UNIVERSIDAD DE TALCA
FACULTAD DE INGENIERÍA
ESCUELA DE INGENIERÍA CIVIL EN COMPUTACIÓN

Asignación de horarios académicos para la Escuela de Ingeniería Civil en Computación de la Universidad de Talca utilizando algoritmos genéticos

YARIXA GÁLVEZ TOLEDO

Profesor Guía: RODRIGO ANDRÉS PAREDES MORALEDA

Memoria para optar al título de
Ingeniera Civil en Computación

Curicó – Chile
Diciembre, 2021

CONSTANCIA

La Dirección del Sistema de Bibliotecas a través de su encargado Biblioteca Campus Curicó certifica que el autor del siguiente trabajo de titulación ha firmado su autorización para la reproducción en forma total o parcial e ilimitada del mismo.

Curicó, 2022

Dedicado a mis padres Andrea y Richard.

AGRADECIMIENTOS

Agradezco a mi madre Andrea Toledo, por el apoyo incondicional y los valores que me supo entregar. Y a mi padre Richard Gálvez por enseñarme que en la vida el éxito se consigue con esfuerzo y dedicación. Quiero agradecer también a mi hermano Benjamín Gálvez, a mi familia y a todas aquellas personas que creyeron en mí.

También agradezco a la Directora de la Escuela de Ingeniería Civil en Computación, la profesora Ruth Garrido, por aconsejarme y orientarme en mi educación profesional. Por su parte, también a Marcela Pacheco, secretaria de la Escuela de Ingeniería Civil en Computación, por su disposición a ayudarme en cada duda o conflicto que se me presentó a lo largo de la carrera.

Agradezco a cada profesor que estuvo presente en mi educación profesional, en especial al profesor Rodrigo Paredes por guiarme en este proyecto y orientarme a raíz de su conocimiento.

Por último quisiera agradecer a mi compañero de casa y primo Roberto Concha por apoyarme a lo largo de los años de formación universitaria, por estar presente en cada momento difícil y aportar ese toque de alegría en nuestro hogar, y agradecerle también cada chocolate que me regaló cuando estaba triste por no encontrar el error en mis programas.

TABLA DE CONTENIDOS

página

ÍNDICE DE FIGURAS

página

ÍNDICE DE TABLAS

página

RESUMEN

En las instituciones de educación, ya sea de enseñanza básica, media o superior, se realiza un proceso en el cual se planifica un horario de actividades docentes (clases de cátedra, laboratorios, etc), que puede estar distribuido en periodos de distinta longitud (bimensual, trimestral, semestral o anual) dependiendo de la cultura organizativa local. Realizar esta tarea no es un proceso sencillo, debido a que está sujeta a una serie de restricciones físicas o reglamentarias, entre otras.

Esta investigación tiene como objetivo ofrecer una alternativa de solución para el problema de asignación de horarios para la carrera de Ingeniería Civil en Computación de la Universidad de Talca.

El problema se encuentra dentro de los problemas de tipo *timetabling* y pertenece a la clase NP-Completa. Dado su alto costo computacional, se utilizan heurísticas que lo resuelven de forma aproximada o probabilística. Dentro de estas heurísticas se tienen la programación lineal entera y los algoritmos genéticos, entre otras.

En la actualidad existen diversos algoritmos que se basan en comportamientos relacionados a la naturaleza y los algoritmos genéticos no son la excepción. Este algoritmo se basa en la teoría de la evolución de Charles Darwin. En el algoritmo se presentan diversos conceptos que la teoría de Darwin señala, como lo son la reproducción, mutación y supervivencia, entre otros.

En el presente documento se muestra el diseño de un algoritmo genético, llevando los conceptos de la teoría de la evolución de Darwin al problema de la asignación de horarios académicos.

Experimentalmente se verifica que la variable que más afecta en el desempeño del proceso de asignación de horarios es la disponibilidad horaria de los profesores. En efecto, considerando que si un profesor tiene que dictar n horas de cátedra o laboratorio, es necesario de que disponga de $2n$ horas disponibles en su jornada laboral para poder encontrar una asignación de horarios factible. Naturalmente, mientras más horas disponibles para la docencia, es más rápido encontrar una planificación factible para los horarios académicos.

1. Introducción

Los establecimientos de educación superior periódicamente se ven enfrentados a problemas de administración. Estos problemas consideran la asignación de múltiples recursos académicos, dentro de ellos, uno de los más importantes es la asignación de horarios de clases, también llamados horarios académicos. Cabe destacar que muchas veces se hace uso del mismo recurso para satisfacer múltiples necesidades; por ejemplo, una misma aula puede ser usada por varias asignaturas o un mismo profesor puede dictar más de un curso. Realizar la toma de decisiones de cómo irán destinados esos recursos en un semestre determinado no es una tarea fácil. En el caso particular de recursos como lo son los docentes, asignaturas y horarios, no quedan ajenos a la difícil tarea de asignación, ya que corresponde a un proceso matemáticamente complejo, debido a la gran cantidad de combinaciones resultantes posibles.

La creación de un horario académico en una institución de educación superior es un procedimiento administrativo, consistente en realizar la asignación de cada uno de los módulos a un docente, para que posteriormente sea asignado a un horario en específico (día y hora determinada). Dado que los periodos académicos varían constantemente, las restricciones que éstos poseen también, por lo que dichos horarios se ven afectados por ellas.

Esta investigación y su implementación en una aplicación de escritorio pretenden dar solución al problema de la creación de horarios académicos de la carrera de Ingeniería Civil en Computación de la Universidad de Talca. Lo que se pretende es reemplazar el proceso manual que se ejecuta actualmente por los directores de escuela, con un proceso asistido por la aplicación de escritorio.

Dentro de los principales objetivos de esta investigación está construir una solución que ayude a reducir el tiempo destinado actualmente en la asignación de horarios

de Ingeniería Civil en Computación y que además se generen horarios con la menor cantidad de violaciones de restricciones posibles.

Es importante mencionar que se espera diseñar una solución que entregue un horario válido en un tiempo "razonable" para este tipo de problema, con el fin de que esta aplicación sea utilizada en la práctica. Vale decir, sólo interesa encontrar soluciones factibles para el problema de asignación de horarios.

El marco de este trabajo se realiza bajo la mención de los diferentes métodos que han sido utilizados con mayor o menor éxito para resolver el problema, siendo los con mayor relevancia la programación lineal entera y el uso de algoritmos genéticos.

1.1. Contexto del proyecto

En cada periodo académico las instituciones de educación se ven enfrentadas a la tarea de estructurar los horarios académicos, sean estos periodos bimestrales, trimestrales, semestrales o anuales. Este proceso consiste en administrar cómo está constituida cada asignatura, asignando un profesor encargado de dictarla, así como también, una programación semanal, especificando el o los días y la o las horas en que se imparte la asignatura.

Este proceso tarda tiempo en completarse, debido a que es una labor difícil de realizar manual e individualmente. Se deben considerar una serie de restricciones y condiciones que varían cada periodo.

En el caso particular de la Universidad de Talca, la realización de este trabajo se encuentra en manos de los Directores de Escuela, adecuándose a los horarios predefinidos que entrega cada Facultad. Los horarios predefinidos son aquellos que pertenecen a los cursos comunes que son dictados en la Facultad, o bien aquellos de otras Facultades o Departamentos (Mátemáticas, Física e Ingles, entre otros).

Existen diversos factores que limitan la distribución de las asignaturas que corresponden a conflictos de planificación, es por ello que es necesario considerar diversas restricciones. Hay restricciones de tipo físicas, por ejemplo, que un profesor no puede dictar dos cursos en horarios simultáneos; restricciones curriculares, como que dos cursos correspondientes a un mismo semestre y año de determinada malla no deben tener topes horarios. Por otro lado, podrían existir también restricciones contractuales, que impidan que un docente trabaje más de la cantidad de horas acordadas.

El proceso de asignación de módulos se realiza de forma manual, en donde se

deben tener en consideración las restricciones correspondientes al semestre que se cursa. Esta tarea tarda alrededor de tres días, en donde en el primer día se asignan los módulos en horarios que el/la Director/a de Escuela estime conveniente según experiencias pasadas. En el segundo día, se realiza una revisión de lo realizado y se completan algunos horarios que aún no están definidos. Finalmente, en el tercer día se realizan correcciones para eliminar topes de módulos pertenecientes a un mismo semestre o topes de horarios de profesores que estén dictando dos cursos en un mismo horario, entre otras restricciones que deben cumplirse.

1.2. Definición del problema

En el proceso de asignación horaria se deben considerar una gran cantidad de variables para llevar a cabo la organización de horarios, lo que implica un excesivo consumo de tiempo y una carga extra para los Directores de Escuela. A pesar del tiempo y trabajo invertido por ellos, muchas veces no es posible lograr una alternativa favorable para los estudiantes y que al mismo tiempo sea conveniente para los profesores.

En el mejor de los casos, es posible llegar a una buena alternativa de asignación, sin embargo, no es posible determinar si realmente la alternativa que entregan los Directores de Escuela satisface con las restricciones que en un principio se desean cumplir.

Es por ello que entregar una alternativa que pudiese evaluar un escenario amplio de posibilidades, y que entregue, al mismo tiempo, una o varias soluciones finales que se adapten a condiciones dadas con anterioridad, podría ser una buena opción para mejorar la organización de este proceso.

1.3. Conceptos básicos

Para la Escuela de Ingeniería Civil en Computación, el problema radica en la programación de asignación de asignaturas en un marco de horarios de seis días por semana, con un máximo de once períodos por día. Cada periodo se compone de una hora. Cada asignatura requiere de dos a cinco períodos a la semana. Este problema presenta una serie de características que son comunes en los sistemas de elaboración de horarios, como lo son:

- Cada profesor está encargado de la docencia de una o más asignaturas.
- Las asignaturas se imparten dentro de bloques de horarios, el primero de estos se lleva a cabo entre las 8:30 hrs a 9:30 hrs y el último de las 20:00 hrs a 21:00 hrs.
- Las asignaturas pueden estar ligadas entre sí. Cuando existe una relación estrecha entre dos asignaturas, se dice que una es pre-requisito de la otra. Esto implica que no es posible cursar la segunda si no se ha aprobado la primera. Además, existen pre-requisitos que son "leves", estos corresponden a que la segunda asignatura puede ser cursada siempre y cuando se haya cursado la primera, sin necesidad de que la primera haya sido aprobada.
- Existen algunas asignaturas que son ajenas a las que imparte la Escuela. Dichas asignaturas son denominadas comunes, es decir, que todos los estudiantes de la Facultad de Ingeniería de la Universidad de Talca las cursan. Estas asignaturas tienen su horario previamente definido y no es posible realizar modificaciones.

1.4. Objetivos

Con el propósito de asistir computacionalmente el proceso de planificación de horarios académicos, se definen los siguientes objetivos para este trabajo.

1.4.1. Objetivo general

- Desarrollar una herramienta que permita planificar los horarios de las asignaturas por semestre de la carrera de Ingeniería Civil en Computación, de forma tal que se minimicen los conflictos de planificación.

1.4.2. Objetivos específicos

- Investigar el estado del arte de los métodos de optimización apropiados para el problema de planificación de horarios.
- Proponer un modelo de restricciones del problema para la generación de horarios de Ingeniería Civil en Computación, con el fin de orientar la toma de

decisiones de la dirección de la Escuela de Ingeniería Civil en Computación de acuerdo a las limitaciones impuestas por los recursos a considerar.

- Diseñar y construir una herramienta que permita llevar a cabo la organización de los horarios, minimizando la cantidad de restricciones violadas.
- Validar la herramienta de organización de horarios para verificar la calidad de la solución obtenida y la usabilidad se la aplicación.

1.5. Alcances

- Durante el desarrollo de esta memoria se espera implementar una aplicación de escritorio muy simple pero funcional que agilice el proceso de programación de horarios de clases.
- Esta aplicación incluye un algoritmo que calcula los horarios académicos considerando las restricciones y consideraciones determinadas por la Escuela de Ingeniería Civil en Computación.
- Este algoritmo no considera la asignación de salas de clases ni por consiguiente la cantidad de alumnos por secciones.
- Esta aplicación se ejecuta en un sólo computador.
- Para el ingreso de datos al sistema se considera el uso de planillas Excel, sujetas a un formato establecido para este propósito.
- El algoritmo que calcula el horario cuenta con restricciones automáticas y manuales.
- La elección de la metodología de optimización a utilizar se realiza a partir de métodos utilizados anteriormente para resolver problemas similares de planificación de horarios.

2. Antecedentes

El problema de asignación de horarios está vinculado con el problema *Timetabling*, específicamente a su variante *Class Scheduling*. Luego de discutir estos problemas, se aborda su complejidad temporal y posteriormente las alternativas de solución existentes. Más adelante se indican los criterios que permiten seleccionar la alternativa de Algoritmos Genéticos. Por último, se desarrolla con mayor detalle la alternativa seleccionada.

2.1. Timetabling y class scheduling

Para entender los conceptos de *timetabling* y *class scheduling* es necesario realizar una contextualización con respecto a la asignación horaria. Una asignación horaria consiste en un proceso en el cual se asignan diversos recursos a un determinado periodo, el cual se ve enfrentado a una serie de restricciones. Dado esto, existen estos dos conceptos claves en la literatura que engloban las investigaciones en torno al problema.

Timetabling consiste en realizar una asignación de eventos a distintos bloques horarios. Este proceso se realiza considerando una serie de restricciones o condiciones que deben cumplirse. Dentro de los problemas de Timetabling, existe una rama llamada Class Scheduling, la cual estudia problemas que están relacionados con la programación horaria de instituciones educativas. Existen tres tipos de problemas relacionado a esta área [7]:

- Programación de horarios de evaluaciones y exámenes (Examination Timetabling).
- Programación de horarios de clases para colegios (School Course Timetabling).

- Programación de horarios de clases para instituciones de educación superior o universidades (University Course Timetabling).

Generalmente en problemas de este tipo y relacionados a la educación, se utiliza el manejo de restricciones, en donde Javier Larrosa [3] las describe de la siguiente forma:

- **Restricciones Duras (Obligatorias):** Estas restricciones son de cumplimiento obligatorio, de tal manera que la violación a alguna de ellas conlleva al origen de un horario no válido. Un ejemplo de ello es que un docente no debe tener asignado dos o más cursos en un mismo bloque de tiempo. Esta restricción no puede ser violada, es por ello que se dice que toda restricción dura se debe satisfacer.

- **Restricciones Blandas (Deseables):** Estas restricciones denotan preferencias del usuario, es decir, que se intentan cumplir en la medida de lo posible. Un ejemplo de ello es la asignación de módulos a horarios dentro del horario protegido, que actualmente es el día jueves de 12:00 hrs a 14:10 hrs. La violación de alguna de estas seguirá ocasionando un horario factible, pero no de la calidad deseada.

Dada las restricciones de un problema de timetabling, se podría decir que su objetivo es eliminar el incumplimiento de las restricciones duras y minimizar el incumplimiento de las restricciones blandas.

2.2. Complejidad del problema

Dentro del área de la computación se han resuelto diversos problemas, en donde cada uno de ellos presenta una dificultad diferente, siendo algunos más difíciles que otros de resolver. Generalmente, para conocer la complejidad de un problema se tiene en cuenta el tiempo de procesamiento y la cantidad de espacio en memoria que se requiere para resolver el problema. Dado lo anterior, es posible clasificar en tres tipos principales la complejidad: P, NP y NP-Completo [3], que corresponden a los problemas polinomiales, no polinomiales y no polinomiales completos.

- **Problemas P:** Son aquellos que es posible darles una solución exacta en tiempo polinomial, es decir, aquellos problemas que son "sencillos" de resolver, en

donde es posible generar una solución de forma práctica. Como por ejemplo las multiplicaciones, funciones lineales, funciones cuadráticas, entre otras.

