, tiene que ser" puesta con una figura de mérito (premiar), para indicar lo cerca que llegó al cumplimiento de la especificación general, y esto es generado por la aplicación de la función de aptitud a la prueba, o simulación, de los resultados obtenidos a partir de esa solución.

3. Metodología de la Investigación.

La revisión bibliográfica de los diferentes modelos y trabajos realizados por otros autores permitió establecer las similitudes y diferencias que poseían algunos con la situación en particular, por lo cual, se consideraron para el estudio los dos algoritmos más usados recientemente en la resolución de los TTP: Búsqueda Tabú y Algoritmos Genéticos, siendo este último el elegido para modelar la solución de nuestro problema en particular.

Para el desarrollo del algoritmo elegido se deberán definir los parámetros, las variables de decisión, restricciones y la integración de todos los elementos claves y críticos frente al problema de programación y asignación planteado.

Se debe evaluar la solución entregada en términos de su relación con el problema de asignación actual. Con ello será posible explorar las diferentes alternativas de solución que pueda entregar el modelo para posteriormente calibrar los diferentes parámetros que lo controlan y finalmente comprobar que aun habiendo realizado cambios en el modelo este sigue entregando la solución más óptima.

3.1. Estructura de los datos de entrada.

3.1.1. Descripción de los datos de entrada.

Los principales datos que influyen en la asignación docente-sala-asignatura-periodo son los siguientes:

- Docente.
    - Horario de los docentes: La disponibilidad de dictar cualquier asignatura de su preferencia en los respectivos periodos de clases.
    - Perfil académico del docente: La preferencia de las asignaturas que el docente desea dictar y que son de su competencia.

  - Carga reglamentaria: La cantidad máxima de horas disponibles de cada docente para dictar un conjunto de asignaturas.
- Salas de Clases.
  - Número de salas de clases: Espacios físicos destinados a la impartición de las clases.
  - Capacidad de las salas de clases: Cantidad máxima de alumnos permitidos para ocupar el espacio físico.
- Asignaturas.
  - Plan de estudios: Totalidad de asignaturas correspondiente a cada semestre incluyendo horas totales y su modalidad..
- Periodos.
  - Bloques de horario: Corresponde al periodo de tiempo donde son programadas cada una de las asignaturas en un horario definido.

3.1.2. Organización de los datos de entrada.

A continuación se muestran algunos datos que fueron obtenidos de la Escuela y que formarán parte de los recursos del modelo.

Tabla 3. Capacidad de Salas de Clases Edificio Brasil

#	Capacidad	Tipo
11	45	Sala de Clases
12	53	Sala de Clases
13	72	Sala de Clases
14	41	Sala de Clases
21	37	Aula Inteligente[4]
22	26	Aula Inteligente
23	37	Sala de Clases
24	51	Sala de Clases
31	60	Sala de Clases
32	50	Sala de Clases
33	36	Sala de Clases
34	50	Sala de Clases

[4] Aula Inteligente es usada para talleres o para ciertas ayudantías.

La tabla 3 muestra la capacidad de las salas de clases que son ocupadas para dictar cada una de las diferentes asignaturas. Se considera la capacidad como la cantidad de alumnos que puedes estar distribuidos en la sala ocupando cada uno un asiento.

Además de la capacidad de las salas de clases, es necesario conocer también la cantidad de ramos ofrecidos por el actual plan de estudios vigente, incluyendo de esta forma cátedras, talleres, ayudantías y laboratorios según se muestra a continuación,

Tabla 4. Distribución de Asignaturas para semestre 2016

Semestre	# Asignaturas
1	5
2	5
3	6
4	6
5	6
6	6
7	6
8	6
9	7
10	7
11	3
	63

En la siguiente tabla se muestra la distribución de los diferentes periodos de clases disponibles para la programación actual de horarios en una semana.

Tabla 5. Distribución de periodos de clases por bloque horario semanal.

Inicio	Termino	Lunes	Martes	Miércoles	Jueves	Viernes
8:30	**10:00**	Periodo 1	Periodo 1	Periodo 1	Periodo 1	Periodo 1
10:15	**11:45**	Periodo 2	Periodo 2	Periodo 2	Periodo 2	Periodo 2
12:00	**13:30**	Periodo 3	Periodo 3	Periodo 3	Periodo 3	Periodo 3
13:45	**15:15**	Periodo 4	Periodo 4	Periodo 4	Periodo 4	Periodo 4
15:30	**17:00**	Periodo 5	Periodo 5	Periodo 5	Periodo 5	Periodo 5
17:15	**18:45**	Periodo 6	Periodo 6	Periodo 6	Periodo 6	Periodo 6
19:00	**20:30**	Periodo 7	Periodo 7	Periodo 7	Periodo 7	Periodo 7

3.2. Diseño del Modelo de Aplicación.

3.2.1. Parámetros.

Se definen los siguientes parámetros que corresponden a cada recurso involucrado en el modelo:

r = 35, cantidad de periodos de tiempo semanales.

u = 12, cantidad salas de clases.

w = 63, cantidad de asignaturas.

y = 36, cantidad de docentes.

3.2.2. Conjunto de datos.

Los siguientes conjuntos de datos forman parte de la creación del Modelo General del problema, para luego poder establecer el Algoritmo Genético en el caso particular de TTP en la Escuela de Ingeniería Industrial.

1. Se define $\boldsymbol{s} = \{1, \dots, 11\}$ como el conjunto de los semestres académicos a los cuales está asociada cada asignatura.

2. Se define $\boldsymbol{a} = \{1, \dots, 63\}$ como el conjunto que contiene la totalidad de asignaturas de la malla académica del plan de estudios vigente de la carrera.

3. Se define $\boldsymbol{m} = \{1, \dots, 4\}$ como el conjunto que indica la modalidad en que será dictada cada asignatura. Dos tipos de modalidad diferente no pueden tener tope de horario para la misma asignatura en el semestre en cuestión. Las modalidades de cada asignatura son catedra, taller, ayudantía y laboratorio respectivamente.

4. Se define $\boldsymbol{b} = \{1, \dots, 3\}$ como el conjunto de las secciones disponibles de las asignaturas para cada semestre.
5. Se define $\boldsymbol{h} = \{1, \dots, 3\}$ como el conjunto de la cantidad de horas semanales que se le asigna a cada una de las modalidades de las diferentes asignaturas. La cantidad de horas va en el orden de 1.5, 3.0 y 4.5.

6. Se define $\boldsymbol{t} = \{1, \dots, 35\}$ como el conjunto de periodos de tiempo en el que se puede dictar una asignatura cualquiera.

7. Se define $\boldsymbol{d} = \{1, \dots, 5\}$ el conjunto de los días de la semana que agrupa los diferentes bloques de horario. Cada elemento del vector antes nombrado corresponde a los días disponibles para realizar las clases, es decir, lunes, martes, miércoles, jueves y viernes respectivamente.

8. Se define $\boldsymbol{p} = \{1, \dots, 36\}$ como el conjunto de docentes que puede dictar una asignatura cualquiera o de su preferencia.

9. Se define $\boldsymbol{c} = \{1, \dots, 12\}$ como el conjunto de salas de clases donde se puede dictar una asignatura cualquiera.

10. Se define $\boldsymbol{e} = \{1, \dots, 12\}$ como el conjunto de la capacidad de las diferentes salas de clases.

11. Se define $\boldsymbol{pp}$ como la matriz de docentes p que dictan la asignatura c correspondiente a la preferencia de las asignaturas de cada uno.

12. Se define $\boldsymbol{dp}$ como la matriz de periodo de tiempo t correspondiente a la disponibilidad horaria del docente para dictar una asignatura de su preferencia.

3.2.3. Variable de decisión.

Se define $X_{i,j,k,l}$ como una variable binaria con valor 1, si asignan una sala $\boldsymbol{i}$, a la asignatura $\boldsymbol{j}$, el docente $\boldsymbol{k}$ en el periodo $\boldsymbol{l}$.

1 si la sala i es asignada a la asignatura j con el docente k en el periodo l.

0, si no, es decir $X_{i,j,k,l} = 0$

$$\forall\, i \in = \{1, \dots, 12\}, \qquad \forall\, j \in = \{1, \dots, 63\}, \qquad \forall\, k \in = \{1, \dots, 35\}, \qquad \forall\, l \in = \{1, \dots, 35\}$$

3.2.4. Modelamiento de Restricciones

Para el modelamiento de restricciones se considera la sala $\boldsymbol{i}$, asignatura $\boldsymbol{j}$, docente $\boldsymbol{k}$ y periodo $\boldsymbol{l}$ como índices.

R1 Toda asignatura j debe tener una cantidad máxima de $\boldsymbol{q}$[5] alumnos matriculados que no sobrepase la capacidad $\boldsymbol{e}$ de la sala i

$$q \cdot X_{i,j,k,l} <= \ e_i$$

$$\forall\, X_{i,j,k,l} \in (0,1), \qquad \forall\, i \in c, \qquad \forall\, j \in a, \qquad \forall\, k \in p, \qquad \forall\, l \in t$$

R2 Toda sala i en un periodo específico l debe tener asignado a lo más una asignatura j y un docente k

$$\sum_{j=1}^{63} \sum_{k=1}^{36} X_{i,j,k,l} \leq 1$$

$$\forall\, i \in c, \qquad \forall\, l \in t$$

[5] La cantidad máxima de alumnos permitidos por sección es de 45 estudiantes.

R3 Todo docente k debe tener asignado a lo más una asignatura j que sea de su preferencia.

$$\sum_{i=1}^{12}\sum_{l=1}^{35} X_{i,j,k,l} = pp_{jk}$$

$$\forall\, j \in a, \qquad \forall\, k \in p$$

R4 Todo docente k debe tener asignada una asignatura dentro de su disponibilidad de tiempo l.

$$\sum_{i=1}^{12}\sum_{j=1}^{63} X_{i,j,k,l} = dp_{kl}$$

$$\forall\, k \in p, \qquad \forall\, l \in t$$

R5 Toda asignatura j que se dicta en un periodo específico l debe tener asignado a lo más un docente k en la sala i.

$$\sum_{i=1}^{12}\sum_{k=1}^{36} X_{i,j,k,l} \leq 1$$

$$\forall\, j \in a, \qquad \forall\, l \in t$$

R6 Todo docente k que dicta en un periodo específico l debe tener asignado a lo más una asignatura j que en la sala i.

$$\sum_{i=1}^{12}\sum_{j=1}^{63} X_{i,j,k,l} \leq 1$$

$$\forall\, k \in p, \qquad \forall\, l \in t$$

R7 Toda asignatura j en sus diferentes modalidades debe cumplir con una cantidad de horas semanales h

$$\sum_{i=1}^{12}\sum_{k=1}^{36}\sum_{l=1}^{35} X_{i,j,k,l} = h_j$$

$$\forall\, j \in a$$

3.2.5. Función Objetivo.

$$f_{(o)} = \frac{g - g_c}{g}$$

Donde:

g: corresponde al total de actividades[6].

g_c: corresponde al número de actividades que presenta algún tipo de conflicto.

En donde un valor igual a 1 representará que se ha encontrado una solución óptima.

El principal propósito de nuestra función es minimizar el número de conflictos de asignación en las distintas actividades.

En nuestro caso, la función objetivo castiga aquellas malas asignaciones haciendo que estas vuelvas a entrar en la iteración hasta conseguir una buena solución (según el sistema) y premia aquellas buenas asignaciones permitiéndoles pasar de una generación a la otra.

[6] Ver Tabla 6, Página 56.

4. Implementación del Algoritmo Genético.

4.1.1. Población Inicial y cromosomas.

La población inicial se puede generar usando una solución existente del problema (antecedente de horario histórico), aunque en nuestro caso la población inicial será generada de manera aleatoria. Es importante señalar que el uso de una población al azar puede generar que la primera generación sea de muy baja calidad y provoque algún impacto en las generaciones futuras.

Cada población se compone de un número de individuos, o cromosomas. Aunque un individuo puede tener más de un cromosoma, en nuestro enfoque cada individuo tiene un solo cromosoma. Un cromosoma se compone de un conjunto de genes (que es la información más pequeña que lleva un cromosoma). Para nuestro caso, al interior del cromosoma, hay un gen para cada actividad en el horario. De ahora en adelante se considerará una **actividad** como la **combinación de un docente, con su asignatura, modalidad y una sección**, además de una duración especifica de periodos. Cada una de estas actividades se programa según un día de inicio y periodos que ocupa dentro de ese día.

A modo de ejemplo se muestra en la Figura 7 de la representación binaria de un cromosoma el cual corresponde a la actividad con ID 15 con inicio el día miércoles (3) en el periodo 4

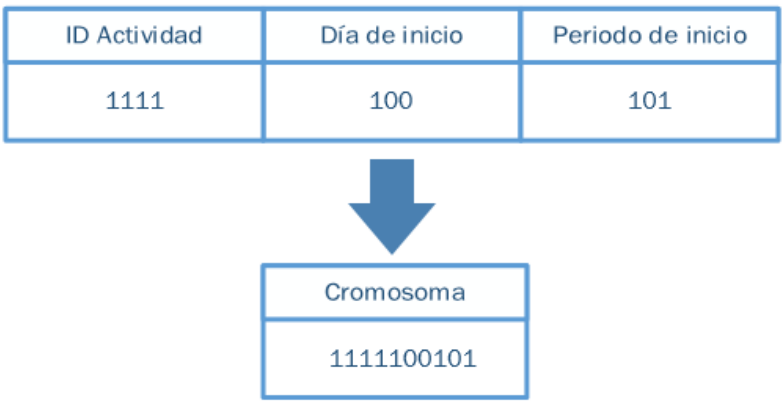

Figura 7 Ejemplo de Cromosoma

4.1.2. Selección, cruce y mutación.

El método evolutivo es la función principal de la población. La estrategia para la generación de una nueva población a partir de una antigua se define de aquí en adelante mediante el uso de la función de mutación, cruce o simple propagación de viejos cromosomas. Una nueva población se va a generar de forma iterativa (los cromosomas se generan uno por uno a partir de los antiguos).

Tomando en consideración la bibliografía estudiada el método de selección que utilizaremos en nuestra la aplicación es el *Three tournament selection* (Selección por torneo de tres) el cual selecciona un cromosoma candidato a cruce, mutación o propagación eligiendo a tres cromosomas al azar, cada uno con probabilidades iguales, a partir de la población inicial. Si se necesita una propagación (conservación) o mutación de un cromosoma, el mejor de los tres es elegido para estas operaciones. Si la operación considerada fuera el cruce, el programa elige automáticamente los dos mejores candidatos. Las razones de elegir este método de selección son porque mantiene la diversidad de la población y además guía a la solución a converger en el óptimo global. El cruce tiene un papel menor en la evolución para nuestro caso, siendo la mutación más importante. Esto es porque el método de mutación utilizado introduce un alto grado de aleatorización dentro de los cromosomas.

4.1.3. Principales categorías de la aplicación.

La creación de nuestra aplicación se llevó a cabo mediante un programa orientado al lenguaje C ++ tomando como base el programa de horarios de uso libre *FET*[7][8] de licencia GNU AGPL v3[9]. *FET* utiliza un algoritmo llamado "*Clever Algorithm" (algoritmo inteligente).* Es por ello que se utilizó como base ciertas partes del código fuente de la aplicación para así mediante las respectivas modificaciones crear un programa personalizado para nuestro caso con fuerte fundamento en la programación genética, más específico, en los Algoritmos Genéticos. Algunas de las categorías más importantes de nuestro programa y sus funciones son:

[7] Free Evolutionary Timetabling.
[8] Disponible en http://lalescu.ro/liviu/fet/
[9] http://www.gnu.org/licenses/agpl-3.0.html

- HGenético: la categoría principal, que encapsula un conjunto de reglas y una población. Tiene todas las funciones de interfaz necesarias para leer/editar/guardar reglas, el inicio/término de la simulación y también para guardar/ver los resultados.
- Reglas: el conjunto de normas que conforman el horario. Incluye el conjunto de los docentes, las asignaturas (clases lectivas) que son impartidas, el conjunto de cursos de los estudiantes (organizadas jerárquicamente en semestre, curso y sección), las actividades y el conjunto de restricciones obligatorias y flexibles.
- Población: representa un conjunto de cromosomas y tiene diferentes funciones para la aplicación de los métodos de evolución presentados.
- Cromosoma: representa un candidato a solución. Su interfaz incluye las funciones de mutación y de cruce, la función de evaluación de la aptitud y la función para la obtención de una programación según información almacenada en los genes.

El diagrama de clases que refleja de manera experimental el programa se muestra en la Figura 8.

