UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

TESIS:

DISEÑO DE UNA METODOLOGÍA E IMPLEMENTACIÓN DE PROTOTIPO BASADO EN INTELIGENCIA ARTIFICIAL PARA INCREMENTAR LA EFICIENCIA DE ASPECTOS ACADÉMICOS UNIVERSITARIOS

PARA OBTENER EL GRADO ACADÉMICO DE MAESTRO EN CIENCIAS EN INGENIERÍA ELECTRÓNICA CON MENCIÓN EN TELEMÁTICA

ELABORADO POR

Ing. MARCIAL ANTONIO LÓPEZ TAFUR

ASESOR:

MSc. Lic. JOSÉ PARIS MIGUEL CAÑAMERO

LIMA – PERÚ

2019

DEDICATORIA

Dedico la presente tesis de maestría a Dios, mi familia y mi Alma Mater

AGRADECIMIENTOS

Un agradecimiento muy especial a mí asesor de tesis el MSc. Lic. José Paris Miguel Cañamero y al profesor MSc. Ing. Arturo Vilca Román, por los consejos y el apoyo brindado, los cuales fueron muy útiles para poder culminar este trabajo de investigación, así como al profesor Mag. Ing. Domingo Pedro Lazo Ochoa, Director de la sección de posgrado de la FIEE; a mi Alma Mater, que mediante los estudios de posgrado me brindó la oportunidad de aprender temas más avanzados y aplicarlos para poder aportar soluciones a los problemas de organización académica de una facultad en una universidad.

TABLA DE CONTENIDOS

ÍNDICE DE ILUSTRACIONES

ÍNDICE DE TABLAS

RESUMEN

Los problemas de elaborar horarios académicos, la asignación de cargas académicas y el uso eficiente de la infraestructura; son problemas difíciles de resolver con métodos tradicionales y todo se complica por las múltiples restricciones.

En este trabajo de investigación presentamos un modelo que va a analizar y/o integrar las metodologías basadas en la inteligencia artificial, tales como los algoritmos genéticos y los heurísticos, el modelado será con UML y trata de asignar horarios considerando restricciones, para ello usaremos métodos de optimización combinatoria multi-restringida, NP-hard con la aplicación en la realidad, así como la programación con restricciones, generando los modelos del problema a resolver usando el UML con sus diferentes diagramas y usando Python como lenguaje de programación. El producto final será una aplicación informática que estará integrada con una base de datos usando MySQL ubicada dentro del programa principal, donde se tenga información de cursos, profesores, aulas, laboratorios, etc., la cual será llamada para entregar los datos requeridos por ésta y en un esquema administrativo organizado que permita solucionar este problema que es el de entregar horarios sin cruces en el mismo ciclo y minimizar los cruces para el ciclo anterior y uno posterior. Los resultados obtenidos y la evaluación, así como las conclusiones, se muestran en los capítulos respectivos.

Palabras claves: inteligencia artificial, algoritmo genético, UML, Python, NP-hard.

ABSTRACT

The problems of developing academic schedules, the allocation of academic charges and the efficient use of infrastructure are difficult problems to solve with traditional methods and everything are complicated by the multiple restrictions.

In this research we present a model that will analyze and / or integrate methodologies based on artificial intelligence, such as genetic algorithms and heuristics, the modeling will be with UML and tries to assign schedules considering restrictions, for this we will use methods of combinatorial optimization multi-restricted, NP-hard with the application in the reality, as well as the programming with restrictions, generating the models of the problem to be solved using the UML with its different diagrams and using Python as a programming language. The final product will be a computer application that will be integrated with an MySQL database located within the main program, where information about courses, teachers, classrooms, laboratories, etc. will be available, which will be called to deliver the data required by this and in an organized administrative scheme that allows to solve this problem that is to deliver schedules without intersections in the same cycle and to minimize the crosses for the previous cycle and a subsequent one. The results obtained and the evaluation, as well as the conclusions, are shown in the respective chapters

Keywords: artificial intelligence, genetic algorithm, UML, Python, NP-Hard

CAPÍTULO I
ANTECEDENTES Y DESCRIPCIÓN DEL PROBLEMA

1.1 Antecedentes bibliográficos

Vamos a citar varios antecedentes de problemas similares en otras universidades nacionales y de otros países parecidos al nuestro.

"Se debe hacer una automatización de los mismos, [...] Existen básicamente dos tipos de restricciones, restricciones blandas y restricciones duras. Las restricciones blandas son aquellas que, si las violamos en la programación, la salida sigue siendo válida, pero las limitaciones duras son aquellas que si las violamos; el calendario ya no es válido". "Orientación dedicada a producir sistemas de alta calidad con rapidez, principalmente mediante el uso de iteración por prototipos (en cualquier etapa de desarrollo), promueve la participación de los usuarios y el uso de herramientas de desarrollo computarizadas. Estas herramientas pueden incluir constructores de Interfaz gráfica de usuario (GUI), *Computer Aided Software Engineering* (CASE), el lenguaje unificado de modelaje (UML), los sistemas de gestión de bases de datos (DBMS), lenguajes de programación de cuarta generación, generadores de código, lógica difusa, inteligencia artificial y técnicas orientadas a objetos. El control de proyecto implica el desarrollo de prioridades y la definición de los plazos de entrega. Si el proyecto empieza a aplazarse, se hace hincapié en la reducción de requisitos para el ajuste, no en el aumento de la fecha límite" [10].

"A medida que aumenta el costo de la educación superior, es cada vez más importante para los colegios y universidades examinar qué tan bien están satisfaciendo las necesidades de los estudiantes. Además de la calidad de la instrucción que se ofrece, esto incluye la disponibilidad de cursos requeridos para cumplir con los objetivos de los títulos de los estudiantes. Uno de los mayores costos imprevistos que enfrentan muchos estudiantes universitarios es tener que pagar la matrícula de un semestre o año extra porque no pudieron tomar todos los cursos requeridos para graduarse dentro del plazo previsto (generalmente de cuatro años). Por lo tanto, un objetivo principal de los procesos de programación académica de la universidad debe ser maximizar la probabilidad de que

todos los estudiantes reciban los cursos que requieren para cumplir con los requisitos de sus títulos de manera oportuna.

Es esencial para los administradores universitarios que son responsables de la creación de los horarios de clases y la asignación de estudiantes a estas clases para comprender los efectos de sus decisiones sobre el funcionamiento de la universidad. Pueden ser de gran alcance: desde las oportunidades educativas creadas o negadas a los estudiantes, hasta el balance presupuestario de hacer un uso eficiente de los recursos. El propósito de este artículo es relacionar algunos de estos efectos para que el lector pueda satisfacer mejor las necesidades de los estudiantes" [4].

Como se menciona en [16]: "Ya en 1986, Suleiman K. Kassicieh, Donald K. Burleson y Rodrigo J. Lievano de la Universidad de Nuevo México, escribían sobre el complejo escenario de la asignación de salas de clases. "Encontraron conflictos de interés:

- Los administradores buscaban una asignación de recursos más efectiva y eficiente.
- Los jefes de departamento tenían distintos objetivos, pues debían equilibrar "puntos de vista encontrados entre administrativos, profesores y estudiantes".

Concluyeron que esta tarea consumía tanto tiempo que era muy difícil de realizar sin la ayuda de un computador. "Incluso con un modesto número de componentes y destinos, el rango de búsqueda de soluciones factibles es demasiado grande.

Los académicos John J. Dinkel, John Mote, y M. A. Venkataramanan "… recuerdan que en 1985 la planificación horaria de la Escuela de Administración de Empresas de Texas A&M University se hacía de tal modo que los profesores debían finalmente "negociar" el uno con el otro las secciones, salas y bloques horarios no planificados.

Los investigadores usaron un sistema computarizado de toma de decisiones en red para hacer calzar los horarios, y permitir que quienes tomaban las decisiones volvieran a tener el control del proceso.

Su experimento mejoró sus horarios, permitió cambiar la asignación de prioridades y preferencias y entregar fácilmente soluciones. Mejoraron la tasa de uso de salas, redujeron los cursos sin asignar y permitieron generar un mejor modelo de gestión para los bloques horarios. Todo en una menor cantidad de tiempo" [16].

"Muchas universidades continúan resolviendo problemas de creación de horarios manualmente, aunque hay una variedad de sistemas de horarios aplicables a estos

problemas. El proceso a menudo tedioso de establecer inicialmente un sistema automatizado y hacer ajustes para las necesidades específicas de una institución es una de las razones críticas para esto. Discutiremos una aplicación del sistema integral de horarios UniTime2 en una gran universidad con 10,800 estudiantes. Aquí los horarios generados se publicaron ocho semanas después de nuestra primera reunión con el administrador de horarios. Nuestro objetivo es describir cómo se puede lograr una implementación tan rápida de un sistema automatizado de programación de cursos universitarios incluso para grandes problemas“ [25].

Según www.unex.es: “… procedimiento es establecer las actividades a realizar por la Facultad de Ciencias de la Universidad de Extremadura (UEx) para elaborar y aprobar los horarios de clase y calendario de exámenes, incluida la asignación de aulas, de los títulos de Grado y Máster que se imparten en la Facultad de Ciencias. Debe realizarse de forma que se garantice la disponibilidad de medios docentes (recursos humanos y materiales) que aseguren la calidad de las enseñanzas impartidas y favorezcan el aprendizaje de los estudiantes. Este procedimiento se articula como una serie de acciones necesarias para el desarrollo de las enseñanzas y, por tanto, viene a ser un procedimiento auxiliar al Proceso de Coordinación de las Enseñanzas. Fuente: manual de calidad procesos y procedimientos” [1].

En el caso de la Universidad Peruana de Ciencias Aplicadas UPC de Lima: “… procesos se realizan cada cierto lapso de tiempo. Uno de ellos es la construcción de horarios, la cual se realiza cada periodo académico por coordinadores o responsables de dicho proceso. Se requiere la formalización del proceso de construcción de horarios, como es su documentación; asimismo, que siga las mejores prácticas de los usuarios tales como orden, facilidad y agilidad de realización y finalmente, reducción de los niveles de error. Por todo este diagnóstico es que resulta útil, desde el punto de vista práctico, contar con la información necesaria de manera rápida, confiable y de fácil administración” [28].

De acuerdo a la referencia [11]: “¿Qué elementos limitan nuestra planificación horaria?

… *Syracuse University*, han comparado distintas técnicas de planificación horaria y han dividido estas limitaciones en términos de prioridad y costo alternativo para la institución:

Consideran conflictos horarios y de espacios que no pueden violarse físicamente, pero que pueden ser previstos con ciertos ajustes de cómo “se explicita el problema”, como

las preferencias de clases de los estudiantes. Las universidades deben equilibrar Conflictos horarios con estudiantes en común. Criterios de admisibilidad a un curso" [11].

1.2 Descripción de la realidad problemática

El tema de administración de la programación de horarios y de recursos es común en muchos países en vías de desarrollo y los problemas de asignación de horarios y aulas llevan desde el cambio de la ley universitaria del año 1984 y la creación de las facultades, haciendo ellas sus propios procesos de matrícula con criterios discrecionales no uniformes.

Requerimientos de diseño:

1. Generar horarios sin cruces de cursos del mismo ciclo, minimizar los cruces de un ciclo anterior y uno posterior, es decir en tres ciclos consecutivos,
2. Los primeros ciclos hasta el sexto con horarios diurnos hasta las 6 pm, con un mínimo de una hora y hasta dos horas máximo para almorzar (entre 12 am a 2 pm, 11 am a 1 pm o 1 pm a 3 pm),
3. Las clases son de lunes a viernes para los profesores a tiempo completo;
4. En la tarde, noche y sábados para los profesores a tiempo parcial,
5. A partir del 8vo ciclo las clases a partir de las 4 pm dentro de lo posible, sábado todo el día.
6. Para todos los docentes, solo se puede dictar hasta un máximo de 5 horas por día sean horas consecutivas o no,
7. Los profesores a tiempo completo deberán distribuir su carga académica durante tres días a la semana de lunes a viernes,
8. Los horarios de los tiempos completos no deberán distribuirse en franjas que sean superiores a las 8 horas diarias consecutivas,
9. El orden de prelación para la asignación de horarios, la tienen primero los profesores Principales, luego los Profesores Asociados y finalmente los Profesores Auxiliares,
10. No deberá excederse la cantidad de salones disponibles por horarios
11. Los horarios de las experiencias de laboratorio deberán ser hechas de lunes a viernes de 8 am a 8 pm

No hay antecedentes escritos en la facultad sobre el tiempo que demora el elaborar horarios, cargas académicas, ya que todo se hace manualmente, usándose programas de Office en su forma básica, y mayormente para la impresión de los documentos, los horarios normalmente se referencian al ciclo anterior, del cual se modifican manualmente en los

casos de cruces de horarios realizados durante el ciclo anterior, nuevos horarios, horarios cambiados por alguna razón de fuerza mayor y por la inclusión de nuevos docentes. Por ello es que el tiempo promedio en los últimos 5 años se ha estimado que es de por lo menos tres semanas para la elaboración de los horarios, (con la premisa de que deberían estar listos al menos 15 días antes de la fecha de la matricula) y se corrigen hasta el mismo día de la matrícula, ya que siempre aparecen alumnos con cruces de horarios en cursos que no son del mismo ciclo ni de ciclo consecutivo inferior o superior, algunos alumnos que se reincorporan a los estudios, y otros alumnos en la condición de "promoción" o que egresarían ese ciclo. La solución propuesta permitirá automatizar los procedimientos manuales y podrá medir el tiempo de reducción en la elaboración de horarios comparados con el tiempo manual. Esto supondrá un ahorro de tiempo y envío oportuno (con al menos quince días de anticipación) a la Oficina de Registro (ORCE) para la respectiva matrícula.

1.3 Formulación del problema

Los problemas de índole administrativos que influyen en la eficiencia académica de una universidad, en particular el problema de horarios, asignación de cargas, salones y laboratorios; son cruciales tanto para la matrícula como para el ciclo académico y por lo tanto es posible afirmar que el diseño de una metodología basada en inteligencia artificial incrementará la eficiencia del funcionamiento de una institución universitaria.

La problemática académica de administración de recursos es común en muchos países del tercer mundo y los problemas de asignación de horarios y de aulas llevan décadas, desde los años 80s hasta la actualidad. Cambios en las leyes y/o en autoridades han implicado muchas veces la pérdida de lo avanzado en el procesamiento de información académica, tal fue el caso en la Facultad de Ingeniería Eléctrica y Electrónica (FIEE) de la Universidad Nacional de Ingeniería (UNI) que experimentó estas pérdidas. Es prioritario y necesario generar conocimiento y plantear una solución a esta problemática, para darle eficiencia al empleo de los recursos tales como docentes e infraestructura disponible. Por ejemplo no hay antecedentes escritos en la facultad sobre el tiempo que demora el elaborar horarios, cargas académicas, ya que todo se hace manualmente, usándose programas de Office en su forma básica. Por ello es que el tiempo promedio en los últimos 5 años se ha estimado que es de por lo menos tres semanas para la elaboración de horarios, (con la premisa de que deberían estar listos al menos 15 días antes de la matricula) y se corrigen prácticamente hasta el mismo día de la matrícula, ya que siempre aparecen alumnos con cruces de horarios en cursos que no son del mismo ciclo ni de ciclo consecutivo, muchos de ellos en la condición de "promoción" o que egresarían ese ciclo. La metodología

propuesta permitirá automatizar los procedimientos manuales y podrá medir el tiempo de reducción en la elaboración de horarios comparados con el tiempo manual. Esto supone un ahorro de tiempo y envío oportuno (con al menos quince días de anticipación) a la Oficina de Registro (ORCE) para la matrícula respectiva.

1.4. Justificación e importancia de la investigación

La importancia de solucionar problemas administrativos que impactan en lo académico en una institución universitaria ha sido abordada por diferentes autores lo cual justifica la importancia del desarrollo del presente trabajo de investigación. En el caso específico de la FIEE-UNI, todo es manual, se hace una plantilla en Excel y se corrige y así se hace ciclos tras ciclo, luego aparecen los cruces de horario en ciclos consecutivos, inclusive durante la matrícula y hasta dos o tres semanas después de ella siendo todo un trámite el cambiar de secciones a los alumnos con cruce en un determinado horario, la falta de salones en determinadas horas la asignación de salones grandes a cursos con pocos alumnos y viceversa así como el cierre de salones en los que ningún alumno se ha matriculado y demás problemas relacionados.

Este también es problema común en países del tercer mundo.

"el tema del horario de clase es un problema típico de programación, que parece ser un trabajo tedioso en cada instituto académico una o dos veces al año". [10]

"actualmente toda empresa, compañía, o entidad de cualquier tipo que maneje cierta cantidad de información o que requiera gestión de determinados procesos para su desenvolvimiento en la respectiva área, debe contar con herramientas tecnológicas para estar a la vanguardia. Es por eso que la mayoría de estas organizaciones han ido incursionando en el mundo de la tecnología; que, si bien puede llegar a ser complejo, el hecho de estar inmersos en él, se convierte casi en una necesidad, si no se quiere llegar a estar obsoletos en el mercado correspondiente, además que facilita muchas de las tareas inherentes a los diferentes procesos que se realizan". [1]

[...] "el proceso de asignación de la carga académica se ha convertido en una tarea complicada debido a los diversos cambios que se ven obligados a realizar para poder llevar a cabo con éxito dicho proceso y teniendo en cuenta que se realiza manualmente. De ahí surge entonces la necesidad de implementar un sistema informático que a través de reportes sirva como herramienta de apoyo a los Directivos del programa para que puedan desempeñar eficientemente esta ardua tarea".

"Diversos métodos se han propuesto para resolver estos problemas basados en métodos de programación adecuados para una institución universitaria. La bibliografía que a continuación describimos realizan algunas propuestas para la elaboración de horarios".

"La programación con lógica difusa. La "característica básica" es el concepto de grado de afiliación en conjuntos difusos. El uso de límites difusos en lugar de límites agudos como en los conjuntos clásicos ha hecho posible el uso de términos lingüísticos cotidianos en el desarrollo de sistemas informáticos". [12]

"Al automatizar este proceso con la ayuda del generador de cronogramas de asistencia de computadora puede ahorrar mucho tiempo valioso para los administradores que están involucrados en la creación y administración de varios horarios de las entidades educativas" [5]

1.5. Objetivos

General: El empleo de una metodología basada en la Inteligencia Artificial permitirá mejorar los procedimientos académicos de una institución universitaria.