- **Problemas NP:** Estos problemas pueden ser resueltos en un tiempo polinomial utilizando máquinas de turing no deterministas. Una definición alternativa dice que si se dispone de un testigo (solución factible) del problema, este se puede verificar en tiempo polinomial. El tiempo de procesamiento de este tipo de problemas depende de la cantidad de datos de entrada y de la combinatoria de posibilidades que puede ocurrir.

  Usualmente, para resolver estos problemas en una máquina determinista se requiere de tiempo exponencial.Por lo que a medida que crece el tamaño del problema el tiempo requerido aumenta vertiginósamente.

- **Problemas NP-Completo:** Es el subconjunto de los problemas de decisión en NP tal que todo problema en NP se puede transformar polinomialmente en cada uno de los problemas de NP-Completos. Es posible decir que los problemas NP-Completos son los más difíciles de resolver dentro del conjunto NP y no están presentes dentro de los problemas P.

Cabe destacar que para enfrentar los problemas NP y NP-Completos, se suelen utilizar alternativas aproximadas o probabilísticas, dependiendo de la naturaleza del problema en particular.

2.3. Alternativas de solución

El problema Timetabling tiene varias aplicaciones prácticas, por lo que se ha estudiado en muchas ocasiones. En efecto, presenta una amplia área de investigación, esto debido a la complejidad que este problema presenta, ya que al tener una gran cantidad de variables y restricciones, se vuelve un problema difícil de resolver. A continuación se muestran algunas metodologías que se han utilizado para buscar solución a la asignación de horarios.

2.3.1. Meta-heurística

La meta-heurística es un método heurístico para resolver problemas computacionales. Dicho método parte de una solución inicial, la cual se mejora a medida que

el algoritmo se va ejecutando. Es por ello que este tipo de soluciones se basan en un principio denominado "aumento sucesivo". Este método se ha utilizado con bastante éxito resolviendo problemas de complejidad NP-Completa. Existen diversas meta-heurísticas, dentro de los cuales se pueden mencionar las siguientes:

- Búsqueda Tabú
- Algoritmos Genéticos
- Algoritmos de Enfriamiento Lento Simulado
- Colonia de Hormigas

En este grupo de heurísticas se tienen las que intentan obtener una solución que se aproxime a la óptima (que suele ser el caso de enfriamiento simulado o de colonia de hormigas) o que con alguna probabilidad sea factible (por ejemplo para búsqueda tabú o algoritmos genéticos).

Búsqueda tabú

La búsqueda tabú es un método heurístico que busca resolver problemas de optimización combinatoria. Este método intenta dar inteligencia a los algoritmos de búsqueda local. Según Fred Glover, quien fue el primero que la definió, "la búsqueda tabú guía un procedimiento de búsqueda local para explorar el espacio de soluciones más allá del óptimo local". La búsqueda tabú intenta dirigir su búsqueda mediante la implementación de estructuras simples. Con ello trata de obtener información de lo que va sucediendo y de esta forma aprender de ello y poder actuar en consecuencia de lo ocurrido. Es por eso que se dice que existe un aprendizaje y se considera una búsqueda inteligente.

Para el desarrollo de esta búsqueda es necesario comenzar con una población inicial, la cual a través de las iteraciones va explorando el espacio de búsqueda. El proceso iterativamente considera que dado un espacio actual de soluciones (s) se produce el subconjunto de vecinos $N(s)$. El vecino que presente una mejor función objetivo se convierte en la nueva solución actual, independiente que su función objetivo sea mejor o peor que el valor de la función objetivo en s.

El método utiliza una lista llamada lista tabú, que es una estructura de memoria de corto plazo que contiene los movimientos que fueron realizados en las últimas n

iteraciones, de forma de no repetirlos inmediatamente y así favorecer una exploración más amplia a las posibles soluciones.

Algoritmos genéticos

Los algoritmos genéticos son métodos de búsqueda que están inspirados en el comportamiento de la naturaleza. La principal ventaja que se obtiene al utilizar algoritmos genéticos reside en su capacidad de simular la adaptación biológica. De esta manera, es posible explorar una mayor cantidad de soluciones distintas y escoger la o las mejores de ellas.

La evolución actúa sobre los individuos de una población, en donde los individuos mejor adaptados son los que tienen más probabilidad de dar descendencia y así transmitir su material genético. Este material genético está guardado codificado en forma de genes. Durante la reproducción es posible que se produzcan cambios llamados mutaciones, las cuales pueden ser positivas, negativas o neutras, lo que permite entregar nuevas soluciones en donde explorar.

Los individuos generados son sometidos a una función de evaluación llamada *Fitness*, la cual entrega un valor que refleja hasta qué punto la solución es buena.

Los algoritmos genéticos utilizan algunos términos básicos, los que son utilizados en su funcionalidad. Uno de ellos es la *población* que corresponde a un subconjunto de todas las soluciones posibles que se encuentran codificadas para un problema dado. La población se puede ver como un conjunto de individuos, en donde cada individuo es un candidato a solución del problema que se desea resolver. En esta nomenclatura, al individuo también se le llama cromosoma. Un cromosoma está compuesto por varios genes. Un gen es la posición de un elemento de un cromosoma y codifica una porción de información del individuo. Los genes participan de las operaciones de mutación y cruce.

Algoritmos de enfriamiento lento simulado

Corresponde a un algoritmo de búsqueda por trayectorias que intenta mejorar a la búsqueda local. Este algoritmo tiene como objetivo encontrar un valor relativamente óptimo de una función dado un espacio grande de búsqueda. Este algoritmo se inspira en los principios de la mecánica estadística, en donde se requiere de un proceso de recocido, el cual consiste en calentar un metal para luego enfriarlo lentamente y de

esta manera obtener una estructura fuerte y cristalina. Viéndolo desde un punto más científico, este proceso tiene un funcionamiento basado en el comportamiento físico de los átomos, en donde un enfriamiento demasiado rápido implicaría que los átomos se acomodan rápidamente, llevando así a imperfecciones en el metal. Sin embargo, un enfriamiento lento puede generar que los átomos tengan mayor cantidad de tiempo para acomodarse y así formar una estructura más cristalina y perfecta. El método de enfriamiento simulado es utilizado para dar solución a problemas de optimización combinatorio, en donde la función objetivo del problema es minimizar la energía del material, con la ayuda de una temperatura (parámetro de control del algoritmo) con el fin de conducir al estado óptimo. Primeramente se establece una temperatura inicial en un valor alto, la que se reduce en cada iteración del algoritmo mediante un mecanismo de enfriamiento de la temperatura hasta alcanzar una temperatura final. Si el parámetro de la temperatura disminuye controladamente, es posible alcanzar el mínimo global, pero por el contrario, si la temperatura disminuye bruscamente es posible encontrarnos con un mínimo local.

Colonia de hormigas

Este algoritmo se basa en el comportamiento de las hormigas, ya que estos insectos al momento de buscar su alimento exploran la superficie alrededor de su nido de forma aleatoria y en caso de encontrar una fuente de alimento, proceden a llevarla a su nido. A medida que se dirigen de regreso, las hormigas liberan una sustancia llamada feromona sobre el camino, la que permite que otras hormigas puedan encontrar la ruta donde está el alimento, la cantidad de feromona liberada por la hormiga depende de la calidad y la cantidad de alimento. Este algoritmo de optimización se basa en lo anteriormente descrito, en donde agentes computacionales (hormigas) trabajan en conjunto para poder llegar a una solución a través de la comunicación por feromonas. En este caso cada hormiga puede descubrir una solución del problema, ya que al marcar el camino que transita y encontrar el alimento se convierte en una especie de mapa denominado grafo, en donde cada uno de las aristas corresponde a un camino posible a tomar y cada nodo en un punto donde encontrar alimento.

2.3.2. Programación lineal entera

La programación lineal entera es una técnica que se utiliza en la investigación de operaciones. En esta, las variables sólo pueden tomar valores enteros. Esta técnica permite optimizar una función objetivo dada una serie de restricciones en sus variables. Tiene como principal objetivo maximizar la utilidad o minimizar los costos. Su funcionamiento se basa en formular los problemas utilizando un sistema de inecuaciones lineales y definir una función objetivo.

2.3.3. Redes neuronales

Las redes neuronales se basan en algoritmos predecibles, en donde la computación neuronal permite el desarrollo de sistemas que resuelven problemas complejos. Una red neuronal es un sistema de procesamiento de información, el cual se compone por varios elementos de procesamiento denominadas neuronas, las cuales se encuentran conectadas entre sí a través de canales de comunicación. Las redes neuronales utilizan algunos principios para su funcionamiento, como lo son el aprendizaje adaptativo, la auto organización, la tolerancia a fallos, entre otros.

2.4. Selección de alternativa de solución

Para resolver el problema de timetabling han surgido diversos métodos de solución. Tras realizar una investigación acerca de los distintos métodos utilizados para darle solución al problema que se plantea, se encontraron diversos artículos, tesis y documentos, entre otros, en donde se resolvía el problema de timetabling, siendo los métodos más frecuentes la programación lineal entera y los algoritmos genéticos. A continuación se presentan algunos trabajos relacionados y que se encuentran involucrados en la investigación:

- **El problema de timetabling para colegios chilenos. Solución mediante Algoritmos Genéticos [2]:** En este trabajo se diseñó un método heurístico basado en Algoritmos Genéticos para la solución de asignación de horarios de asignaturas y profesores para colegios. Para ello se utilizó la herramienta Visual Basic en donde diseñó una aplicación. Finalmente se obtuvieron resultados factibles en tiempos computacionales deseados.

- **Algoritmo para la asignación de horarios académicos en la Universidad Francisco de Paula Santander Ocuña utilizando técnicas de inteligencia artificial [6]:** En este trabajo se realiza una investigación acerca de distintas técnicas utilizadas para abordar el problema de asignación de horarios, en donde se selecciona la de Algoritmos Genéticos, dada la flexibilidad que estos entregan al momento de adaptarse a la generación de horarios del problema en particular. Además se construyó un prototipo de software donde se implementaron las reglas y las restricciones analizadas para este problema en particular, evidenciando las ventajas en la disminución del tiempo destinado.

- **Programación de Horarios de Clases y Asignación de Salas para la Facultad de Ingeniería de la Universidad Diego Portales mediante un Enfoque de Programación Entera [4]:** En este trabajo se presenta un modelo de programación entera el cual decide simultáneamente los horarios de los cursos y la asignación de salas.

- **Solving Timetable Problem by Genetic Algorithm and Heuristic Search Case Study: Universitas Pelita Harapan Timetable [5]:** En este trabajo se propone un método para resolver el problema de asignación de horarios mediante el uso de un algoritmo genético combinado con búsqueda heurística. También analiza cómo el algoritmo genético y la búsqueda heurística también pueden resolver problemas de programación. Por último, se propone el diseño arquitectónico de un sistema para resolverlo.

- **Desarrollo de un modelo de asignación de horarios en el entorno educativo mediante la programación lineal [8]:** En este trabajo se presenta un modelo matemático de programación lineal que contribuye con la óptima asignación de clases, teniendo en cuenta diversas restricciones. Por medio del modelo matemático se logró optimizar la asignación horaria y la distribución de la carga laboral de los profesores, obteniendo un aumento 91,66 % de efectividad en tiempo de ejecución.

- **Generación de horarios académicos en INACAP utilizando algoritmos genéticos [1]:** En este trabajo se muestra el diseño y la implementación de un algoritmo genético que logra resolver el problema de asignación de horarios y salas de lNACAP, el cual lo realiza en tiempos muy razonables y con

una muy buena calidad de las soluciones reflejada en el bajo porcentaje de choques de horarios. Finalmente, se concluye que los algoritmos genéticos son una alternativa muy rápida y confiable para los problemas de optimización.

- **Genetic Algorithm as a General Approach to Time Tabling Problem [9]:** Este artículo describe la implementación de un algoritmo genético en un problema de programación de horarios en instituciones educativas. El objetivo principal del algoritmo es encontrar una solución factible, vale decir, que satisfaga todas las restricciones del problema particular.

  Tras realizarse el estudio del caso, el documento revela el éxito del algoritmo demostrando que es posible encontrar una solución factible utilizando operadores genéticos avanzados.

- **SchedulExpert: Scheduling Courses in the Cornell University School of Hotel Administration [10]:** Hotel Administration at Cornell University revela que la programación de cursos es un problema importante para la escuela. Por lo que diseñaron y desarrollaron un programa de computadora, SchedulExpert, para automatizar el proceso de programación. Este programa utiliza la lógica de recocido simulado estándar, en base a un cronograma, en donde la heurística altera iterativamente el cronograma para determinar si se debe reemplazar el cronograma actual por otro, esto ocurre siempre y cuando el cronograma evaluado sea mejor que el actual.

Tras la investigación de los métodos más utilizados, se pudo observar que el empleo de la programación lineal entera no se encuentra tan presente en términos computacionales, ya que las implementaciones de soluciones asociadas a la resolución del problema de timetabling estaban principalmente relacionadas con los algoritmos genéticos. Por otro lado existían antecedentes en donde la programación lineal entera no ha sido efectiva en la resolución de problemas de grandes magnitudes, sino que solo ha tenido buenos resultados en entornos medianos y pequeños. Es importante que la solución al problema que aquí se plantea sea escalable, ya que podría en un futuro ser aplicado no sólo a la Escuela de Ingeniería Civil en Computación, sino que también a nivel de la Facultad de Ingeniería u otras facultades.

Por otro lado, Mauricio Andrés Guerra Cubillos, Erwin Hamid Pardo Quiroga y Roberto Emilio Salas Ruiz en el 2014, utilizaron los Algoritmos Genéticos (AG) para

resolver el problema de calendarización de horarios. Los resultados comparan los AG con recocido simulado, Grasp y colonia de hormigas. Obteniendo que los resultados obtenidos con AG por la complejidad del problema son los más eficientes, gracias a la diversidad de soluciones que puede generar.

Dada la investigación de las soluciones implementadas por diversos autores y los buenos resultados obtenidos por ellos, se ha decidido utilizar algoritmos genéticos para darle solución al problema de la asignación de horarios académicos de la Escuela de Ingeniería Civil en Computación de la Universidad de Talca.

2.5. Elementos y operaciones del algoritmo genético

Existen dos elementos centrales en un algoritmo genético, que corresponden a la:

- Representación de los individuos (soluciones), denominados cromosomas; y a la
- Función objetivo o función de aptitud que mide si el individuo es una posible solución.

Además, existen tres operaciones fundamentales dentro de los algoritmos genéticos, las cuales son aplicadas a los cromosomas, estas son:

- Selección: elige los mejores cromosomas para reproducirse.
- Cruza: realiza la reproducción entre dos cromosomas.
- Mutación: modifica un cromosoma en algún punto con el fin de explorar espacios que serían inalcanzables sólo utilizando la cruza.

2.5.1. Cromosoma

El cromosoma corresponde a una representación de una solución al problema, la cual está compuesta por una estructura de datos que contiene la información relevante del problema que se desea resolver.

2.5.2. Función objetivo

La función objetivo o función de fitness indica qué tan apta es una solución para el problema que se está resolviendo. Esta función evalúa una solución (cromosoma) y le asigna un valor dependiendo de las restricciones violadas que cuente dicha solución.

2.5.3. Población inicial

Como se ha mencionado con anterioridad, el algoritmo se basa en la naturaleza y específicamente en la selección natural. Por este motivo, durante la ejecución del método se van revisando poblaciones de cromosomas con el objetivo de paulatinamente encontrar soluciones cada vez más apropiadas para el problema. Para esto, En la primera iteración se genera una población aleatoriamente y en las siguientes se van utilizando los cromosomas pasados para producir la población siguiente.

Para determinar la calidad de los cromosomas de la población, cada uno de ellos se somete a una evaluación, en donde se determina qué soluciones (cromosomas) son candidatas a reproducirse (más aptos según la función objetivo). Los cromosomas (soluciones) más aptos podrán dar origen a nuevos cromosomas que corresponden a nuevas soluciones.

Es importante considerar que la población inicial debe tener un tamaño lo suficientemente grande como para garantizar que exista diversidad en las soluciones, de forma de evitar caer en óptimos locales y explorar el espacio de soluciones lo más ampliamente posible.