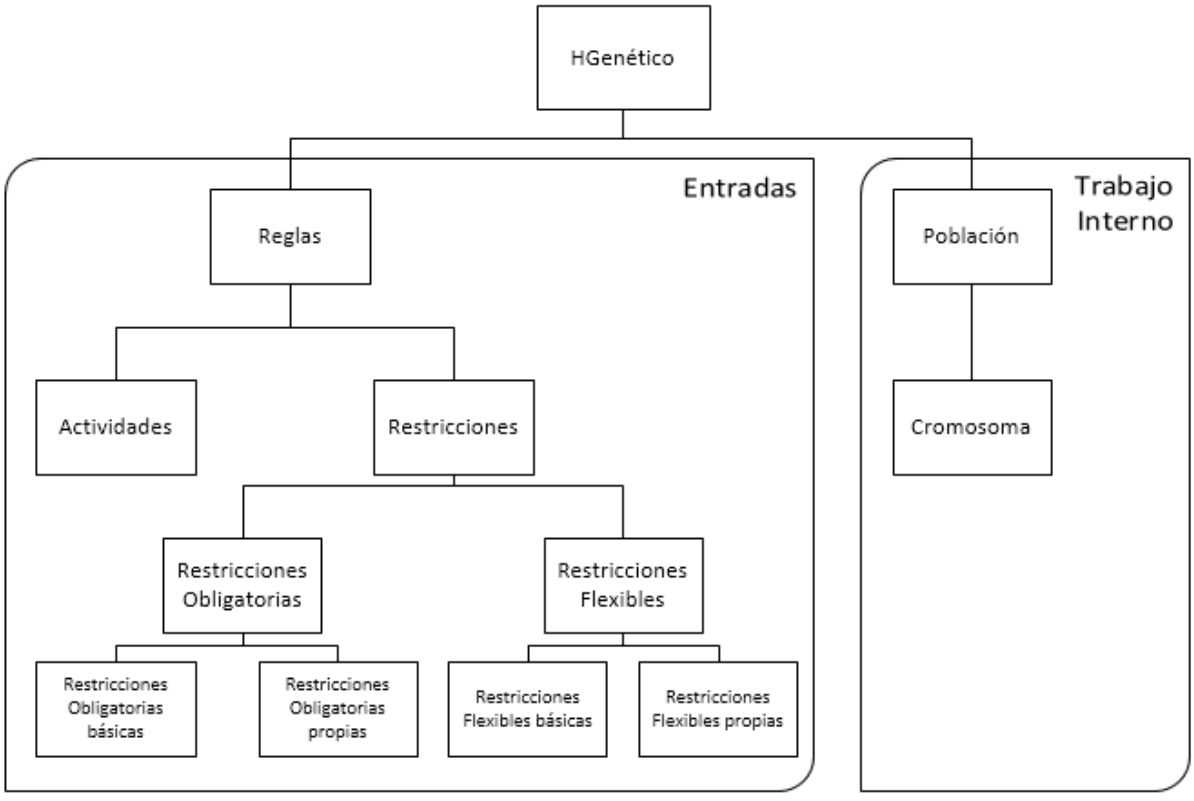

Figura 8 Diagrama de clases del Programa

El diseño del programa se realizó con los siguientes objetivos: la facilidad de uso y flexibilidad para la experimentación. También fue diseñado para permitir la fácil adición de restricciones más personalizadas (ya sean obligatorias o flexibles).

El usuario dispone de la posibilidad de asignar manualmente algunas de las actividades, permitiendo de esta forma al programa asignar automáticamente sólo las actividades restantes. Esto con el propósito, si es que el usuario lo desea, de guiar a la aplicación a una mejor solución. Esta característica apunta a lo que se denomina un sistema "semi-automático".

Esta característica parece tener un escaso impacto en el tiempo de ejecución para obtener un horario. Los resultados parecen no mejorar mediante el uso de la instalación semiautomática. Esto se debe principalmente al reciente aumento de la potencia de cálculo de los computadores para realizar todos los cálculos necesarios en un tiempo aceptable, con o sin la ayuda del usuario.

4.1.4. Entorno de Desarrollo.

La aplicación fue programada en el lenguaje C++ a través de la plataforma QT[10] bajo la versión 4.8.6[11] para Windows y compilada en conjunto con MinGW[12]. Las pruebas de desarrollo y compilación fueron realizadas en un equipo con las siguientes características:

- Procesador: Intel® Core™ i5-2430M CPU 2.40GHz.
- Memoria RAM: 4GB.
- Sistema Operativo: Microsoft Windows 7 Professional x64.

La razón de utilizar la plataforma QT es porque ofrece una interfaz bastante amigable, accesible para todos los usuarios (software libre y de código abierto), tiene estabilidad y sobretodo es más ligera que otras plataformas de este tipo.

[10] http://www.qt.io/
[11] Disponible en https://download.qt.io/archive/qt/4.8/4.8.6/
[12] http://www.mingw.org/

4.1.5. Resultados de prueba experimental.

En este apartado se mostrarán las capturas más significativas de la aplicación y resultados de una prueba experimental. Un mejor detalle del ingreso de datos y configuraciones se puede ver en el Anexo 2.

La aplicación se denominó con las siglas SASPHEII, lo cual quiere decir “Sistema de Asignación de Salas y Programación Horaria Escuela de Ingeniería Industrial UV”.

En la Figura 9, se muestra la ventana principal de la aplicación la cual contiene principalmente los menús de: Archivo, Datos, Población y Horario.

Figura 9 Ventana principal aplicación SASPHEII

Después del ingreso de los datos básicos de entrada se debe realizar la programación de las actividades. Es aquí en donde se forma el conjunto docente-asignatura-sección además de su duración determinada.

La vista previa de la ventana en donde se realiza en ingreso respectivo de las actividades así también como sus modificaciones es la que se muestra en la figura 10.

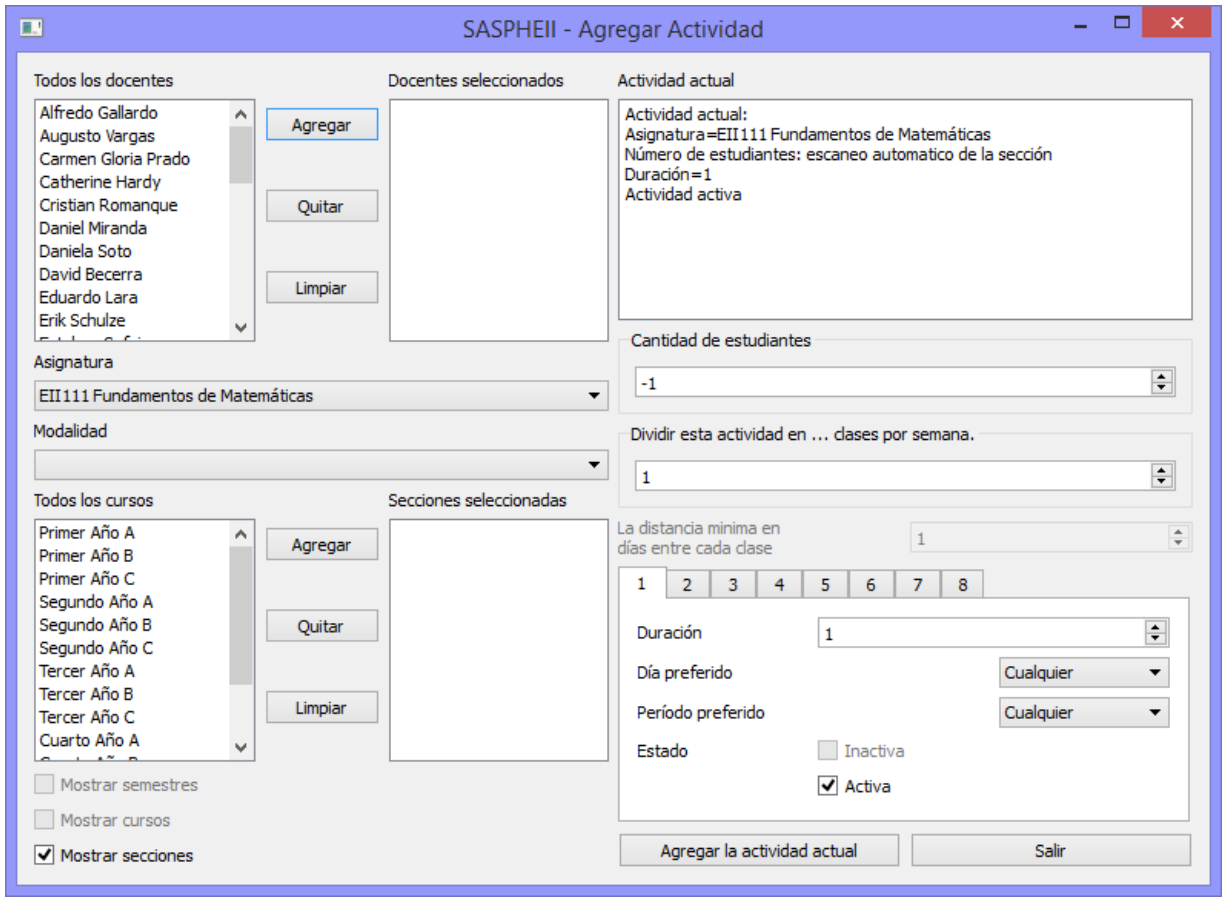

Figura 10 Ventana de Actividades

En las pruebas experimentales se consideraron:

Tabla 6 Valores pruebas experimentales

Periodos por día	7
Días de trabajo	5
Docentes	34
Asignaturas	30
Modalidades	2
Secciones	14
Salas de clase	12
Nº Actividades *(g)*	157

El resultado de haber ingresado la totalidad de las actividades correspondiente al semestre a programar es el mostrado a continuación en la Figura 11.

A Cada bloque de horario para cualquier asignatura en cualquier día se le va asignando un identificador de actividad y para cada grupo de actividad (bloques de una misma asignatura en una sola sección) también se le fue asignado un identificador. Es muy importante que en este paso queda registrado el número de estudiantes que componen cada actividad.

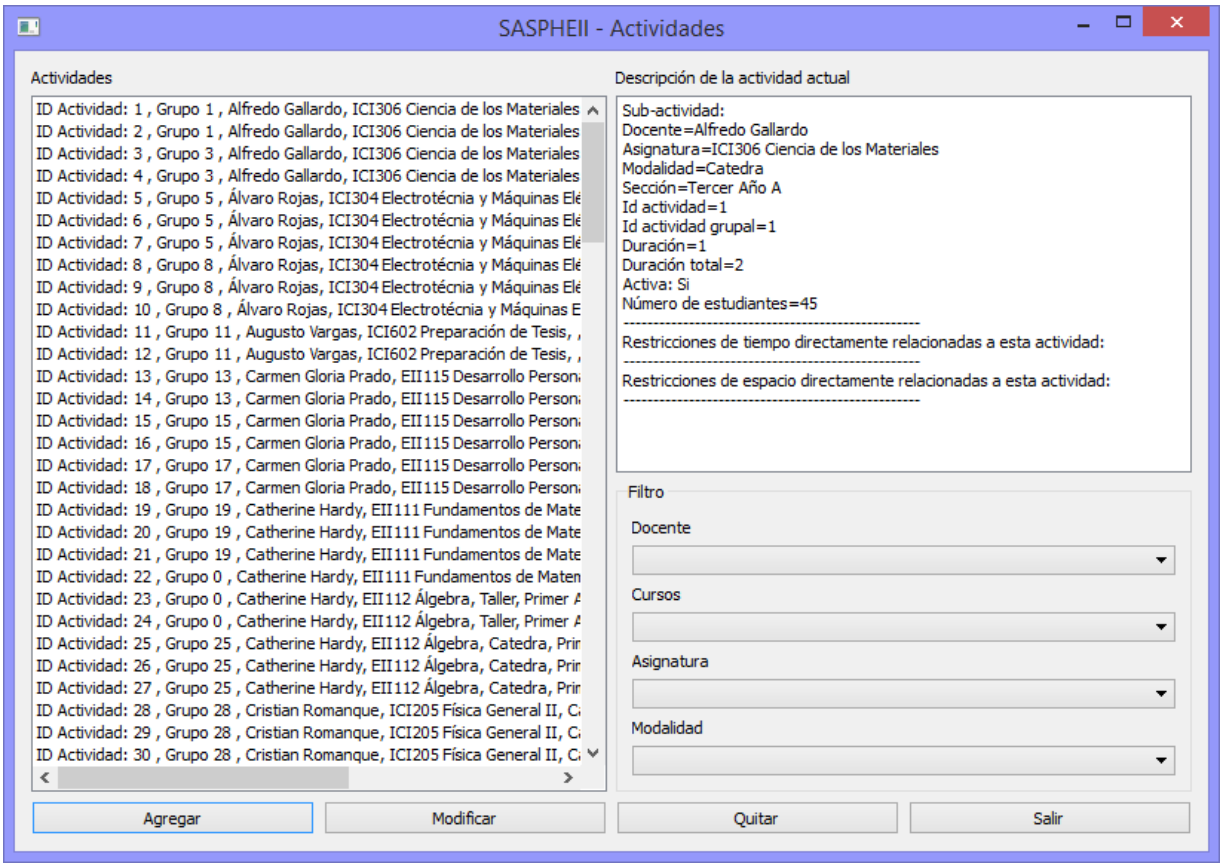

Figura 11 Vista previa cuadro Actividades

Para el ingreso de las diferentes restricciones que ofrece la aplicación cada una de ellas tiene asignada una *ponderación* (o peso según la literatura estudiada) la cual tiene una puntuación máxima de 1. Un valor inferior a este significa que la restricción “debería” respetarse, pero no necesariamente cumplirse.

En la siguiente tabla se aprecia la cantidad de reintentos que ejecuta la aplicación para algunos valores asignados según la ponderación de una restricción:

Tabla 7 Ponderación de Restricciones

Ponderación	Número de reintentos
1	Ilimitados
0.995	200
0.99	199
0.95	20
0.9	10
0.8	5
0.75	4
0.5	2

El número de reintentos corresponde a las veces en que la aplicación tratará de asignar una actividad sin que presente conflictos, una vez superado el número de reintentos posibles la aplicación mantiene el conflicto para esa actividad y se salta automáticamente a la siguiente. Es importante que se asigne **siempre** a las restricciones obligatorias una ponderación de 1.

En la Figura 12 y Figura 13 se muestra la configuración básica de las restricciones de tiempo y espacio respectivamente. Estas tienen un carácter obligatorio y siempre deben estar presentes en cada una de las asignaciones que se deseen realizar.

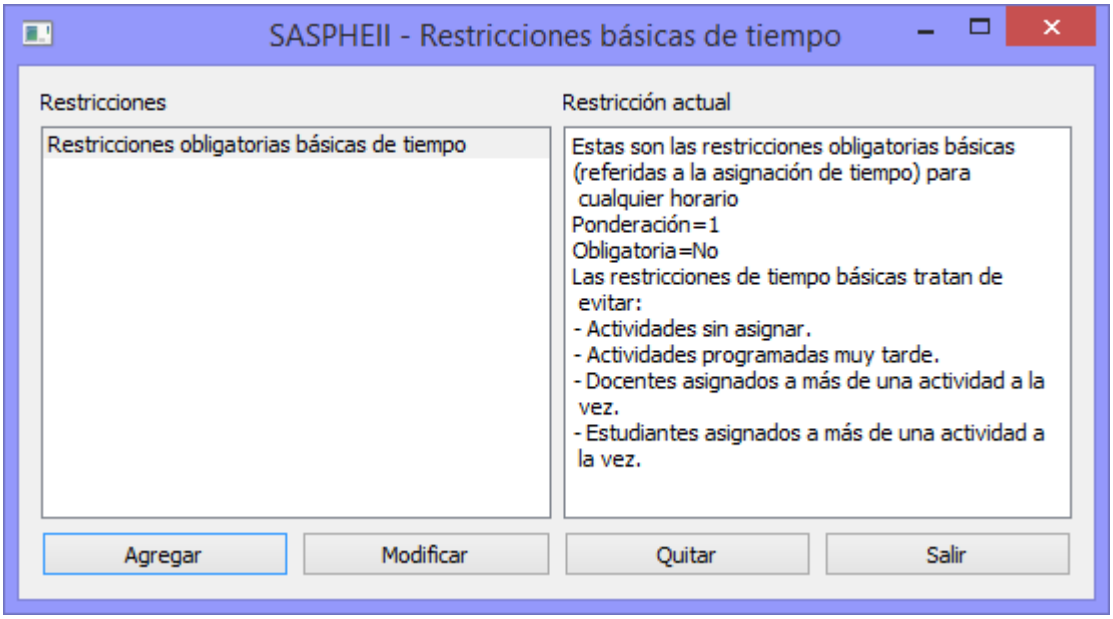

Figura 12 Ventana de Restricciones básicas de tiempo

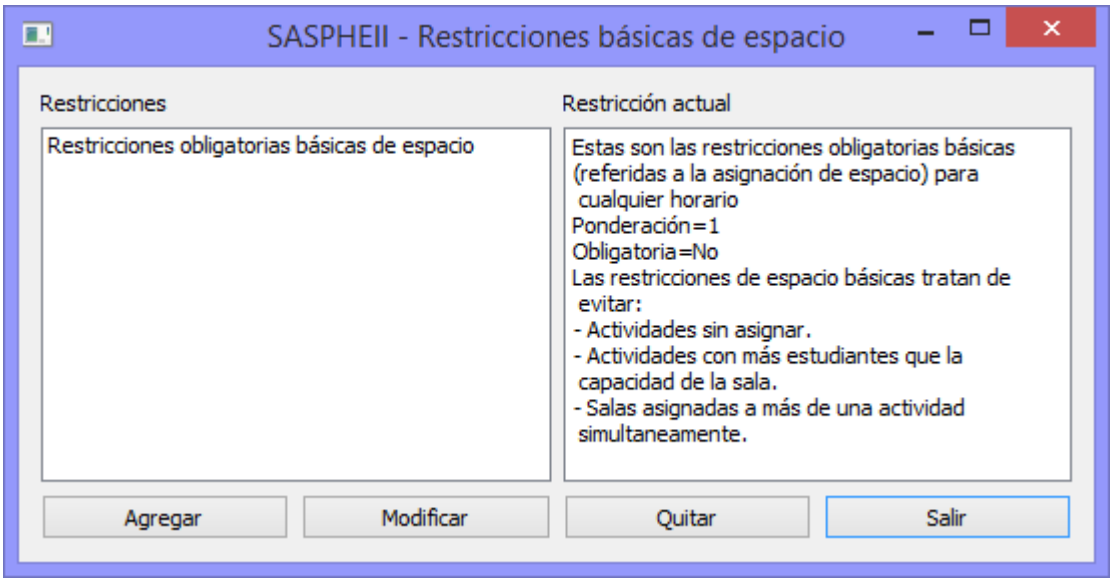

Figura 13 Ventana de Restricciones básicas de espacio

El cuadro que se muestra abajo forma parte de una de las restricciones obligatorias que han sido impuestas por la institución: La disponibilidad del docente. En ella es posible configurar los días y bloques en que un docente puede o no dictar alguna asignatura. Recordar que esta es una de las restricciones importantes de nuestro caso de estudio y es por ello que se tuvo que ajustar a carácter obligatorio con una ponderación de 1.