Específicos:

1. Reducir el tiempo de la elaboración de los horarios en base a un algoritmo de inteligencia artificial.
2. Evitar tener horarios saturados en base a un algoritmo de inteligencia artificial.
3. Evitar el cruce de horarios en al menos dos ciclos consecutivos aplicando un algoritmo de inteligencia artificial.
4. Mejorar en la asignación de cargas horarias.
5. Optimizar en el uso de la infraestructura.

1.6. Hipótesis

General: El empleo de una metodología basada en IA permitirá mejorar la eficiencia de la asignación de la carga académica de una facultad en una institución universitaria.

Secundarias:

1. El empleo de la metodología permitirá reducir el tiempo de la elaboración de los horarios.

2. La aplicación de la metodología evitará tener horarios con cruces y saturados.

1.7. Variables e Indicadores

1.7.1. Variables

Variable independiente: La metodología basada en inteligencia artificial

Variable dependiente: La eficiencia de la asignación de la carga académica de una facultad en una universidad

1.7.2. Indicadores

1. El tiempo de elaboración de los horarios
2. Porcentaje de ocupación de las aulas
3. La complejidad del algoritmo

1.8. Unidad de análisis

La base de datos es de los docentes, en sus diferentes categorías (Principal, Asociado y Auxiliar) y para los laboratorios y prácticas dirigidas y calificadas a los Jefes de Práctica. Asimismo con el listado de salones y laboratorios disponibles, la población de docentes es de 120 docentes. El listado de cursos que las Direcciones de las Escuelas profesionales indican que cursos se van a dictar en el semestre respectivo y el número de vacantes por curso-sección para una población de cerca de 1,400 alumnos.

1.9. Tipo y nivel de investigación

La investigación es de tipo aplicada experimental para tesis de maestría. Es aplicada porque se pretende resolver el problema de elaboración de horarios que es un dolor de cabeza recurrente a los Directores de Departamento Académicos que tienen que hacerlo antes de que empiece el nuevo ciclo. Es experimental porque se pretende validar los algoritmos genéticos como técnica para elaborar los horarios. El nivel es para una maestría en ciencias pues se utiliza conocimientos específicos de inteligencia artificial, algoritmos genéticos para resolver el problema.

1.10. Periodo de análisis

El periodo es de tipo semestral, los datos se recolectan una vez finalizado el semestre académico, realizándose desde agosto del 2016 a enero del 2017, en la finalización de cada ciclo (2016-1 y 2016-2)

1.11. Fuentes de información e instrumentos utilizados

Las fuentes de información son tomadas de artículos científicos o papers, relacionados a técnicas de solución de problemas para la asignación de horarios en instituciones universitarias; algoritmos genéticos y sus aplicaciones en casos de NP-hard; y libros de algoritmos genéticos. Se utilizó la herramienta Google Schoolar® para la búsqueda de artículos como fuente de información, así como artículos de la IEEE.

1.12. Técnicas de recolección y procesamiento de datos

Para la recolección de datos se usaron

Las tablas de asignación de horarios en Excel suministrados por las Direcciones de los Departamentos Académicos de la FIEE – UNI.

Para el procesamiento de datos se usó:

El lenguaje de programación Python con el cual se desarrolló el algoritmo genético tipo NP-hard, se usó Python versión 3.7 con el IDE PyCharm en su versión 18-2,

El Qt que es un entorno de desarrollo que permite integrar aplicaciones de código abierto ya desarrolladas, las cuales son convocadas por el programa ahorrando tiempo en su desarrollo.

CAPÍTULO II
MARCO TEÓRICO Y MARCO CONCEPTUAL

2.1 Marco teórico

En este capítulo se trata las distintas formas de atacar el problema de la generación de horarios, estos están basados en las diferentes formas de aplicar los conceptos de inteligencia artificial, los cuales son:

- Algoritmos genéticos,
- Heurística; y
- La programación con restricciones.

2.2 Marco conceptual

2.2.1 Algoritmos genéticos

Son llamados así porque se inspiran en la evolución biológica y su base genético-molecular. Estos algoritmos hacen evolucionar una población de individuos sometiéndola a acciones aleatorias semejantes a las que actúan en la evolución biológica (mutaciones y recombinaciones genéticas), así como también a una selección de acuerdo con algún criterio, en función del cual se decide cuáles son los individuos más adaptados, que sobreviven, y cuáles los menos aptos, que son descartados.

"Una implementación de algoritmos genéticos en el problema de programación de horarios. El problema de programación de horarios es común a todas las instituciones educativas. El objetivo principal del algoritmo es minimizar el número de conflictos en el cronograma. También la reducción a la codificación del espacio de búsqueda. El algoritmo se probó en problemas horarios pequeños y grandes en la Facultad de Ingeniería Eléctrica y Computación (FER) en Zagreb. La interfaz del programa fue desarrollada en C #. El núcleo del algoritmo genético se desarrolló en C ++ con soporte STL (*Standard Template Library*). Los problemas de programación son esencialmente los problemas que tienen que ver con la distribución efectiva de los recursos. Durante el proceso de programación, se deben considerar muchas restricciones. Los recursos generalmente son limitados y no hay dos tareas que ocupen un recurso en particular al mismo tiempo. Para la mayoría de los

problemas de programación se ha demostrado que son NP-hard, y que no se pueden resolver en tiempo polinomial utilizando un algoritmo determinista. El problema de programación de horarios presenta un conjunto de tareas (clases) y un conjunto de recursos (salas, grupos, instructores). Cada tarea requiere algunos recursos para su realización y tiene la longitud exacta. También se determina el conjunto de rangos de tiempo cuando se puede programar una clase. El objetivo es asignar esas tareas a sus recursos a la vez que se satisfacen todas las restricciones difíciles; no se deben asignar recursos por varias tareas al mismo tiempo. El problema inicial de programación con un gran número de variables binarias se ha reducido al tamaño aceptable eliminando ciertas dimensiones del problema, incorporando esas dimensiones en las restricciones. La agrupación de varias variables binarias en un valor de gen redujo significativamente el tamaño individual. Ahora es posible tratar de resolver el problema de tamaño completo (problema de todo el cronograma de la facultad) con el enfoque del algoritmo genético" [6].

"En este estudio, hemos abordado el problema NP-hard de la programación de clases académicas (o *timetabling*) a nivel universitario. Hemos investigado una variedad de enfoques basados en el recocido simulado (*Simulated Annealing*, es un método de búsqueda local que explota la analogía entre la optimización combinatoria y los problemas de mecánica estadística) incluido el recocido de campo medio, el recocido simulado con tres programas de enfriamiento diferentes y el uso de un preprocesador basado en reglas para proporcionar una buena solución inicial para el recocido. Los mejores resultados se obtuvieron utilizando el recocido simulado con refrigeración adaptativa y recalentamiento como una función del costo, y un preprocesador basado en reglas. Este enfoque nos permitió obtener calendarios válidos para el problema del calendario de una universidad grande, utilizando una función de costos complejos que incluye las preferencias de los estudiantes. Ninguno de los otros métodos pudo proporcionar un horario válido completo" [11].

"Se ha elegido este problema, ya que es representativo de la clase de problemas de optimización combinatoria multi-restringida, NP-hard con la aplicación en el mundo real. Primero presentamos nuestra modelo, incluida la definición de una estructura jerárquica para la función objetivo y los operadores genéticos generalizados que se pueden aplicar a las matrices que representan los horarios. Luego informamos sobre los resultados de la utilización del sistema implementado para el caso específico de la generación de un horario escolar. Comparamos dos versiones del algoritmo genético (AG), con y sin búsqueda local, tanto en un horario hecho a mano como en otros dos enfoques basados en el recocido simulado y la búsqueda tabú. Nuestros resultados muestran que AG con la búsqueda local

y la búsqueda tabú con relajación superan el recocido simulado y los horarios hechos a mano. Utilizaremos la construcción de un horario o un horario de clases para una escuela secundaria italiana como medio para nuestra investigación. Las ideas presentadas en este documento pueden aplicarse, por supuesto, a la solución de otras instancias del problema del calendario, y posiblemente muy diferentes. La posibilidad de realizar pruebas "sobre el terreno" ha sido la razón principal para la elección de este ejemplo de problema particular. (En una escuela secundaria típica de Italia, una clase recibe cinco horas de clases, seis días a la semana. Los profesores pueden enseñar una o más materias, por lo general en dos o más clases. Además de su demanda de enseñanza de dieciocho horas, tienen otras clases actividades, como se describe en el documento. Además, cada maestro tiene el derecho de tomarse un día de descanso por semana, además de los domingos.) La construcción del horario de clases para una escuela secundaria italiana puede descomponerse en la formulación de varias relaciones interrelacionadas de horarios De hecho, las secciones siempre se acoplan de a pares, con un par de secciones que comparten muchos profesores y recursos (por ejemplo, laboratorios). Por lo tanto, dos secciones acopladas pueden procesarse como una "unidad atómica", no descomponerse más debido a sus altas dependencias internas, pero relativamente aisladas de otras secciones. Comparamos nuestro enfoque basado en AG con varias versiones de recocido simulado y búsqueda tabú. En nuestros experimentos, las AG produjeron mejores horarios que el recocido simulado, pero tiempos un poco peores que la búsqueda tabú. Una ventaja de las AG sobre SA (*Simulated Annealing)* y TS (*Tabu Search*) es que los AG otorgan al usuario la flexibilidad de elegir dentro de un conjunto de diferentes horarios. Esta es una característica importante ya que la evaluación de un horario se realiza mediante una función objetiva que puede fallar caracterizando aspectos. Esto, a su vez, puede hacer que un cronograma con un costo ligeramente más alto sea más deseable que uno con un costo menor. Creemos que nuestro enfoque es una generalización útil de la AG y se puede aplicar a otros problemas de optimización combinatoria altamente restringidos. Comparamos nuestro enfoque basado en GA con varias versiones de recocido simulado y búsqueda tabú. En nuestros experimentos, las AG produjeron mejores horarios que el recocido simulado, pero tiempos un poco peores que la búsqueda tabú. Una ventaja de las AG sobre SA y TS es que los AG otorgan al usuario la flexibilidad de elegir dentro de un conjunto de diferentes horarios" [9].

"La creación de horarios es una tarea ardua y lenta. Para crear un horario, se necesita mucha paciencia y horas de trabajo. El horario se crea con diversos fines, como organizar conferencias en la escuela y universidades, crear diagramas de tiempo para el horario de

trenes y autobuses y muchos más. Para crear un cronograma se requiere mucho tiempo y poder humano. En nuestro documento, hemos tratado de reducir estas dificultades de generación de cronograma mediante el uso de algoritmo genético, podemos reducir el tiempo requerido para generar tabla de tiempos y generar un horario que sea más preciso, preciso y libre de errores humanos. La primera fase contiene todas las clases obligatorias comunes del instituto, que están programadas por un equipo central. La segunda fase contiene las clases departamentales individuales. Actualmente, este calendario se prepara manualmente, manipulando los de años anteriores, con el único objetivo de producir un calendario factible. Con el fin de lidiar con problemas de calendario, estamos proponiendo un sistema que generaría mecánicamente un calendario para el instituto. El curso y las conferencias se programarán de acuerdo con todas las restricciones posibles y las entradas dadas, y así se generará un cronograma. Además, los horarios generados son mucho más precisos que los creados manualmente. Hemos utilizado C # junto con .NET *framework* para desarrollar nuestra aplicación. Hemos utilizado datos reales de varios departamentos de nuestro instituto para probar el método y la eficacia con que está funcionando. El proyecto reduce el consumo de tiempo y el dolor al enmarcar el horario manualmente. Los beneficios de este enfoque son un diseño simplificado y un tiempo de desarrollo reducido" [19].

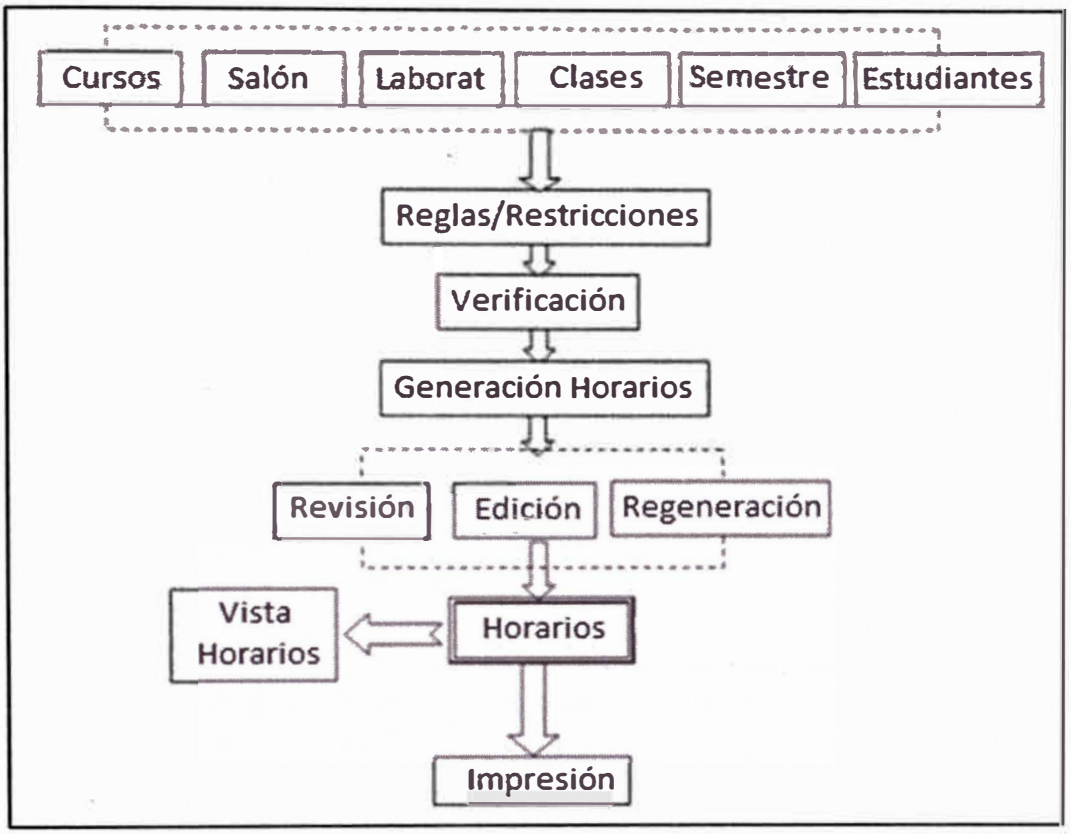


Figura 1 Vista general de un generador de horarios [19]

"...proponen una técnica optimizada para automatizar el sistema de generación de tablas de tiempo. El sistema de generación de tablas de tiempo implica varias restricciones desafiantes de recursos, incluyendo facultades, salas, intervalos de tiempo, etc. La técnica propuesta filtra las mejores reglas activas y algoritmos genéticos para generar la solución optimizada. El Algoritmo Genético y las Reglas Activas juntas forman una esfera completa para desarrollar un sistema que necesita satisfacer varias restricciones. Las Reglas activas proporcionan un modelo de "acción-condición-evento" para la implementación de cualquier sistema basado en reglas. En el algoritmo genético, cada individuo se caracteriza por una función de aptitud. Después del análisis, si hay una mayor aptitud, significa una mejor solución y luego, en función de su estado físico, los padres son seleccionados para reproducir descendencia para una nueva generación en la que los individuos más en forma tienen más posibilidades de reproducirse. El objetivo del trabajo es crear un modelo utilizado para generar el horario aceptable utilizando operadores probabilísticos" [21].

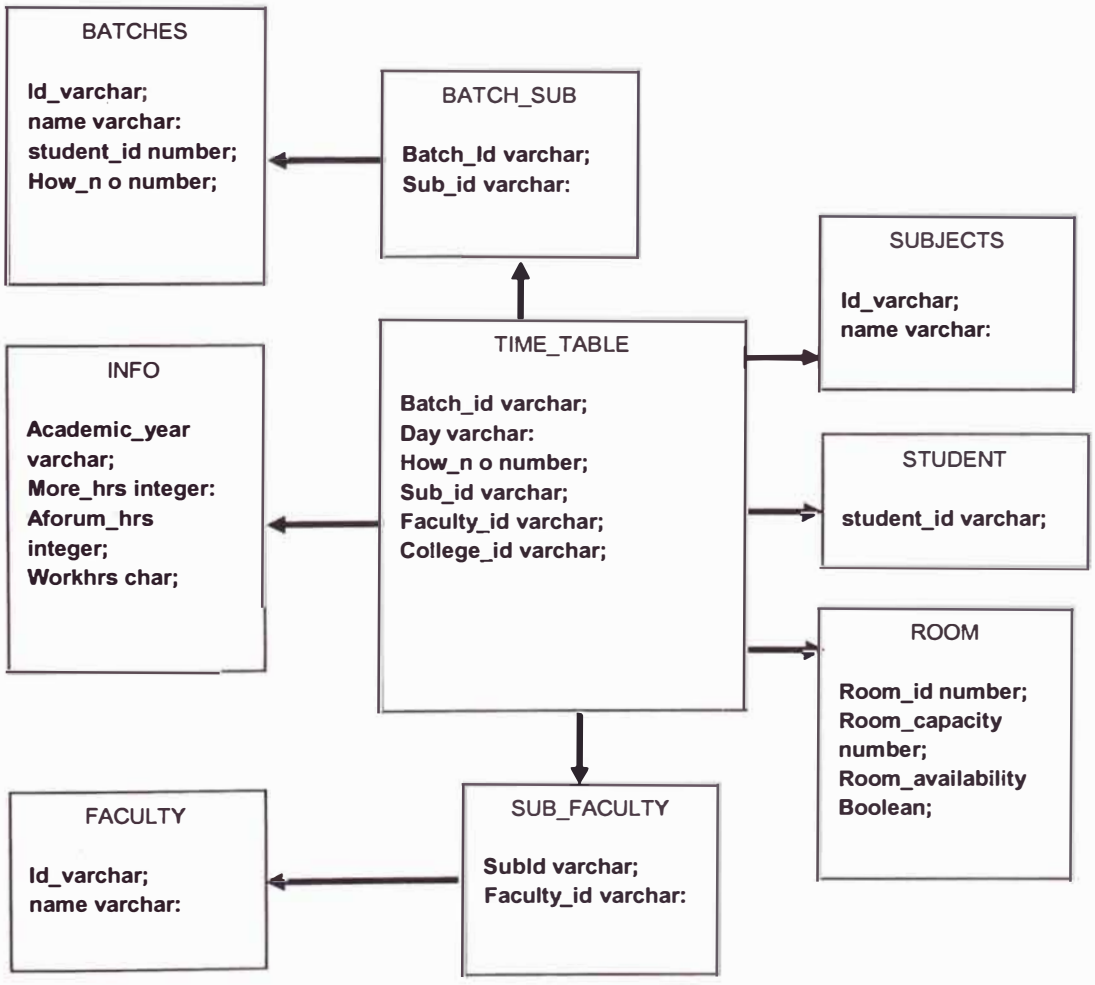


Figura 2 Modelo E-R de un sistema de horarios [21]

“Los algoritmos genéticos son métodos para resolver problemas basados en una abstracción del proceso de Selección Natural. Intentan imitar la naturaleza desarrollando soluciones a los problemas en lugar de diseñarlos. Los algoritmos genéticos funcionan por analogía con la selección natural de la siguiente manera. Primero, se mantiene un grupo poblacional de cromosomas. Los cromosomas son cadenas de símbolos o números. Hay una buena precedencia para esto, ya que los humanos se definen en el ADN usando un alfabeto de cuatro símbolos. Los cromosomas también se llaman el genotipo (la codificación de la solución), a diferencia del fenotipo (la solución misma). En el algoritmo genético, se mantiene un grupo de cromosomas, que son cadenas. Estos cromosomas deben evaluarse para determinar la aptitud. Las soluciones deficientes se purgan y se realizan pequeños cambios en las soluciones existentes y luego permiten que la "selección natural" siga su curso, evolucionando el conjunto de genes para que se descubran soluciones cada vez mejores” [21].