2.5.4. Operación de selección

La repoducción de los cromosomas se realiza después de llevar a cabo la operación de selección. Por lo tanto, la intuición dice que hay que seleccionar a los cromosomas más aptos. Sin embargo, esta estrategia puede llevar al método a caer en óptimos locales. Una alternativa para evitar esto es conservar algunos de los otros cromosomas ya que esto le brinda diversidad a la población. Para ello, una alternativa es por un lado, selecccionar a un grupo de los individuos (cromosomas) más aptos y, por otro, seleccionar aleatoriamente a un grupo de cromosomas dentro del resto de la población.

2.5.5. Operación de cruza

La cruza es una operación de reproducción sexual. Para buscar nuevas y mejores soluciones, un algoritmo genético utiliza funciones de cruza, que corresponde a una recombinación de individuos, dando origen a nuevas generaciones de soluciones. Una función de cruza usa dos cromosomas progenitores y los combina generando nuevos individuos que a su vez son evaluados y seleccionados como progenitores de nuevas generaciones.

Desde el punto de vista de la exploración del espacio de soluciones, esta operación permite explorar zonas intermedias al par de cromosomas que participan en la cruza.

Cruce de un punto

La cruza de un punto consiste en dividir los cromosomas en dos partes, en algún punto escogido de manera aleatoria. Se debe considerar que el punto elegido debe ser un punto intermedio, ya que es importante asegurarse que el cromosoma sea dividido en dos porciones. Una vez realizada la división del cromosoma, queda como resultado una primera parte, llamada cabeza, y una segunda parte, llamada cola. De esta forma se realiza un intercambio entre cabeza y cola de ambos cromosomas progenitores. Esto se ilustra en la207- Figura 2.1.

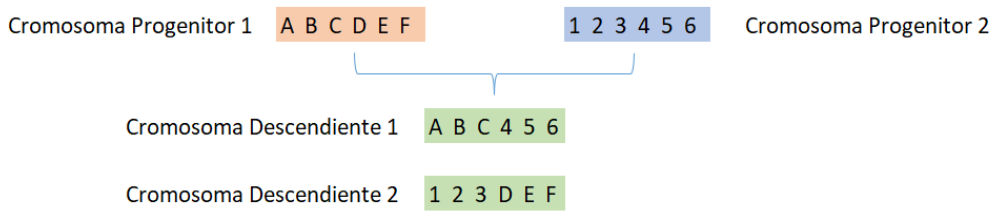


Figura 2.1: Ejemplo cruce de un punto

Cruce de dos puntos

La cruza de dos puntos consiste en dividir los cromosomas en tres porciones. Para esto, se escogen dos puntos distintos de forma aleatoria. Es importante asegurarse que el cromosoma sea dividido en tres porciones. Una vez realizada la división del cromosoma, queda como resultado una primera parte, la cual también es llamada cabeza, una parte intermedia llamada cuerpo, y una última parte, llamada cola.

De esta forma se realiza un cruce asignando el cuerpo de uno de los cromosomas progenitores y en los extremos (cabeza y cola) del otro. Esto se ilustra en la Figura 2.2.

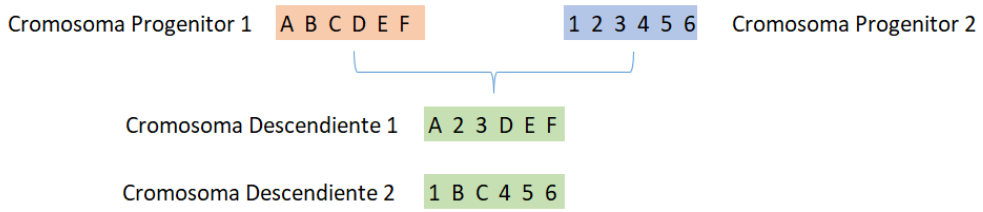


Figura 2.2: Ejemplo cruce de dos puntos

Cruce uniforme

La cruza uniforme consiste en generar una máscara para cada cromosoma progenitor, la cual está constituida por una secuencia binaria aleatoria. La máscara decide qué gen de los cromosomas progenitores se hereda a sus descendientes. En caso de que la máscara posea un 1 dentro de su secuencia, entonces quiere decir que el gen que se encuentra en la posición del binario será considerado. De lo contrario, si posee un 0 significa que no se considera. Esto se ilustra en la Figura 2.3.

Naturalmente se puede producir un segundo descendiente considerando la asignación complementaria.

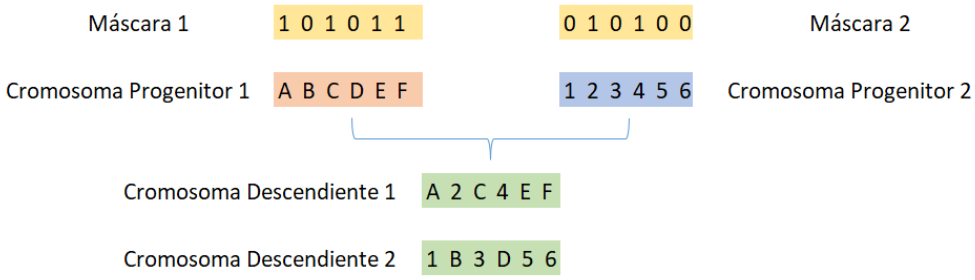


Figura 2.3: Ejemplo cruce uniforme

2.5.6. Operación de mutación

Una función de mutación se encarga de realizar una modificación en un cromosoma sin combinarlo con otro. La modificación de un individuo consiste en que alguno o algunos de sus genes, cambie su valor aleatoriamente.

Dentro del diseño del algoritmo genético es posible decidir si la operación de mutación se realiza seleccionando directamente los individuos y posteriormente mutarlos, o se realiza dicha mutación en conjunto con la operación de cruce.

Desde el punto de vista de la exploración del espacio de soluciones, esta operación permite explorar el vecindario del cromosoma que sufre la mutación.

3. Metodología

En este capítulo se detallan las fases que se contemplan para el desarrollo de esta memoria y las actividades correspondientes. También se explica el bosquejo de la solución a desarrollar, explicando los componentes del algoritmo.

3.1. Metodología de investigación

En cuanto a la metodología que se utiliza en este trabajo, es la metodología de investigación cuantitativa [11], la cual consiste en un proceso sistemático y ordenado que se lleva a cabo mediante el seguimiento de determinados pasos. La razón por la cual se utiliza esta metodología es porque para resolver el problema planteado en esta memoria es necesario identificar una serie de elementos comunes que puedan proporcionar una dirección y una guía al momento de realizar la investigación. Es necesario también, identificar fases y etapas en las que se divide el proyecto, ya que de esta forma es posible generar un proceso investigativo. A continuación se nombran las etapas a considerar.

3.1.1. Fase conceptual

En esta fase se formula y delimita el problema que se está abordando, a partir de la definición del contexto y basándose en la revisión literaria de tópicos relacionados al tema abordado. Las actividades comprendidas en esta fase son:

- Investigación del estado del arte de timetabling y class scheduling.
- Estudio de la situación actual del problema.

- Definición y delimitación del problema que se aborda.
- Estudio de las principales restricciones del problema.
- Construcción del estado del arte.
- Construcción del marco teórico.
- Formulación de objetivos.

3.1.2. Fase de planeación y diseño

En esta fase se definen los métodos y estrategias que se emplean para resolver el problema. Se realizan tareas como:

- Diseño e implementación de una estructura de datos.
- Diseño e implementación de una estructura lógica del problema.

3.1.3. Fase empírica y analítica

Esta fase corresponde a pasar a la ejecución del estudio, donde en primer lugar se lleva a cabo una recolección de datos y la preparación de estos para su análisis. Esto significa que se lleva a cabo una serie de experimentos en los que se generan diversas soluciones, en las que su posterior análisis permite sacar conclusiones generales que apuntan a esclarecer el problema formulado al inicio de esta memoria.

3.1.4. Fase de difusión

En esta fase se divulgan los resultados obtenidos, por lo que esta etapa comprende la formalización de todo el proceso realizado en el presente documento de memoria, por lo que se da cuenta de los objetivos propuestos, el diseño metodológico utilizado, resultados, dificultades y limitaciones. Se concluye el trabajo realizado junto con sugerencias o recomendaciones para nuevos estudios.

3.2. Metodología de validación

Se propone utilizar la metodología para pruebas unitarias, con el fin de comprobar el correcto funcionamiento de cada una de las unidades de código. Estas pruebas

se aplican a los elementos más pequeños del sistema (nombres de variables, tipo de parámetros, variables de retorno, entre otras) asegurándose que cada unidad funcione correctamente y eficientemente por separado. También es necesario verificar que el código cumple con lo que debería hacer. Por su parte también se realizan pruebas de integración, en donde se comprueba el funcionamiento global del algoritmo, esto normalmente se realiza mediante una única función cohesiva que llama a todas las funcionalidades pequeñas del algoritmo.

3.3. Bosquejo de solución

A continuación se definen los elementos y operaciones a utilizar en el algoritmo genético, esta vez desde un enfoque distinto al capítulo anterior ya que este se enfoca en dirigir cada elemento y operación del algoritmo genético al problema que se está planteando, describiendo a grandes rasgos como podía ese elemento u operación funcionar dentro del escenario del problema de asignación de horarios.

3.3.1. Definición de cromosoma

Como se mencionó en el capítulo anterior, un cromosoma corresponde a una representación de una solución al problema. Llevándolo a la resolución del problema de asignación de horarios, un cromosoma corresponde a una posible asignación horaria, en donde esta podría ser correcta o no.

Este cromosoma está compuesto por una estructura de datos que almacena información con respecto a los cursos que se desean organizar. Cabe destacar que cada cromosoma que se crea posee una misma estructura, con la misma cantidad de información y elementos.

3.3.2. Definición de función objetivo

La función objetivo evalúa al cromosoma, contando la cantidad de conflictos que este posee. Para ello toma en consideración una serie de restricciones que en el caso de que una de ellas sea violada, la función objetivo agrega un conflicto.

Dentro de las restricciones se encuentran:

- Que la planificación presente incumplimiento a la restricción de que un profesor esté dictando dos asignaturas dentro del mismo horario.

- Que la planificación presente incumplimiento en la restricción de que dos módulos pertenecientes a un mismo semestre se dicten en el mismo horario.
- Que la planificación presente incumplimiento en la restricción de la disponibilidad horaria de un docente.
- Que la planificación presente incumplimiento en la restricción de que dos módulos de distinto semestre se dicten en el mismo horario (Esto hace referencia a que los módulos de un semestre x no tenga tope de horarios con un porcentaje de las secciones de los módulos de los semestres $x+1$, $x+2$ y $x-1$, $x-2$).

3.3.3. Población

Como el algoritmo itera un número x de veces, esto implica que se generen tantos cromosomas como se defina el tamaño de la población y la cantidad de generaciones también definidas. Es por ello que es necesario identificar cada cromosoma, junto con sus datos relevantes. Para ello se define una población de cromosomas, que varía a medida que se crea una generación nueva.

3.3.4. Definición de población inicial

Los cursos comunes de la Facultad de Ingeniería de la Universidad de Talca son planificados por departamentos, áreas o facultades en específico. Es por ello que la organización o modificación de estos módulos se encuentra fuera de las posibilidades de este algoritmo. Los Directores de escuela reciben la información acerca de la planificación de estos módulos y a partir de esta información es posible rellenar una planilla en donde se almacenen los módulos, con sus determinadas secciones y horarios. A raíz de esta información el algoritmo crea una población inicial, la cual es generada al azar, asignando un día y un horarios aleatorio a cada bloque de cada sección de un módulo. Sin embargo,al generar la población inicial se tienen las siguientes consideraciones:

- A una sección se debe asignar siempre el o los mismos docentes en todos sus horarios.
- Sólo se asignan docentes con disponibilidad horaria a las secciones.

- Los horarios generados dentro de una sección son válidos.

3.3.5. Preservación de mejores soluciones

Para preservar las soluciones que son consideradas como las mejores, se utiliza una cruza destructiva, esto quiere decir que se aceptan los individuos que su función objetivo no es lo suficientemente buena en comparación a sus progenitores.Es decir, a pesar de que estos individuos poseen un resultado deficiente en cuanto a la función objetivo, sirven para preservar la diversidad en la búsqueda de horarios factibles. En efecto, estos cromosomas son considerados como válidos porque si bien no son los mejores, estos podrían generar descendencia que pudiese mejorar con el tiempo, ampliando el área de búsqueda.

Por su parte los individuos que se consideran buenos según su función objetivo, se propagan en las siguientes generaciones, para ello se define que una parte de la población se traspase directamente a la siguiente generación.

3.3.6. Selección de progenitores

La selección de los progenitores se realiza escogiendo dos cromosomas. En la solución de este problema la selección de estos individuos se realiza de tres formas:

- Selección aleatoria: Consiste en la selección aleatoria de dos cromosomas de la población.
- Selección dirigida: Consiste en la selección de dos cromosomas cuya función objetivo sea prometedora.
- Selección mixta: Consiste en la selección de dos cromosomas, uno de ellos cuya función objetivo sea prometedora y el otro de forma aleatoria.

3.3.7. Cruza

Para aumentar la diversidad de los cromosomas, se considera utilizar tres tipos de cruza, las cuales son:

- Cruza de un punto.
- cruza de dos puntos.

- cruza uniforme.

3.3.8. Mutación

Para la operación de mutación se calcula un factor de mutación, en donde este factor puede tomar valores entre 1 y 100, ya que corresponde a la probabilidad de que se produzca o no la mutación. Si el valor del factor es menor o igual al designado como probabilidad de mutación, se aplica un cambio en los genes de una sección del cromosoma que se desea mutar, modificando los horarios aleatoriamente con el fin de crear nuevas posibilidades de solución. Este proceso se realiza para cada uno de los genes del cromosoma pertenecientes a una sección de un módulo. Esta operación podría o no modificar la sección de genes ya que no garantiza que los valores aleatorios generados para realizar la modificación sean distintos a los que tenían originalmente.

4. Desarrollo de la solución

En este capítulo se presenta el desarrollo de la solución. Para esto se explica la estructura de los datos y las decisiones tomadas, en conjunto con las consideraciones y funcionalidades de la estructura del algoritmo genético.

4.1. Lectura de datos de entrada

Para el diseño de un algoritmo genético es de suma importancia organizar los datos que se consideran en el algoritmo. Dentro de estos se encuentran: los cursos que se dictan en un semestre determinado, la cantidad de secciones que posee cada curso y la asignación de horarios de los cursos que están predefinidos. También es necesario contar con información respecto a los prerrequisitos de los cursos, junto con la cantidad de horas de cátedra y laboratorios que estos poseen. Otros datos relevantes a considerar son la información con respecto a qué profesores están encargados de dictar los cursos que se organizan mediante el algoritmo, es decir, aquellos que no están definidos con anterioridad. Por último es necesario contar con la disponibilidad de los profesores.

Para realizar la lectura de estos datos se debe rellenar una planilla excel que es posible descargar desde el propio programa. Dicha planilla debe rellenarse con la información necesaria para que el algoritmo tenga la información suficiente para ejecutarse.