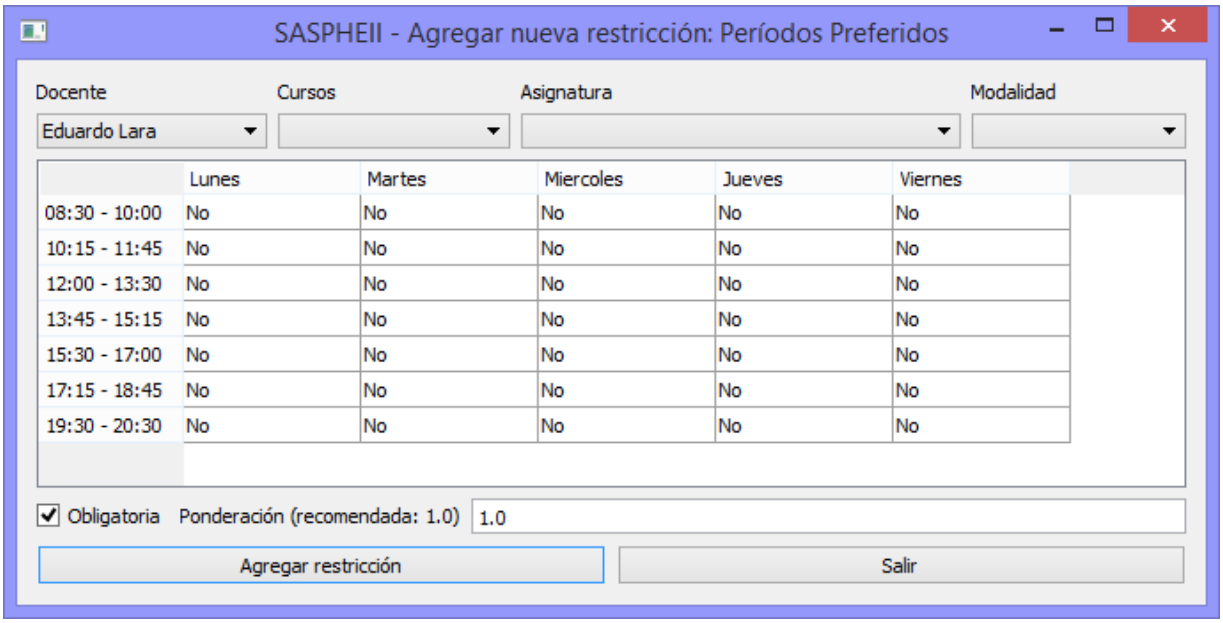

Figura 14 Ventana de Disponibilidades

Luego de ingresar la totalidad de restricciones es necesario ahora definir el tamaño de la población en donde se tiene la libertad de optar por la velocidad o el rendimiento. La velocidad se refiere a que las soluciones serán entregadas en un lapso de tiempo menor pero no siempre será la mejor y el rendimiento indica que la aplicación puede demorar en entregar una solución pero el resultado será cercano al óptimo.

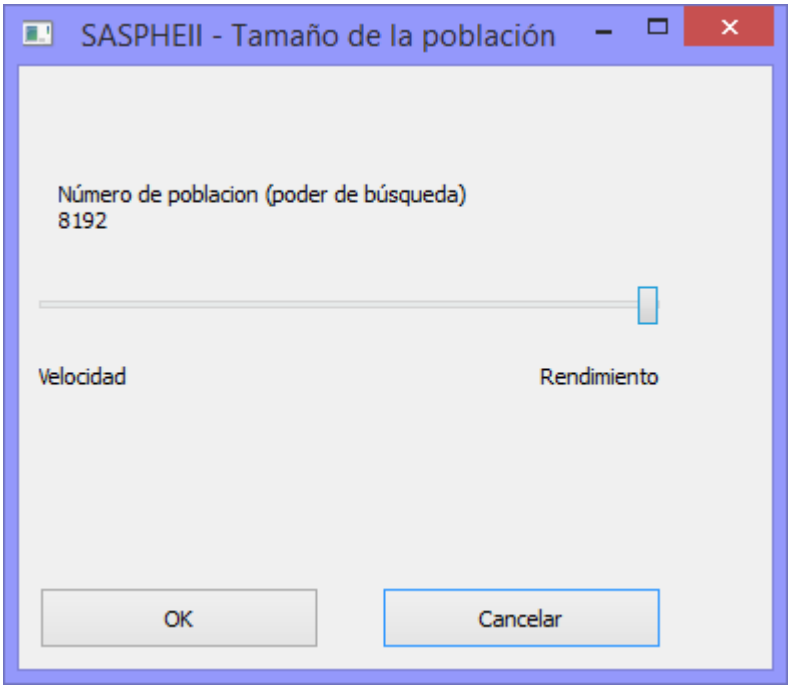

Figura 15 Cuadro configuración de la población

Finalmente luego de haber ingresado las restricciones necesarias se deben realizar ciertas configuraciones vitales para la ejecución del programa. Para ello se pone a disposición del usuario el archivo “saspheii.ini” en el cual puede ajustar los valores a su conveniencia.

Los valores de configuración disponibles y los que fueron utilizados para nuestra prueba son los siguientes (predeterminados por los autores y la aplicación original de FET):

Tiempo límite de búsqueda	1800 (segundos)
Tamaño de población	8192
Método de Selección	Basada en un torneo de 3
Probabilidad de Mutación (intercambio al azar)	70%
Probabilidad de cruce	20%
Probabilidad de Propagación	10%

Con esta configuración se procede a realizar la simulación del programa. La Ventana de simulación es la que se muestra en la Figura 16.

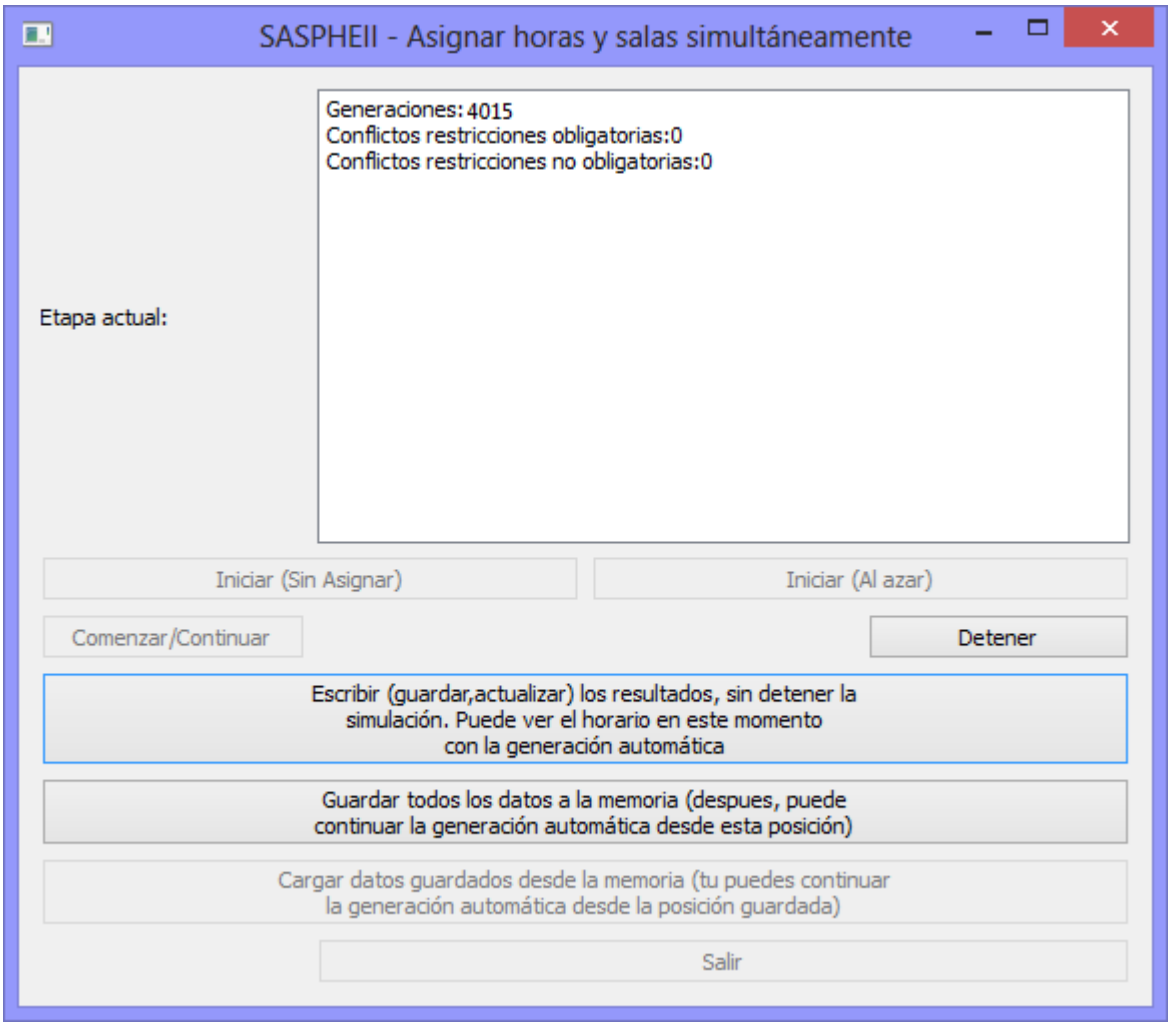

Figura 16 Cuadro de simulación del Algoritmo

En esta ventana se observa el número de generaciones que le toma a la aplicación hasta llegar a la solución final, el número de restricciones obligatorias que presentan conflictos y el número de restricciones flexibles que presentan conflictos.

Los resultados finales se dieron en un tiempo de 10 minutos, con 0 conflictos en restricciones obligatorias y no obligatorias (restricciones flexibles) y un total de 4015 generaciones.

El resultado gráfico a modo de ejemplo de la Sección A de los alumnos de primer año se muestra en la figura 17.

Escuela de Ingenieria Industrial UV

Primer Año A

	Lunes	Martes	Miercoles	Jueves	Viernes
08:30 - 10:00	Máximo Pérez EII114 Informática I Taller Sala 33	Carmen Gloria Prado EII115 Desarrollo Personal I Catedra Sala 11	Manuel Ramos EII113 Introducción a la Ingeniería Catedra Sala 31		Giglia Gómez EII114 Informática I Catedra Sala 11
10:15 - 11:45	Patricio Suzarte EII112 Álgebra Catedra Sala 23	Patricio Suzarte EII112 Álgebra Catedra Sala 13	Manuel Ramos EII113 Introducción a la Ingeniería Catedra Sala 31	Catherine Hardy EII111 Fundamentos de Matemáticas Catedra Sala 23	Catherine Hardy EII111 Fundamentos de Matemáticas Catedra Sala 11
12:00 - 13:30	Lorena Quinteros ICI105 Historia Contemporánea Catedra Sala 12	Patricio Suzarte EII112 Álgebra Catedra Sala 13	Lilian Arancibia EII111 Fundamentos de Matemáticas Taller Sala 11	Giglia Gómez EII114 Informática I Catedra Sala 32	Catherine Hardy EII111 Fundamentos de Matemáticas Catedra Sala 11
13:45 - 15:15	Lorena Quinteros ICI105 Historia Contemporánea Catedra Sala 12		Carmen Gloria Prado EII115 Desarrollo Personal I Catedra Sala 31	Catherine Hardy EII112 Álgebra Taller Sala 11	
15:30 - 17:00					
17:15 - 18:45					
19:30 - 20:30					

Figura 17 Ejemplo de Horario generado para Primer Año A

Estos resultados están en formato HTML e incluye tres archivos generados: el horario de los estudiantes (con sus salas), el horario de los docentes (con sus salas) y el horario propio de las salas de clases.

5. Conclusión.

En general, todos los métodos de búsqueda metaheurística son guiados por un método de decisión aleatoria. El algoritmo genético es un ejemplo de estos métodos. Las características principales del algoritmo genético son la búsqueda aleatoria de una solución, las operaciones de mutación, de cruce y también la evolución de las soluciones basadas en el valor del *fitness* o aptitud.

En este trabajo de título, nuestro problema se centró en la programación de horarios y asignación de salas en la Escuela de Ingeniería Industrial de la Universidad de Valparaíso cuya solución se puede obtener con un Algoritmo Genético lo cual fue comprobado a lo largo de los diferentes capítulos. A pesar de que los Algoritmos Genéticos son un método bastante antiguo, existe una amplia documentación que soporta el uso de este método cómo fue posible apreciar en la bibliografía estudiada. El problema consistía en la programación de una serie de asignaturas con sus respectivas modalidades en periodos de tiempo y un conjunto de salas de clases sin violar restricciones obligatorias y minimizar en lo posible el incumplimiento de las flexibles. Las restricciones obligatorias deben cumplirse con el fin de obtener una solución factible. Hemos abordado este problema mediante la introducción de operadores genéticos con la capacidad de ser modificadas sus probabilidades.

Un problema presente con el uso de los Algoritmos Genéticos es la determinación de los mejores valores de cada uno de los parámetros de evolución, tales como tamaño de población, probabilidades de mutación y cruce, método de selección, etc. Esto es difícil de determinar teóricamente, es por ello que se hicieron unas pruebas cambiando estos parámetros y mostrando a modo de resultado el mejor de ellos. Los operadores de cruce y mutación son necesarios para evitar que la búsqueda de soluciones se quede atrapada en mínimos locales. A valores de probabilidad más altos para estos operadores se aumenta la eficiencia del algoritmo. También fue posible darse cuenta de que el aumento del tamaño de la población aumenta la convergencia del algoritmo con respecto a la número de las generaciones pero eso si sacrificando un poco de rendimiento del programa (tiempo).

La aplicación entregada asegura que los horarios generados cumplen con las restricciones impuestas y entrega la flexibilidad de poder modificaciones en cualquier momento.

6. Bibliografía

Abramson, D. (1991). Constructing school timetables using simulated annealing: sequential and parallel algorithms. En *Management Science* (págs. 98-113).

Alvarez-Valdes, R., Crespo, E., & Tamarit, J. (2000). Assigning students sections using tabu search. Annals of Operations Research.

Alvarez-Valdes, R., Crespo, E., & Tamarit, J. (2002). Design and implementation of a course scheduling systems using tabu search: Production, manufacturing and logistics. En *European Journal of Operational Research* (págs. 512-523).

Arbib, M. (2003). *Handbook of brain theory and neural networks.* Cambridge: MIT Press.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms.* USA: Oxford University Press.

Brailsford, S., Potts, C., & Smith, B. (1999). "Constraint Satisfaction Problems: Algorithms and Applications". En *European Journal of Operational Research* (págs. 557-581).

Burke, E., Bykov, Y., & Petrovic, S. (2001). Multi-criteria Approach to examination timetabling. En *Practice and Theory of Automated Timetabling: Third International Conference* (págs. 118-131). Berlin: Springer-Verlag.

Burke, E., Kendall, G., Newall, J., Hart, E., Ross, P., & Schulenberg, S. (2003). Hyper-heuristic: An Emerging Direction in Modern Search Technology. En *Handbook of Meta-Heuristics* (págs. 457–474). Kluwer.

Carter, M. (1986). A Survey of Practical Applications of Examination Timetabling Algorithms. En *Operations Research* (págs. 193-202).

Carter, M., & Laporte, G. (1996). Recent developments in practical examination timetabling. En *Practice and Theory of Automated Timetabling: First International Conference* (págs. 1-21). Berlin: Springer-Verlag.

Colorni, A., Dorigo, M., & Maniezzo, V. (1990). Genetic algorithms - A new approach to the timetable problem. En *Computer Science - NATO ASI Series* (págs. 235-239). Springer-Verlag.

Darwin, C. (2007). *Descent of Man: Nuvision Publications.* NuVision Publications.

Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. En *European Journal of Operational Research* (págs. 117-135).

Dorigo, M., Birattari, M., & Stützle, T. (2006). Ant colony optimization: Artificial ants as a computational intelligence technique. *IEEE Computational Intelligence Magazine*, 28-39.

Easton, K., Nemhauser, G., & Trick, M. (2001). The travelling tournament problem: Description and Benchmarks. En *Principles and Practice of Constraint Programming,Lecture Notes in Computer Science* (págs. 580-585). T. Wlash.

Elmohamed, M., Coddington, P., & Fox, G. (1998). A comparison of annealing techniques for academic course scheduling. En *Practice and Theory of Automated Timetabling: Third International Conference* (págs. 92-112). Toronto, Canada: Springer-Verlag.

Fogel, L., Owens, A., & Walsh, M. (1966). *Artificial Intelligence through Simulated Evolution.* New York: John Wiley & Sons.

Franses, P., & Post, G. (2003). Personnel Scheduling in Laboratories. En *Practice and Theory of Automated Timetabling: Fourth International Conference* (págs. 113–119). Berlin: Springer-Verlag.

Gestal, M., Rivero, D., Rabuñal, J. R., Dorado, J., & Pazos, A. (2010). *Introducción a los Algoritmos Genéticos y la Programación Genética.* A Coruña.

Glover, F., & Laguna, M. (1998). *Tabu Search.* Boston, Massachusett.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Boston, MA.: Addison-Wesley Longman Publishing Co.