“… presentan un potente algoritmo genético híbrido basado en un marco de calendario heurístico. Esto combina una representación directa del horario con los operadores de cruce heurísticos para garantizar que nunca se violen las restricciones más fundamentales. Explicamos cómo se siembra la población para producir una solución que no puede ser superada solo por el método heurístico. También vemos cómo se garantiza que el algoritmo siempre produce una solución factible mediante restricciones de codificación rígida que no deben romperse. Se utilizan operadores híbridos de cruce, que propagan las características más deseables del horario para producir soluciones de buena calidad, incluso para problemas grandes y muy restringidos. Finalmente, presentamos los resultados de aplicar el algoritmo a un problema particularmente difícil y demostramos la variedad de horarios posibles que se pueden generar, dependiendo de los requisitos del usuario.

Los algoritmos evolutivos son muy prometedores en el área del horario educativo, particularmente en su capacidad para considerar y optimizar la gran variedad de restricciones que se pueden encontrar en las universidades. Hemos demostrado aquí que al considerar los operadores híbridos de cruce, incorporando las técnicas de coloración de gráficos ya conocidas, podemos producir horarios de buena calidad, incluso a partir de problemas extremadamente limitados” [6].

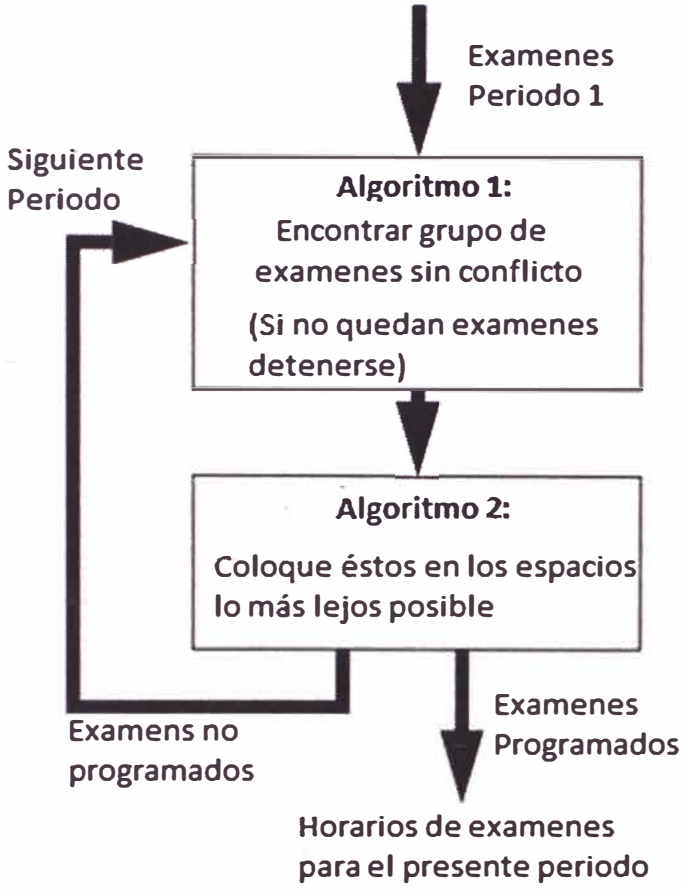


Figura 3. Estructura de un horario heurístico [6]

"... es el trabajo describe un enfoque de algoritmo genético para la programación de recursos limitados utilizando una representación directa basada en el tiempo. Mientras que los métodos de solución tradicionales suelen basarse en secuencias, esta representación codifica la información de programación como una matriz dual de tiempos de demora relativos y modos de ejecución enteros. La representación admite múltiples modos de ejecución, apropiación, disponibilidad / uso de recursos no uniformes, una variedad de tipos de recursos, modelos de rendimiento de recursos probabilísticos, relaciones de precedencia superpuestas y restricciones temporales en tareas y recursos. Además, la representación incluye disponibilidad y requisitos de recursos variables en el tiempo. Se pueden definir muchas medidas objetivas, como la minimización de *makespan* (que es la distancia en tiempo que transcurre desde el inicio del trabajo hasta el final), la maximización del valor presente neto o la minimización de la tardanza promedio. Múltiples objetivos, posiblemente conflictivos, son compatibles. El algoritmo genético se adapta a factores dinámicos como cambios en el plan del proyecto o alteraciones en la ejecución del cronograma" [29].

"Se sabe que la identificación temporal es un problema completo no polinómico, es decir, que no existe una forma eficiente conocida de localizar una solución. Además, la característica más llamativa de los problemas NP-completos es que no se conoce la mejor solución para ellos. Por lo tanto, para encontrar una solución a un problema de calendario, se elige un enfoque heurístico. Este enfoque heurístico, en el mismo, conduce a un conjunto de buenas soluciones (pero no necesariamente la mejor solución). En un problema de planificación general de horarios educativos, un conjunto de eventos (por ejemplo, cursos y exámenes, etc.) se asignan en un cierto número de intervalos de tiempo (períodos de tiempo) sujetos a un conjunto de restricciones, que a menudo hace que el problema sea muy difícil de resolver en la realidad circunstancias mundiales. De hecho, los horarios a gran escala, como los horarios universitarios, pueden requerir muchas horas de trabajo por parte de personas o equipos calificados para producir horarios de alta calidad con una satisfacción de restricciones óptima y la optimización de los objetivos del cronograma al mismo tiempo. El algoritmo incorpora una serie de técnicas, destinadas a mejorar la eficiencia de la operación de búsqueda. También aborda la importante limitación de los enfrentamientos entre la disponibilidad de docentes. Las restricciones blandas no rígidas, es decir, los objetivos de optimización para la operación de búsqueda, también se manejan de manera efectiva. Dada la generalidad de la operación del algoritmo, puede adaptarse aún más a escenarios más específicos, p. Universidad, programación de exámenes y mejorar aún más para crear horarios de trenes. Por lo tanto, a través del proceso de automatización del problema de la tabla de tiempos, muchas horas de creación de un calendario efectivo se han reducido con el tiempo" [20].

"La instancia particular con la que nos enfrentamos aquí es el problema del calendario de exámenes universitarios. Las universidades difieren de las escuelas en que, en general, en las escuelas, las clases son todas de un tamaño similar y las salas son lo suficientemente grandes como para albergar cualquier clase. En las universidades, las clases pueden variar de cuatro o cinco estudiantes hasta unos doscientos. La programación de los cursos difiere de la del examen en que es deseable que los cursos no entren en conflicto con los estudiantes, pero en algunos casos, esa solución no existe. Esto significa que los estudiantes tendrían que elegir entre cursos conflictivos. En la programación de exámenes, ningún estudiante puede ser asignado a dos exámenes a la vez. El sistema que describimos aquí puede ampliarse para incluir otros problemas relacionados con el calendario. Es bien sabido que el problema de coloración de gráficos es NP-completo. *Timetabling* y el problema de coloración de gráficos fueron vinculados por Welsh y Powell en 1967. Sigue de inmediato que los problemas de tiempo son en general

NP-completos. Los vértices de un gráfico se consideran equivalentes a los exámenes, mientras que los bordes entre los vértices representan conflictos. Colorear los vértices es lo mismo que dividir los exámenes en períodos, cada color representa un período diferente" [20].

Figura 4. Método para hacer horarios [20]

"El calendario de planificación es una de las aplicaciones más complejas y propensas a errores. Todavía hay problemas serios, como la generación de tablas de tiempo de alto costo que se producen durante la programación y estos problemas se repiten con frecuencia [28].

"Por lo tanto, existe un gran requisito para una aplicación que distribuya el curso de manera uniforme y sin colisiones. El objetivo es desarrollar una aplicación simple, fácilmente comprensible, eficiente y portátil que pueda generar automáticamente una tabla de tiempos de buena calidad en un segundo. El esquema de este documento es el siguiente: se describen reglas activas para el conocimiento de agentes inteligentes (es decir, Restricciones), se describen GA y se propone su uso en la optimización del agente basado en reglas, los métodos se aplican al problema de optimizar algunos se presentan los resultados de esta aplicación y, finalmente, se presentan algunas conclusiones y posibles orientaciones para futuras investigaciones.

Un problema de horario de conferencia se refiere a encontrar la asignación de tiempo exacta dentro del período de tiempo limitado del número de eventos (cursos-conferencias) y asignarles el número de recursos (profesores, estudiantes y salas de conferencias) al tiempo que se satisfacen algunas limitaciones. Las restricciones se clasifican en Restricciones duras y Restricciones blandas. Las limitaciones difíciles son aquellas que deben cumplirse, mientras que las Restricciones suaves pueden ser violadas si es necesario. La ventaja de GA es que pueden explorar el espacio de solución en múltiples direcciones a la vez. En casos extremos donde solo hay una buena solución, el algoritmo genético puede fallar, pero nuevamente puede reiniciarse con las Reglas Activas con muchas posibilidades de encontrar una mejor solución" [2].

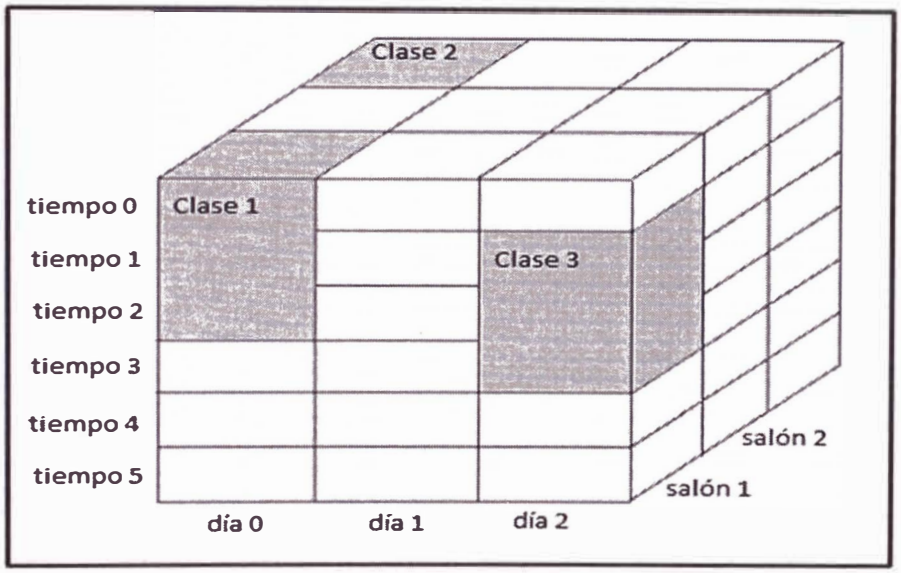


Figura 5 El horario presentado como una estructura en 3D [2]

"El problema de la programación del examen de hoja perenne se ha estudiado ampliamente en la literatura con una amplia gama de soluciones. Sin embargo, parece haber un retraso en la adopción e implementación de estas soluciones en la práctica. Este problema se vuelve de suma importancia ya que los institutos educativos se están moviendo hacia la automatización para hacer frente a la gran cantidad de estudiantes, clases e instructores.

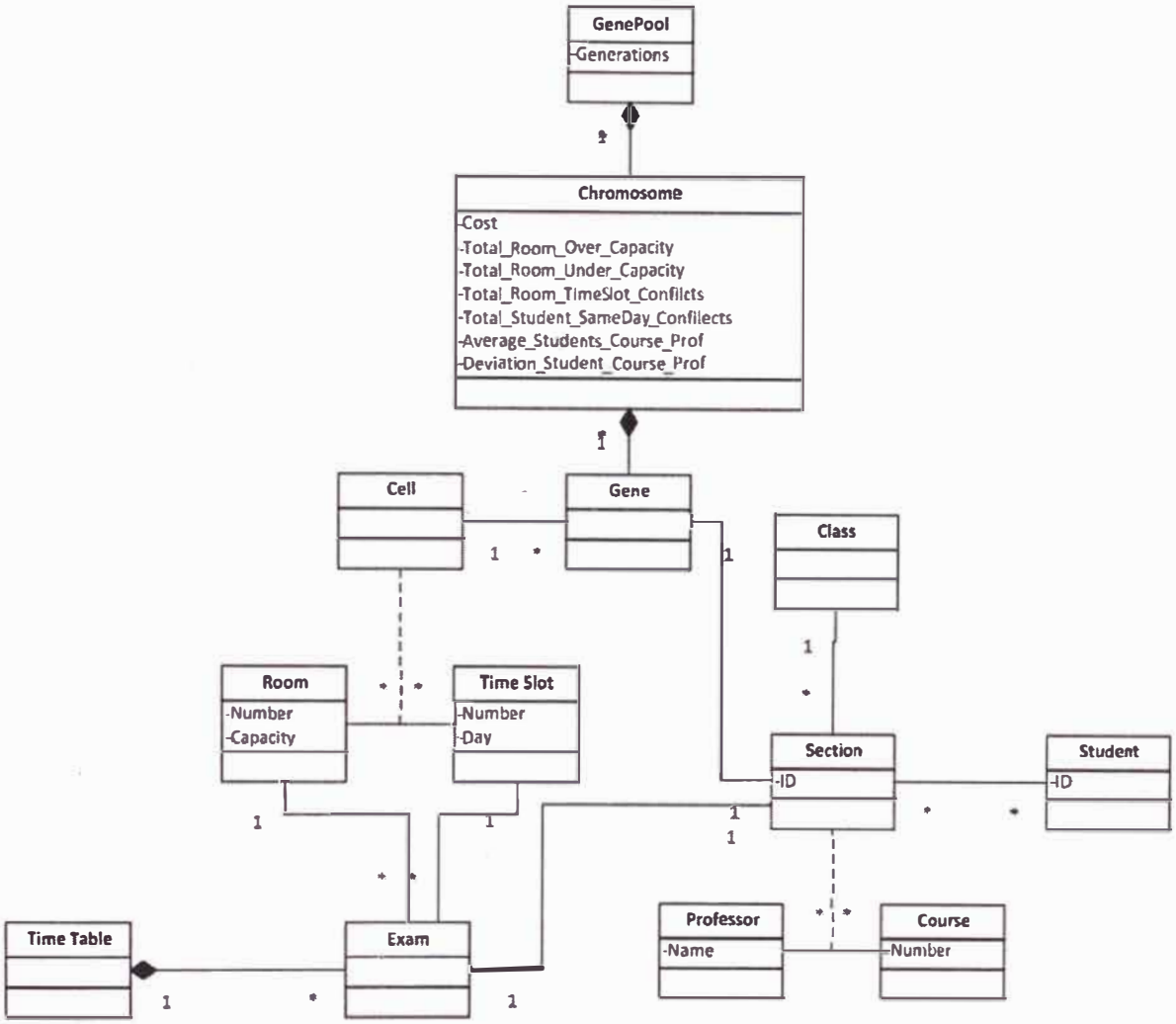


Figura 6 El diagrama UML que muestra el diseño del software de la solución [27]

El problema de programación del examen generalmente consta de las tres partes principales; 1. Los espacios de tiempo, durante los cuales se pueden programar los diversos exámenes. Típicamente, estos espacios son uniformes en duración y limitados en número. 2. Recursos: estos son los elementos humanos y logísticos necesarios para completar el examen de manera correcta (por ejemplo, salas, supervisores, etc.). 3. Restricciones de programación: son condiciones que impiden la programación concurrente de exámenes. A veces no es posible programar un examen en un intervalo de tiempo dado debido a que los estudiantes tienen exámenes múltiples al mismo tiempo, un cierto número de exámenes en el mismo día o cualquier otra restricción ordenada por la institución académica" [27].

"Crear horarios escolares es problemático en muchas escuelas debido a la complejidad de los modelos que necesitan ser resueltos. Si las escuelas no tienen suficientes recursos (habitaciones, maestros), se ven obligados a trabajar en turnos dobles o varios. Este

documento trata sobre los tipos de cambio y las formas de escribirlos en los formatos utilizados por las computadoras para generar horarios de forma automática. A pesar de que muchas escuelas trabajan en turnos, la restricción para incluir cambios en la creación de horarios rara vez se discute en la literatura. Este documento discute una forma de formular cambios escolares usando el formato XHSST, una forma de modificar un lenguaje específico de dominio para soportar cambios, y un ejemplo de una manera de modificar el modelo matemático de la programación lineal entera para apoyar el trabajo en varios turnos" [24].