4.1.1. Pre-Requisitos de los cursos

Se define un conjunto de cursos correspondientes a la carrera de Ingeniería Civil en Computación, en donde cada curso posee una duración, un código, un semestre,

un nombre del curso, dos pre-requisitos y la cantidad de horas de cátedra y de laboratorio. La representación de los pre-requisitos es una matriz de n filas, donde cada columna representa un dato relevante para el problema, tal como muestra la Figura 4.1:

Matriz Pre-Requisitos							
Frecuencia	Código	Semestre	Curso	Pre-Requisito1	Pre-Requisito2	Horas Cátedra	Horas Laboratorio
	3407B112	1	Introducción a la ICC			2	2
X	3407B111	1	Introducción a la Programación			3	2
	3407B113	1	Teoría de Sistemas			2	2
X	3407B114	1	Introducción a la Matemática			3	0
	3407B115	1	Cominicación Oral y Escrita I			2	0
X	3407B116	1	Idioma Extranjero I			2	0
X	3407B122	2	Interfaz Humano Computador	3407B112		2	1
X	3407B121	2	Pensamiento Computacional			2	2
X	3407B123	2	Algebra			3	0
X	3407B124	2	Calculo I	3407B114		3	0
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
X	3407B612	11	Proyecto de Titulación	3407B523		2	0
X	3407B611	11	Electivo IV			3	0

Figura 4.1: Matriz pre-requisitos

Donde:

- **Frecuencia:** Indica si el curso es semestral (que se imparte todos los semestres) o que es anual (que se imparte una vez al año) .
- **Código:** Identificador único de cada curso de la carrera de Ingeniería Civil en Computación.
- **Semestre:** Indica a qué semestre pertenece un curso determinado.
- **Curso:** Indica el nombre del curso.
- **Pre-requisito1:** Referencia al curso que es pre-requisito del curso en cuestión.
- **Pre-requisito2:** Referencia al curso que es pre-requisito del curso en cuestión.

4.1.2. Semestre

Se define un conjunto de secciones correspondiente a los cursos que se dictan dentro del semestre. En donde cada sección pertenece a un curso y cada curso pertenece a un semestre. Además, cada sección contiene diversos bloques horarios en donde son dictados. Las secciones son representadas en una matriz de n filas, donde cada

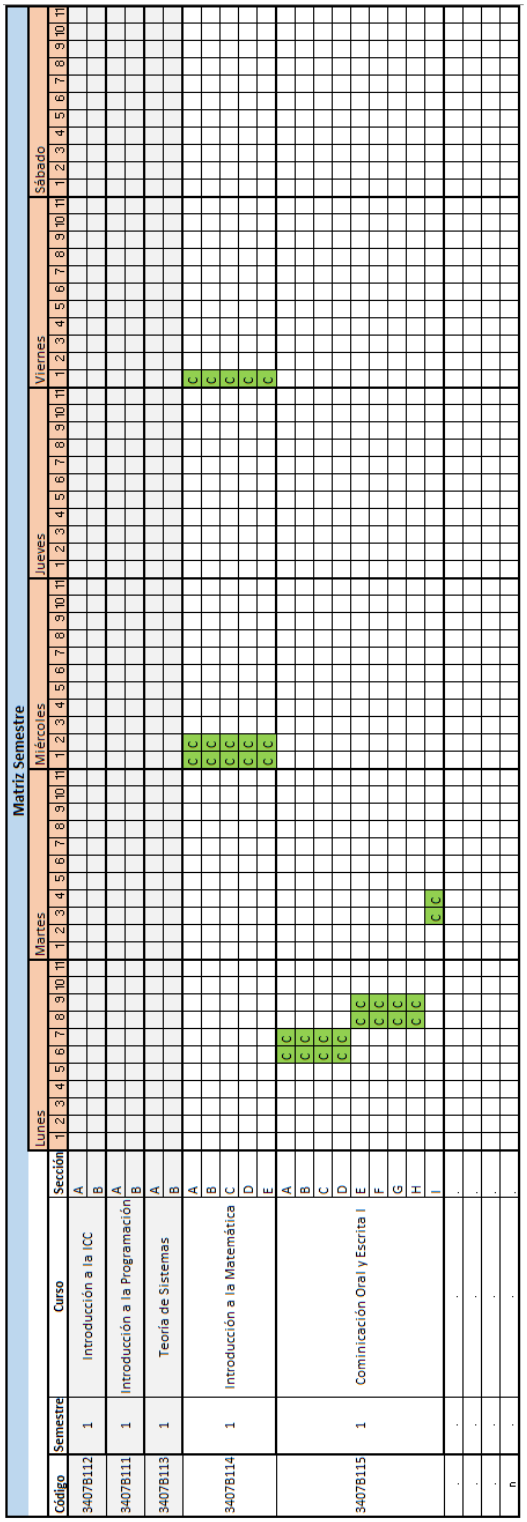


Figura 4.2: Matriz semestre

columna representa un dato relevante para el problema, tal como muestra la Figura 4.2:

Donde:

- **Código:** Referencia al código de la asignatura.
- **Semestre:** Corresponde al número del semestre al cual pertenece el curso.
- **Curso:** Corresponde al nombre del curso.
- **Sección:** Identificador de la sección del curso en cuestión.
- **Horario:** Corresponde al horario en el que se dicta el curso. Este puede ser en más de uno, siempre y cuando se encuentren dentro de los bloques 1 al 11 y dentro de los días Lunes a Sábado.

4.1.3. Disponibilidad Profesor

Se define un conjunto de profesores, los cuales son los encargados de dictar los cursos propios de la carrera de Ingeniería Civil en Computación. Dentro de este conjunto de datos se encuentran los bloques en los cuales el profesor tiene o no disponibilidad de realizar una clase.

Los profesores son representados en una matriz de n filas, donde cada columna representa un dato relevante para el problema, tal como muestra la Figura 4.3:

Donde:

- **Código:** Indicador único de cada profesor de la carrera de Ingeniería Civil en Computación.
- **Nombre Profesor:** Corresponde al nombre del profesor.
- **Horario:** Corresponde al horario en el que el profesor tiene o no disponibilidad para dictar un curso. Este puede ser en más de uno, siempre y cuando se encuentren dentro de los bloques 1 al 11 y dentro de los días Lunes a Sábado.

4.1.4. Curso Profesor

Se define un conjunto de secciones correspondiente a los cursos que se dictan dentro del semestre. En donde cada sección pertenece a un curso. Cada una de estas

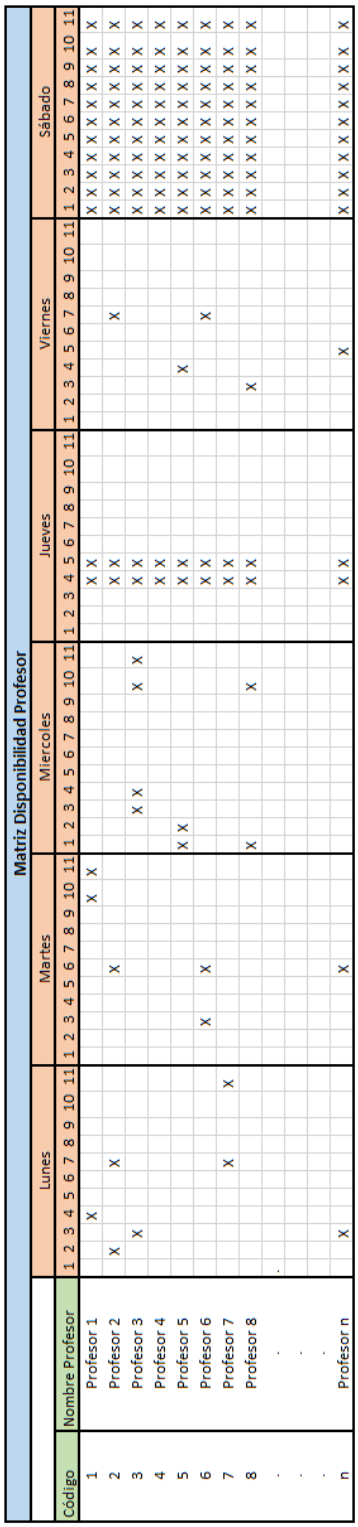


Figura 4.3: Matriz disponibilidad profesor

secciones tiene asignado a lo más dos profesores y a lo menos uno. Esta asignación se realiza referenciando el código del profesor que dicta una sección en específico.

Esta asignación es representada en una matriz de n filas, donde cada columna representa un dato relevante para el problema, tal como muestra la Figura 4.4:

Matriz Curso Profesor				
Código	Curso	Sección	Profesor 1	Profesor 2
3407B112	Introducción a la ICC	A	3	
		B	3	
3407B111	Introducción a la Programación	A	1	8
		B	1	8
3407B113	Teoría de Sistemas	A	2	
		B	2	
3407B121	Pensamiento Computacional	A	5	
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
n	.	.	.	.

Figura 4.4: Matriz curso profesor

Donde:

- **Código:** Identificador único de cada curso de la carrera de Ingeniería Civil en Computación.
- **Curso:** Indica el nombre del curso.
- **Sección:** Identificador de la sección del curso en cuestión.
- **Profesor 1:** Identificador único de cada profesor de la carrera de Ingeniería Civil en Computación.
- **Profesor 2:** Identificador único de cada profesor de la carrera de Ingeniería Civil en Computación.

4.2. Gen

En el caso particular de esta solución, se considera el gen como la estructura que constituye a un cromosoma para la transmisión de caracteres hereditarios. El

gen está compuesto por información relevante para la solución del problema que se presenta.

Cada gen se compone de distintos atributos, dentro de ellos se encuentran:

- **ID:** Corresponde al identificador único del gen, en un rango desde 0 a N_g, donde N_g depende de la cantidad de secciones que la planificación horaria tenga por la cantidad de horas de cátedra más la cantidad de horas de laboratorio posea dicha sección.
- **Asignatura:** Corresponde a la asignatura a la cual pertenece el gen.
- **Sección:** Corresponde a la sección de la asignatura a la cual pertenece el gen.
- **Horario:** Corresponde al horario del gen. Este puede ser de Lunes a Sábado entre los bloques 1 y 11.
- **Modificable:** Indica si el gen es modificable o no. En caso de corresponder a una sección cuyos horarios hayan sido ingresados manualmente, esta no podrá ser modificable ya que se encuentra definida con anterioridad. En caso de no ser completada esa información, entonces se considera modificable.
- **TipoClase:** Indica si el gen corresponde a una hora de cátedra o una de laboratorio.

Los genes siguen una estructura definida, en donde cada uno de ellos corresponde a un bloque horario de una sección perteneciente a una asignatura. Cabe destacar que los genes no cambian su estructura. Si el gen cuyo ID es 0 y corresponde a la sección "A" del curso de "Requisitos de Software", este siempre es así. En caso de que este gen sea modificable, solo cambia el horario del gen, pero el resto de su estructura se mantendrá igual.

4.3. Cromosoma

El cromosoma corresponde a una posible solución del problema. Es una combinación de genes generada aleatoriamente, en donde esta solución puede ser correcta o no. Para ser más específicos, un cromosoma está compuesto por un listado de genes generados al azar, el cual más adelante irá siendo evaluado para saber qué tan bueno es.

El cromosoma está compuesto por información relevante para la solución del problema. Se compone de distintos atributos, dentro de ellos se encuentran:

- **ID:** Corresponde al identificador único del Cromosoma, el rango va desde 0 a N_c, donde N_c depende de la cantidad de la población y de iteración que posea el algoritmo.
- **Generación:** Corresponde a la generación a la cual pertenece el cromosoma.
- **Conflictos:** Corresponde a la cantidad de conflictos que posee el cromosoma, los cuales se determinan aplicando la función objetivo que se explica más adelante. Se separan entre conflictos obligatorios y conflictos deseables.
- **ListadoGenes:** Corresponde al listado de genes que posee el cromosoma.

4.4. Población

La población corresponde a dos listados de cromosomas, uno de ellos almacena los cromosomas más prometedores, es decir, que tengan una menor cantidad de conflictos, mientras que el otro listado corresponde al resto de la población, o denominada población normal. El tamaño de esta población es de T cromosomas, dividiéndose P de ellos en la población prometedora y R en la población normal, donde $P + R = T$. Esta cantidad puede ser modificada por el usuario, pudiendo éste aumentar o disminuir la cantidad de la población.

$$T = P + R$$

4.5. Población inicial

Como se mencionó en la Sección 3.3.4, a través de todo el procesamiento, el algoritmo puede generar tantos cromosomas como se defina en el tamaño de la población multiplicado por la cantidad de generaciones. En principio, se genera una población inicial de T individuos. Una vez generada esta población inicial se procede a evaluar a los individuos mediante la función objetivo, la cual calcula la cantidad de conflictos que estos tengan. Una vez realizado lo anterior se ordena el listado de cromosomas

creados inicialmente de menor a mayor según la cantidad de conflictos. Para finalmente agregar a la población prometedora los P individuos cuya función objetivo sea menor y a la población normal los R individuos restantes.

4.6. Función objetivo

Como se mencionó en la Sección 3.3.2, la función objetivo evalúa al cromosoma, contando la cantidad de conflictos que este posee. Para ello considera una serie de restricciones que en el caso de que una de ellas sea violada, la función objetivo suma un conflicto. Para realizar esta evaluación se recorre cada gen del cromosoma, evaluando uno a uno la cantidad de restricciones violadas por el gen. Dentro de las restricciones se encuentran:

- **los profesores pertenecientes al gen deben estar disponibles en el horario del gen:** Para realizar esta verificación se agrupan los genes cuya asignatura corresponde a la de un profesor. Verificando una a una si esta está dentro del horario disponible del profesor.
- **los genes que tengan el mismo profesor no deben tener el mismo horario:** Para realizar esta verificación se agrupan los genes cuya asignatura corresponde a la de un profesor. Utilizando un arreglo auxiliar se almacenan los horarios que ya están siendo ocupados por los genes. En caso de que al verificarse uno de ellos se encuentre ocupado, entonces se cuenta un conflicto.
- **los genes que pertenecientes a un mismo semestre no deben tener el mismo horario:** Para realizar esta verificación se agrupan los genes por semestre, posterior a ello se hace uso de un HashMap, donde se almacenan todos los cursos con sus respectivas secciones. Una vez realizado lo anterior, se revisa una a una las secciones del HashMap y se cuenta la cantidad de conflictos que existen entre los cursos de un semestre, verificando que los cursos que se imparten dentro de un mismo semestre no choquen entre sí. Cabe mencionar que esta verificación cuenta un conflicto siempre y cuando ambos cursos que se estén revisando tengan conflicto con más del 50 % de las secciones del otro curso. Por otro lado, la revisión de los cursos considera que la asignatura que se está revisando sea modificable, de esta forma se evita contar conflictos innecesarios ya que los cursos que vienen predefinidos son imposibles de modificar.

- **los genes que pertenecientes a semestres consecutivos no deben tener el mismo horario:** Para realizar esta verificación se agrupan los genes por semestre. Posterior a ello se hace uso de un HashMap, donde se almacenan todos los cursos con sus respectivas secciones. Una vez realizado lo anterior, se realiza una revisión por semestre, en donde se toman en cuenta a lo más dos semestres anteriores y dos semestres posteriores al que se está revisando. Para ello se revisa una a una las secciones del HashMap y se cuenta la cantidad de conflictos que existen entre los cursos de tales semestres, verificando que los cursos que se imparten dentro de los semestres que se están revisando no choquen entre sí. Cabe mencionar que esta verificación cuenta un conflicto siempre y cuando ambos cursos que se estén revisando tengan conflicto con más del 80 % de las secciones del otro curso. Por otro lado, la revisión de los cursos considera que la asignatura que se está revisando sea modificable, de esta forma se evita contar conflictos innecesarios ya que los cursos que vienen predefinidos son imposibles de modificar. Otro punto a considerar es que no se revisan cursos que son pre-requisitos del curso que se está revisando ya que no existiría un alumno cursando esos módulos simultáneamente.

Cabe destacar que dentro de la función objetivo o fitness se clasifican dos tipos de conflictos, los conflictos débiles y los confictos fuertes. Los conflictos débiles son aquellos que no invalidan una solución, es decir que a pesar de que la solución cuenta con ellos, esta puede ser válida, por lo que los genes que pertenecen a semestres consecutivos podrían eventualmente tener el mismo horario.

Por su parte los conflictos fuertes son aquellos que invalidan la solución, es decir, que al presentarse a lo menos uno dentro de la solución encontrada, esta se considera automáticamente como una solución inválida. Es por esto que los genes que pertenecen a un mismo semestre no podrían tener el mismo horario, los genes que tengan el mismo profesor no podrían tener el mismo horario y los profesores pertenecientes al gen deben estar disponibles en el horario del gen. De lo contrario se suman a la función objetivo un conflicto fuerte.