Goldberg, D. (2002). The Design of Innovation: Lessons from and for Competent Genetic Algorithms. MA: Addison-Wesley, Reading.

Gudes, E., Kuflik, T., & Meisels, A. (1990). Resource Allocation by an Expert System. En *Engineering Applications of Artificial Intelligence* (págs. 101-109).

Gülcü, A., Kuzucuoğlu, A., & Bulkan, S. (2011). A comparison of genetic algorithms & tabu search for a course timetabling problem. En *Technics Technologies Education Management* (págs. 930-938).

Hertz, A. (1991). Tabu Search for Large Scale Timetabling Problems. En *European Journal of Operational Research* (págs. 39-47).

Holland, J. (1975). Adaptation in Natural and Artificial Systems. University of Michigan Press.

Kanagasabapathy, K., Radhakrishnan, R., & Balasubraimanian, S. (2000). Empirical investigation of critical success factor and knowledge management structure for successful implementation of knowledge management system- a case in process industry. En *Hindustan College of Engeneering Review* (págs. 1-13).

Meisels, E., Gudes, & Kuflik, T. (1991). Limited-resource time-tabling by a generalized expert system. En *Knowledge-Based Systems* (págs. 215–224).

Michalewicz, Z., & Fogel, D. (2000). *How to Solve It: Modern Heuristics.* Springer Verlag.

Mitchell, M. (1999). *"An Introduction To Genetic Algorithm". A Bradford Book The MIT Press .* Fifth printing.

Naupari, R., & Rosales, G. (2010). *Aplicación de algoritmos genéticos para el diseño de un sistema de apoyo a la generación de horarios de clases para la Facultad de Ingeniería de Sistemas e Informática de la UNMSM.* Lima-Perú.

Ostermann, R., & Werra, D. (1983). En *Some experiments with a timetabling system* (págs. 199–204).

Papadimitriou, H., & Steiglitz, K. (1982). *Combinatorial Optimization: Algorithms and Complexity.* Englewood Cliffs, New Jersey: Prentice-Hall.

Rahoual, M., & Saad, R. (2006). Solving Timetabling Problems by Hybridizing Genetic Algorithms and Tabu Search. En *Practice and Theory of Automated Timetabling: Sixth International Conference* (págs. 467–472).

Ross, P., & Corne, D. (1995). Comparing Genetic Algorithms, Simulated Annealing. En *Lecture Notes in Computer Science* (págs. 94-122). Berlin: Springer-Verlag.

Rossi-Doria, O., Samples, M. B., Chiarandini, M., Dorigo, M., Gambardella, L., Knowles, J., . . . Stützle, T. (2003). A comparison of the performance of different meta-heuristics on the timetabling problem. En *Practice and Theory of Automated Timetabling: Fourth International Conference* (págs. 329-254). Gent, Belgium.

Schaerf, A. (1997). A survey of automated timetabling. En *Artificial Intelligence Review* (págs. 87-127).

Schmidt, G., & Strohlein, T. (1980). Timetable construction - an annotated bibliography. En *The Computer Journal* (págs. 307–316).

Selim, S. (1988). Split vertices in vertex colouring and their application in developing a solution to the faculty timetable problem. En *The Computer Journal, 30* (págs. 76-82).

Socha, K., Knowles, J., & Samples, M. (2002). A max-min ant system for the university course timetabling problem. En *Practice and Theory of Automated Timetabling: Third International Workshop on Ant Algorithms* (págs. 1-13). Springer-Verlag.

Solotorevsky, G., Gudes, E., & Meisels, A. (1994). RAPS: A rule-based language specifying resource allocation and time-tabling problems. En *Transactions on Knowledge and Data Engineering* (págs. 681-697).

Stallert, J. (1997). En *Automated Timetabling Improves Course Scheduling at UCLA,Interfaces* (págs. 67-81).

Tomassini, M. (1995). A survey of genetic algorithms. Annual Reviews of Computational Physics.

Tripathy, A. (1984). School Timetabling - A Case in Large Binary Integer Linear Programming. En *Management Science* (págs. 473-1489).

Tripathy, A. (1992). Computerised decision aid for timetabling – a case analysis. En *Discrete Applied Mathematics* (págs. 35:313-323).

Tsang, E. (1993). *Foundations of Constraint Satisfaction.* Academic Press,London and San Diego,.

Ueda, H., Ouchi, D., Takahashi, K., & Miyahara, T. (2001). A co-evolving timeslot/room assignment genetic algorithm technique for university timetabling.

Werra, D. (1981). Scheduling in sports. En *Annals of Discrete Mathematics 11* (págs. 381-395).

Werra, D. (1985). An Introduction to Timetabling. En *European Journal of Operational Research* (págs. 151-162).

Whitley, D. (1994). A genetic algorithm tutorial. Statistics and Computing 4.

Wren, A. (1981). *Computer Scheduling of Public Transport.* North-Holland.

Wren, A. (1996). Scheduling, timetabling and rostering — A special relationship? En *Practice and Theory of Automated Timetabling: First International Conference* (págs. 46-75). Berlin, Germany: Springer-Verlag.

Anexos.

Anexo 1: Ejemplo de Algoritmo Genético[13].

Para demostrar los componentes básicos de un GA se trabajará con la optimización de una función. La función es definida como

$$f(x) = xsin(\frac{1}{x})$$

El problema consiste en encontrar un mínimo para esta función en el intervalo $[0, 0.5]$. Por ejemplo encontrar un x_0 tal que $f(x_0) \leq f(x)\ \forall x\epsilon\ [0{,}0.5]$. Actualmente, los valores máximos o mínimos de la función pueden ser calculados haciendo 0 la primera derivada donde $f'(x) = \sin\left(\frac{1}{x}\right) - \frac{\cos\left(\frac{1}{x}\right)}{x} = 0.$

Representación:

Un vector binario se utiliza para representar los valores reales de x como un cromosoma. Este vector consta de 32-bits. Esto significa que la longitud del dominio de la variable x, que es $0{,}5$ en nuestro ejemplo, se divide en 32 rangos de igual tamaño.

La conversión de la cadena binaria a un número real se completa de la siguiente forma. Primero la cadena binaria de longitud 32 se convierte desde la base 2 a la base 10:

$$x^* = ((b_{31}b_{30} \dots b_0))_2 = (\sum_{i=0}^{31} b_i 2^i)_{10}$$

Entonces el correspondiente número real es encontrado por: $x = x^* \frac{1}{2^{32}-1} 0.5$, donde 0.5 es el tamaño del dominio.

[13] Obtenido de "M. Asli Aydin, (2008), Solving University Course Timetabling Problem Using Genetic Algorithm, Istanbul. "

Por ejemplo, un cromosoma $(11110011110010101001011110100111)$ representa el número $0{,}476155032$ en $[0, 0.5]$ desde $x^* = (11110011110010101001011110100111)_2 = (4090140583)_{10}$, siendo el número real $x = (4090140583)\frac{1}{2^{32}-1}0.5 = 0{,}476155032$

Se puede notar fácilmente que $(00000000000000000000000000000000)$ y $(11111111111111111111111111111111)$ representa los límites del dominio 0 y 0.5 respectivamente.

Población Inicial:

Se construyó aleatoriamente 10 cromosomas de cada uno es los vectores binario de 32 bits como población inicial. Los 10 cromosomas c_i donde $1 \leq i \leq 10$ y su correspondiente valor real x_i en el intervalo $[0, 0.5]$ se listan a continuación.

$$
\begin{aligned}
c_1 &= (01101110010010110001111111101001)\ x_1 = 0{,}21541691\\
c_2 &= (11000000001100100111010011011111)\ x_2 = 0{,}37538495\\
c_3 &= (11110011110010101001011110100111)\ x_3 = 0{,}47615503\\
c_4 &= (10001100010110101011110101110100)\ x_4 = 0{,}27412979\\
c_5 &= (10010100101100110001000010100000)\ x_5 = 0{,}29042866\\
c_6 &= (10000110110110000100100000000000)\ x_6 = 0{,}26336884\\
c_7 &= (01100101000110110101101011110000)\ x_7 = 0{,}19747433\\
c_8 &= (11010110010111110111001010110111)\ x_8 = 0{,}41869696\\
c_9 &= (01011001000010001110001110010111)\ x_9 = 0{,}17389594\\
c_{10} &= (11001010100101001111100010101010)\ x_{10} = 0{,}39566781
\end{aligned}
$$

Evaluación de la función:

La evaluación de la función $eval$ para los cromosomas es equivalente a la función f original, es decir, $eval(c) = f(x)$ donde c representa los cromosomas y x representa el valor real correspondiente. La evaluación de la función juega un papel importante ya que se utiliza para evaluar las posibles soluciones en términos de aptitud. Por ejemplo, la aptitud de los cromosomas en nuestra población es la siguiente:

$$
\begin{aligned}
eval\,(c_1) = f(x_1) &= -0{,}214885912\\
eval\,(c_2) = f(x_2) &= \ \ 0{,}172565550\\
eval\,(c_3) = f(x_3) &= \ \ 0{,}410983902\\
eval\,(c_4) = f(x_4) &= -0{,}132941257\\
eval\,(c_5) = f(x_5) &= -0{,}086269622
\end{aligned}
$$

$$eval\,(c_6) = f(x_6) = -0{,}160509512$$
$$eval\,(c_7) = f(x_7) = -0{,}185396112$$
$$eval\,(c_8) = f(x_8) = \ \ 0{,}286388252$$
$$eval\,(c_9) = f(x_9) = -0{,}088302975$$
$$eval\,(c_{10}) = f(x_{10}) = 0{,}228031778$$

El cromosoma c_1 es el cromosoma más apto en la población porque desde su evaluación devuelve el valor más pequeño. Además c_3 es el peor cromosoma puesto que devuelve valor más alto

Operadores Genéticos:

En algoritmo genético hay dos operadores clásicos: cruce y mutación. Aplicar un punto de cruce se eligen los primeros dos cromosomas más aptos c_1y c_7. También un número aleatorio, se selecciona 21 genes para el punto de cruce. Es decir, los cromosomas son divididos en genes y entonces las partes se intercambian entre los cromosomas como se ve a continuación:

$c_1 =$	011011100100101100011	11111101001
$c_7 =$	011001010001101101011	01011110000

Figura 18 Cromosomas padres ejemplo de aplicación

Entonces los hijos resultantes son:

$o_1 =$	011011100100101100011	01011110000
$o_2 =$	011001010001101101011	11111101001

Figura 19 Cromosomas hijos ejemplo de aplicación

Se calculan los correspondientes valores reales y la aptitud de estos nuevos cromosomas de esta forma:

$$f(o_1) = f(0{,}21541676) = -0{,}214885813$$
$$f(o_2) = f(0{,}19747604) = -0{,}185400708$$

La primera descendencia o_1 tiene un valor de aptitud mejor que sus padres sin embargo la mejora sólo se puede ver en el noveno punto decimal.

Entonces se aplica una mutación a los descendientes para obtener mejores cromosomas. Para la mutación, se toman 6 genes seleccionados al azar y luego son cambiaron. El bit de valor “1” se cambia a “0” o viceversa. Suponiendo que las alteraciones se realizan de esta forma:

$$
\begin{array}{rl}
o_1 = & 011011100100101100011010111110000 \\
o_{1m} = & 011\mathbf{100}10\mathbf{1}1001011000110101111\mathbf{11}0 \\
\hline
o_2 = & 011001010001101101011111111101001 \\
o_{2m} = & 011\mathbf{110}01\mathbf{1}001101101011111111101\mathbf{11}1
\end{array}
$$

Figura 20 Cromosomas mutados ejemplo de aplicación.

Se calculan los correspondientes valores reales y la aptitud de estos nuevos cromosomas a continuación:

$$f(o_{1m}) = f(0{,}22420582) = -0{,}217113164$$
$$f(o_{2m}) = f(0{,}23751354) = -0{,}208197846$$

Hay que tener en cuenta que, después de la mutación ambos descendientes mutados tienen mejor valor de aptitud que no sólo de sus padres, sino también las crías que no están mutados. Los descendientes mutados actúan como padres para la próxima generación, es decir, cada iteración comienza con una mejor población.

Aquí sólo se ilustra una generación y las soluciones de esta son prometedores, los Algoritmos Genéticos encuentran soluciones óptimas o casi óptimas gradualmente por esta forma.

Anexo 2: Manual de Aplicación SASPHEII

En el presente apartado se muestra algunas configuraciones y requerimientos básicos para poder lograr una asignación de horarios.

En la ventana principal, en el menú ***Archivo*** se puede acceder a las siguientes opciones tal como se ve en la figura 18

Figura 21 Manual SASPHEII: Archivo.

1. **Nuevo:** Permite crear un horario con nuevos datos y restricciones.
2. **Abrir:** Abre un horario generado con anterioridad que tenga un formato compatible.
3. **Guardar:** Permite al usuario guardar el horario actual bajo el formato predeterminado (*.sph)
4. **Guardar Como:** Permite al usuario guardar el horario actual bajo el formato predeterminado (*.sph) y otros formatos (*.html)
5. **Salir:** Cierra la aplicación solicitando al usuario si desea guardar el progreso actual.

1. Configuración básica datos de entrada.

El menú *Datos* incluye todo lo referente a la configuración básica de la aplicación, es decir, los datos de entrada (días, periodos, docentes, asignaturas, etc.), las actividades y restricciones del programa. En esta parte no se incluye la solución generada para un horario de clases.

En la figura 19 se muestran las opciones accesibles para el usuario en el menú de *Datos*.

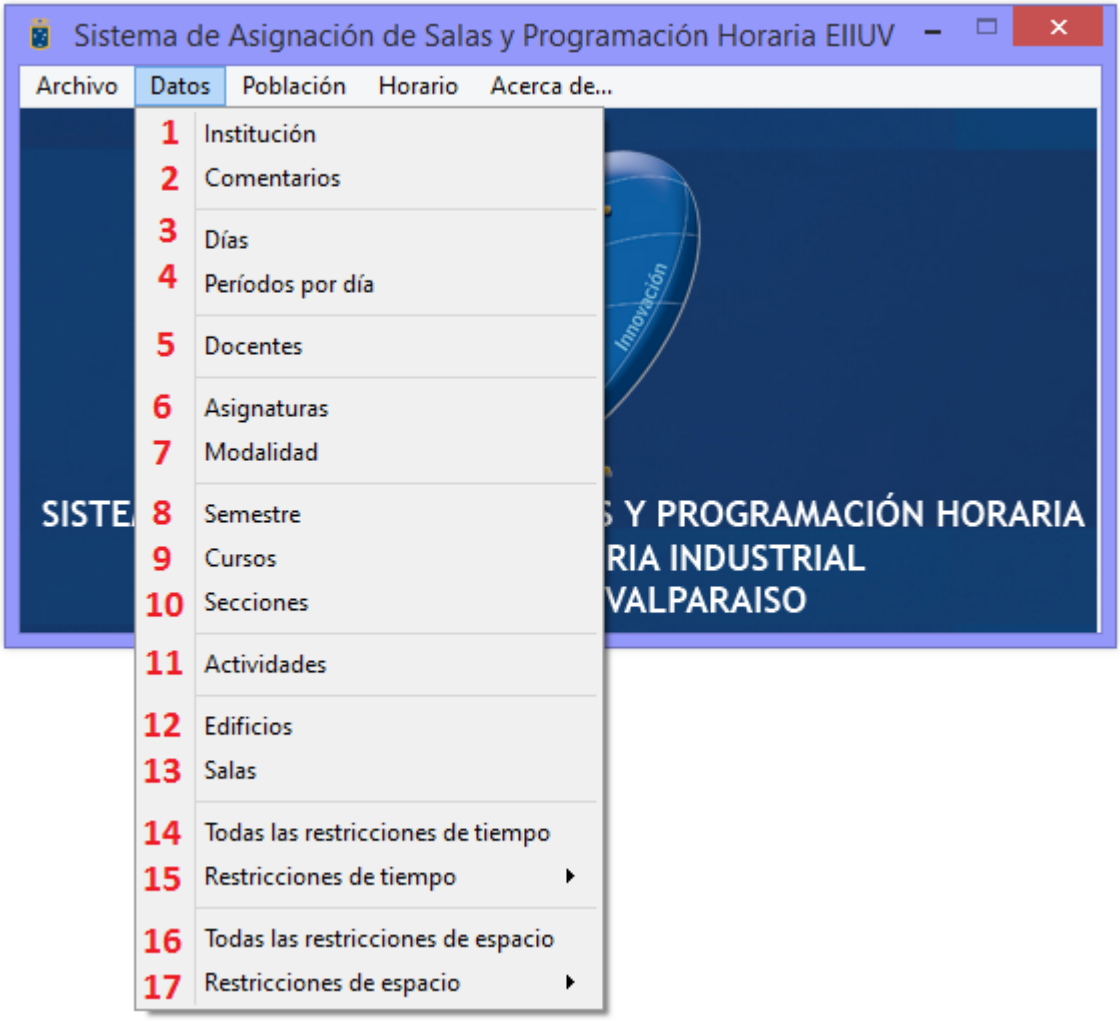

Figura 22 Manual SASPHEII: Datos.

1.1. Institución.

Esta opción es únicamente con el objeto de que el usuario lleve el registro sobre cual institución está trabajando. Cabe señalar que el nombre de la institución será mostrado en el archivo correspondiente al horario generado en la parte superior del mismo.