Tabla 1 - Ejemplo de un horario sin espacios en blanco [24]

	Lunes	Martes	Miércoles	Jueves	Viernes
1	Química	Música	Artes	Química	Lengua nativa
2	Leng. nativa	Matemática	Geografía	Leng. Nativa	Historia
3	Historia	Matemática	Biología	Religión	Matemática
4	Matemática	Física	Inglés	Matemática	Comunidad
5	Religión	Leng. nativa	Tecnología	Alemán	Física
6	Informática	Deportes		Alemán	Deportes

"La mayoría de los métodos heurísticos utilizados para resolver problemas de establecimiento de horarios (alpinismo, búsqueda de vecinos grandes, colonia de hormigas, ...) se basan en permutaciones en lecciones de horario y en la medición del grado de satisfacción de restricciones. Los coeficientes de ponderación significativamente mayores se toman generalmente para restricciones duras que para restricciones blandas. Dependiendo del algoritmo y su implementación, los cambios se pueden integrar en el conjunto existente de restricciones, como se describe en la sección III-A para todos los algoritmos heurísticos que se implementan utilizando el motor KHE. En caso de que el conjunto existente de restricciones para el algoritmo heurístico no sea directamente compatible con los cambios, es necesario definir y programar restricciones adicionales" [25].

"El problema del calendario universitario del curso (UCTP) es un tema importante desde siempre. Los diferentes investigadores consideran y prueban NP-Complete los problemas de programación. Se debe a un cambio continuo de restricciones para cumplir con los datos que cambian rápidamente. En este estudio se investigan diferentes técnicas híbridas del estado del arte y su uso para el problema del horario del curso universitario. Este documento también analiza la ocurrencia de restricciones y la relación de similitud en la tendencia de investigación reciente sobre el problema de la programación de cursos universitarios. Las restricciones varían mucho de un departamento a otro. Este nivel de varianza de restricciones hace que el problema de la asignación de horarios sea difícil de resolver y NP-complete. En los últimos años, el concepto de hibridación de diferentes métodos aumentó. Este estudio analizó el uso de métodos híbridos heurísticos y metaheurísticos para el problema del calendario de cursos universitarios. En este estudio, categorizamos esta hibridación en dos categorías principales: hibridación de búsqueda local con enfoques basados en búsquedas locales e hibridación basada en la población con enfoques basados en búsquedas locales. Se observa que los métodos basados en la población (algoritmo genético (GA), optimización de enjambre de partículas (PSO: *Particle Swarm Optimization*) y colonia de abejas artificiales (ABC: *Artificial Bee Colony*) etc.) se prefieren en combinación con métodos basados en búsqueda local (LS: *Local Search*) para problemas de horarios universitarios. El concepto de hibridación de métodos basados en la población con métodos locales y otros basados en la población se adopta para eliminar los inconvenientes de ambos métodos. Aunque, los métodos híbridos son difíciles y también requieren un mayor costo computacional. Aún la hibridación es útil para encontrar mejores soluciones" [15].

2.2.2 Heurística

La heurística es un método basado en la experiencia que puede utilizarse como ayuda para resolver problemas de diseño, desde calcular los recursos necesarios hasta en planear las condiciones de operación de los sistemas. Mediante el uso de heurísticas, es posible resolver más rápidamente problemas conocidos o similares a otros conocidos.

"La programación con métodos heurísticos y evolutivos permite resolver problemas de gran tamaño en tiempos computacionales razonables y satisfaciendo niveles de calidad deseados" [16].

"La definición de una estructura jerárquica para la función objetiva y los operadores genéticos generalizados que se pueden aplicar a las matrices que representan los horarios" [5].

"El problema de horarios (*High School Timetabling*, HSTT) tiene diferentes estructuras en diferentes escuelas secundarias, incluso dentro del mismo país o sistema educativo. El problema HSTT en varios países se ha estudiado para encontrar un conjunto común de restricciones y objetivos. El problema HSTT representado en formato XML (XHSTT) se diseñó con el fin de modelar mejor el problema completo y facilitar el intercambio de datos entre investigadores de horarios de la escuela secundaria. En este proyecto maestro, se extiende un programa de programación de la escuela secundaria, para facilitar la construcción y aplicación de algoritmos a varias instancias de la escuela secundaria. Se implementan restricciones, una evaluación de costos de soluciones y varios algoritmos. Hacemos uso de combinaciones de algoritmos, que pueden mejorar la calidad de los horarios dentro de un cierto límite de tiempo de cálculo. Se aplica una hiperheurística que encuentra la mejor combinación de algoritmos en diferentes etapas del proceso de solución. La hiperheurística se prueba en varias instancias reales de la escuela secundaria de diferentes partes del mundo" [5].

"En este trabajo describimos una heurística que llena el intervalo de tiempo (TFH) para crear los horarios de la escuela secundaria. Esta heurística se basa en rellenar iterativamente ranuras de tiempo seleccionadas con conjuntos de eventos. El enfoque más común, con el que comparamos nuestra heurística, es asignar iterativamente eventos únicos. Estos enfoques se evaluaron utilizando instancias artificiales, así como las instancias del mundo real del Proyecto de Benchmarking para Horario escolar (alto). Aunque nuestro algoritmo en ocasiones es superado por algoritmos hechos a medida para casos particulares, demostramos la aptitud general de la heurística que llena el intervalo de tiempo. La idoneidad de un enfoque depende en gran medida de las características de la instancia a la que se aplica, lo que impide declarar un ganador claro. Como el enfoque de llenado del intervalo de tiempo es mucho menos explorado, será necesario realizar más investigaciones para evaluarlo con más detalle. En particular, se requiere una comparación con otros algoritmos líderes de horarios de escuela que sean lo suficientemente flexibles como para manejar las limitaciones de las instancias. El principal desafío definitivamente radica en el desarrollo de una función de clasificación adecuada. Tiene que mantener el equilibrio entre las diversas restricciones suaves y duras, entre eventos de diferente tamaño y la necesidad de crear un calendario completamente lleno. El trabajo futuro puede enfocarse en conceptos más avanzados para la función de calificación. Los parámetros autoadaptables o la conmutación entre varias funciones de clasificación para una restricción son posibles mejoras. Esto podría hacerse aplicando algoritmos de búsqueda locales a los cronogramas creados por la heurística que llena el intervalo de tiempo.

Entonces uno podría concentrarse en adaptar los parámetros o las funciones de clasificación de la parte (restricción-violación) que la búsqueda local pudo mejorar. Esto haría que la búsqueda de conjuntos de parámetros adecuados y funciones de graduación inapropiadas sea más eficiente" [22].

"El problema del horario del curso universitario (UCTP) representa una clase importante de problema de optimización en la investigación operativa. Es considerado como uno de los problemas más difíciles que enfrentan las universidades y colegios hoy en día. El problema puede definirse como la asignación de recursos dados (maestros, estudiantes y aulas) a objetos (cursos) que se colocan en el espacio tiempo satisfaciendo todas las restricciones universitarias y optimizando la utilización de las instalaciones existentes de modo que se satisfagan un conjunto de objetivos deseables. Básicamente, el problema del horario universitario existe en dos formas: formatos de calendario de cursos y exámenes. Aquí nuestro enfoque se centra únicamente en el problema del calendario del curso. El horario del curso universitario requiere varios espacios y con diferentes categorías como conferencias, tutoriales y sesiones prácticas, que se ajusta dentro de una semana y se repite durante todo el semestre. Dado el creciente número de estudiantes en las universidades, se ofrecen una gran cantidad de cursos cada trimestre. Cada curso tiene un número diferente de estudiantes matriculados y cada aula tiene diferentes capacidades que dificultan la asignación de cursos a las aulas. Además, no solo es suficiente programar un curso en el aula con mayor capacidad que el número de estudiantes matriculados, ya que esto puede llevar a una utilización ineficiente de las aulas, lo que puede causar dificultades para los docentes y los estudiantes. El problema de la automatización del calendario es, por lo tanto, una tarea importante ya que ahorra muchas horas de trabajo a las instituciones y proporciona soluciones óptimas con una satisfacción limitada que puede impulsar la productividad, la calidad de la educación y los servicios. Sin embargo, los horarios a gran escala, como los horarios universitarios, pueden requerir muchas horas de trabajo por parte de personas o equipos calificados para producir horarios de alta calidad con una satisfacción óptima de las restricciones y la optimización de los objetivos del cronograma al mismo tiempo" [7].

TABLA 2 - ESPECIFICACIÓN DEL PROBLEMA DE LOS HORARIOS [7]

Número	Descripción del Parámetro	Cantidad
1	Cantidad de Cursos	90

2	Cantidad de diferentes conferencias	200
3	Cantidad de eventos programados	210
4	Número de semestres	11
5	Tipo de conferencias (teoría / laboratorio)	2
6	Número de profesores	50
7	Número de aulas / laboratorios	19
8	Cantidad de días	5
9	Número de períodos dentro de un día	10

En este trabajo, se presenta el algoritmo FGH para UCTP. La técnica utiliza una representación indirecta que presenta las prioridades de asignación de eventos e invoca la rutina del generador de horarios para construir el cronograma completo. El algoritmo incorpora una cantidad de técnicas y operadores de búsqueda locales heurísticos específicos del dominio para mejorar la eficiencia de la búsqueda. Las restricciones blandas no rígidas implicadas en el problema son básicamente objetivos de optimización para el algoritmo de búsqueda" [7].

"Este documento estudia el problema de la programación de cursos universitarios en un estudio de caso relacionado con el sistema de gestión de la facultad, que se ocupa de asignar estudiantes / conferenciantes a clases y horarios. Para aumentar las opciones de flexibilidad del intervalo de tiempo para "repetidores", el problema se ha vuelto más apretado en la programación. Se presenta un enfoque heurístico de dos etapas, donde la etapa inicial agrupa los cursos que pueden conducir simultáneamente. Luego, la segunda etapa asigna los intervalos de tiempo semanales para cada grupo de cursos, seguidos del lugar para cada curso. Los resultados computacionales se presentan para la solución propuesta utilizando datos reales. Muestra que la solución propuesta es efectiva para manejar el horario del curso de la facultad" [8].

2.2.3 Programación con restricciones

Es un paradigma de la programación en informática, donde las relaciones entre las variables son expresadas en términos de restricciones (ecuaciones). Actualmente es usada como una tecnología de software para la descripción y resolución de problemas

combinatorios particularmente difíciles, especialmente en las áreas de planificación y programación de tareas (calendarización).

"La programación con restricciones. Se ajusta muy bien a la resolución de este tipo de problemas, puesto que las características del modelo matemático formulado guardan una similitud con las del modelo propio de la técnica, lo que significa una reducción de la complejidad del proceso de traducción de un modelo a otro [11].

"El problema de programar exámenes y cursos es de mucho interés y preocupación para las instituciones académicas. El problema básico es asignar un intervalo de tiempo y una sala para todos los eventos (exámenes, conferencias, seminarios, tutoriales) dentro de un número limitado de espacios de tiempo y salas permitidas para encontrar un horario factible. Este proceso de asignación está sujeto a restricciones 'duras' que deben cumplirse para obtener un cronograma factible. Un ejemplo de tal limitación es que no se requiere que ningún estudiante asista a dos eventos al mismo tiempo. Además, también es importante construir un cronograma de conferencias de buena calidad que considere no solo los requisitos administrativos, sino que también tenga en cuenta las preferencias de los profesores y los estudiantes. En general, es deseable (pero no esencial) satisfacer estas preferencias y, como tales, se denominan restricciones "blandas". Como esta tarea lleva mucho tiempo y es tediosa de llevar a cabo manualmente, se ha dedicado mucho esfuerzo en las últimas décadas para generar horarios de forma automática. Con la necesidad de asignar un gran número de eventos a los recursos (espacios de tiempo y habitaciones) y una lista de restricciones (tanto físicas como dinámicas), existe una gran cantidad de posibles soluciones a este problema. Además, el proceso de generación de horarios es complejo, con una serie de puntos clave de decisión. Dos puntos principales de decisión son cómo construir soluciones factibles y cómo evaluar su efectividad (esencialmente, cómo decidir cuál de las diversas soluciones alternativas es "la mejor"). Se deben considerar muchos factores en ambas áreas clave de decisión, con mucha información disponible. Hasta la fecha, ha habido relativamente poca investigación sobre cómo se puede combinar la información disponible, con el objetivo de lograr mejores soluciones" [3].

"Se presenta un sistema de información basado en la necesidad de respaldar la producción y el mantenimiento del cronograma. Además de los resultados muy prácticos que se esperan de la investigación del cronograma, el sistema de información fue diseñado para permitir que toda la gama de funciones administrativas realizadas por los docentes sean apoyadas directamente o modificadas fácilmente para probar dicho apoyo. Se da la

implementación de este sistema en particular y se presentan y se discuten los calendarios resultantes. El sistema generó horarios manuales y automatizados y estos fueron producidos al final de una serie de funciones objetivas. Se observó que la determinación de la función objetivo óptima está dominada por criterios institucionales individuales específicos. Se sugiere que esto haría un proyecto más significativo para la investigación futura de sistemas de información. De la literatura se observa que el problema de la generación del cronograma, como se ha informado una y otra vez en la literatura, se ha resuelto. Tales afirmaciones conducen a un punto de referencia que se propone para permitir una comparación inicial de la efectividad de las soluciones propuestas por diferentes investigadores" [26].

"Podemos ver el problema de la relación de horarios como un conjunto de variables y restricciones. El principio de *timetabling* basado en restricciones es una ubicación de restricciones entre variables y sus dominios. Una restricción define restricciones en los valores de las variables o posibles relaciones entre variables y sus valores. Las restricciones se dividen en restricciones duras y restricciones blandas, aunque una restricción suave bajo ciertas circunstancias podría comportarse como una restricción difícil. Restricciones duras Las limitaciones duras representan limitaciones absolutas impuestas en el calendario. Su satisfacción es necesaria para que el horario sea válido. Las limitaciones de recursos, como los instructores disponibles y la capacidad de la sala, pueden marcarse como restricciones duras típicas. Por ejemplo, una ubicación de tiempo o una ubicación de habitación no se pueden asignar a más de un curso. Un curso debe ser enseñado en un lugar con suficiente capacidad para los estudiantes. Un instructor no puede enseñar más cursos al mismo tiempo. Un calendario con todas las restricciones difíciles satisfechas y todas las clases asignadas a las ubicaciones de tiempo y sala se denomina horario factible. Restricciones suaves La satisfacción de las restricciones suaves se refleja en una función objetiva de una solución, que expresa el valor y la idoneidad del horario computado. Los criterios de optimización típicos de la solución de un problema son las preferencias de tiempo, las preferencias de sala y las preferencias de distribución (restricciones entre varias clases, como la precedencia, etc.). La motivación para las preferencias podría provenir de instructores con preferencias específicas en el momento de la enseñanza, o salas, donde deberían impartirse sus cursos. Ciertamente, los requisitos del curso también deben tenerse en cuenta. Por ejemplo, la enseñanza con equipo especial puede ser necesaria para ciertas clases, una clase magistral debería tener lugar antes de que los seminarios o una clase en particular se enseñen uno tras otro. Por lo general, los estudiantes y los maestros no recomiendan los turnos de tiempos tempranos y tardíos.

Además, una clase debe asignarse a una sala con una capacidad adecuada, las salas demasiado grandes para clases pequeñas también se desalientan. El número de conflictos estudiantiles en un horario es otro criterio de optimización común. Los cursos inscritos por un estudiante (o cursos en un plan de estudios) no deben tener lugar al mismo tiempo, de modo que el alumno pueda asistir a todos ellos. De lo contrario, los cursos están en el duro conflicto estudiantil. Si los cursos se llevan a cabo en habitaciones distantes entre sí, es posible que un alumno no pueda asistir a ellas aunque no se superpongan debido a la distancia entre el alumno y el alumno. Un conflicto suave de estudiantes surge entre cursos que consisten en clases más equivalentes de las cuales el estudiante debe asistir al menos a uno. Si las clases a las que asiste el alumno se superponen, existe la posibilidad de que el alumno asista a otra clase equivalente" [17].

"En la programación de restricciones, los problemas combinatorios se especifican declarativamente en términos de restricciones. Las restricciones son relaciones sobre variables problemáticas que definen el espacio de soluciones al especificar restricciones sobre los valores que las variables pueden tomar simultáneamente. Para resolver los problemas planteados en términos de restricciones, el programador de restricciones típicamente combina el retroceso cronológico con la propagación de restricciones que identifica combinaciones de valores no factibles y poda el espacio de búsqueda en consecuencia. En los últimos años, la programación de restricciones se ha convertido en una tecnología clave para la optimización combinatoria en aplicaciones industriales. En este éxito, las restricciones globales han estado desempeñando un papel vital. Las restricciones globales [AB93] son abstracciones cuidadosamente diseñadas que, de forma concisa y natural, permiten modelar problemas que surgen en diferentes campos de aplicación. Por ejemplo, la restricción *alldiff* (una restricción completamente diferente) permite establecer que las variables deben tomar valores distintos por pares; tiene numerosas aplicaciones en horario y programación. En horario escolar, se nos exige que programemos un conjunto dado de reuniones entre alumnos y profesores a la misma hora, los horarios resultantes son factibles y aceptables para todas las personas involucradas. Dado que las escuelas difieren en sus políticas educativas, el problema del horario escolar se produce en varias variaciones. Sin embargo, existe un conjunto de entidades y restricciones que son comunes a estas variaciones. Este núcleo común todavía da lugar a problemas combinatorios NP-completos" [18].