Es importante destacar que cuando se habla de una solución inválida, no significa que el programa la descarte, ya que lo que se pretende es encontrar una solución con la menor cantidad de conflictos violados posibles. Por lo que si bien, encontrar soluciones que posean conflictos fuertes no es lo ideal, podría ser esta la única solución encontrada dadas las restricciones impuestas inicialmente.

4.7. Preservación de mejores soluciones

Para preservar las mejores soluciones, la población posee un arreglo en donde se almacenan los cromosomas que tienen una menor cantidad de conflictos. Esta población es denominada población prometedora P y está ordenada de menor a mayor cantidad de conflictos.

4.8. Selección de progenitores

La selección de los progenitores es un método que selecciona pares de progenitores. Para ello se utilizan tres métodos de selección, el método de selección aleatoria, el método de selección prometedora y el método de selección mixta. Cada uno de ellos con un porcentaje distinto, es decir, que del 100 % de las parejas de cromosomas que son seleccionados como progenitores, a % son pares escogidos de la población normal, b % de la población prometedora y c % de forma mixta, es decir, un cromosoma de la población normal y otro de la población prometedora, donde, $a\,\% + b\,\% + c\,\% = 100\,\%$. Estas parejas de cromosomas seleccionadas son almacenadas en un arreglo. De allí se seleccionan posteriormente las parejas de cromosomas para realizar la cruza entre ellas.

4.9. Función de cruza

La cruza corresponde a una combinación de dos cromosomas, estas pueden ser de tres tipos, cruza de un punto, cruza de dos puntos o cruza uniforme (estas se encuentran descritas en la Sección 2.5.5). Cada una de estas cruzas se aplican en el algoritmo, claro que cada una de ellas con distinto porcentaje, es decir, el x % de las cruzas se realiza con el método de cruza de un punto, el y % de las cruzas se realiza con el método de cruza de dos puntos y el z % de las cruzas se realiza con el método de cruza uniforme, donde $x\,\%+y\,\%+z\,\% = 100\,\%$.

Cabe mencionar que cada proceso de cruza tiene como resultado dos cromosomas descendientes, los cuales forman parte de la población de la siguiente generación.

4.10. Función de mutación

La función de mutación consiste en la modificación de una pequeña parte del listado de genes de un cromosoma. Para ello se calcula un factor de mutación, el cual puede tomar valores entre 1 y 100, ya que corresponde a la probabilidad, vista como porcentaje, de que se produzca o no la mutación. En caso de que se produzca, se seleccionan todos los genes que pertenecen a una sección de un módulo (vale decir que esto modifica todos los horarios de la sección que corresponde a esa asignatura), a los cuales se les asignan un horario aleatorio. Este nuevo horario puede ser distinto al horario que poseía el gen anterior a la mutación, sin embargo, existe una posibilidad muy baja de que al realizar la mutación los nuevos horarios designados sean los mismos que poseían los genes antes de realizar la mutación.

4.11. Algoritmo genético

El algoritmo genético que se muestra en el Algoritmo X realiza como primera ejecución la creación de la población inicial (Linea 4). Continúa con una dinámica iterativa (Lineas 5 a 15), en donde a partir de la población creada con anterioridad se realiza la selección de progenitores (Linea 7). Una vez realizada esta selección se procede a realizar la cruza entre los pares de cromosomas progenitores seleccionados (Linea 8). Cabe mencionar que al finalizar la cruza este genera dos cromosomas descendientes los cuales son sometidos (de acuerdo al factor de mutación) a una posible mutación, para luego ser evaluados con la función de fitness (también en Linea 8). Una vez finalizada la cruza, se agregan los cromosomas descendientes a la nueva población (Linea 9), la cual será utilizada para realizar una nueva generación siguiendo el mismo procedimiento anteriormente descrito. Finalmente, se selecciona la mejor solución encontrada hasta el momento (Linea 10). Este proceso se repite hasta alcanzar el máximo número de generaciones consideradas para la ejecución del algoritmo genético (Linea 5). No obstante lo anterior, el procedimiento se podría interrumpir cuando la mejor solución encontrada ya no presente conflicto alguno (Línea 13).

Algoritmo 1: Algoritmo Genético

Entrada: CG, CPN, CPP, $PVCMS$, $PVCDS$, PSA, PSM, PSD, $PCUP$, $PCDP$, PCU

Salida : $mejorSolucionEncontrada$

Resultado: Se devuelve la mejor solución encontrada $mejorSolucionEncontrada$ dada la configuración de Cantidad de Generaciones CG, Cantidad Población Normal CPN, Cantidad de Población Prometedora CPP,Porcentaje Validación Cursos Mismo Semestre $PVCMS$, Porcentaje Validación Cursos Distinto Semestre $PVDS$, Porcentaje Selección Aleatoria PSA, Porcentaje Selección Mixta PSM, Porcentaje Selección Dirigida PSD, Porcentaje Cruza Un Punto $PCUP$, Porcentaje Cruza Dos Puntos $PCDP$ y el Porcentaje Cruza Uniforme PCU.

```
1  listadoProgenitores ⟵ [ ];
2  nuevaGeneracion ⟵ [ ];
3  Cromosoma mejorSolucionEncontrada ⟵ ∅;
4  Poblacion poblacion ⟵ generarPoblacionInicial(CPN, CPP);
5  while i < CG do
6  |  if mejorSolucionEncontrada.getTotalConflictos()! = 0 then
7  |  |  listadoProgenitores ⟵ seleccion(poblacion, PSA, PSM, PSM);
8  |  |  nuevaGeneracion ⟵
   |  |    cruza(listadoProgenitores, PCUP, PCDP, PCU);
9  |  |  agregarDescendientesPoblacion(poblacion, nuevaGeneracion);
10 |  |  mejorSolucionEncontrada ⟵ poblacion.getMejorSolucion();
11 |  end
12 |  else
13 |  |  return mejorSolucionEncontrada;
14 |  end
15 end
16 return mejorSolucionEncontrada;
```

4.12. Funcionamiento del programa

El programa de asignación de horarios consiste en una herramienta que permite llevar a cabo la organización de horarios de clases, minimizando la cantidad de restricciones violadas hasta encontrar, si es posible, una asignación factible o agotar el número de iteraciones permitidas. Este es un programa de escritorio que funciona a partir de una asignación horaria inicial, la cual puede tener asignadas algunas asignaturas y secciones o no. En adelante, los términos asignatura, cursos y módulos son usados de forma indistinguible. A partir de esta asignación inicial que toma en consideración los cursos ya designados, el programa organiza los horarios de los cursos incompletos, manteniendo la asignación de los cursos organizados inicialmente. En caso de que la organización horaria inicial se encuentre vacía, el programa genera la planificación sin considerar asignaciones de cursos predefinidos, por lo que cuenta con total libertad para realizar la asignación de todos los cursos que se encuentren sin asignación horaria.

4.12.1. Datos de entrada

Para que el programa realice la organización horaria, es necesario entregarle información inicial, por lo que cuenta con una planilla en formato Microsoft Excel en donde se almacena información acerca de los distintos cursos y sus respectivas secciones. Es necesario indicar los cursos y secciones de cada uno de ellos que se desean planificar. También es necesario indicar los profesores que dictan cada una de las secciones de los distintos cursos. Por ello el archivo de la planilla Excel cuenta con distintas hojas en donde se especifica distinta información relevante para la planificación de horarios. Dentro de una de las hojas del Excel es necesario indicar la disponibilidad horaria de los profesores. A continuación, se explica en detalle cada una de las hojas del documento en formato MS Excel

Hoja Semestre

Existen dos hojas semestre, la del semestre par y la del semestre impar. En ellas es necesario indicar qué cursos se dictan en dicho semestre, además de las secciones que corresponden a cada curso. Dentro de esta hoja se puede indicar también aquellos cursos y secciones que se encuentran predefinidos, es decir, dichas secciones van

a mantener sus bloques horarios definidos con anterioridad. Esto permite que el planificador de horarios tome en cuenta dicha información y genere los horarios de los cursos cuyo horario se encuentre sin definir.

Para definir los horarios de cada sección es necesario marcar dentro de la hoja semestre el bloque en el cual se va a dictar la sección, En la hoja semestre se encuentran definidos los bloques de cada día de la semana en los que se realizan clases. Para definir el horario en que se va a dictar una clase de una sección es necesario marcar el bloque en que se va a dictar. Para ello es necesario marcar el bloque definiendo a qué tipo de clase pertenece ese bloque. Por ejemplo, si se desea definir una hora de clases que corresponde a cátedra es necesario marcar el bloque con la letra "C" o "c" y si se desea definir una hora de clases que corresponde a un laboratorio es necesario marcar el bloque con la letra "L" o "l". En la Figura 4.2, página 38 se muestra un ejemplo de una hoja semestre.

Hoja CursoProfesor

Existen dos hojas CursoProfesor, la correspondiente a semestre par y otra al impar. En ellas es necesario indicar qué profesor o profesores dictan cada sección de cada uno de los cursos. Para ello es necesario referenciar al código del profesor que está indicado en la hoja DisponibilidadProfesor del Excel. Es posible asignar a lo más dos profesores por sección. En la Figura 4.4, página 41 se muestra un ejemplo de la hoja CursoProfesor.

Hoja DisponibilidadProfesor

Existen dos hojas DisponibilidadProfesor, una para el semestre par y otra para el semestre impar. En ellas es necesario indica el nombre y el código de los profesores, además de los horarios en que un profesor se encuentra disponible o no. Para indicar que un profesor no se encuentra disponible en un bloque es necesario marcar con una "X" o una "x"; en caso de que se encuentre disponible en cierto bloque este debe permanecer vacío. En la Figura 4.3, página 40 se muestra un ejemplo de la hoja DisponibilidadProfesor.

4.12.2. Parámetros

El programa viene con una configuración predefinida, en donde los valores de algunos parámetros están asignados de forma tal de que la aplicación se comporte eficientemente. Sin embargo, el usuario puede definir y modificar los parámetros dentro de la ejecución del programa. Cabe destacar que la configuración definida por el usuario se mantendrá sólo en la ejecución en la cual se realiza la modificación. Esto para garantizar que los parámetros definidos por el programa no sean alterados definitivamente y así mantener una configuración estándar para futuras organizaciones de horarios. La configuración puede ser modificada sólo dentro de la ejecución. Una vez cerrado el programa y vuelto a iniciar, la configuración mantendrá sus parámetros establecidos originalmente.

Dentro del programa es posible configurar los siguientes parámetros:

Cantidad Generaciones

Esta sección recibe como parámetro un número entero, el cual define la cantidad de generaciones del algoritmo genético. Indica la cantidad de veces que este itera.

Cantidad Población

Esta sección recibe como parámetro un número entero, el cual define la cantidad de la población normal del algoritmo.

Cantidad Población Prometedora:

Esta sección recibe como parámetro un número entero, el cual define la cantidad de la población prometedora del algoritmo.

Porcentaje Validación Cursos Mismo Semestre

Esta sección recibe como parámetro un porcentaje, el cual se utiliza para medir la función objetivo. Para ello evalúa la cantidad de secciones de un curso con la cantidad de secciones de otro (de un mismo semestre) y si esta colisiona con el tanto porciento definido de secciones, entonces se cuenta un conflicto.

Porcentaje Validación Cursos Distinto Semestre

Esta sección recibe como parámetro un porcentaje, el cual se utiliza para medir la función objetivo. Para ello evalúa la cantidad de secciones de un curso con la cantidad de secciones de otro (de semestres consecutivos) y si esta colisiona con el tanto porciento definido de secciones, entonces se cuenta un conflicto.

Cantidad Cromosomas Selección Aleatoria

Esta sección recibe como parámetro un número entero, el cual define cuántos pares de cromosomas de la población serán escogidos de la población normal.

Cantidad Cromosomas Selección Mixta

Esta sección recibe como parámetro un número entero, el cual define cuántos pares de cromosomas de la población serán escogidos desde la población normal y de la población aleatoria.

Cantidad Cromosomas Selección Dirigida

Esta sección recibe como parámetro un número entero, el cual define cuántos pares de cromosomas de la población serán escogidos de la población prometedora.

Cantidad Cromosomas Cruza Un Punto

Esta sección recibe como parámetro un número entero, el cual define cuántos pares de cromosomas descendientes serán creados mediante la cruza de un punto.

Cantidad Cromosomas Cruza Dos Puntos

Esta sección recibe como parámetro un número entero, el cual define cuántos pares de cromosomas descendientes serán creados mediante la cruza de dos puntos.

Cantidad Cromosomas Cruza Uniforme

Esta sección recibe como parámetro un número entero, el cual define cuántos pares de cromosomas descendientes serán creados mediante la cruza uniforme.

4.12.3. Criterio de parada

Al momento de ejecutar el programa se toma en consideración los parámetros configurados, dentro de estos parámetros se encuentra la Cantidad de Generaciones, la cual como se menciona anteriormente es el número de iteraciones que el programa realiza para encontrar una solución. Es decir, como máximo el programa realiza tantas generaciones como esté configurado el programa para encontrar una solución. Sin embargo, es posible que el programa tarde menos generaciones, ya que si en una generación la cantidad de conflictos totales (cantidad de conflictos débiles más la cantidad de conflictos fuertes) es igual a cero, entonces el programa deja de iterar y es posible descargar la solución encontrada.

Cabe destacar que el programa podría no encontrar la solución, en este caso el programa entrega la mejor solución encontrada, es decir, la solución cuya cantidad de conflictos fuertes sea menor cuando se agote la cantidad de generaciones.

5. Análisis y evaluación de resultados

El objetivo del análisis y evaluación de resultados es probar el funcionamiento del algoritmo genético explicado en el Capítulo 4. Junto con esto, también se verifica que los resultados obtenidos consideren los requisitos de respetar la disponibilidad de profesores, la asignación de profesores a un solo curso dentro de un mismo horario, la asignación de cursos de un mismo semestre en horarios distintos y la asignación de cursos de distinto semestre en horarios distintos dado un porcentaje de aceptación.

Para ello se realizan pruebas al código, en donde se consideran pruebas unitarias y de integración. También se hace una evaluación experimental, donde se analiza la eficiencia de la aplicación. Por último, se realizan pruebas de usuario, en donde se evalúa la experiencia del usuario.

Por economía de espacio, dentro de este informe se omite la documentación de las pruebas al código, por lo que sólo se muestra la evaluación experimental y las pruebas de usuario.

5.1. Evaluación Experimental

Dada la naturaleza de este problema, es posible realizar diversas pruebas de eficiencia o desempeño al programa. Existen diversos parámetros que se pueden estudiar dentro del algoritmo genético implementado. Dentro de ellos se encuentran: la cantidad de generaciones, el tamaño de las generaciones, la cual está constituido por la cantidad de la población normal más la cantidad de población prometedora. También es posible estudiar el porcentaje de los tipos de selección para el algoritmo,

ya sea selección aleatoria, selección mixta o selección dirigida. Por otro lado podría estudiarse el porcentaje de mutación y el porcentaje para la cruza, evaluando que proporción de ellos entrega mejores resultados. También se podría estudiar el efecto que tiene la disponibilidad horaria de los profesores, etc.

Por restricciones de tiempo para el desarrollo y después de una serie de experimentos preliminares, se decide tomar un enfoque hacia el estudio de la disponibilidad de los profesores. En efecto, los experimentos preliminares mostraron que el factor determinante para encontrar una solución factible es justamente la disponibilidad horaria de los profesores. El resto de los parámetros tienen relevancia desde el punto de vista del desempeño de la aplicación, pero no de su eficacia. Explicado de otra forma, realizar el estudio de los parámetros anteriormente mencionados y encontrar los valores que se ajusten de mejor manera al problema puede conllevar a que el algoritmo funcione más rápido o más lento, pero no que éste encuentre la solución. En cambio, si existe poca disponibilidad de profesores podría no existir solución.