El ingreso se hace en la opción *Institución* del menú de *Datos*. (Ver figura 20)

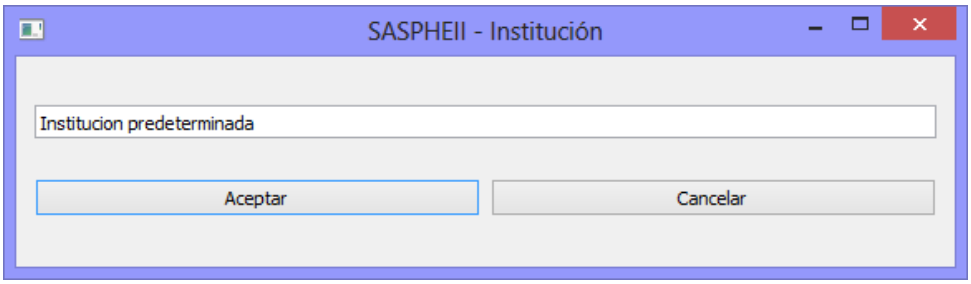

Figura 23 Manual SASPHEII: Ingreso institución

1.2. Comentarios (opcional).

Esta opción es más que nada para que el usuario pueda registrar algún evento en el horario, cambios en las restricciones o cualquier otra información que el considere necesaria apuntar.

El ingreso se hace en la opción *Comentarios* del menú de *Datos*. (Ver figura 21)

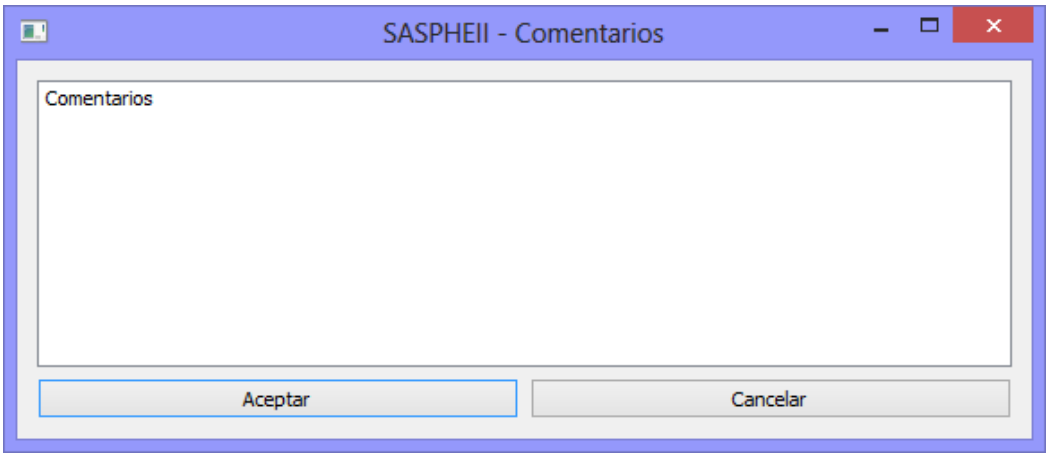

Figura 24 Manual SASPHEII: Ingreso comentarios.

1.3. Días de la semana.

Corresponde a los días de trabajo que permite la institución para la realización de clases, laboratorios y talleres.

El ingreso se hace en la opción *Días* del menú de *Datos*. (Ver figura 22). De manera predeterminada vienen los 5 días de la semana que normalmente se utilizan. Si desea agregar algún otro día debe cambiar la cantidad de días de trabajo y posteriormente presionar el botón aceptar.

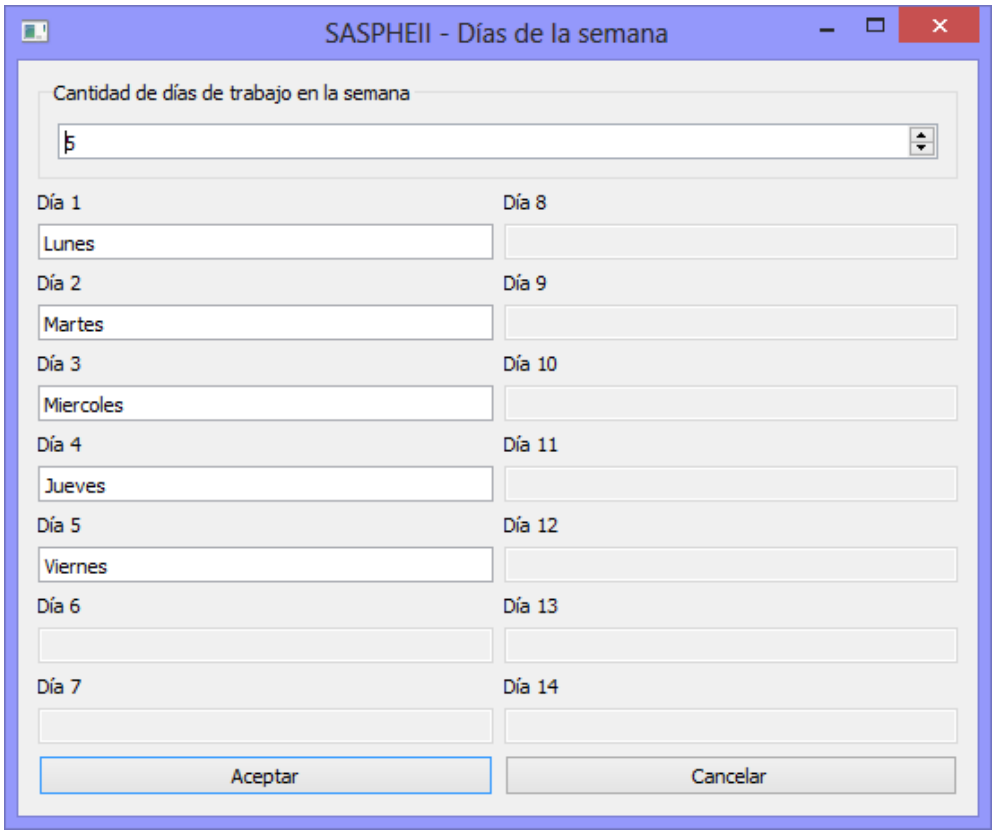

Figura 25 Manual SASPHEII: Ingreso días.

1.4. Horas por día.

Corresponde a los periodos destinados para la programación del horario de clases.

El ingreso se hace en la opción *Períodos por día* del menú de *Datos*. (Ver figura 23). De manera predeterminada vienen los 7 períodos que normalmente se utilizan en la programación del horario. El número de periodos a ingresar siempre debe ser n+1, es decir, si se desea utilizar 7 periodos, en la cantidad de horas de inicio deben ser ingresados 8.

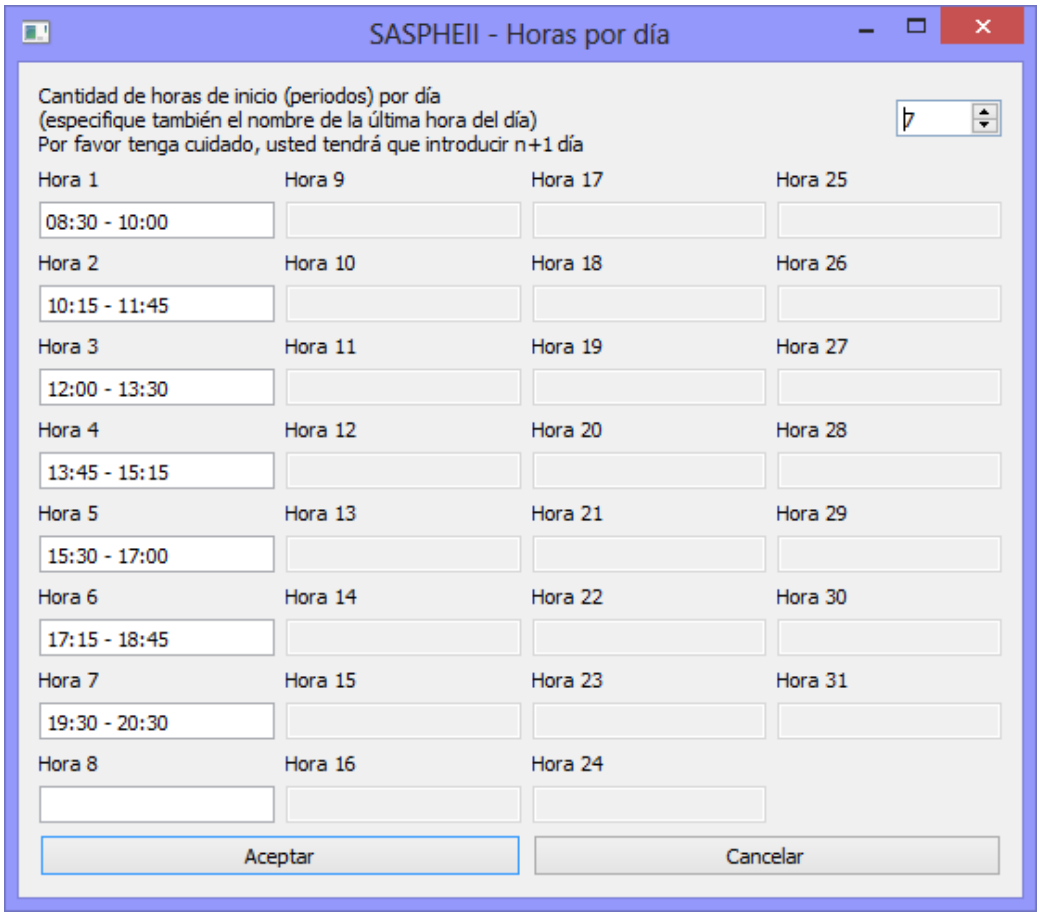

Figura 26 Manual SASPHEII: Ingreso de Períodos.

1.5. Docente.

El ingreso de los docentes se realiza en la opción *Docentes* del menú *Datos*. La ventana principal es la que se muestra en la figura 24.

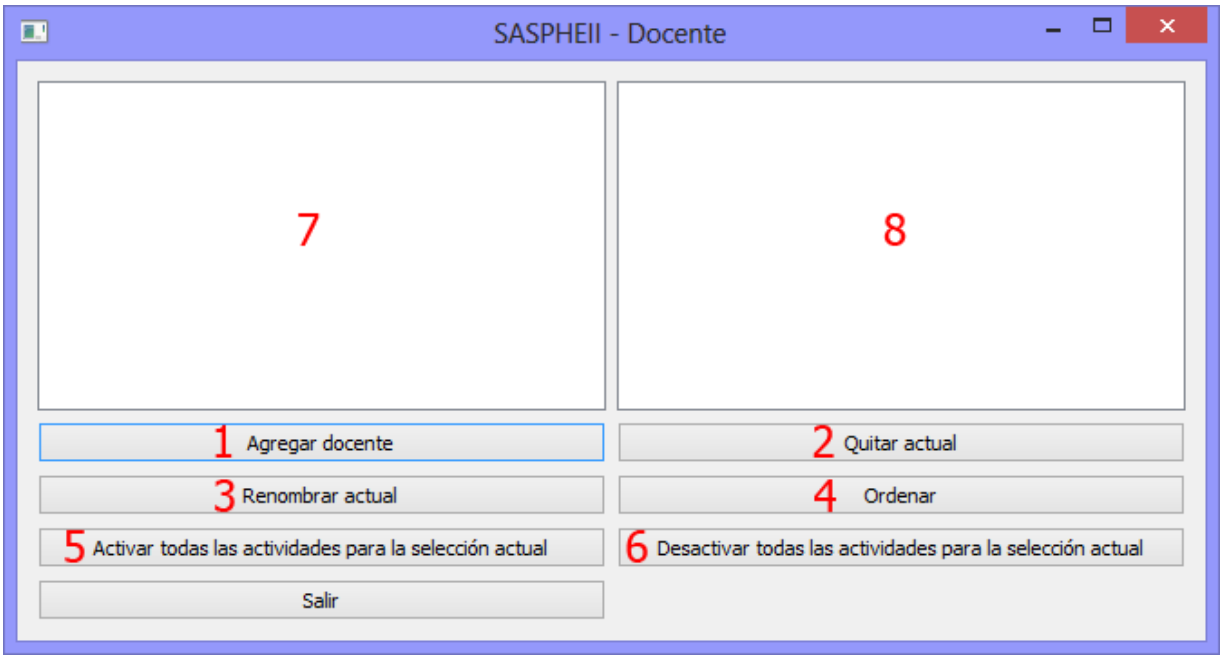

Figura 27 Manual SASPHEII: Docentes.

1. **Agregar docente:** Permite agregar a la lista un nuevo docente.
2. **Quitar actual:** Elimina al docente seleccionado de la lista actual.
3. **Renombrar actual:** Permite cambiar el nombre al docente seleccionado.
4. **Ordenar:** Ordena los docentes de forma alfabética según nombre o apellido (el que sea ingresado primero)
5. **Activar todas las actividades para la selección actual:** Activa al docente seleccionado para poder formar parte de la asignación actual. (predeterminado)
6. **Desactivar todas las actividades para la selección actual:** Desactiva al docente seleccionado para no formar parte de la asignación actual.
7. **Docentes ingresados:** En esa ventana se puede ver la lista de docentes ya ingresados al sistema (no necesariamente participen todos de la asignación).
8. **Detalle docente:** Permite visualizar el detalle de un docente y conocer más adelante si tiene asignada alguna restricción de tiempo o espacio.

1.6. Asignatura.

El ingreso de las asignaturas se realiza en la opción *Asignatura* del menú *Datos*. La ventana principal es la que se muestra en la figura 25.

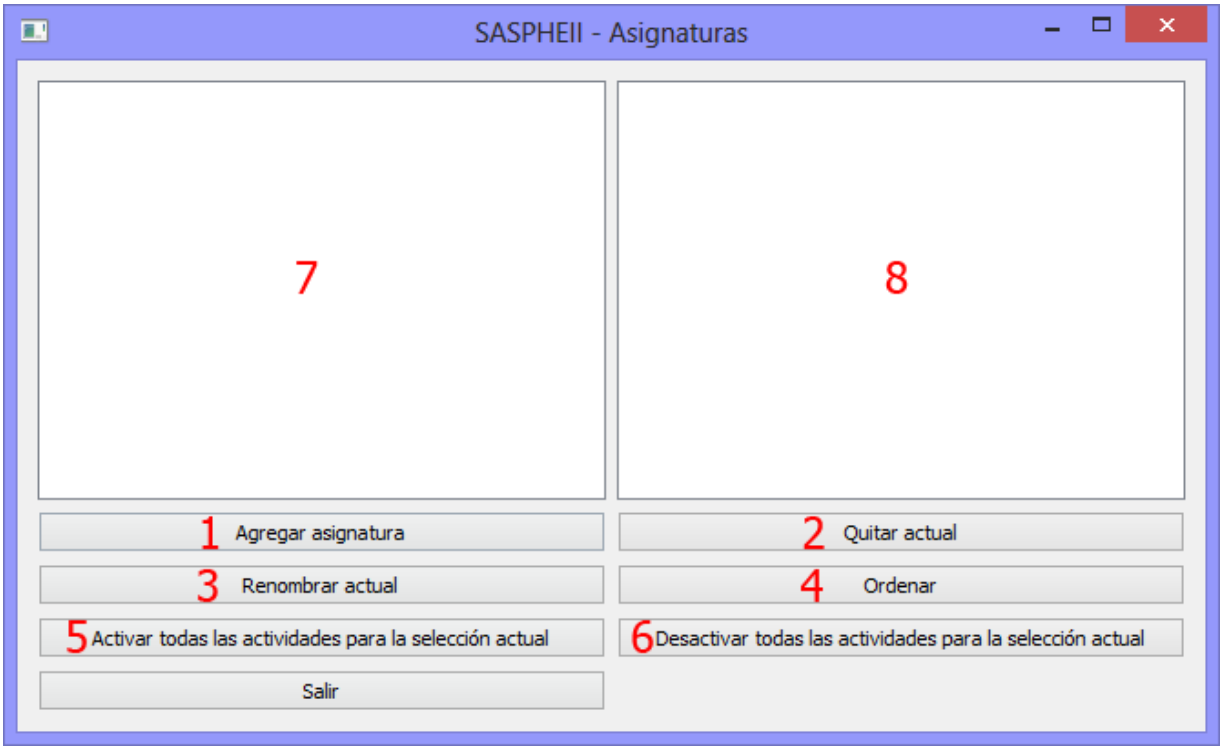

Figura 28 Manual SASPHEII: Asignaturas.

1. **Agregar asignatura:** Permite agregar a la lista una nueva asignatura.
2. **Quitar actual:** Elimina a la asignatura seleccionada de la lista actual.
3. **Renombrar actual:** Permite cambiar de nombre a la asignatura seleccionada.
4. **Ordenar:** Ordena las asignatura de forma alfabética según nombre o código (el que fue ingresado primero)
5. **Activar todas las actividades para la selección actual:** Activa la asignatura seleccionada para poder formar parte de la asignación actual. (predeterminado)
6. **Desactivar todas las actividades para la selección actual:** Desactiva la asignatura seleccionada para no formar parte de la asignación actual.
7. **Asignaturas ingresadas:** En esa ventana se puede ver el listado de asignaturas ya ingresadas al sistema (no necesariamente participen todas de la asignación).
8. **Detalle asignatura:** Permite visualizar el detalle de una asignatura y conocer más adelante si tiene asignada alguna restricción de tiempo o espacio.

1.7. Modalidad.

El ingreso de las respectivas modalidades de cada asignatura se realiza en la opción *Modalidad* del menú *Datos*. La ventana principal es la que se muestra en la figura 26.