"Todos estamos haciendo un horario que ayuda a organizar nuestro día a día. Haciendo un cronograma para cualquier actividad organizada, uno considera una secuencia de tareas y una lista de recursos. Los recursos incluyen herramientas, máquinas, materiales,

fuerza de trabajo, etc. No solo podemos hacer nuestro propio cronograma. Nuestros empleadores están haciendo un cronograma para nosotros. Y es más difícil hacer un cronograma de organización, que hacer un cronograma para nuestra vida cotidiana. Una falla al hacer un cronograma puede traer muchos resultados negativos. Como ejemplo, estamos usando la programación de la escuela tradicional. Es una tarea muy cercana y real de la vida real. Aquí se puede cambiar la secuencia de asignaturas docentes, consideradas como herramientas. Uno necesita reducir la suma de las brechas (horas "vacías") en los horarios del maestro. No debe haber lagunas para los estudiantes. Las diferentes clases se consideran tareas diferentes. Las aulas, incluidas las salas y estudios de informática y física, son recursos limitados. Lo más difícil en la escuela de hoy es hacer un horario para la escuela secundaria. Aquí los alumnos de undécimo y duodécimo grado eligen varias materias de la lista de las disponibles (por ejemplo, 10-14 de 60). Las horas máximas, que los alumnos pueden elegir para una asignatura, se describen en la Tabla 1. Esto significa que cada alumno trabaja según su propio horario. Buscamos el horario factible más conveniente. Los puntos de penalización evalúan los inconvenientes. Consideramos el caso donde el objetivo es el número de horas "vacías" cuando los profesores o alumnos esperan las próximas clases programadas. Estas horas vacías en la escuela se llaman "brechas" para abreviar. Buscamos un horario tal que reduzca la suma de todas las brechas considerando los horarios de la undécima y duodécima clase de la escuela integral. El "programa de optimización del horario escolar" se crea de una manera que ayuda al usuario, que solo debe ingresar los datos iniciales, elegir los parámetros de optimización y el algoritmo de optimización. Un usuario puede poner datos iniciales en el programa "MS Excel" fácilmente. El programa en sí hará toda la optimización. Si un usuario modifica los datos iniciales, no tiene que preocuparse por restablecer la programación o asumir una forma más óptima que la que muestra el programa. Todo esto se puede hacer reiniciando los nuevos parámetros de optimización. Una vez finalizado el trabajo, el programa mostrará los resultados no solo en la pantalla sino también en el archivo "zip". Este archivo "zip" que un usuario puede enviar a su computadora y analizarlo cuando lo desee. Ayuda a elegir el mejor resultado de muchos de ellos” [23]

“Programación asociada con actividades para asignar recursos en una sola vez. El recurso especificado es recursos humanos (como profesores y estudiantes), recursos físicos (como espacio, equipo y capacidad) y, por supuesto, una variedad de sus limitaciones en los recursos del tiempo. En otras palabras, la programación fue un problema debido a que su complejidad se vuelve cada vez más lineal junto con los cambios en la cantidad de recursos que se administran. Ciertamente implica mayores necesidades informáticas, las

necesidades del tiempo de procesamiento, las necesidades de colaboración cuando la programación implica más de un nivel" [14].

• "Los procesos de programación actuales son monolíticos y centralizados en un solo personal. El proceso de programación en sí requiere mucho tiempo y requiere una revisión de las distintas partes. Además, existe el riesgo de perder los resultados de la programación en daños no resueltos debido a amenazas de seguridad tales como virus informáticos y otros" [14].

• La organización que tiene múltiples estudios de programas pero tiene recursos humanos limitados requerirá un proceso prolongado, recursos informáticos y participación del personal" [14].

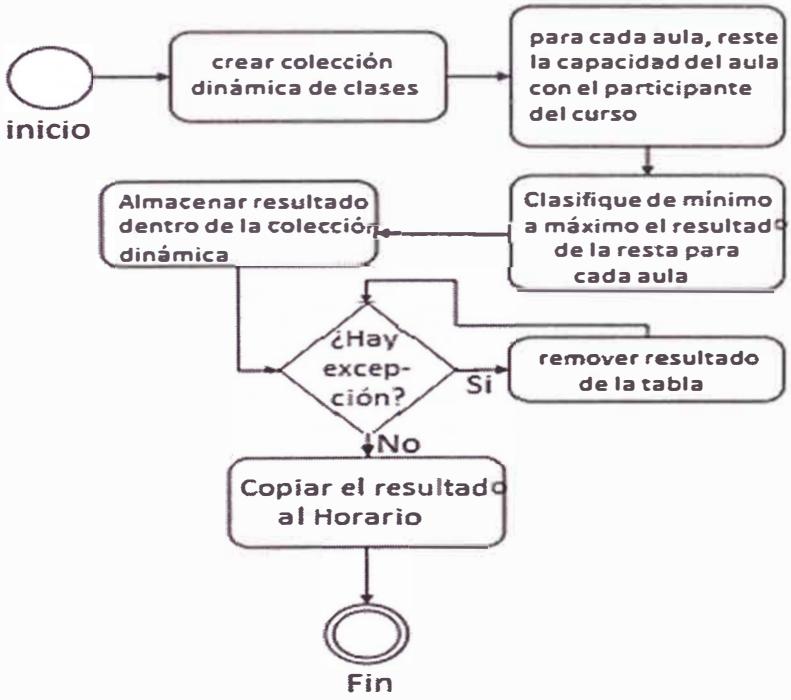


Figura 7 Algoritmo pragmático para la generación de horarios [14]

"El calendario de exámenes (ETT) es una tarea administrativa compleja en las instituciones educativas que debe cumplir varias restricciones para generar el ETT para programar sesiones de exámenes dentro de un período preciso. El problema ETT podría modelarse como Problemas de Satisfacción de Restricciones (CSP). Además, podría ser investigado en particular por el enfoque de Programación Lógica de Restricciones (CLP). Este documento utiliza un conjunto de datos de exámenes reales del *Community College* (CC), Universidad de Sharjah (UoS). Este conjunto de datos tiene datos muy abundantes, como la gran cantidad de inscripciones de estudiantes relacionadas con muchos departamentos, un número acumulado de cursos combinados, escaso número de salas de

examen, escalas de tiempo muy limitadas, bajo número de supervisores y campus distantes.

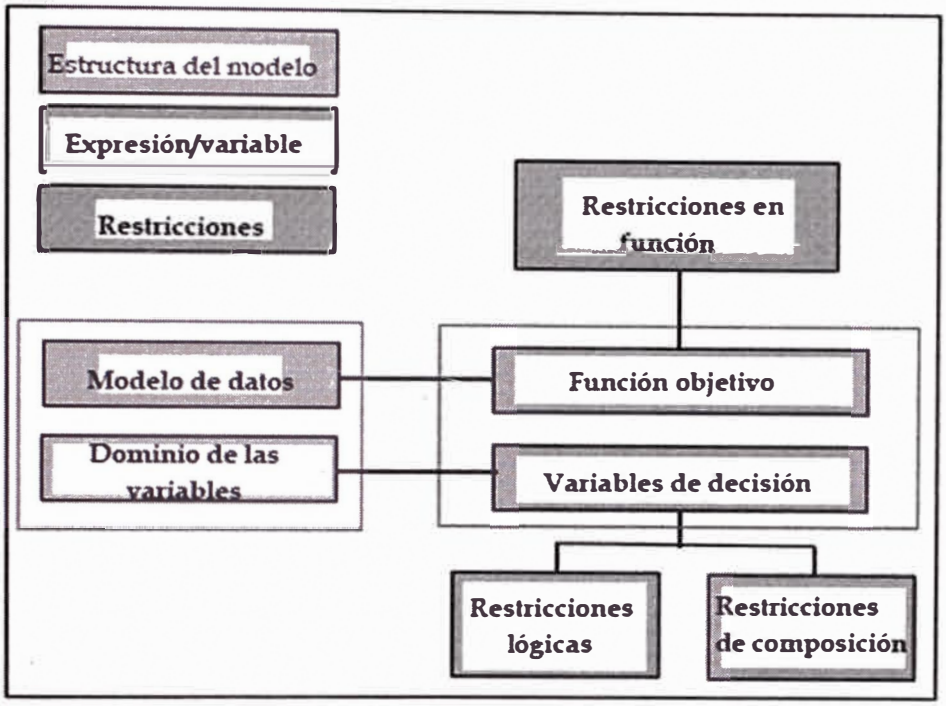


Figura 8 El modelamiento del calendario de exámenes [13]

Este conjunto de datos tiene muchas limitaciones prácticas que cumplir, como un curso impartido en muchos campus debe tener la misma fecha de examen y un vigilante puede participar en cualquier campus. Este documento aplica las definiciones CSP, así como el Lenguaje de programación de optimización (OPL) para modelar el conjunto de datos ETT y generar automáticamente una solución ETT libre de conflictos utilizando un CLP *Solver*. Finalmente, usa los resultados para satisfacer las restricciones propuestas en el modelo" [13].

"En la gestión de los sistemas de transporte público, la programación de horarios busca ajustar los tiempos de salida de los vehículos al flujo dinámico de pasajeros, proporcionar un servicio de alta calidad y optimizar el uso de recursos limitados. En este documento se propone un nuevo método de programación de horarios en un entorno difuso con flujo inverso mejorado, que optimiza todos los servicios de bus. La toma de decisiones se basa en el grado de satisfacción de los pasajeros y el grado de uso de la capacidad del vehículo. En comparación con el diseño de calendario incluso avanzado, el calendario propuesto puede ajustarse al flujo variable de pasajeros, al tiempo que equilibra la satisfacción del pasajero y la carga del vehículo, puede acortar el tiempo de espera de los pasajeros y está optimizado para cualquier período de servicio" [30].

"El problema del horario es un problema bien conocido multidimensional de asignación de restricciones que se enfoca en la asignación de cursos a miembros de la facultad en aulas dentro de espacios de tiempo limitados. Por lo tanto, es un problema desafiante que consume mucho tiempo que enfrentan las universidades y pertenece a la clase de problemas NP-difícil. En particular, las universidades necesitan regularmente una solución óptima para el problema del horario del curso. Sin embargo, una solución manual a este problema teniendo en cuenta todas las limitaciones por lo general requiere un largo tiempo y un trabajo arduo para ofrecer una solución adecuada y optimizada. Específicamente, el escaneo de problemas de tiempo se modelará como problemas de satisfacción de restricción (CSP), que son de naturaleza combinatoria. Particularmente, se utilizan una variedad de enfoques para investigar los CSP, como la programación de restricción de lógica (CLP) que combina la programación declarativa de la lógica con la eficiencia de los métodos de la investigación operativa y la inteligencia artificial. Este documento presenta un modelo que aplica CLP al horario como CSP en el uso de un Lenguaje de Programación de Optimización (OPL). El modelo propuesto se prueba con datos reales obtenidos del *Community College* (CC), de la Universidad de Sharjah (UoS), que tiene un horario tan sofisticado con muchos recursos entrelazados, limitados y espacios de tiempo limitados" [12].

CAPÍTULO III
DESARROLLO DEL TRABAJO DE TESIS

3.1 Solución planteada

Estado del arte para resolver el problema de horarios:

Existen a la fecha varias formas de resolver el problema de asignar horarios e infraestructura para fines universitarios, entre ellos tenemos los Algoritmos Genéticos (AG) que son métodos para resolver problemas basados en una abstracción del proceso de selección natural, los métodos heurísticos y la programación con restricciones.

NP es el acrónimo de *Nondeterministic Polynomial Time* ("tiempo polinomial no determinista"). Es el conjunto de problemas que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista.

NP-hard es el conjunto de los problemas de decisión que contiene los problemas H tales que todo problema L en NP puede ser transformado polinomialmente en H. Esta clase puede ser descrita como aquella que contiene a los problemas de decisión que son como mínimo tan difíciles como un problema de NP. Esta afirmación se justifica porque si podemos encontrar un algoritmo A que resuelve uno de los problemas H de NP-hard en tiempo polinómico, entonces es posible construir un algoritmo que trabaje en tiempo polinómico para cualquier problema de NP ejecutando primero la reducción de este problema en H y luego ejecutando el algoritmo A..

Podemos mencionarlos:

1. Algoritmos genéticos (AG) tales como el NP-Hard usan optimización combinatoria multi-restringida, pero tiene la ventaja de la flexibilidad de elegir dentro de un conjunto de diferentes horarios [9] y el NP-complete es útil para problemas de tiempos, donde la coloración de gráficos es de este tipo, permitiendo presentar los horarios como una estructura en 3D [2]
2. Enfoque heurístico, que conduce a un conjunto de buenas soluciones, pero no necesariamente la mejor solución [20]
3. Algoritmo genético híbrido basado en un marco de calendario heurístico [6]

4. En la programación con restricciones, los problemas combinatorios se especifican declarativamente en términos de restricciones, éstas son relaciones sobre variables problemáticas que definen el espacio de soluciones al especificar restricciones sobre los valores que las variables pueden tomar simultáneamente.

Vamos a emplear los algoritmos genéticos, la heurística y la programación con restricciones, integrándolas mediante una serie de pasos para elaborar un horario universitario en una aplicación para PC, que manejará en una forma integrada en un esquema administrativo organizado basado en la telemática que es modelando la realidad, crear un modelo y éste deberá ejecutarse en el computador y que tenga una estructura que permita solucionar los problemas relacionados con la estructuración de horarios.

3.2 Restricciones

TABLA 3 - RESTRICCIONES

Restricciones suaves	Restricciones medias	Restricciones duras
Los estudiantes deben tener un descanso de 1 hora por cada cuatro horas de clase por día	Las materias de las secciones se colocan en el horario	Docentes enseñan una sola materia en un horario
Los docentes deben tener una carga distribuida basada en el grupo de asignaturas del docente	Las secciones deben tener al menos 1 hora de tiempo libre entre las 11:00 AM y la 3:00 PM para el almuerzo	El docente enseña en su horario disponible
A los cursos se les asignará un docente si hay un conjunto de docentes en su línea de cursos	Las secciones deben tener al menos 1 hora de tiempo libre por cada 4 horas consecutivas de clase	Los docentes solo pueden tomar N cantidad de cursos dependientes de su cantidad máxima de carga horaria
	Los docentes deben tener al menos 1 hora de descanso por cada 4 horas consecutivas de clase.	Las secciones tienen una clase a la vez
	Los cursos se dividirán según su configuración	Las secciones asisten a su horario disponible

Restricciones suaves	Restricciones medias	Restricciones duras
	Los cursos deberán tener el tipo de salón adecuado	Las secciones permanecerán en un salón a menos que se tome una clase de laboratorio
		Las secciones con temas que se comparten con otras secciones deben producir un horario que sea compatible con todos los participantes que comparten
		Los salones solo pueden tomar una clase en su horario disponible

Es importante seleccionar un diseño de investigación y metodología apropiadas para obtener la viabilidad de la investigación con la ayuda de la inteligencia artificial, que es un campo de la informática que se usa principalmente en problemas que incluyen precisión u optimización. La investigación se enmarca en el problema de optimización.

Metodología de diseño de investigación

El método seleccionado para recopilar y analizar datos para probar el sistema es cuantitativo. Los datos que se recopilarán son conjuntos de configuración de la facultad (docentes, cursos y aulas). Estos conjuntos de datos serán perfilados después de ser utilizados en el sistema. Los resultados de los perfiles determinarán el rendimiento del sistema mediante la realización de sistemas de inteligencia artificial basados en la evaluación.

Método de desarrollo de software

El desarrollo iterativo como método para crear el software se ajusta al entorno continuamente cambiante para el desarrollo de la inteligencia artificial. Esto permite que el desarrollador aprenda cada iteración y la aplique a una instancia futura. Esta metodología es adecuada para la investigación ya que el sistema se basa en ajustes y restricciones de gestión. Cada iteración es una mejora del sistema que se repetirá hasta que el sistema entregue el resultado esperado.

La planificación inicial se ha realizado mediante el diseño de la arquitectura del sistema, el esquema de datos y los modelos. Al ingresar una iteración, la característica que se debe hacer es decidida y planificada. Hay dos tipos de iteraciones utilizadas; agregar una característica o un modelo y ajustar valores. El objetivo de cada iteración se implementa y luego se prueba manualmente.

Método de evaluación

La metodología de evaluación seleccionada para la investigación es una combinación de métodos de Monte Carlo y modelos sustitutos. Es una metodología experimental ya que la evaluación de los sistemas de programación basados en cálculos evolutivos aún está por hacerse.

Los métodos de Monte Carlo son una clase de algoritmos computacionales que usan muestreo aleatorio repetido para obtener resultados numéricos. A menudo se usan bajo los conjuntos de problemas de optimización y probabilidad. Funciona ajustando el valor inicial e intentando producir una solución a partir de él.

El modelo sustituto se usa a menudo en el campo de la ciencia y la ingeniería, donde el resultado no se puede medir fácilmente en forma directa, por lo que se utiliza un modelo en su lugar. Un ejemplo de esto es encontrar el diseño óptimo para la aerodinámica de un automóvil. Compilar el rendimiento del diseño con simulaciones de la vida real es costoso, por lo que se utiliza un modelo de emulador. En general, esto funciona al crear una aproximación de modelos que imita lo más posible el comportamiento del modelo real sin la necesidad de una evaluación más costosa.

La implementación del método de Monte Carlo y modelado sustituto para la evaluación del sistema de programación basado en inteligencia artificial, específicamente el estudio del investigador, debe seguir los siguientes pasos:

1. Generación de un conjunto de soluciones que usan el método de Monte Carlo con ajustes de algoritmos genéticos como variables que pueden modificarse.

2. Creación de un modelo sustituto que solo utiliza la matriz de evaluación como base.

3. Cálculo de cada conjunto de soluciones generadas por la diferencia de método de Monte Carlo al modelo sustituto.

4. Proporcionar la media del resultado como base para el rendimiento de la inteligencia artificial (basada en escenarios).

El uso del método de Monte Carlo en el sistema es para la generación de configuración de algoritmo genético aleatorio. El uso de modelos sustitutos para esta investigación utiliza una métrica imaginaria de 1 dimensión basada en la matriz de evaluación. El modelo servirá como base para evaluar la cercanía de la solución al ideal. Junto con la metodología de evaluación de software utilizando los pasos propuestos anteriormente, la recopilación de datos se inclinaría hacia un enfoque centrado en los datos. Esto es para proporcionar conclusiones correctas hacia la conclusión de la investigación. Los datos propuestos que se recopilarán son los que se utilizarán en la evaluación anterior, que se compondrá principalmente de resultados en rendimiento (aptitud)

3.3 Análisis:

Como nuestra investigación es sobre un problema complejo que no tiene una única solución sino soluciones aceptables se considera el uso de la inteligencia artificial usando algoritmos genéticos.

La inteligencia artificial es un campo de la informática que se usa principalmente en problemas que requieren precisión u optimización. La investigación se enmarca en el problema de optimización y la evaluación del rendimiento en una verificación exhaustiva no es una solución viable. Es importante seleccionar un diseño de investigación y metodología apropiadas para obtener la viabilidad de la investigación.