Con la realización del estudio a la disponibilidad de profesores se pretende encontrar un porcentaje estimado de bloques horarios disponibles que debe poseer un profesor para que el algoritmo pueda encontrar una solución. Para ello se realizan dos experimentos los cuales están descritos en las Secciones 5.1.1 y 5.1.2

5.1.1. Experimento 1

El Experimento 1 considera dos planificaciones horarias idénticas, cada una cuenta con la planificación de los cursos predefinidos de los semestres pares e impares. En la primera planilla, la cual lleva por nombre “Organización Cursos1”, posee una gran disponibilidad horaria de profesores. Por su parte la segunda planilla, la cual lleva por nombre “Organización Cursos2”, posee una menor cantidad de horarios disponibles por profesor, por lo que debiese ser más difícil (en el sentido de tener que revisar varias generaciones de cromosomas) encontrar la solución.

Para este experimento se realizan 10 ejecuciones por cada organización horaria, es decir, 10 iteraciones para el semestre par de la planilla “Organización Cursos1” y 10 iteraciones para el semestre impar de esta misma. De la misma forma se realizan las 10 iteracones para cada semestre de la planilla “Organización Cursos2”. Cabe destacar que para la organización de semestre par e impar de cada planilla se utiliza la misma disponibilidad de profesores. Sin embargo, en cada planilla, la disponibilidad

de profesores es distinta.

Las figuras 5.1y 5.2 muestra la disponibilidad horaria a utilizar en las planillas "Organización Cursos1" y "Organización Cursos2".

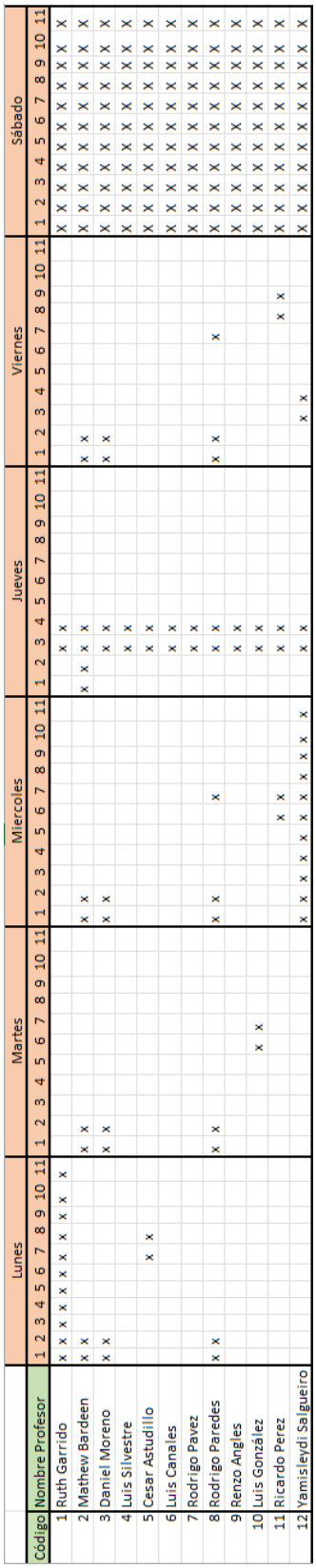
- **Organización Cursos1 Impar:** Esta planilla corresponde a la planificación de una asignación horaria de un semestre impar. Allí se encuentran definidos los cursos que se dictan en un semestre impar y que vienen predefinidos por los departamentos externos a los de la Escuela de Ingeniería Civil en Computación. Esta planificación cuenta con una alta disponibilidad horaria de profesores. Esto se ve reflejado en la Figura 5.1.

- **Organización Cursos1 Par:** Esta planilla corresponde a la planificación de una asignación horaria de un semestre par. Allí se encuentran definidos los cursos que se dictan en un semestre par y que vienen predefinidos por los departamentos externos a los de la Escuela de Ingeniería Civil en Computación. Esta planificación cuenta con una alta disponibilidad horaria de profesores. Esto también se ve reflejado en la figura 5.1.

- **Organización Cursos2 Impar:** Esta planilla corresponde a la planificación de una asignación horaria de un semestre impar. Allí se encuentran definidos los cursos que se dictan en un semestre impar y que vienen predefinidos por los departamentos externos a los de la Escuela de Ingeniería Civil en Computación. Esta planificación cuenta con una disponibilidad razonable de profesores, semejante a la que tendrían dentro de un semestre regular. Esto se ve reflejado en la Figura 5.2.

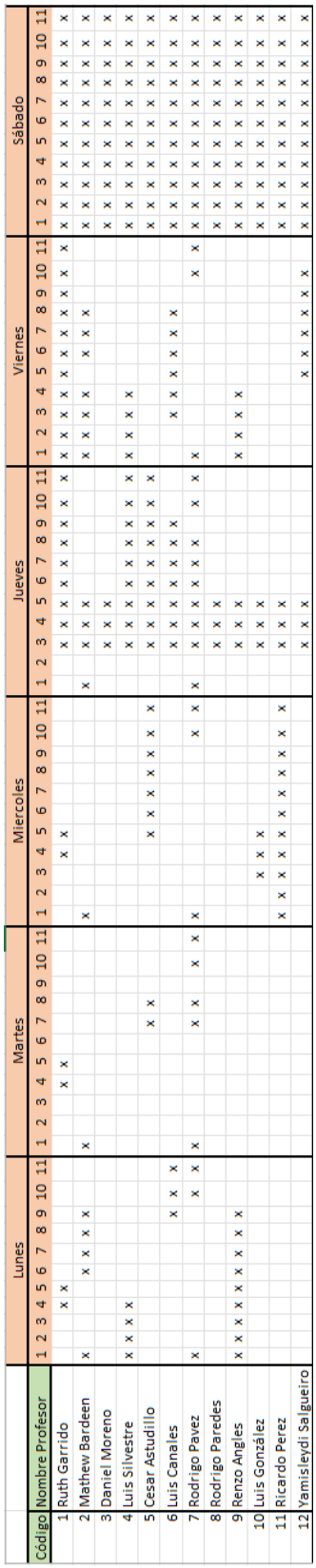
- **Organización Cursos2 Par:** Esta planilla corresponde a la planificación de una asignación horaria de un semestre par. Allí se encuentran definidos los cursos que se dictan en un semestre par y que vienen predefinidos por los departamentos externos a los de la Escuela de Ingeniería Civil en Computación. Esta planificación cuenta con una disponibilidad razonable de profesores, semejante a la que tendrían dentro de un semestre regular. Esto se ve reflejado en la Figura 5.2.

Tomando en consideración las planillas anteriormente mencionadas. Los resultados de las ejecuciones se muestran en los Cuadros 5.1 y 5.2:



Código	Nombre Profesor	Lunes											Martes											Miercoles											Jueves											Viernes											Sábado										
		1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11
1	Ruth Garrido	x	x	x	x	x	x	x	x	x	x	x															x	x																																						
2	Mathew Bardeen	x	x										x	x											x	x	x									x	x											x	x											x	x	x	x	x	x	x	x	x	x	x
3	Daniel Moreno	x	x										x	x												x	x									x	x											x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
4	Luis Silvestre																										x	x																																						
5	Cesar Astudillo							x	x																		x	x																																					
6	Luis Canales																										x	x																																				
7	Rodrigo Pavez																										x	x																																				
8	Rodrigo Paredes	x	x										x	x					x								x	x			x					x	x											x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
9	Renzo Angles																										x	x																																				
10	Luis González							x	x																		x	x																																				
11	Ricardo Perez																		x	x							x	x																																			
12	Yamisleydi Salgueiro																							x	x	x	x	x	x	x	x	x	x	x																														

Figura 5.1: Matriz disponibilidad profesor planilla Organización Cursos1



Código	Nombre Profesor	Lunes											Martes											Miercoles											Jueves											Viernes											Sábado										
		1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11	1	2	3	4	5	6	7	8	9	10	11
1	Ruth Garrido				x	x										x	x										x	x										x	x										x	x										x	x									
2	Mathew Bardeen	x					x	x	x	x			x											x					x	x	x																																				
3	Daniel Moreno		x	x	x																								x	x	x																																			
4	Luis Silvestre	x	x	x																									x	x	x				x	x	x																													
5	Cesar Astudillo								x	x																			x	x	x																																		
6	Luis Canales										x	x																	x	x	x																																	
7	Rodrigo Pavez	x																	x	x		x	x																																									
8	Rodrigo Paredes																																																																	
9	Renzo Angles	x	x	x	x	x	x	x	x	x																			x	x	x																																		
10	Luis González															x	x	x																																														
11	Ricardo Perez													x	x	x	x	x	x	x	x	x	x																																											
12	Yamisleydi Salgueiro																																																																	

Figura 5.2: Matriz disponibilidad profesor planilla Organización Cursos2

En el Cuadro 5.1 se puede observar que para llegar a una solución para el semestre impar de 0 conflictos, como mínimo se necesitaron 39.600 cromosomas y a lo más 66.000 cromosomas, lo que equivale a realizar una revisión de 33 y 55 generaciones respectivamente. Sin embargo, esto no quiere decir que estos valores sean los que se necesitan siempre para encontrar la solución, ya que al generarse de forma aleatoria, es probable que estos valores varíen en cada ejecución. Por lo que buscar en 55 generaciones no garantiza encontrar una solución de 0 conflictos para este problema. Lo mismo ocurre para el semestre par, que en su caso necesitó de 37.200 cromosomas como mínimo para llegar a una solución de 0 conflictos totales y un máximo de 58.800 cromosomas, revisando así 31 y 49 generaciones respectivamente.

En el Cuadro 5.2 se puede observar que para llegar a una solución para el semestre impar de 0 conflictos, como mínimo se necesitaron 43.200 cromosomas y a lo más 234.000 cromosomas, lo que equivale a realizar una revisión de entre 36 y 195 generaciones, respectivamente. Por su parte en el semestre par se puede observar con mayor claridad la diferencia que puede llegar a existir en la cantidad de cromosomas y generaciones necesarias a revisar para poder llegar a una solución. Como se observa en la tabla, para el semestre par como mínimo se necesitó de 57.600 cromosomas para llegar a una solución cuyo conflictos totales fuese de 0. Sin embargo, el máximo de cromosomas que tuvo que revisar fue de 74.330.400, lo que equivale a más de 1.290 veces el mínimo.

Dado los resultados anteriores, en promedio, para la organización del semestre impar de la planilla "Organización Cursos1" se necesitaron 42 generaciones, es decir, la evaluación de 50.400 cromosomas. Por su parte, el semestre par de la misma planilla necesitó un promedio de 38 generaciones, es decir, tuvo que revisar 45.600 cromosomas. Por otro lado, la planilla "Organización Cursos2" que posee una menor disponibilidad horaria de profesores, para el semestre impar tuvo que revisar un total de 82.800 cromosomas, es decir, 69 generaciones, necesitando así un 64,2 % más de generaciones que para la disponibilidad horaria de la planilla "Organización Cursos1". Por su parte el semestre par tuvo mayor dificultades para encontrar la solución, tuvo que revisar en promedio 9.446 generaciones, lo que equivale a 11.335.200 cromosomas. En este caso el el semestre par de la planilla "Organización Cursos2" revisa un 24.757,8 % más que el semestre par de la planilla "Organización Cursos1".

Población total	**Generaciones evaluadas**	**Total cromosomas evaluados**	**Total conflictos**	**Solución encontrada**
Planilla Organización Cursos1 Impar				
1.200	39	46.800	0	Sí
1.200	46	55.200	0	Sí
1.200	41	49.200	0	Sí
1.200	33	39.600	0	Sí
1.200	44	52.800	0	Sí
1.200	42	50.400	0	Sí
1.200	55	66.000	0	Sí
1.200	33	39.600	0	Sí
1.200	38	45.600	0	Sí
1.200	47	56.400	0	Sí
Planilla Organización Cursos1 Par				
1.200	32	38.400	0	Sí
1.200	38	45.600	0	Sí
1.200	36	43.200	0	Sí
1.200	31	37.200	0	Sí
1.200	49	58.800	0	Sí
1.200	35	42.000	0	Sí
1.200	39	46.800	0	Sí
1.200	35	42.000	0	Sí
1.200	44	52.800	0	Sí
1.200	36	43.200	0	Sí

Cuadro 5.1: Resultados ejecución programa organizador de horarios, utilizando la planilla Organización Cursos1

Población total	**Generaciones evaluadas**	**Total cromosomas evaluados**	**Total conflictos**	**Solución encontrada**
Planilla Organización Cursos2 Impar				
1.200	45	54.000	0	Sí
1.200	67	80.400	0	Sí
1.200	85	102.000	0	Sí
1.200	53	63.600	0	Sí
1.200	45	54.000	0	Sí
1.200	195	234.000	0	Sí
1.200	49	58.800	0	Sí
1.200	36	43.200	0	Sí
1.200	53	63.600	0	Sí
1.200	62	74.400	0	Sí
Planilla Organización Cursos2 Par				
1.200	133	159.000	0	Sí
1.200	86	103.200	0	Sí
1.200	61.942	74.330.400	0	Sí
1.200	48	57.600	0	Sí
1.200	117	140.400	0	Sí
1.200	300	360.000	0	Sí
1.200	31.469	37.762.800	0	Sí
1.200	73	87.600	0	Sí
1.200	107	128.400	0	Sí
1.200	185	222.000	0	Sí

Cuadro 5.2: Resultados ejecución programa organizador de horarios, utilizando la planilla Organización Cursos2

5.1.2. Experimento 2

El Experimento 2 consiste en generar una planilla a partir de una solución anteriormente obtenida por el programa organizador de horarios. En esta planilla, la disponibilidad de los profesores es limitada, ya que se encuentran disponibles sólo en el horario en que la solución arroja que están dictando un curso. Para evaluar el comportamiento del algoritmo se crean 6 copias de la planilla donde cada una tiene asignada un 25 % por encima de las horas de disponibilidad que la anterior, obteniendo así disponibilidades del 25 %, 50 %, 75 %, 100 %, 125 % y 150 % más que la planilla original. Para este experimento se utiliza un máximo de 5.000 generaciones por porcentaje evaluado. La información de generaciones evaluadas, total de cromosomas evaluados y la cantidad de conflictos, corresponden a un promedio de 10 pruebas realizadas a cada disponibilidad horaria.

A continuación se muestra el Cuadro 5.3 con los resultados obtenidos.

Porcentaje disponibilidad extra	**Población total**	**Generaciones evaluadas**	**Total cromosomas evaluados**	**Conflictos fuertes**	**Conflictos débiles**
0 %	1.200	5.000	6.000.000	9	6
25 %	1.200	5.000	6.000.000	7	6
50 %	1.200	5.000	6.000.000	5	6
75 %	1.200	5.000	6.000.000	2,7	4
100 %	1.200	4.861	5.833.440	1,5	1
125 %	1.200	2.623	3.147.120	0,4	1
150 %	1.200	1.395	1.674.240	0	0

Cuadro 5.3: Resultados ejecución programa organizador de horarios, utilizando distinta disponibilidad horaria de profesores

El detalle de las pruebas de cada porcentaje de disponibilidad extra se encuentran en los anexos.

Como es posible observar en el Cuadro 5.3, cuando la disponibilidad de los profesores coincide exactamente con la planificación, el programa no logra encontrar una solución dentro de las 5.000 generaciones establecidas para este experimento, evaluando así un total de 6.000.0000 de cromosomas. Sin embargo, alcanza un promedio total de 9 conflictos fuertes y 6 conflictos débiles. Al aumentar en un 25 % la

disponibilidad horaria de los profesores, se puede observar que la cantidad de conflictos disminuye, pero no lo suficiente como para encontrar una solución dentro de las 5.000 generaciones. Lo mismo pasa al aumentar la disponibilidad en un 50 % y en un 75 %. Cuando se realiza un aumento del 100 % de la disponibilidad, se puede observar que ya ha disminuido considerablemente la cantidad de conflictos encontrados y que el promedio de las generaciones evaluadas ya no corresponden a 5.000. Esto es porque al realizar las 10 iteraciones para este porcentaje de disponibilidad, hubo una solución con 0 conflictos débiles y 0 conflictos fuertes, la cual fue encontrada en la generación 3.612. Lo que indica que si bien el aumento del 100 % de disponibilidad podría encontrar una solución factible, no se recomienda ya que la probabilidad de que ocurra es muy baja. Lo mismo ocurre con un aumento de disponibilidad del 125 %.