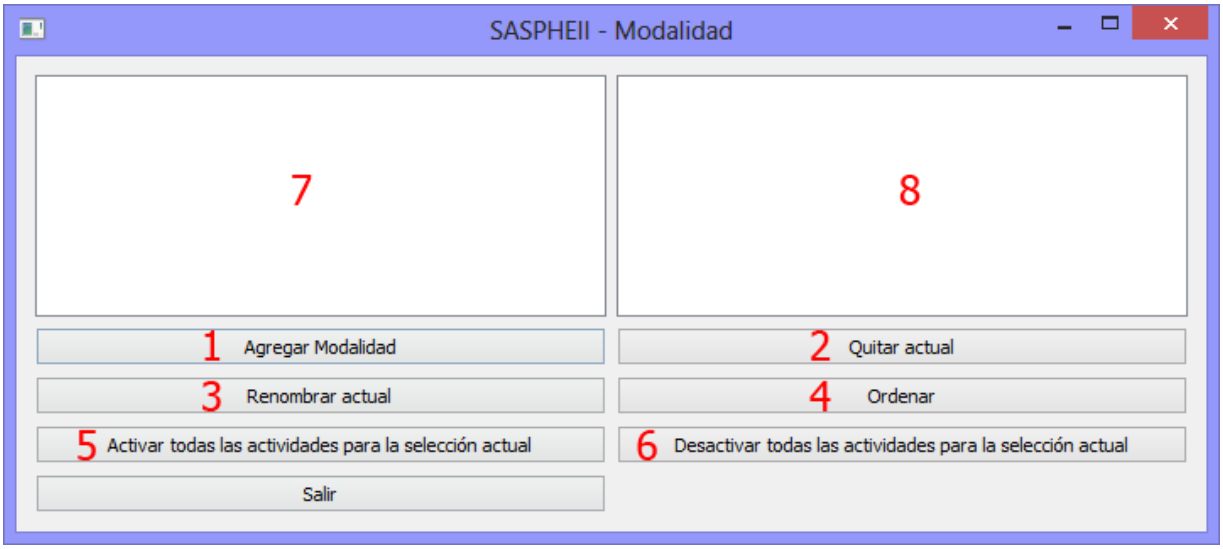

Figura 29 Manual SASPHEII: Modalidad.

1. **Agregar Modalidad:** Permite agregar a la lista una nueva modalidad definida por el usuario.
2. **Quitar actual:** Elimina a la modalidad seleccionada de la lista actual.
3. **Renombrar actual:** Permite cambiar de nombre a la modalidad seleccionada.
4. **Ordenar:** Ordena las modalidades de la lista de forma alfabética según nombre.
5. **Activar todas las actividades para la selección actual:** Activa la modalidad seleccionada para poder formar parte de la asignación actual. (predeterminado)
6. **Desactivar todas las actividades para la selección actual:** Desactiva la modalidad seleccionada para no formar parte de la asignación actual.
7. **Modalidades ingresadas:** En esa ventana se puede ver el listado de modalidades ingresadas al sistema (no necesariamente participen todas de la asignación).
8. **Detalle Modalidad:** Permite visualizar el detalle de una modalidad y conocer más adelante si tiene asignada alguna restricción de tiempo o espacio.

1.8. Semestre.

El ingreso de los respectivos semestres se realiza en la opción *Semestre* del menú *Datos*. La ventana principal es la que se muestra en la figura 27. Acá solamente se ingresa el semestre en curso, por ejemplo, Primer o Segundo semestre año 2016.

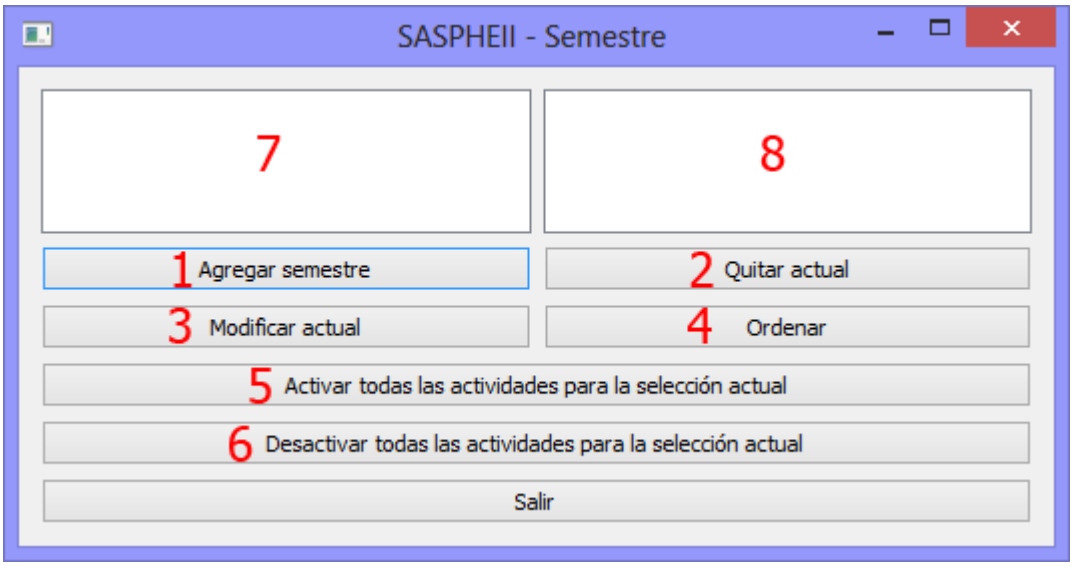

Figura 30 Manual SASPHEII: Semestres.

1. **Agregar Semestre:** Permite agregar a la lista un nuevo semestre. En esta opción es posible ingresar la cantidad de estudiantes. Se recomienda el valor “0”.
2. **Quitar actual:** Elimina el semestre seleccionado de la lista actual.
3. **Renombrar actual:** Permite cambiar de nombre el semestre seleccionado.
4. **Ordenar:** Ordena los semestres de forma alfabética según nombre.
5. **Activar todas las actividades para la selección actual:** Activa el semestre seleccionado para poder formar parte de la asignación actual.
6. **Desactivar todas las actividades para la selección actual:** Desactiva el semestre seleccionado para no formar parte de la asignación actual.
7. **Semestres ingresados:** En esa ventana se puede ver el listado de semestres ya ingresadas al sistema. Se recomienda ingresar solamente un semestre por horario. Para ingresar otro semestre es preferible crear un nuevo archivo e ingresar nuevamente los datos de entrada del horario.
8. **Detalle semestre:** Permite visualizar el detalle de un semestre y conocer más adelante si tiene asignado alguna restricción de tiempo o espacio.

1.9 Cursos.

El ingreso de los respectivos cursos se realiza en la opción *Cursos* del menú *Datos*. La ventana principal es la que se muestra en la figura 28.

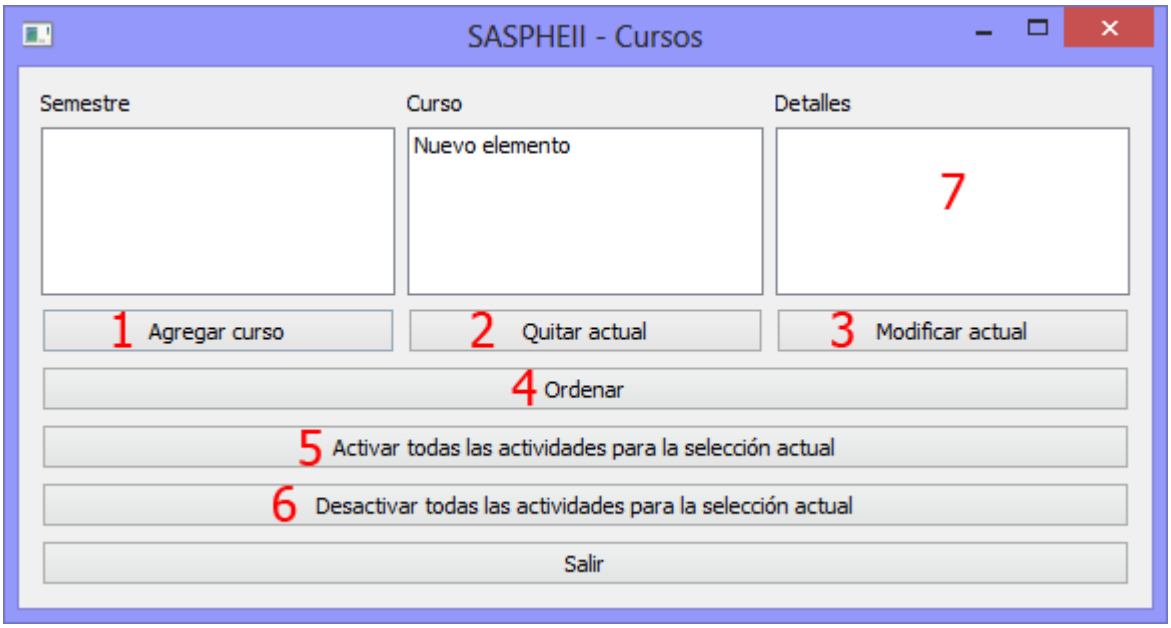

Figura 31 Manual SASPHEII: Cursos.

1. **Agregar curso:** Permite agregar a la lista un nuevo curso. En esta opción si el usuario lo requiere es posible ingresar la cantidad de estudiantes de la sección en particular. Se recomienda dejarlo en el valor “0”.
2. **Quitar actual:** Elimina el curso seleccionado de la lista actual.
3. **Renombrar actual:** Permite cambiar de nombre el curso seleccionado.
4. **Ordenar:** Ordena los cursos de forma alfabética según nombre.
5. **Activar todas las actividades para la selección actual:** Activa el curso seleccionado para poder formar parte de la asignación actual.
6. **Desactivar todas las actividades para la selección actual:** Desactiva el curso seleccionado para no formar parte de la asignación actual.
7. **Detalle curso:** Permite visualizar el detalle de un curso y conocer más adelante si tiene asignado alguna restricción de tiempo o espacio.

1.9. Secciones.

El ingreso de las respectivas secciones por cada curso se realiza en la opción *Secciones* del menú *Datos*. La ventana principal es la que se muestra en la figura 29.

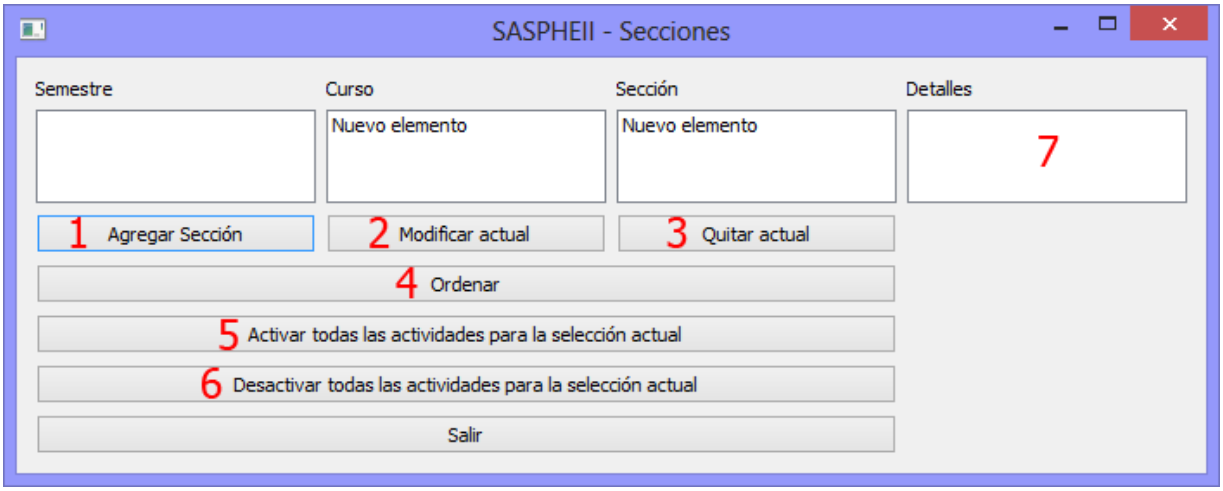

Figura 32 Manual SASPHEII: Secciones.

1. **Agregar sección:** Permite agregar a la lista una nueva sección para el curso seleccionado. En esta opción si el usuario lo requiere es posible ingresar la cantidad de estudiantes de la sección en particular.
2. **Quitar actual:** Elimina la sección seleccionada de la lista actual.
3. **Renombrar actual:** Permite cambiar de nombre a la sección seleccionada.
4. **Ordenar:** Ordena las secciones que componen el curso seleccionado de forma alfabética según nombre.
5. **Activar todas las actividades para la selección actual:** Activa la sección seleccionada para poder formar parte de la asignación actual. (predeterminado)
6. **Desactivar todas las actividades para la selección actual:** Desactiva la sección seleccionada para no formar parte de la asignación actual.
7. **Detalle sección:** Permite visualizar el detalle de una sección y conocer más adelante si tiene asignada alguna restricción de tiempo o espacio.

1.10. Actividades.

Una actividad contiene por lo general una asignatura, su modalidad, un docente y una duración determinada por el usuario. Cuando una actividad contiene una asignatura que debe ser dividida en dos o más clases por semana el sistema creará automáticamente un grupo de sub-actividades con su ID (identificador) correspondiente.

El ingreso de las actividades se realiza en la opción *Actividades* del menú *Datos*. La ventana principal es la que se muestra en la figura 30.

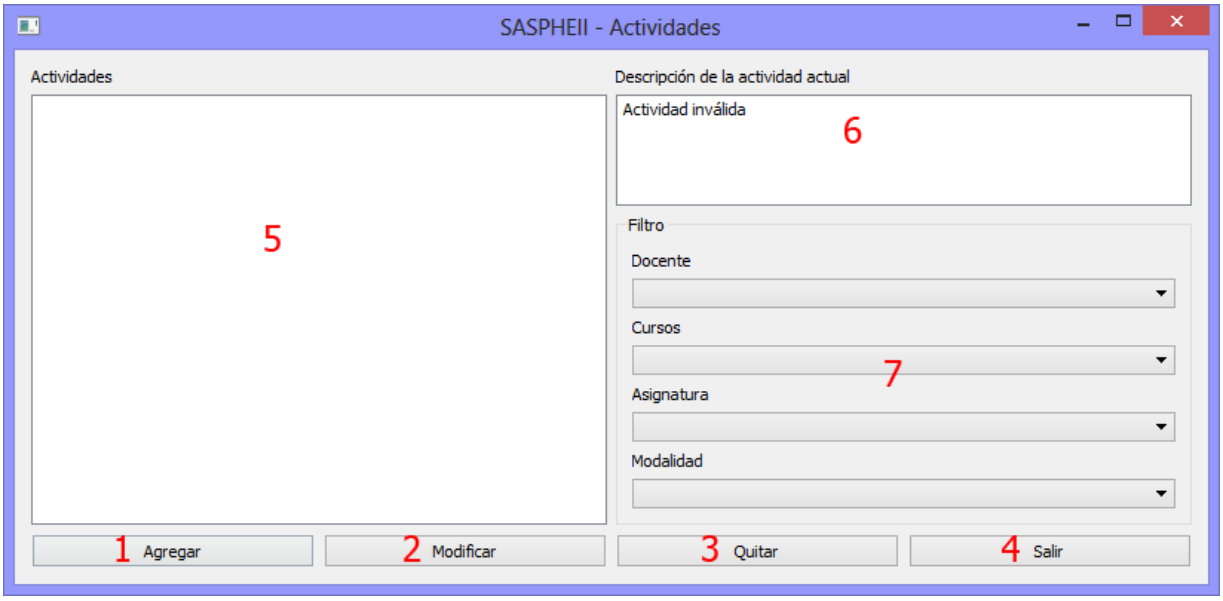

Figura 33 Manual SASPHEII: Actividades.

1. **Agregar:** Abre la ventana correspondiente para poder agregar una actividad
2. **Modificar:** Modifica la actividad (o sub-actividad) seleccionada.
3. **Quitar:** Elimina la actividad seleccionada así también como las sub-actividades y restricciones relacionadas.
4. **Salir:** Cierra la ventana de actividades.
5. **Lista de actividades:** Muestra todas las actividades y sub-actividades que han sido ingresadas hasta el momento.
6. **Detalle actividad:** Muestra un detalle completo de cada sub-actividad. Incluye docente, asignatura, modalidad, sección, duración y restricciones asociadas.
7. **Filtro de búsqueda:** Permite buscar una actividad específica aplicando uno de los filtros de búsqueda que aparecen. Se puede buscar por docente, curso, asignatura o modalidad.

En la figura 31 se muestra el cuadro correspondiente para agregar una nueva actividad. Esta ventana es muy similar a la que se muestra al momento de modificar una actividad.