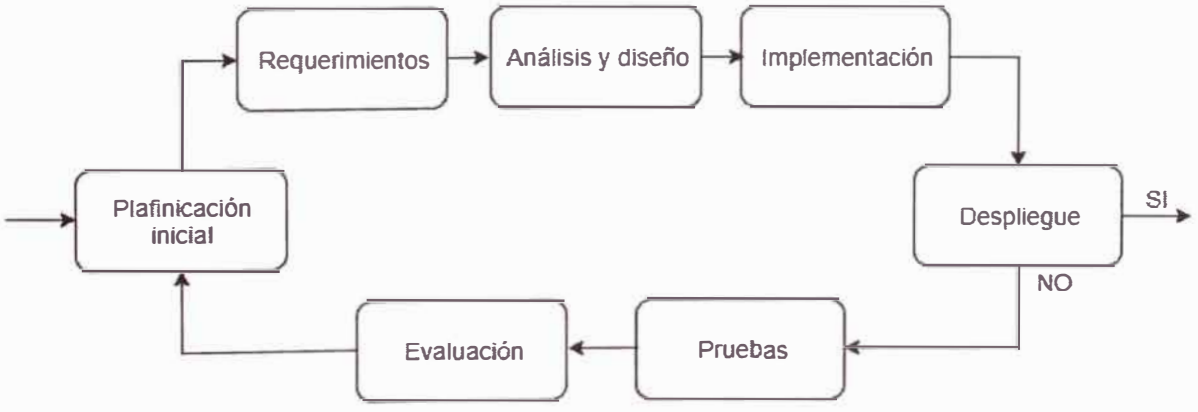


Figura 9. Desarrollo iterativo

3.3.1 Diagramas de bloques

A continuación, mostraremos los diferentes diagramas de bloques:

El diagrama de bloques muestra la secuencia de trabajo se alimenta con los docentes, cursos y aulas, luego van las restricciones, se verifica y se genera el horario por ciclos, luego viene de ser necesario la edición de algún parámetro, la revisión y la regeneración del horario final según sea el caso, finalmente se imprime y/o visualiza el horario final.

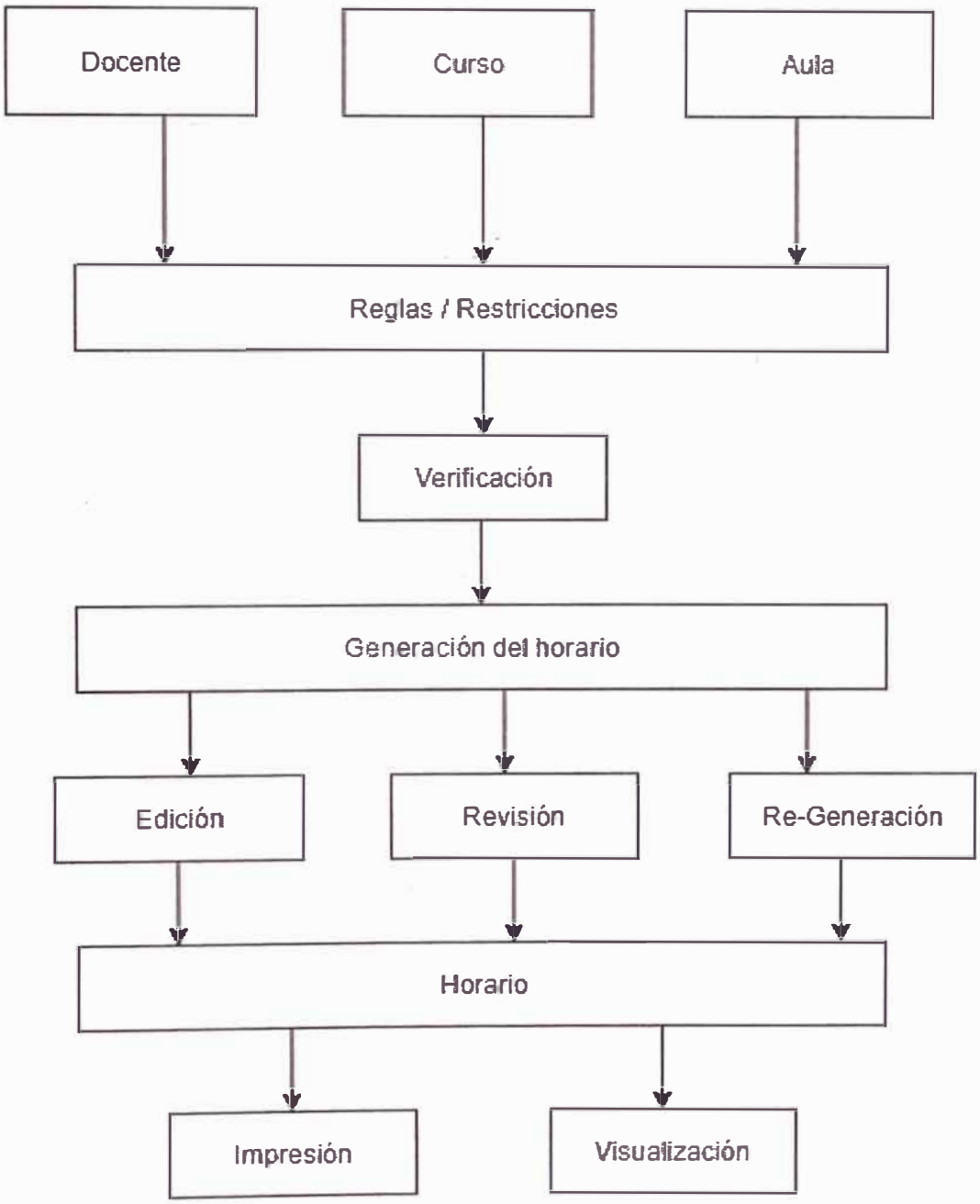
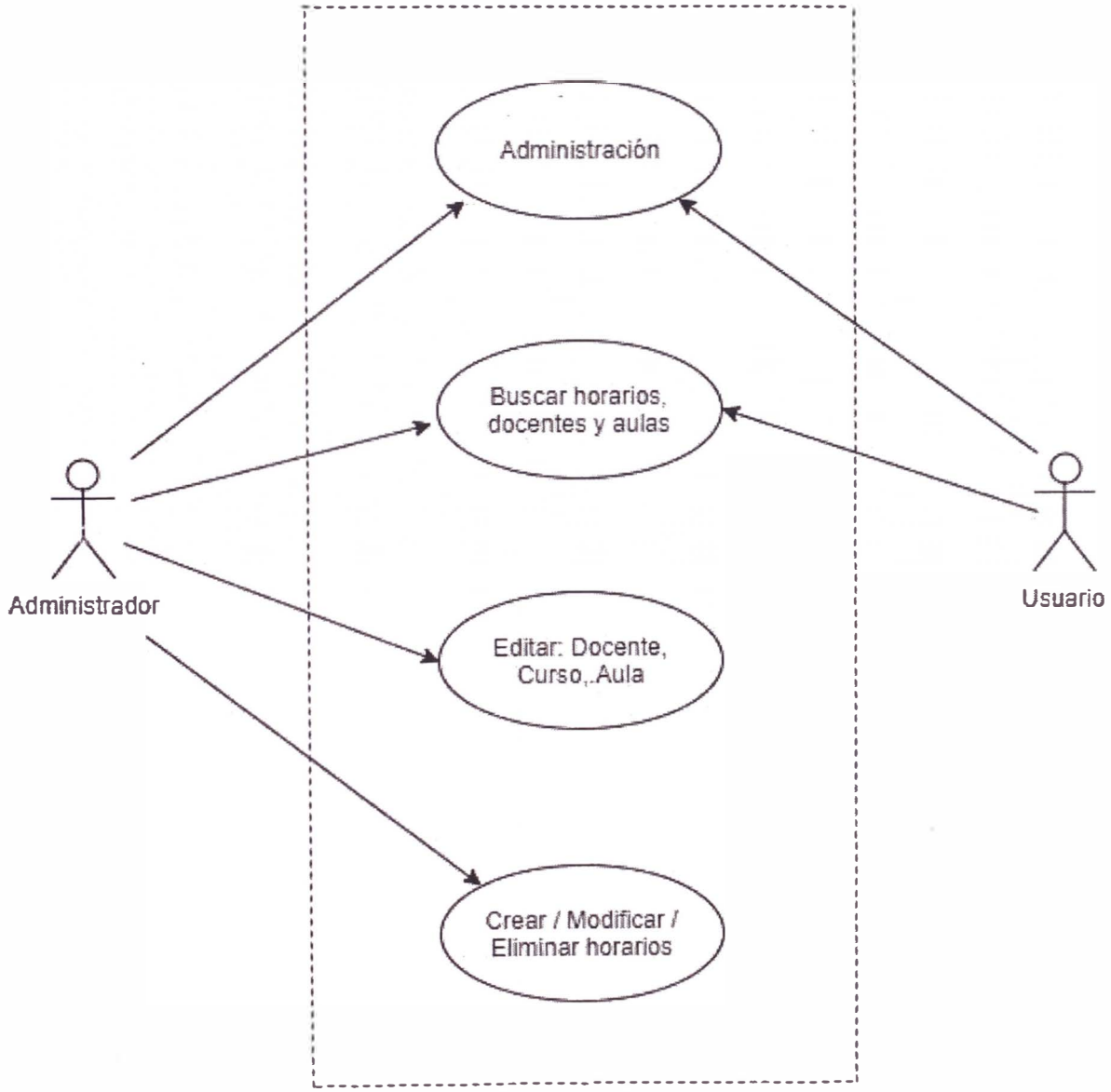


Figura 10 Diagrama de bloques del sistema de horarios

3.3.2 Diagramas UML usados en el modelaje de la solución

Para este diagrama será de caso de uso del sistema de administración de los horarios, en el cual los dos actores considerados, el administrador (docente) que tiene las atribuciones de conceder acceso a nuevos usuarios, cambiar contraseñas a usuarios actuales, editar parámetros del sistemas como son los atributos de los docentes, de los cursos y de las aulas, buscar horarios; crear modificar y eliminar horarios, y el usuario (docente), que puede: ingresar al sistema, visualizar los horarios terminados (buscar sus horarios e información al respecto)



En la siguiente página se muestra el diagrama de flujo, que muestra el proceso que inicia con ingresar el docente, el curso y el aula, luego establece las conexiones entre ellos

Figura 11 Diagrama de casos de usos del sistema de horarios

y considera los intervalos de tiempo en los que se dictará el curso, después sigue insertar las franjas horarios con sus tiempos de inicio y fin para cada una de ellas, luego se aplican las reglas para cada restricción, tales como que el docente solo debe tener su carga tres días a la semana de lunes a viernes para el caso de los docentes tiempo completos y a partir de las 4pm y sábados todo el día para los profesores a tiempo parcial, también que no debe haber cruces en el mismo ciclo ni en un ciclo anterior ni uno posterior, que debe haber al menos una hora para almorzar entre las 11 y 14 horas, finalmente se aplican los algoritmos genéticos para hallar la mejor solución y luego se imprimen (también se puede visualizar) el resultado de los mismos.

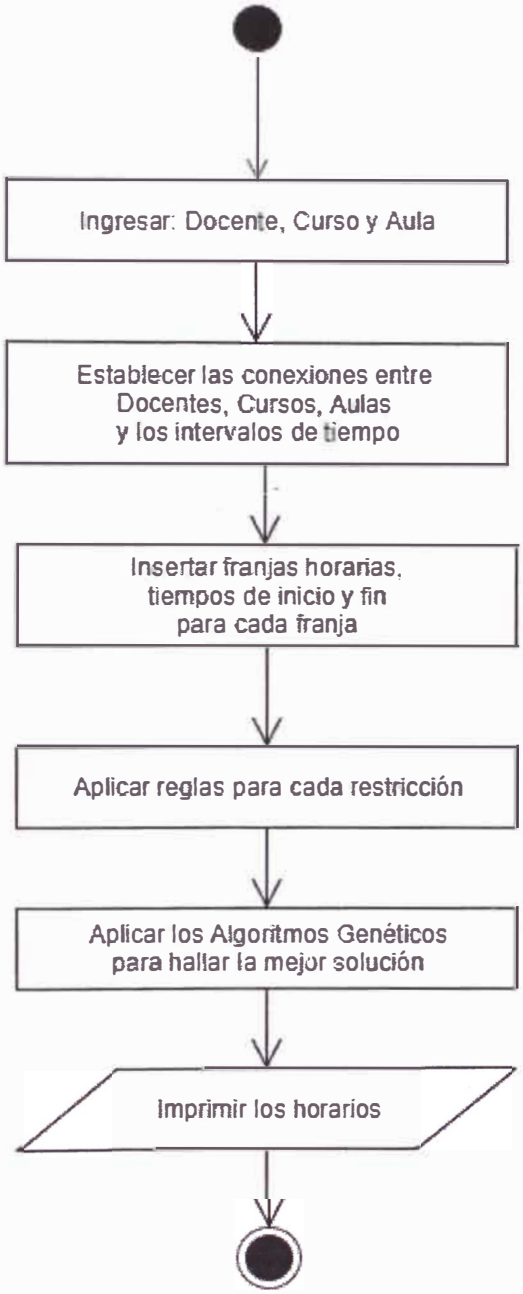


Figura 12 Diagrama de actividades del sistema de horarios

El "horario" puede ser editado teniendo las opciones de seleccionar cursos, salones, franjas, también la de agregar o eliminar alguna franja horaria, algún horario de clase cuando no haya suficientes alumnos para abrir o cerrar una sección, aumentar una nueva sección, etc.

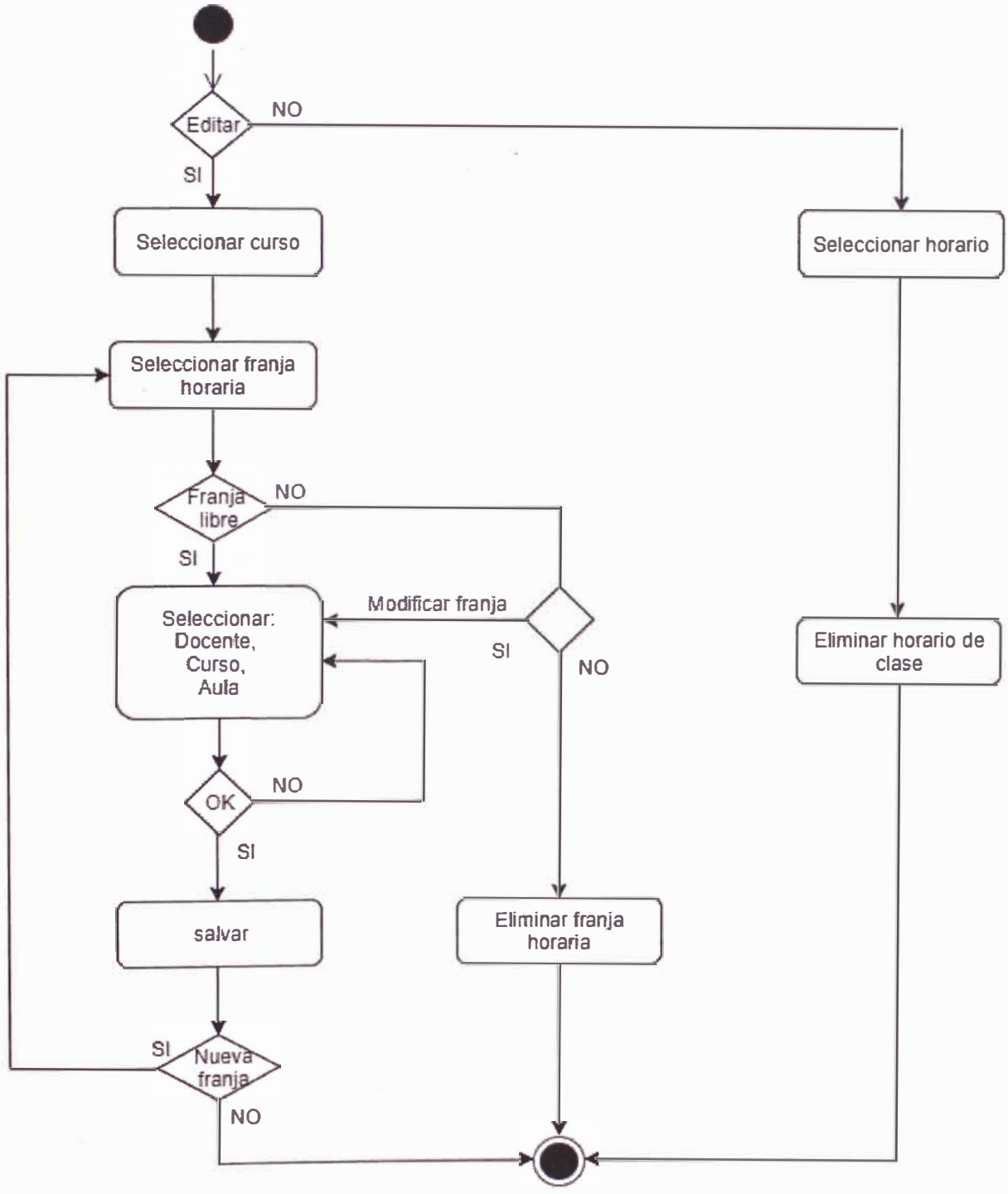


Figura 13 Diagrama de actividades para editar el horario

El bloque "horario" es el corazón del sistema donde está el algoritmo genético y puede crear, modificar, eliminar y buscar horarios tiene como insumos del docente, el curso y el aula así como las restricciones que deberá cumplir, cada módulo muestra su nombre , sus características y las operaciones que realiza.