Por último, se evalúa el aumento del 150 % de disponibilidad horaria extra de profesores, en donde es posible observar que encuentra la solución con 0 conflictos totales en un promedio de 1.395 generaciones. Por lo que evalúa un total de 1.674.240 cromosomas. A raíz de esto es posible concluir que para que el programa pueda encontrar una solución depende principalmente de la disponibilidad horaria de profesores, ya que si la disponibilidad horaria es baja, el programa podría encontrar una solución que se acerque a una factible, pero no que entregue una que los sea.

5.2. Pruebas de usuario

Las pruebas de usuario se realizan para verificar que las necesidades del cliente están siendo consideradas y se encuentra satisfecho con el resultado del programa. Para realizar esta verificación se aplican distintas pruebas al cliente. Estas pruebas consisten en que el cliente debe ejecutar ciertas tareas en el programa. Para evaluar el resultado de la prueba y saber si esta ha sido concretada con éxito o no, se aplica un test por cada prueba, en donde el cliente evalúa si cumple o no con el requisito solicitado. Para ello existe una pauta de evaluación, en donde el cliente debe indicar el resultado obtenido al realizar la prueba que se le indica. Además es posible escribir algunas observaciones con respecto a la prueba realizada. Finalmente el cliente debe marcar una alternativa de evaluación, donde indique si se encuentra satisfecho con el resultado de la prueba, medianamente satisfecho o insatisfecho con el resultado. Se realizan 7 pruebas al usuario, las cuales verifican el funcionamiento de la aplicación.

De estas 7 pruebas, el usuario final de la aplicación indicó estar satisfecho en 6 de ellas y medianamente satisfecho sólo en 1. A continuación, en el Cuadro 5.4 se muestra la prueba en la que el usuario se encuentra medianamente satisfecho con el resultado.

Existen otras pruebas en donde el usuario ha estado satisfecho con el resultado. Sin embargo, ha realizado observaciones las cuales deben ser consideradas e implica realizar una modificación en el programa para garantizar que el usuario esté completamente satisfecho con el resultado.

Por otro lado, existe una prueba realizada en donde el usuario indica que se encuentra satisfecho con el resultado obtenido. Sin embargo, hubo un resultado anormal en una de las iteraciones. Por lo que es necesario realizar un seguimiento a ese comportamiento inusual del programa para encontrar el motivo del error. A continuación, en el Cuadro 5.5 se puede observar en la observación que el programa pudo encontrar una solución al tener libertad al organizar los cursos propios de la carrera de Ingeniería Civil en Computación. Sin embargo, no pudo encontrar una alternativa con cero confictos totales cuando se asignaron los cursos de primer año propios de la carrera de Ingeniería Civil en Computación. También es posible observar que hubo un comportamiento anormal en una de las iteraciones, ya que se menciona que se completaron los cursos de matemáticas en el Excel como horario preestablecido. Pero se genera un cambio al final, en donde el horario del laboratorio de uno de ellos fue modificado.

Cabe mencionar que los errores observados están siendo corregidos en la aplicación y las observaciones mencionadas están siendo consideradas. Lo que implica que se están realizando adecuaciones en el programa.

Recibir mensaje de error al ingresar una planilla que contenga errores.		
Resultado esperado	**Resultado obtenido**	**Observaciones**
El usuario debe recibir un mensaje claro del error presentado. El mensaje debe entregar alternativas de solución y reflejar con claridad el error cometido por el usuario.	Correcto	No estoy segura de los mensajes que retorna el sistema. No me resulta fácil entender el mensaje y corregirlo. Muchas veces (casi todas) no logro corregir sola y debo enviar de vuelta datos al programa para que se identifiquen y resuelvan a aparecer los errores. Como todos los errores se presentan al cargar la planilla, no tengo claro si los errores son los mismos o son distintos. La ventana emergente en la que hay que aceptar cada error es incomoda, pues hay que presionar aceptar en cada uno y no hay forma de cortar la ejecución del programa. Tal vez una opción sería presentar la ventana y luego generar una que muestre un log con todos los errores.
Evaluación		
Satisfecho	**Medianamente satisfecho**	**Insatisfecho**
-	X	-

Cuadro 5.4: Resultado prueba de usuario: Recibir mensaje de error al ingresar una planilla que contenga errores.

Generar horarios académicos.		
Resultado esperado	**Resultado obtenido**	**Observaciones**
El usuario debe poder presionar el botón "Generar" horarios académicos y de esta manera el programa comenzará a iterar hasta encontrar la mejor solución posible, dada las condiciones del algoritmo.	**Resultados corrida 1:** Se dejó fijo los cursos matemáticos y fundamentales. Generaciones: 3.935 y conflictos: 0-0. **Resultados corrida 2:** Se dejó fijo los cursos matemáticos y fundamentales. Pero también los cursos de la carrera de primer año. Generaciones: 10.000 y conflictos: 5-1. **Resultados corrida 2:** Se dejó fijo los cursos matemáticos y fundamentales. Pero también los cursos de la carrera de primer año, Pero esta vez con más generaciones. Generaciones: 15.000 y conflictos: 5-1	Completé para los cursos de matemáticas en el Excel los horarios preestablecidos. Pero me cambió los horarios de laboratorio.
Evaluación		
Satisfecho	**Medianamente satisfecho**	**Insatisfecho**
x	-	-

Cuadro 5.5: Resultado prueba de usuario: Generar horarios académicos.

6. Conclusiones

En esta memoria se ha abordado el problema de la generación de horarios académicos utilizando el método del algoritmo genético. Con respecto al objetivo específico de *Investigar el estado del arte de los métodos de optimización apropiados para el problema de planificación de horarios*, se puede indicar que ha sido logrado ya que se ha realizado una investigación acerca de diferentes métodos que han sido utilizados para darle solución a este problema, que cabe dentro de la categoría de Timetabling, específicamente dentro de class scheduling. En la investigación se tomaron en consideración los métodos: Búsqueda Tabú, Algoritmos Genéticos, Algoritmos de Enfriamiento Lento Simulado, Colonia de Hormigas, Programación Lineal Entera y Redes Neuronales. Una vez que se obtuvo la información con respecto al funcionamiento de los métodos se realizó un análisis acerca de cúal de ellos podría adecuarse de mejor manera al problema de la asignación horaria. Este análisis ha revelado que el algoritmo genético cuenta con una mayor capacidad de revisión del espacio de búsqueda que los otros métodos, lo que aumentaba la probabilidad de alcanzar un óptimo global antes de un óptimo local, lo que permitía también un aumento en la posibilidad de encontrar una solución.

Con lo que respecta al objetivo de *Proponer un modelo de restricciones del problema para la generación de horarios de Ingeniería Civil en Computación, con el fin de orientar la toma de decisiones de la directora de la Escuela de Ingeniería Civil en Computación de acuerdo a las limitaciones impuestas por los recursos a considerar.* También se puede indicar que se ha logrado con éxito. En efecto para considerar las restricciones de planificación se ha creado una estructura en donde es posible incluir las restricciones, como lo es la planilla Excel necesaria para realizar la organización, en donde se puede definir la restricción horaria de los profesores, como también la

función objetivo que utiliza el algoritmo genético para evaluar la solución, buscando tener una menor cantidad de conflictos.

Para los conflictos se realiza una categorización, en donde se encuentran los conflictos fuertes y los conflictos débiles, la diferencia entre ellos recae en la validez de la solución. Una solución que presenta a lo menos un conflicto fuerte invalida la solución, pero si el o los conflictos presentados corresponden a conflictos débiles, la solución podría ser no tan buena como una con 0 conflictos totales pero seguiría siendo válida. Dentro de los conflictos fuertes se han asignado la disponibilidad horaria de los profesores, es decir, que un profesor no puede dictar un curso en un horario que se encuentra declarado como ocupado en la planilla de organización. Por su parte también, el hecho de que un profesor dicte dos cursos dentro de un mismo horario también se encuentra dentro de los conflictos fuertes, al igual que dos cursos pertenecientes a un mismo semestre estén siendo dictados en un mismo horario. Por último, la asignación de cursos de semestres consecutivos dentro de un mismo horario se encuentra dentro de la categoría de conflictos débiles.

Otro objetivo que se ha logrado es la realización de un *Diseño y la construcción de una herramienta que permite llevar a cabo la organización de horarios, minimizando la cantidad de restricciones violadas*. Para ello se ha realizado una estructura donde se definen los elementos y operaciones del algoritmo genético a utilizar, definiendo así la estructura del gen, el cromosoma, la población, tipos de cruza y selección de cromosomas. También se define cómo se evalúan los cromosomas en función a la cantidad de conflictos que estos poseen. Una vez que ha quedado definida la estructura de cada elemento y operación a utilizar dentro del algoritmo genético se continuó con la construcción de un programa de escritorio, en donde lo primero a realizar fue la creación de la lectura de datos de entrada y el almacenamiento de los datos dentro del programa. Cabe destacar que la herramienta no cuenta con una base de datos ya que la información que recibe la obtiene desde una planilla Excel que recibe el programa y la única información que posteriormente se desea almacenar es la solución final encontrada que es posible descargar desde la aplicación, la cual queda almacenada en el ordenador. Una vez que se realiza la lectura de datos se procede a organizarlos de forma tal que cumplan con la estructura para que el algoritmo haga uso de ellos. Es allí donde se construyen los genes, los que almacenan la información de cada bloque horario de la sección de una asignatura determinada. Ya con los genes creados se procede a crear los cromosomas y las operaciones que se

pueden realizar con ellos, como lo son la selección, la cruza y la mutación, además de aplicarle la función objetivo para saber que tan buena es la solución. De esta forma se van creando y evaluando cromosomas generados mediante el algoritmo, logrando entregar una organización horaria minimizando la cantidad de conflictos mediante la búsqueda de la mejor solución encontrada por el algoritmo (cromosoma cuyo cantidad de conflictos sea menor).

Por último se ha realizado una *Validación de la herramienta de organización de horarios, en donde se verifica la calidad de las soluciones arrojadas por el algoritmo.* Para ello se realizaron pruebas al programa donde se evaluaron las funcionalidades y los métodos que utiliza el programa, tanto de forma unitaria como la integración de ellos. Posteriormente se evalúan las soluciones arrojadas por el programa verificando que las restricciones hayan sido consideradas y que la cantidad de conflictos que la solución dice que posee sean realmente ciertos. También se ha aplicado una encuesta de usabilidad de la aplicación, donde la profesora Ruth Garrido, quien es la usuaria principal de este programa ha utilizado la aplicación, realizando las tareas que la encuesta indica. Dicha encuesta ha arrojado un resultado positivo, ya que la usuaria principal se encuentra satisfecha en 6 de 7 pruebas realizadas. Para la prueba en que se encuentra medianamente satisfecha se está corrigiendo y modificando el programa lo necesario para cumplir con las observaciones mencionadas.

Con respecto al objetivo general, que indica que *Se pretende desarrollar una herramienta que permita planificar los horarios de los módulos por semestre de la carrera de Ingeniería Civil en Computación, de forma tal que se minimicen los conflictos de planificación.* Se concluye que se ha logrado con éxito, ya que el programa implementado entrega una solución que considera las restricciones de disponibilidad horaria de profesores, el tope de horarios entre cursos de un mismo semestre, el tope de horario entre cursos de semestres consecutivos y también considera la restricción de que un profesor no dicte dos cursos simultáneamente. Dadas estas restricciones, el programa genera alternativas de solución contabilizando los conflictos que posee cada alternativa, seleccionando la alternativa cuya suma de conflictos fuertes sea menor y considerando en segundo lugar que la suma de conflictos débiles sea lo más bajo posible.

6.1. Trabajos futuros

Este trabajo de memoria se puede expandir mejorando la eficiencia del programa de planificación de horarios, buscando parámetros adecuados para mejorar el tiempo en que el programa tarda en encontrar la solución. Se podría investigar cuál es la cantidad más adecuada de generaciones para que el programa encuentre una solución. También podría investigarse con respecto a la proporción de cromosomas que son escogidos mediante los distintos métodos de cruza y selección. Cuál es la probabilidad más adecuada de que ocurra una mutación. Es posible también investigar acerca del tamaño de la población y qué proporción de ella pertenece a la población prometedora.

Otro alternativa que podría realizarse sería la investigación de cómo se comportaría el programa tomando en consideración la organización horaria de otros cursos, no sólo tomando en cuenta los de la carrera de Ingeniería Civil en Computación, si no que también los otros cursos de la Facultad u otras carreras.

También podría evaluarse la factibilidad de que la aplicación se pueda utilizar para organizar lo cursos de la universidad completa.

Por otro lado, si bien no tiene que ver directamente con esta memoria, un tema a fin es la organización de salas de clases. Ya que lo que se podría hacer es que dada la planificación de horarios, un planificador de salas podría realizar la asignación, y en caso de arrojar error en la asignación de salas de clases, solicitar una nueva asignación horaria de cursos para evaluar nuevamente una asignación de salas dada esa nueva asignación horaria de cursos.

Otra opción a considerar sería la implementación de un sistema de ingreso y modificación de datos. Esto permitiría reportar los errores a medida que se cometen y mejorar las instrucciones para el ingreso de datos dado que de momento es muy rígido.

Por último, al revisar el programa y los resultados con la profesora Ruth Garrido se ha verificado que aún faltan ciertas consideraciones que ella toma en cuenta al realizar la asignación horaria. Estas consideraciones no fueron descritas en el momento en que se diseñó e implementó el programa por lo que se podrían agregar en una nueva versión. Una de estas consideraciones es el tope de horarios de los Electivos, ya que estos se dictan dentro de los Semestres 8, 9, 10 y 11 por lo que la verificación actual solo evalúa que el Electivo del Semestre 8 no choque con el del Semestre 9 ni

el 10, que el Electivo del Semestre 9 no choque con el del Semestre 10 y 11 y así con los demás. Pero no es posible verificar de momento que el electivo del Semestre 8 no choque con el electivo del Semestre 11. Al igual que esta restricción con los cursos electivos podrían existir para otros cursos por lo que sería necesario agrupar los cursos que no deben chocar entre sí ya que pueden ser cursados de forma simultánea. Otro inconveniente que no se ha considerado es que un Electivo se puede tomar en cualquiera de los Semestres 8, 9, 10 y 11, por consiguiente cada Electivo no tendría que chocar con ninguno de los ramos de estos cuatro semestres.

Bibliografía

[1] Juan Andrés Ahumada Ahumada. Generación de horarios académicos en INACAP utilizando algoritmos genéticos. Technical report, Facultad de Ciencias Físicas y Matemáticas, Departamento de Ciencias de la Computación, Universidad de Chile, 2014.

[2] Víctor Yamil Neira González. El problema de timetabling para colegios chilenos. Solución mediante Algoritmos Genéticos. Technical report, Dirección de postgrado, Universidad de Concepción, Noviembre 2014.

[3] Mauricio Andres Guerra Cubillos, Erwin Hamid Pardo Quiroga, and Roberto Emilio Salas Ruiz. Problema del School Timetabling y algoritmos genéticos: una revisión. *Vinculos*, 10(2):262–263, 2013.

[4] Rodrigo Hernandez and Pablo A. Rey. Programación de Horarios de Clases y Asignación de Salas para la Facultad de Ingeniería de la Universidad Diego Portales Mediante un Enfoque de Programacion Entera. *Revista Ingeniería de sistemas*, 22:122–137, 2008.

[5] Samuel Lukas, Arnold Aribowo, and Milyandreana Muchri. Solving Timetable Problem by Genetic Algorithm and Heuristic Search Case Study: Universitas Pelita Harapan Timetable. Technical report, Faculty of Computer Science, Universitas Pelita Harapan, Indonesia, March 2012.