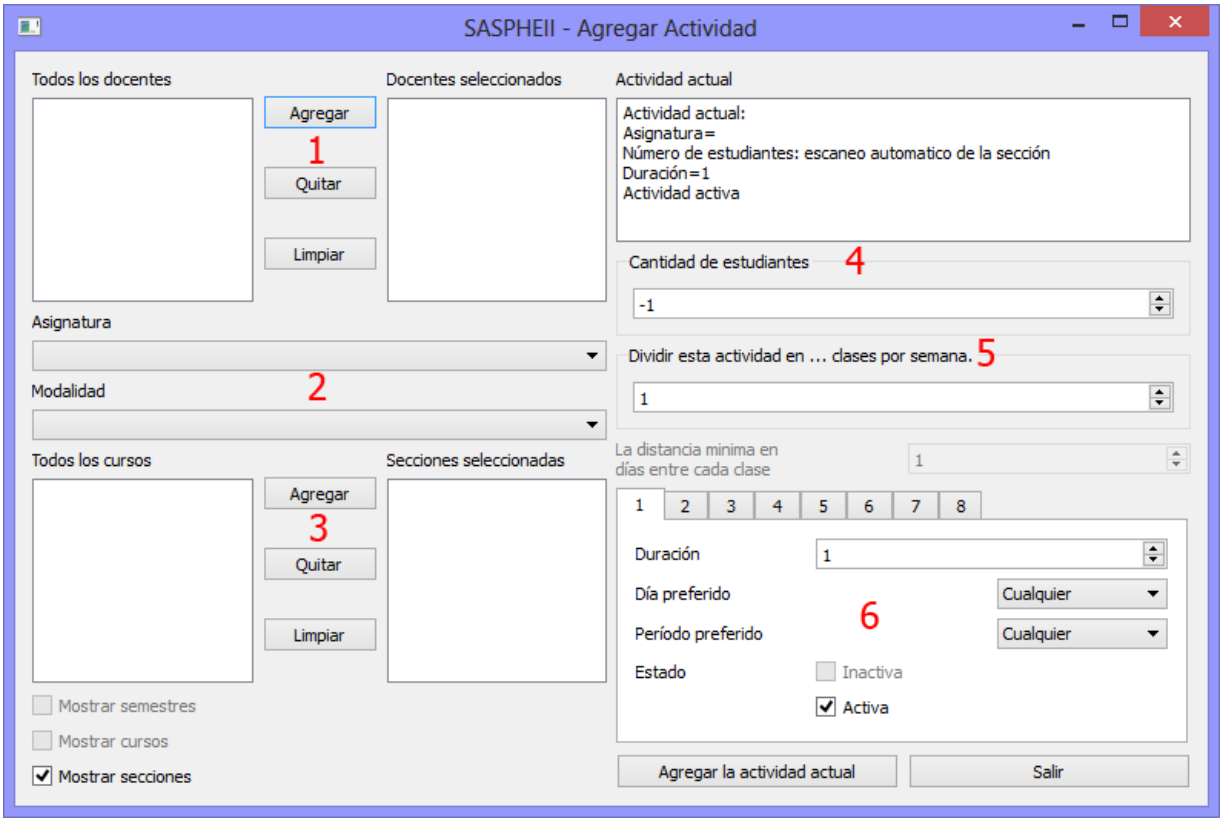

Figura 34 Manual SASPHEII: Agregar una actividad.

1. **Cuadro docentes:** Abre la ventana correspondiente para poder agregar una actividad. Solo se debe ingresar un docente por cada actividad. (Si ingresa dos docentes a la vez podría causar una inestabilidad del sistema)
2. **Cuadro asignatura y modalidad:** muestra el listado de todas las asignaturas y modalidades ingresadas al sistema.
3. **Cuadro Cursos:** Muestra todos las secciones disponibles (paralelos). Solo se debe ingresar una sección por cada actividad. (Si ingresa dos secciones a la vez podría causar una inestabilidad del sistema)
4. **Cantidad de estudiantes:** Corresponde a la cantidad de estudiantes de esa sección, para esa asignatura y dicha modalidad. El valor automático es -1 (debe

dejarlo solamente si anteriormente ingreso el número de estudiantes en el apartado de secciones)

5. **Clases por semana:** Corresponde a la cantidad de sesiones en que se debe realizar esta actividad. Una vez ingresada la cantidad de sesiones se debe ingresar los mínimos días en que se pueden realizar estas. Se recomienda para los días mínimos un valor de 0.
6. **Duración:** Corresponde a la cantidad de periodos totales de una actividad. Esta opción va asociada a lo anterior. Por ejemplo si ingreso que una sesión debe realizarse en dos días y tiene una actividad de 3 periodos de clases (4,5 horas), en una de las pestañas de duración (enumeradas de 1 a 8) debe ingresar como duración dos horas y en la otra solamente una hora. Es muy importante verificar siempre la duración de cada clase.

1.11. Edificios.

El ingreso de los edificios que albergan una cantidad de salas de clases se realiza en la opción *Edificios* del menú *Datos*. La ventana principal es la que se muestra en la figura 32.

Figura 35 Manual SASPHEII: Edificios.

1. **Agregar edificio:** Permite agregar a la lista un nuevo edificio.
2. **Modificar:** Modifica el nombre del edificio seleccionado.
3. **Quitar:** Elimina de la lista el edificio seleccionado.
4. **Ordenar:** Ordena los edificios de forma alfabética según nombre.
5. **Lista edificios:** Muestra los edificios que han sido ingresados al sistema
6. **Detalle edificio:** Permite visualizar el detalle de un edificio y conocer más adelante si tiene asignado alguna restricción de espacio.

1.12.Salas.

El ingreso de las salas de clases se realiza en la opción *Salas* del menú *Datos*. La ventana principal es la que se muestra en la figura 33.

Figura 36 Manual SASPHEII: Salas.

1. **Agregar sala:** Permite agregar a la lista una nueva sala de clases.
2. **Quitar actual:** Elimina la sala selecciona como también todas sus restricciones de espacio asociadas.
3. **Modificar:** Permite modificar la sala seleccionada. Es posible cambiar su nombre, capacidad y tipo.
4. **Ordenar:** Ordena las salas de forma alfabética según nombre.
5. **Lista salas:** Muestra las salas que han sido ingresados al sistema
6. **Detalle sala:** Permite visualizar el detalle de una sala y conocer más adelante si tiene asignada alguna restricción de espacio.

El cuadro que se muestra en la figura 34 corresponde a la ventana que aparece al momento de ingresar una sala de clases.

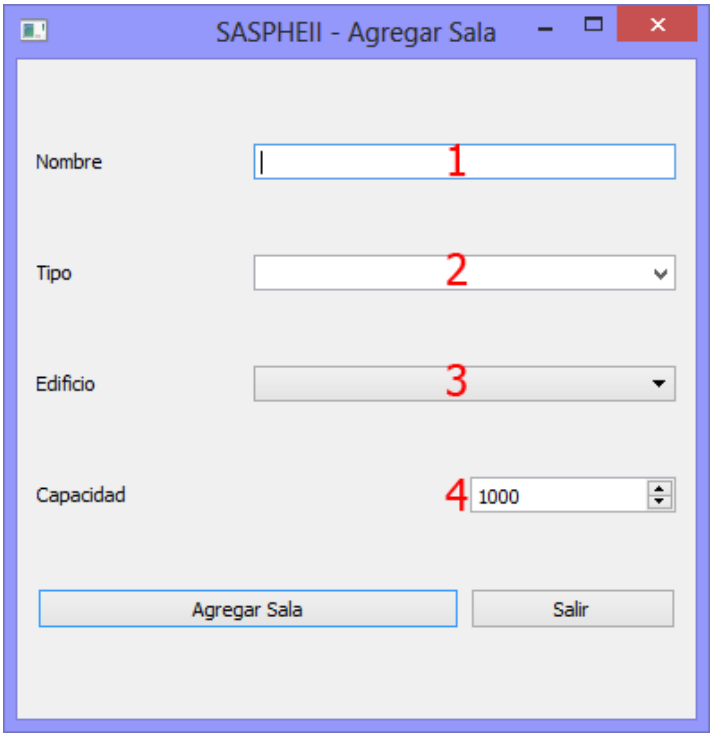

Figura 37 Manual SASPHEII: Ingreso sala de clases.

1. **Nombre:** Corresponde al nombre que se le asigna a una sala.
2. **Tipo:** Se debe ingresar el tipo de sala de la que se trata. Normal o inteligente.
3. **Edificio:** Indica el edificio en el cual está ubicada esa sala de clases en particular.
4. **Capacidad:** Corresponde a la capacidad de la sala que se está ingresando.

2. Configuración Restricciones.

Es importante que vaya verificando si el horario puede resolverse antes de ir agregando nuevas restricciones. Antes de agregar cualquier nueva restricción asegúrese de guardar el archivo en el cual está trabajando.

Cada nueva restricción añade, además de una dificultad al programa, un aumento en el tiempo de búsqueda de una solución. Por lo tanto añada solo las restricciones más importantes y necesarias según su criterio.

A continuación se explica brevemente la función de algunas restricciones de tiempo y espacio. Alguna de ellas comparten una ventana similar en la interfaz es por ello que no se mostrará una figura para cada una de ellas. La decisión de mantener una restricción de forma obligatoria o no debe ser decidida por el usuario.

2.1. Configuración Restricciones de tiempo.

Se puede visualizar el total de restricciones de tiempo asignadas para el horario actual a través del menú *Datos* en la opción *Todas las restricciones de tiempo como se muestra en la figura 35.*

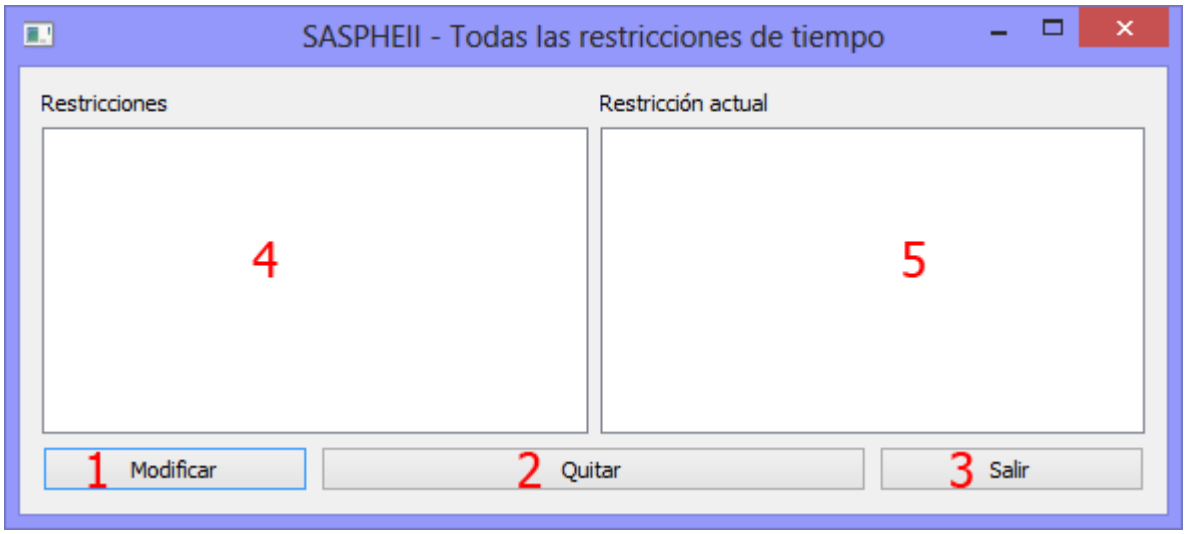

Figura 38 Manual SASPHEII: Todas las restricciones de tiempo.

1. **Modificar:** Modifica la restricción actual sin necesidad de acceder por menú a la misma. Las modificaciones que se pueden realizar corresponde a las propias de cada restricción.
2. **Quitar:** Elimina la restricción seleccionada.
3. **Salir:** Cierra la ventana.
4. **Lista de restricciones actuales:** Listado de todas las restricciones de tiempo.
5. **Restricción actual:** Detalle de la restricción seleccionada.

2.1.1. Restricciones básicas de tiempo.

Este tipo de restricciones debe tener una ponderación de 1 y siempre debe estar incluida en la asignación. Es por ello que el usuario debe verificar que estas restricciones estén activadas.

Para acceder a ella se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Restricciones básicas de tiempo*

La ventana de las restricciones básicas de tiempo se muestra en la figura 36.

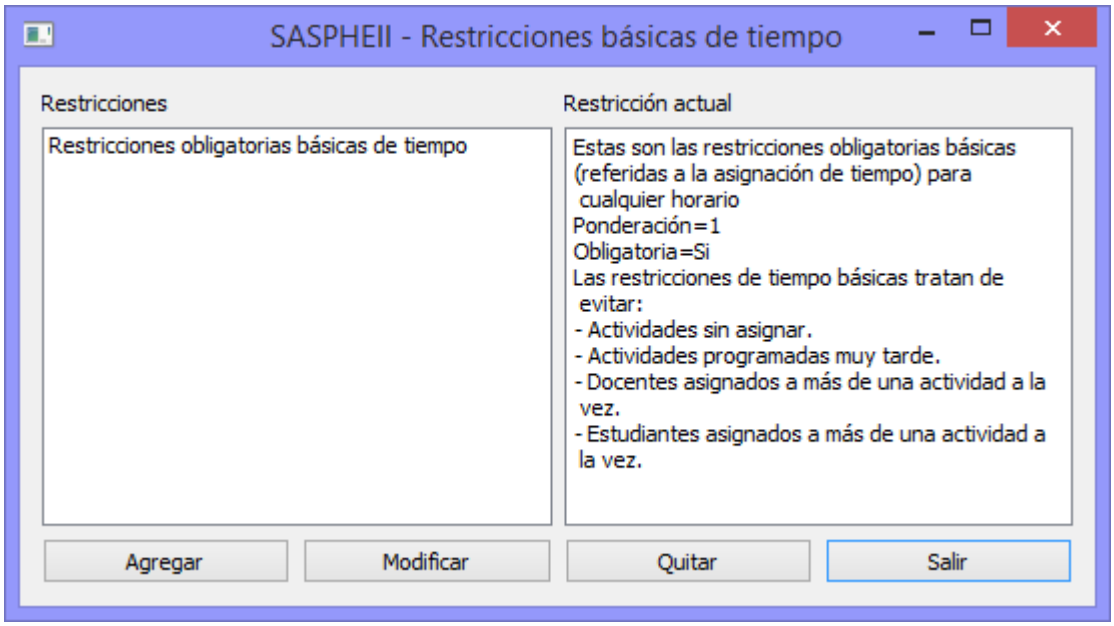

Figura 39 Manual SASPHEII: Restricciones básicas de tiempo.

2.1.2. Break.

Esta restricción señala el periodo tiempo en el que no debe programarse ninguna actividad. Se debe ingresar el día, tiempo de inicio y hora de término de cada break.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Break.*

2.1.3. Períodos Preferidos.

Esta restricción señala los periodos de tiempo disponibles para ciertos recursos. Es posible ingresar una disponibilidad tanto para un docente, curso, asignatura o modalidad pero solo de uno a la vez.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Periodos Preferidos.*

En la Figura 37 se muestra la ventana de los Periodos Preferidos.

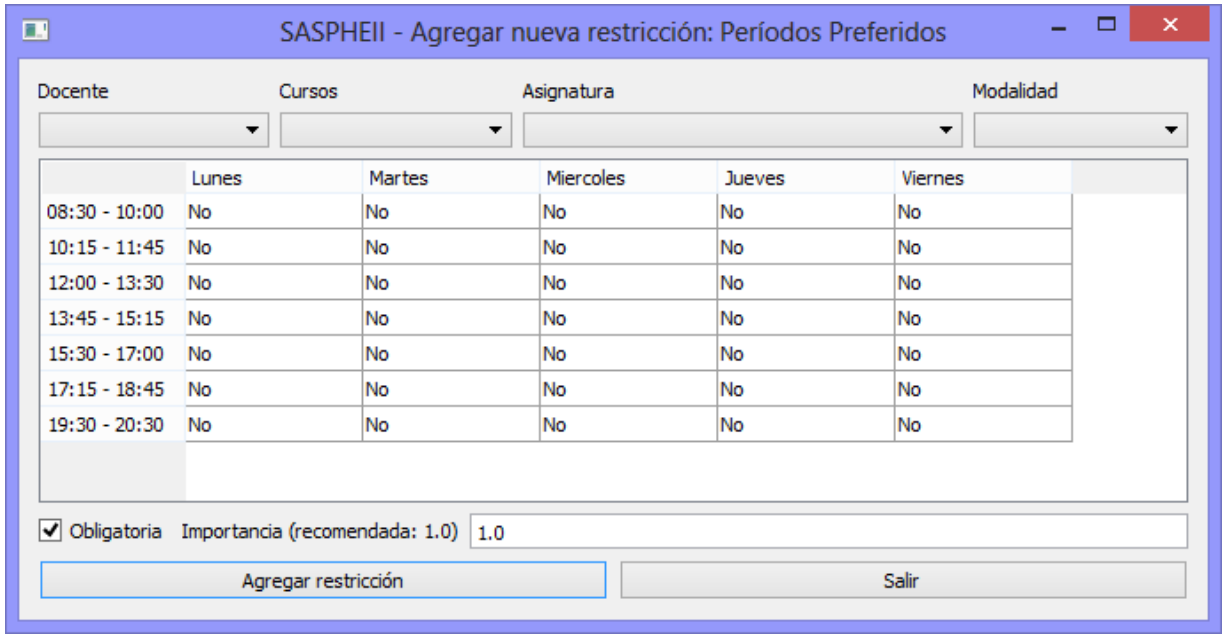

Figura 40 Manual SASPHEII: Periodos Preferidos.

Se recomienda utilizar esta restricción de forma obligatoria con ponderación de 1.

2.1.4. Docente sin ventanas.

Esta restricción nos permite decidir si los docentes pueden o no tener ventanas en su asignación de horario.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Docente sin ventanas*.

Si se va a utilizar se recomienda dejar esta restricción como no obligatoria.

2.1.5. Clases lo más temprano.

Esta restricción nos permite configurar que las clases se traten de programar en los primeros periodos de cada día.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Clases temprano*.

Si se va a utilizar se recomienda dejar esta restricción como no obligatoria.

2.1.6. Cursos N horas al día.

Esta restricción nos permite decidir la cantidad de horas mínimas y máximas que pueden tener **un** curso en particular. Se puede configurar de uno a la vez

Para acceder a esta restricción se hace a través del menú Datos en la opción *Restricciones de tiempo* y luego en el submenú *Cursos con N horas al día*.

Si se va a utilizar se recomienda dejar esta restricción como no obligatoria.

2.1.7. Cursos sin ventanas.

Esta restricción nos permite decidir si las clases programadas deben tener la menor cantidad posible de ventanas por cada sección. Se debe configurar de uno a la vez

Para acceder a esta restricción se hace a través del menú Datos en la opción *Restricciones de tiempo* y luego en el submenú *Clases sin ventanas*.