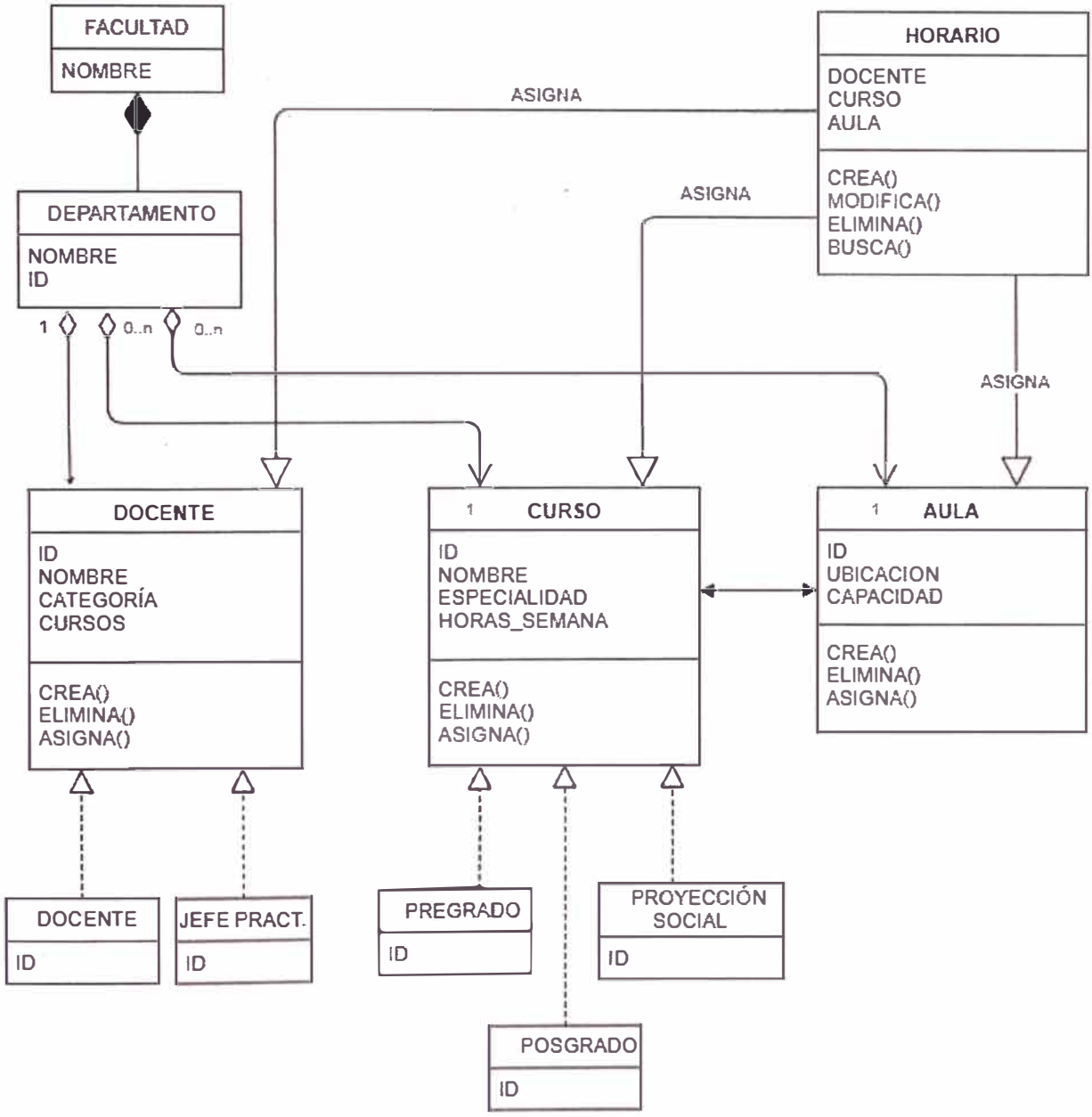


Figura 14 Diagrama de clases para el sistema de administración de los horarios

Dado que el número de docentes y aulas no es significativo por ser solo aplicable a una facultad con tres especialidades y no a toda la universidad, se usa una base de datos relacionales tipo MySQL, , el sistema de horarios asigna: Docente, Curso y Aula.

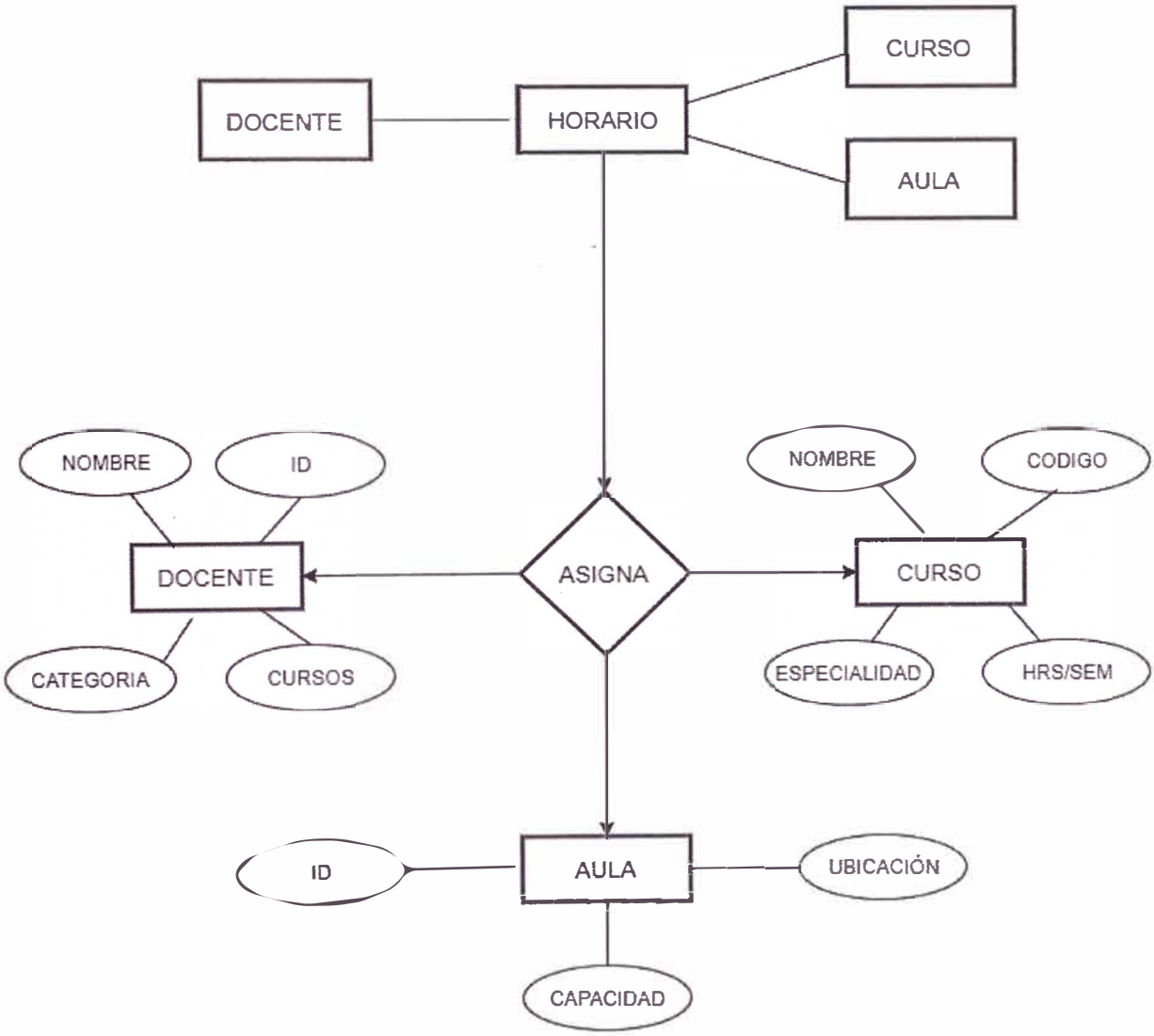


Figura 15 Diagrama entidad relación (E/R)

El proceso básico se incia el algoritmo genético (AG) con generar una población inicial, luego se evalúa cada cromosoma, si la respuesta es satifactoria se termina el proceso, caso contrario, se seleccionan cromosomas para el cruce, se realiza el cruce, luego la mutación y después se evalúa cada cromosoma hasta obtener el resultado adecuado

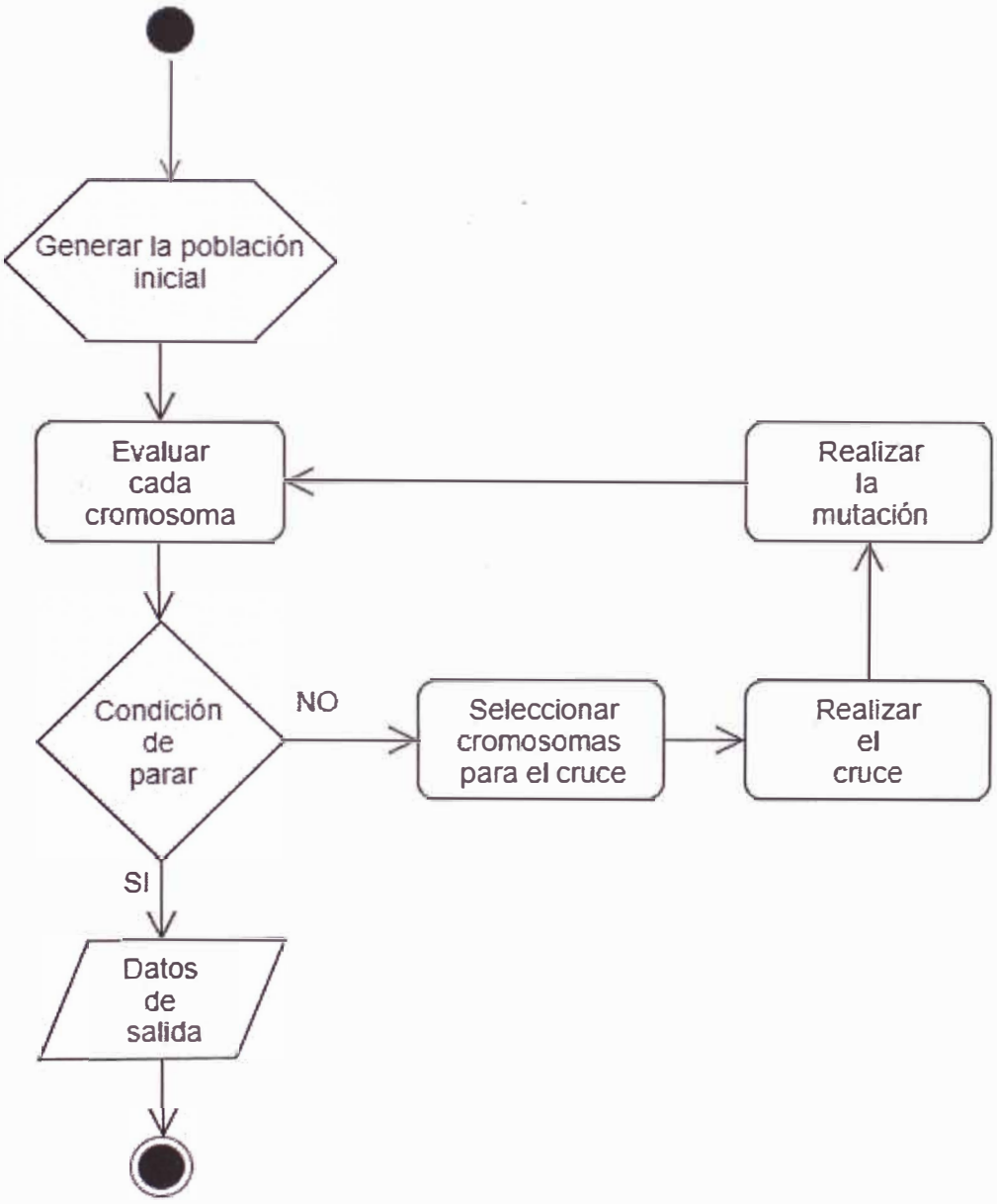


Figura 16 Proceso básico del algoritmo genético

La figura 17 muestra el diagrama de secuencias para generar un nuevo horario en alguna especialidad o en algún ciclo en particular, previamente se tienen que hacer los cambios en listado de datos (archivo en formato csv) para el nuevo horario, primero se selecciona el curso y la franja horaria que se requiere, luego se ingresan los datos del docente, el curso y el aula, luego de validarse los datos en el sistema y el administrador da su conformidad, luego el nuevo horario podrá ser solicitado, en caso no pueda validarse los datos habrá un mensaje de error.

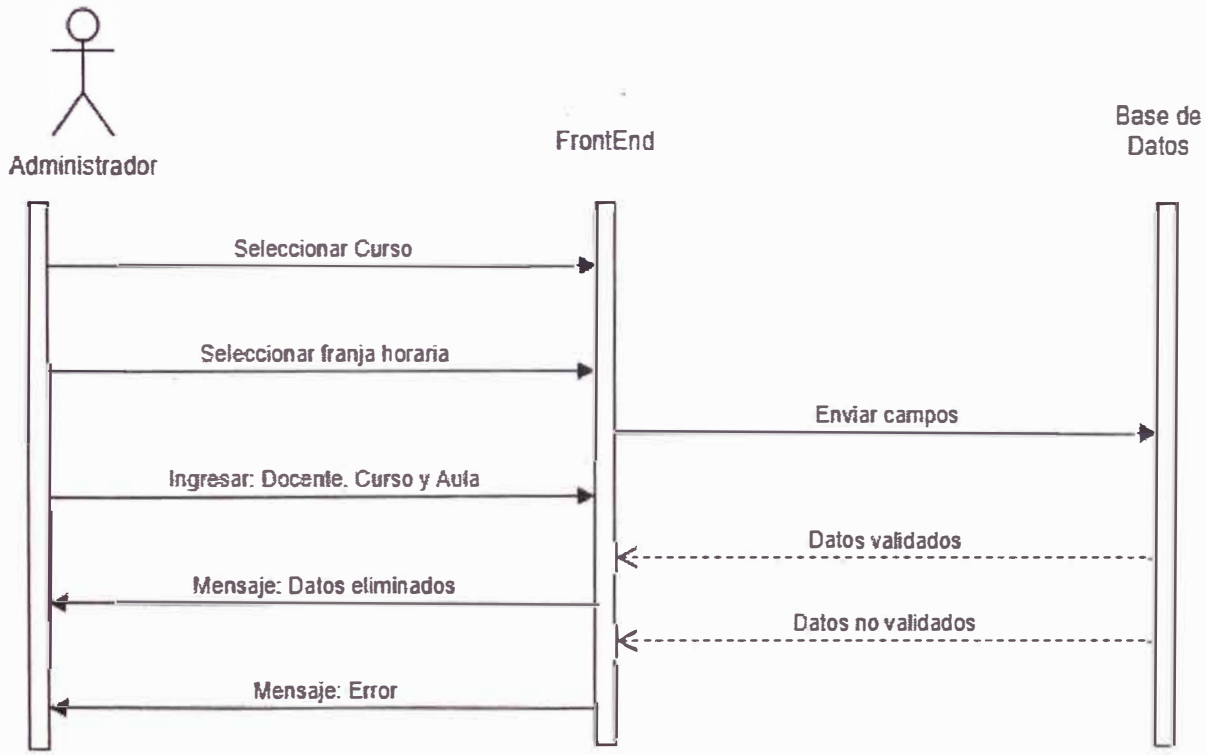


Figura 17 Diagrama de secuencias para crear un nuevo horario

La figura 18 muestra el diagrama de secuencias para borrar a un docente, que es el caso cuando un docente se jubila, renuncia, fallece u otra causal por la cual no va a dictar más, se borran todos los datos del docente como son la especialidad, la categoría, los cursos que dictaba, luego de validarse los datos en el sistema y el administrador da su conformidad el docente quedará borrado del sistema. En caso no puedan validarse los datos habrá un mensaje de error.

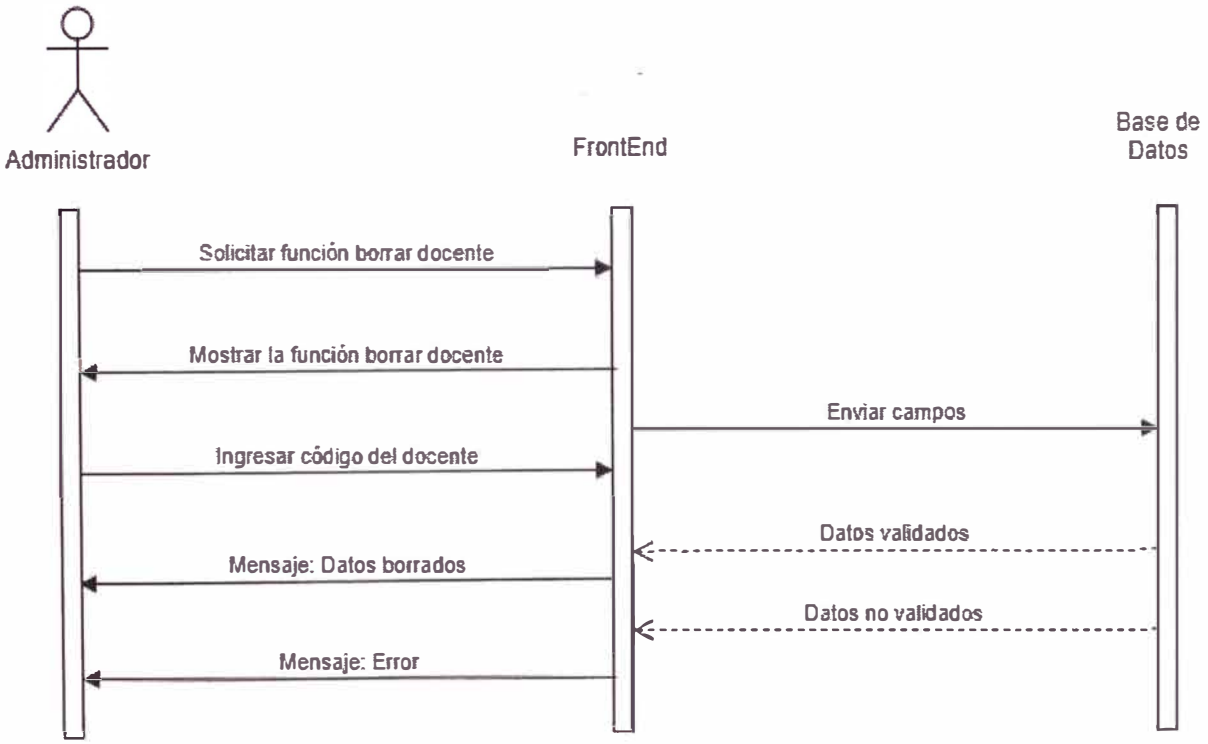


Figura 18 Diagrama de secuencias para borrar un docente

La figura 19 muestra el diagrama de secuencias para agregar un nuevo docente, que es el caso cuando reemplaza a un docente se jubila, renuncia, fallece u otra causal por la cual no va a dictar más, se ponen todos los datos del docente como son la especialidad, la categoría, los cursos que puede o va a dictar, se selecciona el curso y la franja horaria que se dispone para él, según sea docente a tiempo completo o tiempo parcial, luego de validarse los datos en el sistema y el administrador da su conformidad, el nuevo docente estará agregado al sistema. En caso no pueda validarse los datos habrá un mensaje de error.