[6] Andres Mauricio Vergel Miranda. Algoritmo para la asignación de horarios académicos en la Universidad Francisco de Paula Santander Ocuña utilizando técnicas de inteligencia artificial. Technical report, Facultad de Ingeniería, Universidad Autónoma de Occidente, Octubre 2018.

[7] Andrea Schaerf. A Survey of Automated Timetabling. *Artificial Intelligence Review*, 1:1–28, 1996.

[8] Bryan Stiven, Triana Gonzales, and Ana María Suarez. Desarrollo de un modelo de asignación de horarios en el entorno educativo mediante la programación lineal. Technical report, Facultad Ingeniería de Sistemas, Universidad Francisco de Paula Santander Ocuña, Mayo 2012.

[9] Needa Jamil Tarun Jain. Genetic Algorithm Approach to Time Tabling Problem. Technical report, European Journal of Business and Management, 2015.

[10] Gary M. Thompson Timothy R. Hinkin. SchedulExpert: Scheduling courses in the Cornell University School of Hotel Administration. Technical report, School of Hotel Administration, Cornell University, Ithaca, New York, 2002.

[11] Carlos Arturo Monje Álvarez. Metodología de la investigación cuantitativa y cualitativa. Technical report, Universidad Sur Colombiana Facultad de Ciencias Sociales y Humanas, 2011.

ANEXOS

A. Resultados Experimento 2

En este anexo se encuentran los cuadros con la información completa de los resultados del Experimento 2.

A.1. Cuadro de resultados de la organización horaria dado un 0 % extra de disponibilidad

En el Cuadro A.1 se muestran los resultados de las 10 iteraciones para la organización horaria dado un 0 % extra de disponibilidad del Experimento 2.

Porcentaje disponibilidad extra	**Población total**	**Generaciones evaluadas**	**Total cromosomas evaluados**	**Conflictos fuertes**	**Conflictos débiles**
0 %	1.200	5.000	6.000.000	6	4
0 %	1.200	5.000	6.000.000	10	4
0 %	1.200	5.000	6.000.000	8	8
0 %	1.200	5.000	6.000.000	7	9
0 %	1.200	5.000	6.000.000	14	4
0 %	1.200	5.000	6.000.000	4	5
0 %	1.200	5.000	6.000.000	12	6
0 %	1.200	5.000	6.000.000	10	10
0 %	1.200	5.000	6.000.000	4	6
0 %	1.200	5.000	6.000.000	15	6

Cuadro A.1: Resultados ejecución programa organizador de horarios, utilizando 0 % extra de disponibilidad horaria de profesores

A.2. Cuadro de resultados de la organización horaria dado un 25 % extra de disponibilidad

En el Cuadro A.2 se muestran los resultados de las 10 iteraciones para la organización horaria dado un 25 % extra de disponibilidad del Experimento 2.

Porcentaje disponibilidad extra	Población total	Generaciones evaluadas	Total cromosomas evaluados	Conflictos fuertes	Conflictos débiles
25 %	1.200	5.000	6.000.000	5	9
25 %	1.200	5.000	6.000.000	7	6
25 %	1.200	5.000	6.000.000	8	2
25 %	1.200	5.000	6.000.000	5	9
25 %	1.200	5.000	6.000.000	10	4
25 %	1.200	5.000	6.000.000	5	7
25 %	1.200	5.000	6.000.000	7	5
25 %	1.200	5.000	6.000.000	5	8
25 %	1.200	5.000	6.000.000	6	7
25 %	1.200	5.000	6.000.000	12	6

Cuadro A.2: Resultados ejecución programa organizador de horarios, utilizando 50 % extra de disponibilidad horaria de profesores

A.3. Cuadro de resultados de la organización horaria dado un 50 % extra de disponibilidad

En el Cuadro A.3 se muestran los resultados de las 10 iteraciones para la organización horaria dado un 50 % extra de disponibilidad del Experimento 2.

A.4. Cuadro de resultados de la organización horaria dado un 75 % extra de disponibilidad

En el Cuadro A.4 se muestran los resultados de las 10 iteraciones para la organización horaria dado un 75 % extra de disponibilidad del Experimento 2.

Porcentaje disponibilidad extra	Población total	Generaciones evaluadas	Total cromosomas evaluados	Conflictos fuertes	Conflictos débiles
50 %	1.200	5.000	6.000.000	3	4
50 %	1.200	5.000	6.000.000	5	9
50 %	1.200	5.000	6.000.000	7	7
50 %	1.200	5.000	6.000.000	4	7
50 %	1.200	5.000	6.000.000	4	6
50 %	1.200	5.000	6.000.000	3	2
50 %	1.200	5.000	6.000.000	7	9
50 %	1.200	5.000	6.000.000	5	5
50 %	1.200	5.000	6.000.000	6	6
50 %	1.200	5.000	6.000.000	6	7

Cuadro A.3: Resultados ejecución programa organizador de horarios, utilizando 50 % extra de disponibilidad horaria de profesores

Porcentaje disponibilidad extra	Población total	Generaciones evaluadas	Total cromosomas evaluados	Conflictos fuertes	Conflictos débiles
75 %	1.200	5.000	6.000.000	3	7
75 %	1.200	5.000	6.000.000	3	6
75 %	1.200	5.000	6.000.000	2	5
75 %	1.200	5.000	6.000.000	4	6
75 %	1.200	5.000	6.000.000	3	5
75 %	1.200	5.000	6.000.000	2	6
75 %	1.200	5.000	6.000.000	2	4
75 %	1.200	5.000	6.000.000	2	3
75 %	1.200	5.000	6.000.000	3	0
75 %	1.200	5.000	6.000.000	3	2

Cuadro A.4: Resultados ejecución programa organizador de horarios, utilizando 75 % extra de disponibilidad horaria de profesores

A.5. Cuadro de resultados de la organización horaria dado un 100 % extra de disponibilidad

En el Cuadro A.5 se muestran los resultados de las 10 iteraciones para la organización horaria dado un 100 % extra de disponibilidad del Experimento 2.

Porcentaje disponibilidad extra	**Población total**	**Generaciones evaluadas**	**Total cromosomas evaluados**	**Conflictos fuertes**	**Conflictos débiles**
100 %	1.200	5.000	6.000.000	0	2
100 %	1.200	5.000	6.000.000	0	4
100 %	1.200	5.000	6.000.000	1	1
100 %	1.200	3.612	4.334.400	0	0
100 %	1.200	5.000	6.000.000	2	0
100 %	1.200	5.000	6.000.000	4	0
100 %	1.200	5.000	6.000.000	2	2
100 %	1.200	5.000	6.000.000	3	1
100 %	1.200	5.000	6.000.000	1	0
100 %	1.200	5.000	6.000.000	2	1

Cuadro A.5: Resultados ejecución programa organizador de horarios, utilizando 100 % extra de disponibilidad horaria de profesores

A.6. Cuadro de resultados de la organización horaria dado un 125 % extra de disponibilidad

En el Cuadro A.6 se muestran los resultados de las 10 iteraciones para la organización horaria dado un 125 % extra de disponibilidad del Experimento 2.

A.7. Cuadro de resultados de la organización horaria dado un 150 % extra de disponibilidad

En el Cuadro A.7 se muestran los resultados de las 10 iteraciones para la organización horaria dado un 150 % extra de disponibilidad del Experimento 2.

Porcentaje disponibilidad extra	**Población total**	**Generaciones evaluadas**	**Total cromosomas evaluados**	**Conflictos fuertes**	**Conflictos débiles**
125 %	1.200	3.246	3.895.200	0	0
125 %	1.200	1.258	1.509.600	0	0
125 %	1.200	5.000	6.000.000	1	3
125 %	1.200	530	636.000	0	0
125 %	1.200	5.000	6.000.000	1	0
125 %	1.200	5.000	6.000.000	0	1
125 %	1.200	322	386.400	2	2
125 %	1.200	804	964.800	0	0
125 %	1.200	2.057	2.458.400	1	2
125 %	1.200	3.009	3.610.800	1	1

Cuadro A.6: Resultados ejecución programa organizador de horarios, utilizando 125 % extra de disponibilidad horaria de profesores

Porcentaje disponibilidad extra	**Población total**	**Generaciones evaluadas**	**Total cromosomas evaluados**	**Conflictos fuertes**	**Conflictos débiles**
150 %	1.200	327	392.400	0	0
150 %	1.200	752	902.400	0	0
150 %	1.200	1.231	1.477.200	0	0
150 %	1.200	115	138.000	0	0
150 %	1.200	267	320.400	0	0
150 %	1.200	708	849.600	0	0
150 %	1.200	350	420.000	0	0
150 %	1.200	2.136	2.563.200	0	0
150 %	1.200	3.720	4.464.000	0	0
150 %	1.200	4.346	5.215.200	0	0

Cuadro A.7: Resultados ejecución programa organizador de horarios, utilizando 150 % extra de disponibilidad horaria de profesores

B. Resultados pruebas de usuario

En este anexo se encuentran los Cuadros con la información completa de los resultados de las Pruebas de Usuario.

B.1. Cuadro de resultados para la prueba de usuario *Configurar parámetros del algoritmo genético.*

En el Cuadro B.1 se muestra el resultado en la prueba de usuario: Configurar parámetros del algoritmo genético.

B.2. Cuadro de resultados para la prueba de usuario *Descargar planilla excel desde la aplicación.*

En el Cuadro B.2 se muestra el resultado en la prueba de usuario: Descargar planilla excel desde la aplicación.

B.3. Cuadro de resultados para la prueba de usuario *Seleccionar planilla a utilizar en la aplicación.*

En el Cuadro B.3 se muestra el resultado en la prueba de usuario: Seleccionar planilla a utilizar en la aplicación.

Configurar parámetros del algoritmo genético.		
Resultado esperado	**Resultado obtenido**	**Observaciones**
El usuario, en caso de ser necesario debe poder configurar los parámetros del algoritmo genético. Debe poder configurar los parámetros: Cantidad Generaciones, Cantidad Población, Cantidad Población Prometedora, Porcentaje Validación Mismo Semestre, Porcentaje Validación Distinto Semestre, Cantidad Cromosomas Selección Aleatoria, Cantidad Cromosomas Selección Mixta, Cantidad Cromosomas Selección Dirigida, Cantidad Cromosomas Cruza de Un Punto, Cantidad Cromosomas Cruza de Dos Puntos, Cantidad Cromosomas Cruza Uniforme.	Correcto	Me parece que no tiene sentido que el usuario tenga que especificar esto, en particular si el usuario no tiene conocimiento sobre el tipo de algoritmo que hay detrás de la aplicación. Pensaría tal vez en dos tipos de usuarios, uno experto que configure este tipo de parámetros y otro no experto que configure los datos sobre los que trabaja la aplicación. Si se deja al usuario final la configuración de estos parámetros, entonces debería tener junto con la ventana de configuración una explicación del efecto que significa modificar dichos parámetros. No es necesario para mí, no me molesta que esté pero no lo estoy usando.
Evaluación		
Satisfecho	**Medianamente satisfecho**	**Insatisfecho**
x	-	-

Cuadro B.1: Resultado prueba de usuario: Configurar parámetros del algoritmo genético

Descargar planilla excel desde la aplicación.		
Resultado esperado	**Resultado obtenido**	**Observaciones**
El usuario debe poder descargar la planilla desde la aplicación y almacenarse automáticamente en el escritorio del ordenador. Se debe entregar un mensaje de éxito al finalizar la acción o un mensaje de error en caso de no poder realizarse la descarga.	Correcto	Super bien
Evaluación		
Satisfecho	**Medianamente satisfecho**	**Insatisfecho**
x	-	-

Cuadro B.2: Ejemplo de prueba de usuario:Descargar planilla excel desde la aplicación.

Seleccionar planilla a utilizar en la aplicación.		
Resultado esperado	**Resultado obtenido**	**Observaciones**
El usuario debe poder buscar en su ordenador la planilla Excel a utilizar, para posteriormente seleccionarla.	Correcto	Resulta un poco incomodo que la ventana para seleccionar la planilla no recuerde por ejemplo donde estaba la planilla la ultima vez que se ejecutó el programa y siempre haya que buscarla desde cero.Pero está bien.
Evaluación		
Satisfecho	**Medianamente satisfecho**	**Insatisfecho**
x	-	-

Cuadro B.3: Ejemplo de prueba de usuario: Seleccionar planilla a utilizar en la aplicación.

B.4. Cuadro de resultados para la prueba de usuario *Seleccionar semestre a organizar.*

En el Cuadro B.4 se muestra el resultado en la prueba de usuario: Seleccionar semestre a organizar.

Seleccionar semestre a organizar.		
Resultado esperado	**Resultado obtenido**	**Observaciones**
El usuario debe poder seleccionar el semestre que desee organizar.	Correcto	Es simple, bien.
Evaluación		
Satisfecho	**Medianamente satisfecho**	**Insatisfecho**
x	-	-

Cuadro B.4: Ejemplo de prueba de usuario: Seleccionar semestre a organizar.

B.5. Cuadro de resultados para la prueba de usuario *Obtener solución.*

En el Cuadro B.5 se muestra el resultado en la prueba de usuario: Obtener solución.

Obtener solución.		
Resultado esperado	**Resultado obtenido**	**Observaciones**
El usuario debe poder descargar un archivo con la asignación de horarios mejor encontrada. Para luego poder visualizarla.	Correcto	El archivo se ve bien.
Evaluación		
Satisfecho	**Medianamente satisfecho**	**Insatisfecho**
x	-	-

Cuadro B.5: Ejemplo de prueba de usuario: Obtener solución.

C. Programa Organizador de horarios

En este anexo se encuentra el screenshot de la interfaz del Programa Organizador de Horarios.

C.1. Vista principal.

La Figura C.1 corresponde a la vista principal del programa Generador de Horarios. Esta vista es la que se muestra al iniciar el programa.

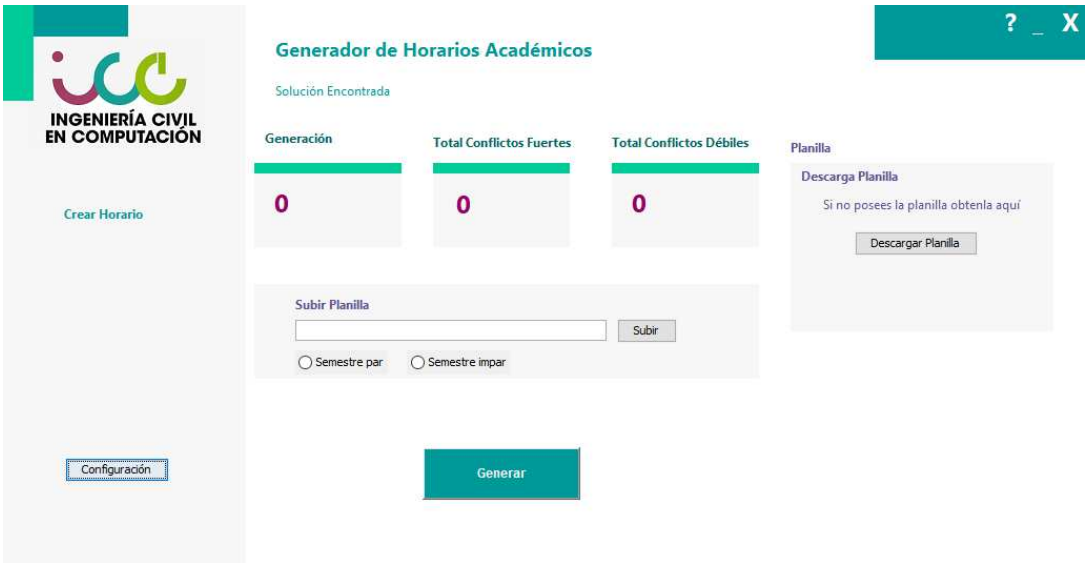


Figura C.1: Vista principal Programa Generador de Horarios