Si se va a utilizar se recomienda dejar esta restricción como no obligatoria.

2.1.8. Actividad con períodos preferidos.

Esta restricción señala los periodos de tiempo en los que se desea programar alguna actividad. Acá se puede elegir más de un periodo a la vez, la opción es similar a la de periodos preferidos vista en el punto 2.1.3 de este apartado.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Actividad con periodos preferidos.*

2.1.9. Actividades consecutivas.

Esta restricción significa que se deben programar dos actividades consecutivas sin que exista ninguna ventana entre las dos. Se debe ingresar el ID de las dos actividades.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Actividades consecutivas.*

2.1.10. Actividades agrupadas.

Esta restricción significa que la primera actividad debe programarse antes que la segunda actividad. Se debe ingresar el ID de las dos actividades de manera ordenada.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de tiempo* y luego en el submenú *Actividades agrupadas.*

2.2. Configuración Restricciones de espacio.

Se puede visualizar el total de restricciones de tiempo asignadas para el horario actual a través del menú *Datos* en la opción *Todas las restricciones de espacio* como se muestra en la figura 38.

Para acceder a ella se hace a través del menú *Datos* en la opción *Restricciones de espacio* y luego en el submenú *Restricciones básicas de espacio*

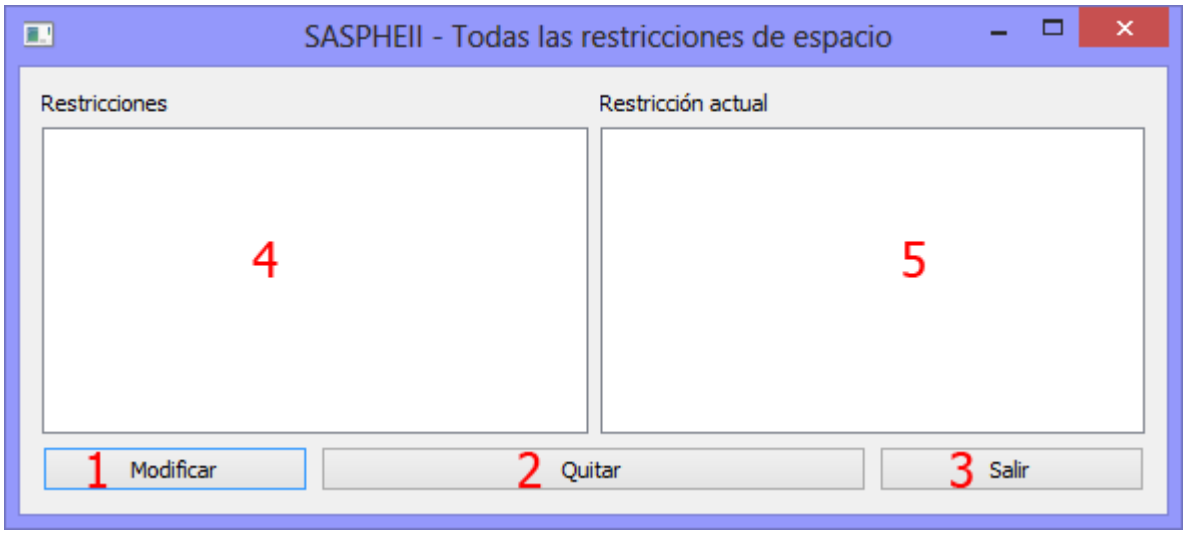

Figura 41 Manual SASPHEII: Restricciones básicas de espacio.

1. **Modificar:** Modifica la restricción actual sin necesidad de acceder por menú a la misma. Las modificaciones que se pueden realizar corresponde a las propias de cada restricción.
2. **Quitar:** Elimina la restricción seleccionada.
3. **Salir:** Cierra la ventana.
4. **Lista de restricciones actuales:** Listado de todas las restricciones de espacio.
5. **Restricción actual:** Detalle de la restricción seleccionada.

2.2.1. Restricciones básicas de espacio

Este tipo de restricciones debe tener una ponderación siempre de 1 y siempre debe estar incluida en la asignación. Es por ello que el usuario debe siempre verificar que estas restricciones estén activadas.

Para acceder a ella se hace a través del menú *Datos* en la opción *Restricciones de espacio* y luego en el submenú *Restricciones básicas de espacio*

La ventana de las restricciones básicas de espacio se muestra en la figura 39.

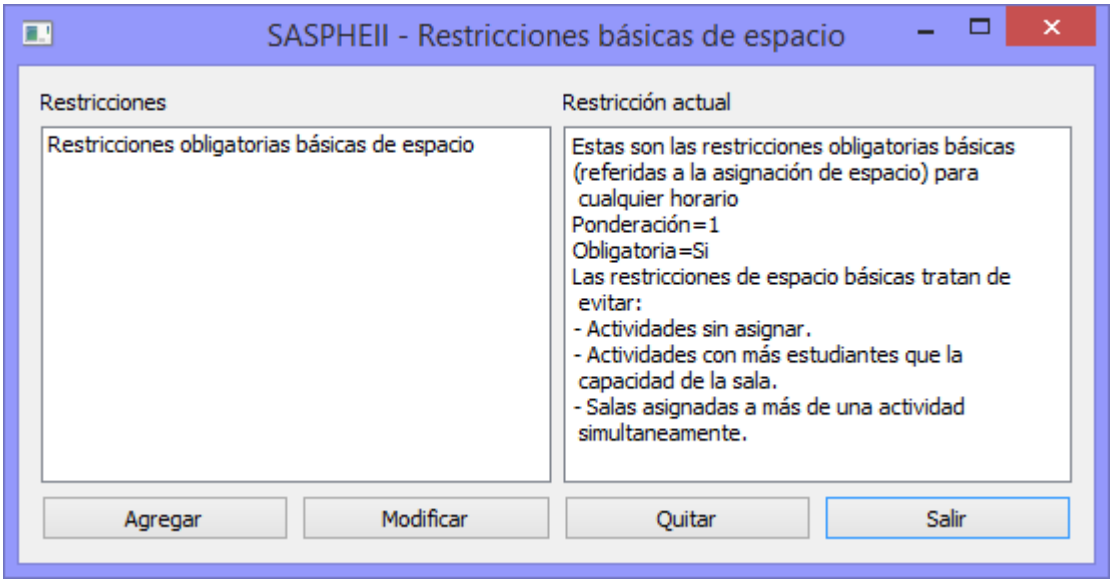

Figura 42 Manual SASPHEII: Restricciones básicas de espacio.

2.2.2. Sala no disponible

Esta restricción señala el periodo de tiempo en el que una sala no se puede utilizar para programar cualquier tipo de actividad. Se debe ingresar el día, tiempo de inicio y hora de término de las horas no disponibles.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de espacio* y luego en el submenú *Sala no disponible.*

2.2.3. Actividad con sala preferida

Esta restricción señala la sala en el que se desea programar cierta actividad. Se debe ingresar el día y la sala preferida.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de espacio* y luego en el submenú *Actividad con sala preferida.*

2.2.4. Minimizar los cambios de salas por curso

Esta restricción nos permite reducir los cambios de sala por cursos de manera global. Lo que hace básicamente es utilizar el menor número de salas para cada curso, pero utilizar esas salas para toda la programación de la semana.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de espacio* y luego en el submenú *Minimizar los cambios de salas por curso.*

Si se va a utilizar se recomienda dejar esta restricción como no obligatoria.

2.2.5. Minimizar los cambios de sala por docente

Esta restricción nos permite reducir los cambios de sala por cursos de manera global. Lo que hace básicamente es utilizar el menor número de salas para cada curso, pero utilizar esas salas para toda la programación de la semana.

Para acceder a esta restricción se hace a través del menú *Datos* en la opción *Restricciones de espacio* y luego en el submenú *Minimizar los cambios de salas por docente.*

Si se va a utilizar se recomienda dejar esta restricción como no obligatoria. Es altamente necesario usar esta restricción debido a que el programa en ciertas oportunidades asigna a una clase de dos periodos consecutivos dos salas distintas.

3. Configuración tamaño de población.

El tamaño de la población define en nuestra aplicación la **velocidad** de búsqueda o la **calidad** de las soluciones (rendimiento) se puede configurar en el menú *Población* en la opción *Tamaño de la población.* La interfaz de esta opción se puede apreciar en la figura 40.

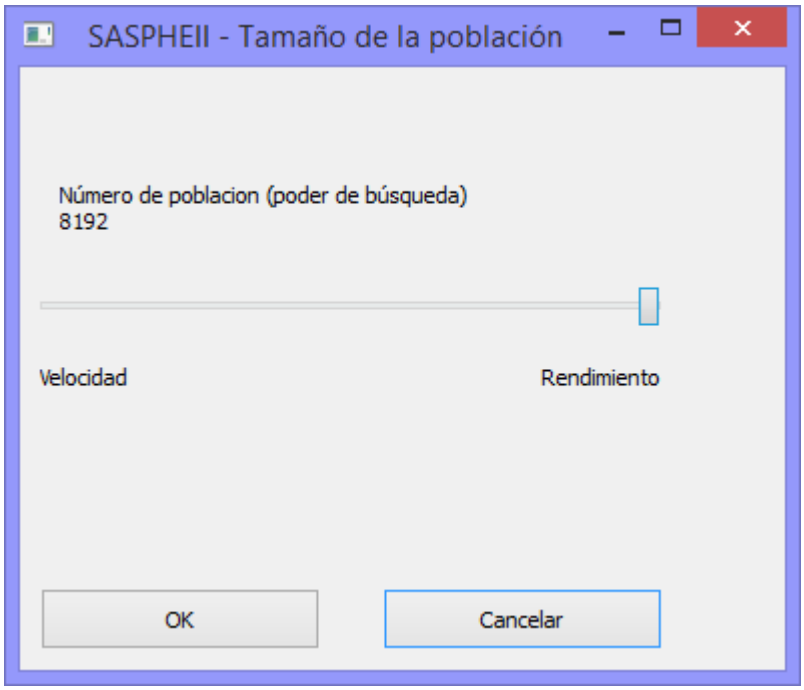

Figura 43 Manual SASPHEII: Tamaño de la población.

Esta opción se puede configurar de igual manera en el archivo *"saspheii.ini"* en la carpeta principal del programa

4. Simulación y visualización de resultados.

En el menú *Horario* es donde se realizan las respectivas asignaciones de horas, de salas o de ambas de manera simultánea. En este menú también es posible realizar una visualización de los horarios generados dentro de la misma aplicación sin necesidad de cerrarla y además permite ver los conflictos de tiempo y espacio (en caso de que hubiera)

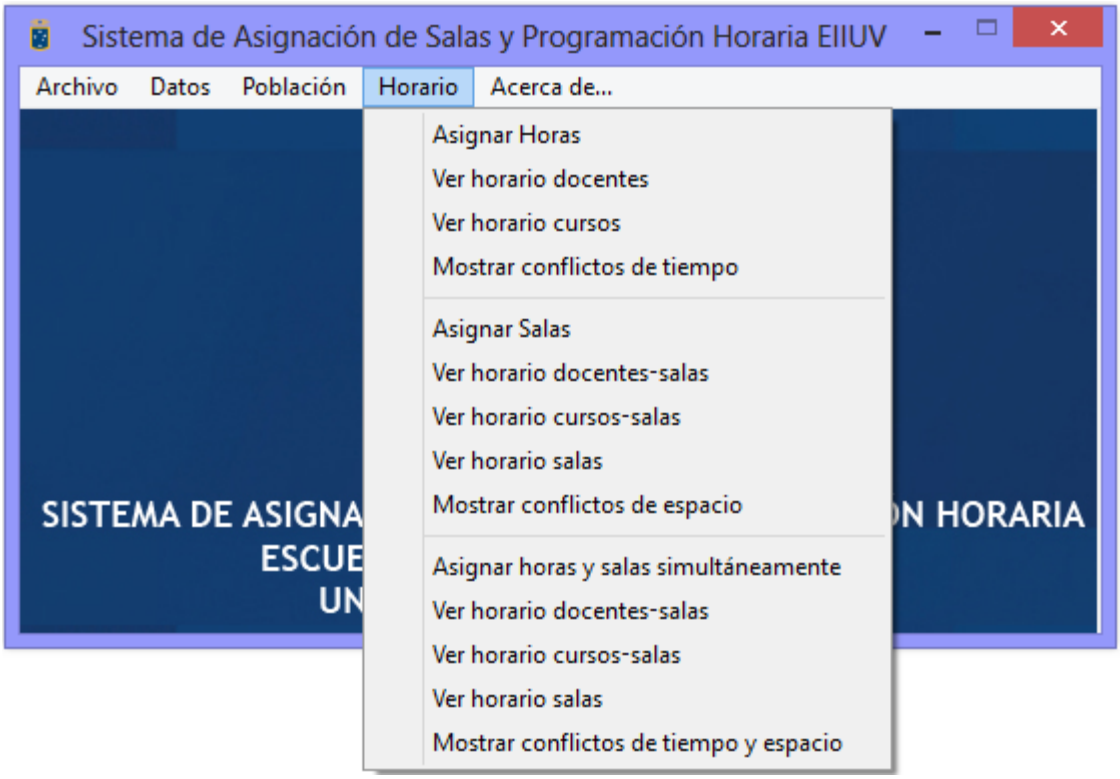

Figura 44 Manual SASPHEII: Horario.

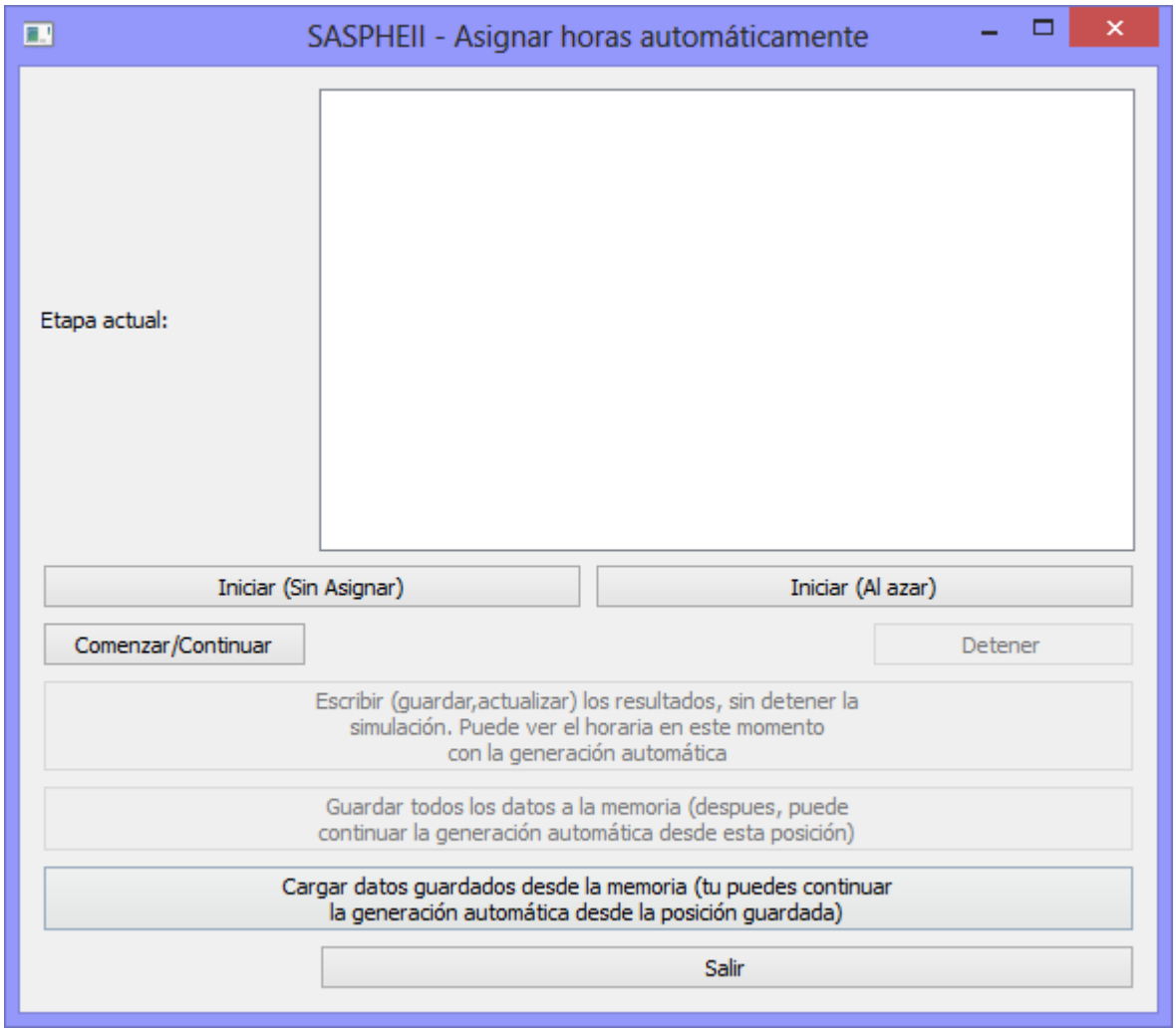

Figura 45 Manual SASPHEII: Simulación.

En la ventana de simulación (opción de las respectivas asignaciones) el usuario tiene dos opciones principales. Iniciar sin asignar que quiere decir que el programa intentará programar las asignaturas una en una para formar las primeras generaciones o Iniciar al azar en donde el programa crea desde la primera generación un horario al azar sin contemplar restricciones para luego ir ajustándolo de acuerdo a los requerimientos del usuario.

Los resultados pueden ser vistos desde la misma aplicación, o bien en la carpeta *"resultados"* que se encuentra en el directorio principal de la aplicación SASPHEII.