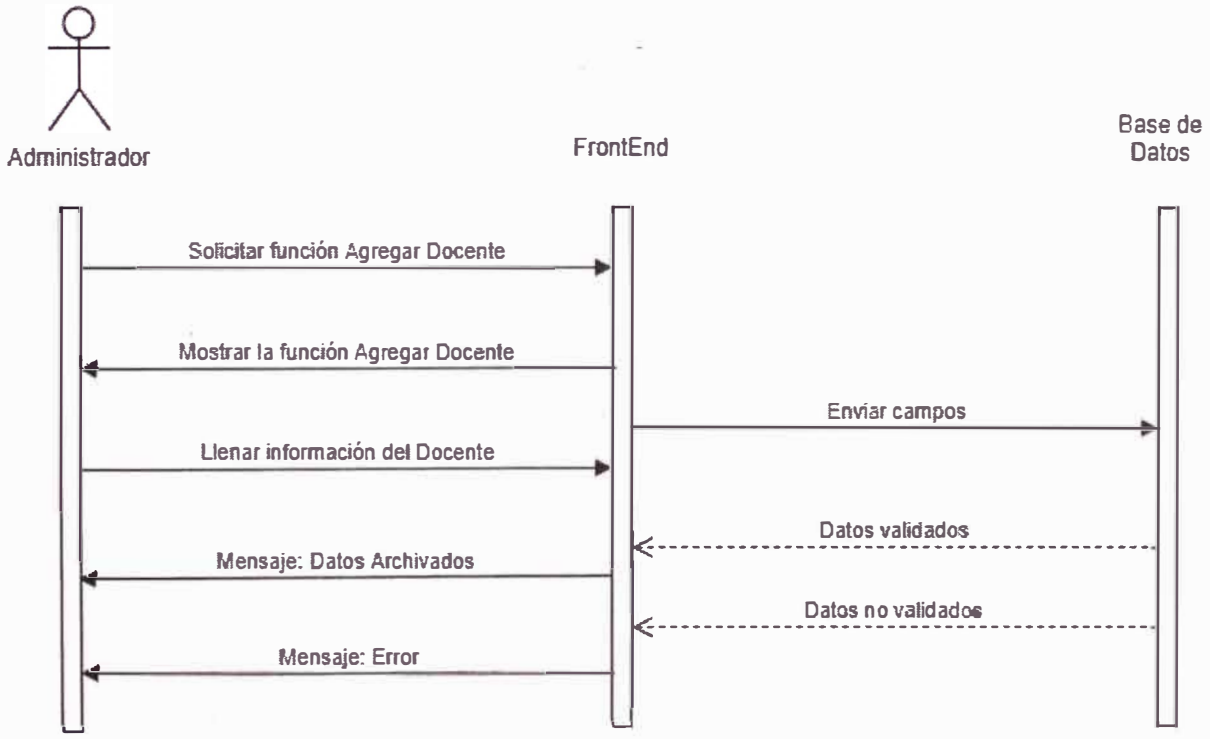


Figura 19 Diagrama de secuencias para agregar un nuevo docente

La figura 20 muestra la cadena de recopilación de datos y la metodología de evaluación para un modelo

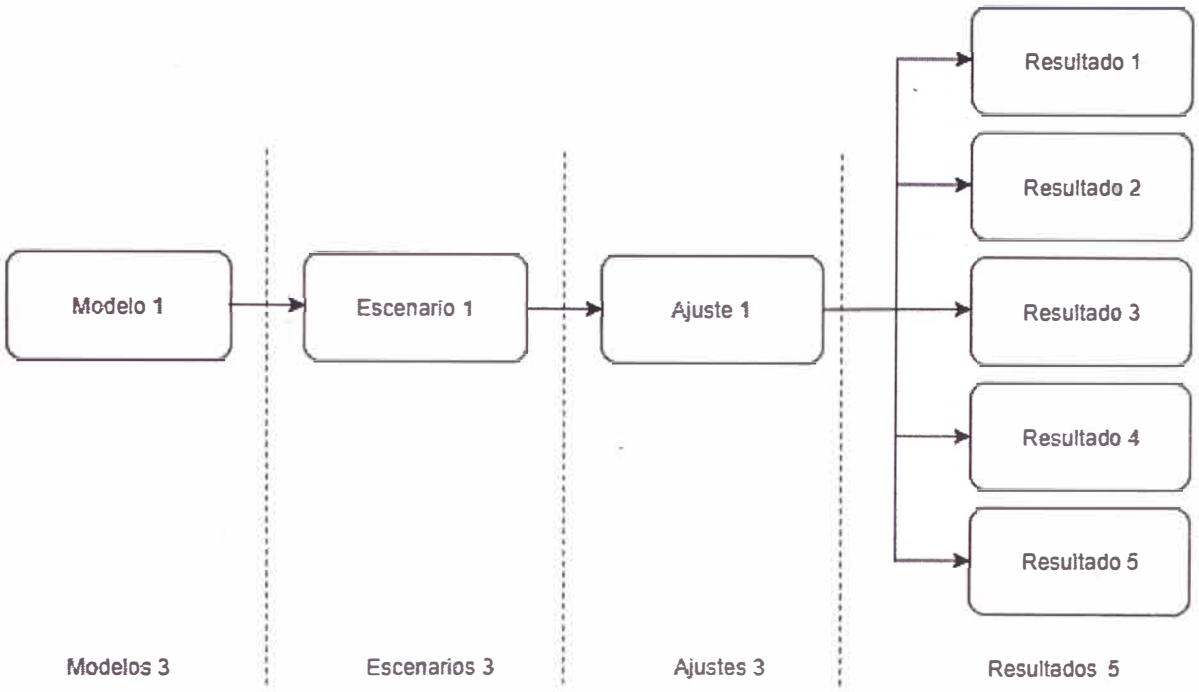


Figura 20 Árbol de evaluación

Habrá tres (3) modelos que se utilizarán enumerados en la tabla a continuación. Para cada modelo, habrá tres (3) escenarios. Para cada escenario, habrá tres (3) valores de configuración generados aleatoriamente. Es importante tener en cuenta que solo habrá una generación de valores aleatorios para la configuración y los tres (3) valores generados se usarán en todos los escenarios para evitar sesgos en el resultado. Para cada configuración, habrá cinco (5) conjuntos de resultados generados. En total, habrá alrededor de cien soluciones establecidas para la evaluación que se abstraerán por nivel.

Tabla 4 - Matriz de evaluación seleccionada

Nombre	**Matriz**	
Ubicación de la asignatura	Ubicación de la asignatura	16%
	Pausa para almorzar	14%
	Descanso	14%
	Tiempo de inactividad	14%
	Descanso del docente	14%
	Equilibrio de la carga del docente	14%

Nombre	Matriz	
Ubicación de la asignatura	Ubicación de la asignatura	100%
	Pausa para almorzar	0%
	Descanso	0%
	Tiempo de inactividad	0%
	Descanso del docente	0%
	Equilibrio de la carga del docente	0%
Restricciones ajustadas	Ubicación de la asignatura	20%
	Pausa para almorzar	0%
	Descanso	20%
	Tiempo de inactividad	20%
	Descanso del docente	20%
	Equilibrio de la carga del docente	20%

El criterio usado en todas las tablas para asignar porcentajes es equilibrar en lo posible estos porcentajes, siendo esta la razón para que los valores iniciales sean los mostrados.

Tabla 5 - Limitación de configuraciones aleatorias

Nombre	Límite
Recuento de población mínimo	50-200
Recuento máximo de población	Número mínimo de habitantes-200
Generaciones máximas	50-150
Máximos intentos de creación	1,500-4,500
Disparador de ajuste de frecuencia de mutación	0.00-1.00 donde el valor está dividido por cinco (5)
Máxima aptitud	90-100
Población Elite	0-10%
Tolerancia a la desviación	50-75%

Usando la aleatorización proporcionada anteriormente, aquí están los tres conjuntos de configuraciones de algoritmos genéticos

Tabla 6 - Configuraciones aleatorias

Entrada	Matriz	
1	Número mínimo de población 30 Número máximo de población 60 generaciones máximas 60 Máximos intentos de creación 1800	Disparador del ajuste de la tasa de mutación .09 Máxima aptitud 96% Población Elite 5% Tolerancia a la desviación 55%
2	Número mínimo de población 75 Número máximo de población 150 Generaciones máximas 90 Máximos intentos de creación 2300	Disparador del ajuste de la tasa de mutación .05 Máxima aptitud 90% Población Elite 8% Tolerancia a la desviación 60%
3	Número mínimo de población 75 Número máximo de población 150 Generaciones máximas 90 Máximos intentos de creación 2300	Disparador del ajuste de la tasa de mutación .10 Máxima aptitud 95% Población Elite 2% Tolerancia a la desviación 63%

Hay tres (3) escenarios preparados para la evaluación de la tabla 4. Cada escenario tiene L1, L2 y L3 entidades respectivamente. Estas entidades son docentes, salones, cursos y secciones. Cada modelo a continuación muestra la versión resumida de cada escenario. El resumen incluye el cálculo del mínimo, máximo y un promedio de la aptitud total de cada resultado de todos los ajustes, que vienen de la tabla 4.

Modelo 1: Distribución balanceada

Tabla 7 - Distribución Balanceada

Nombre	Matriz
Ubicación de la asignatura	16%
Pausa para almorzar	14%
Sección de descanso	14%
Nombre	**Matriz**
Sección tiempo de inactividad	14%
Descanso del docente	14%
Equilibrio de carga del docente	14%
Patrón de reunión	14%

Con el modelo actual, ninguna solución de escenario logró alcanzar la máxima solución posible. La distribución de prioridad ha hecho que la inteligencia artificial tenga dificultades para equilibrar todas las restricciones. La siguiente tabla muestra el rendimiento del algoritmo para cada escenario en la evaluación de configuraciones combinadas.

Tabla 8 - Resumen del modelo 1

Escenario	Alta aptitud	Aptitud promedio	Baja aptitud	Tiempo promedio
1	84.45 (Ajustes 2)	80.32	78.61	41:07
2	84.50 (Ajustes 2)	81.01	78.94	53:46
3	84.48 (Ajustes 2)	80.67	79.0	61:34

Modelo 2: Ubicación de la asignatura

Tabla 9 - Matriz de evaluación de la ubicación de la asignatura

Nombre	Matriz
Ubicación de la asignatura	100%
Pausa para almorzar	0%
Sección de descanso	0%
Sección tiempo de inactividad	0%
Descanso del docente	0%
Equilibrio de carga del instructor	0%
Patrón de reunión	0%

Siendo uno de los modelos más fáciles, la inteligencia artificial pudo adaptarse tan fácilmente como no tiene que preocuparse por otras limitaciones al colocar entidades. Ha demostrado su superioridad en la inicialización de la población para proporcionar soluciones aceptables a pesar de que recién está comenzando.

Tabla 10 - Resumen del modelo 2

Escenario	Alta aptitud	Aptitud promedio	Baja aptitud	Tiempo promedio
1	100.00 (Settings 1)	99.69	97.15	18:04
2	100.00 (Settings 1)	99.16	93.46	19:34
3	100.00 (Settings 1)	99.69	97.15	25:27

Modelo 3: Restricciones ajustadas

Tabla 11 - Matriz de evaluación de las Restricciones ajustadas

Nombre	Matriz
Ubicación de la asignatura	20%
Pausa para almorzar	0%
Sección de descanso	20%
Sección tiempo de inactividad	20%
Descanso del docente	20%
Equilibrio de carga del instructor	20%
Patrón de reunión	0%

El tercer modelo, que puede considerarse como el más cercano al programa real en términos de capacidad de software, pudo evaluar soluciones que son mucho mejores que el primer modelo a pesar de tener la misma naturaleza de equilibrio en la priorización de restricciones. Esto ha demostrado que el sistema era capaz de crear soluciones que no violan demasiadas restricciones.

Tabla 12 - Resumen del Modelo 3

Escenario	Alta aptitud	Aptitud promedio	Baja aptitud	Tiempo promedio
1	88.87 (Settings 2)	87.22	85.74	38:15
2	89.35 (Settings 2)	87.66	85.94	44:45
3	89.09 (Settings 2)	87.44	85.84	53:36

Los modelos han demostrado que la inteligencia artificial puede hacer frente al problema. Cada configuración ha producido una respuesta variable, pero ha demostrado que a partir de los datos recopilados, las configuraciones más altas pueden producir resultados de mayor calidad.

CAPÍTULO IV
RESULTADOS Y ANÁLISIS

4.1 Descripción del escenario de las pruebas

Las pruebas se han realizado ejecutando un programa con un algoritmo genético tipo NP-hard desarrollado en Python el cual resultó bastante extenso por la cantidad de restricciones intermedias y duras que tuvo que cumplir el algoritmo, se usó una pc i7 con 32 GB de RAM y una tarjeta NVIDEA GeForce GT-1030, sistema operativo Windows 10 y Python 3.7 con dos IDEs: Spyder Python y PyCharm versión 18-2

4.2 Métodos de validación empleados y resultados obtenidos de la investigación

Para la validación de los resultados mediante el uso de algoritmos genéticos se puede visualizar que el horario generado tiene cumplimiento de las restricciones duras y en un porcentaje muy alto el de las intermedias, es decir presenta una mejor distribución de la carga académica para el docente y el alumno, permitiéndole tener el tiempo suficiente para almorzar y no agotarse con sesiones mayores a las cuatro horas seguidas, teniendo un lapso de al menos una hora de descanso para poder recuperarse y rendir mucho más, que si solo fuera una secuencia corrida con muy poco tiempo de intermedio.

A continuación, se presentarán las capturas de pantalla de las pruebas realizadas mostrando el resultado de la investigación y las vamos a comparar con lo realizado manualmente, para ello tomaremos como referencia el primer semestre del año 2016-2, usando datos reales, y lo comparamos con el resultado del módulo de horarios generados con el algoritmo genético, luego se hará lo mismo para la presentación del 9no ciclo de la especialidad de electrónica donde compararemos ambas propuestas, la manual y la generada por el algoritmo genético.

El tiempo de ejecución promedio según los tres modelos propuestos (Distribución balanceada, Ubicación de la asignatura y Restricciones ajustadas) fue de una media de cuarenta segundos, realizando un total de cien procesos, esto para un ciclo, considerando diez ciclos y tres especialidades podemos aproximar un tiempo de 20 minutos para tener el horario completo, pudiendo ajustarse manualmente los ciclos de alguna especialidad

que puedan “mejorarse” por alguna razón que tenga algún docente en especial, por ejemplo solo dictar en el primer piso por problemas de locomoción –no poder subir escaleras-, como caso extraordinario. Comparado con la elaboración manual que dura hasta tres semanas, ya que se corrige según van apareciendo los problemas que en su mayoría de casos son debidos a cruces de horarios.

En las siguientes figuras se mostrarán para los ciclos 1° y 9°, en la parte superior la elaboración manual y en la parte inferior la elaborada por el algoritmo genético observándose la diferencia.

CICLO :	1°	SECCION:	N	L2 Electrónica	
HORA	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
08:00 A 9:00	BMA03N (P) VALENCIA	BMA01N (P) SALINAS	BFI01N (P) maria LARRY	BMA01N (T) SALINAS	
09:00 A 10:00					EE150N (P) VILCA
10:00 A 11:00	BFI01N (T) CARO	EE150N (T) VILCA	BRC01N (t) evangelista		
11:00 A 12:00			BRC01N (P) EVANGELISTA		
12:00 A 13:00					BFI01N (T) CARO
13:00 A 14:00				BFI01N (L) maria LARRY	
14:00 A 15:00	BMA01N (T) SALINAS		BMT01N (T) RAQUEL MEDINA		
15:00 A 16:00			BMT01N (P) RAQUEL MEDINA	BMA03N (T) VALENCIA	
16:00 A 17:00					
17:00 A 18:00					

Figura 21 Horario hecho manualmente para la especialidad de electrónica del primer ciclo del semestre 2016-2 para ingresantes

CICLO :	1°	SECCION:	N	L2 Electrónica	
HORA	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES
08:00 A 9:00	BMA01N (T) SALINAS Q1-101	BFI01N (T) CARO Q1-101	BMA01N (T) SALINAS Q1-101	BFI01N (L) M. RODRIGUEZ HUALLPA Lab S	BFI01N (P) M. RODRIGUEZ HUALLPA Q1-102
09:00 A 10:00					
10:00 A 11:00	BFI01N (T) CARO Q1-101	EE150N (T) VILCA LIFIEE-Q1-305	BRC01N (T) EVANGELISTA Q1-102	EE150N (P) VILCA LIFIEE-Q1-305	BMA01N (P) SALINAS Q1-101
11:00 A 12:00			BRC01N (P) EVANGELISTA Q1-102		
12:00 A 13:00					
13:00 A 14:00					
14:00 A 15:00	BMA03N (T) VALENCIA Q1-102				
15:00 A 16:00		BMA03N (P) VALENCIA Q1-102	BMT01N (T) R. MEDINA Q1-102		
16:00 A 17:00			BMT01N (P) R. MEDINA Q1-102		
17:00 A 18:00					

Figura 22 Horario generado por el algoritmo genético del primer ciclo del semestre 2016-2

Se puede observar que el horario está desarrollado de lunes a viernes ya que los docentes son a tiempo completo

CICLO :	IX	SECCION:	M			
HORA	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SÁBADO
08:00 A 9:00		EE515M (T) ALFREDO RODRIGUEZ	EE545M (LAB) CAJAHUARINGA			EE525M (T) GARCIA LOPEZ
09:00 A 10:00				EE625M (T) MACHUCA		
10:00 A 11:00	EE545M (LAB) CAJAHUARINGA		EE641M (L) VALENZUELA/ CABEZAS			
11:00 A 12:00					EE445M (P) FIGUEROA	EE525M (P) GARCIA LOPEZ
12:00 A 13:00						
13:00 A 14:00	EE625M (P) MACHUCA			EE435M2 (L) LAZO	EE435M(T) SAL Y ROSAS	CI106M (T) MONTES BAZALAR
14:00 A 15:00			EE435M(P) SAL Y ROSAS			
15:00 A 16:00	EE515M (P) A. RODRIGUEZ	EE435M1 (L) LAZO				
16:00 A 17:00						
17:00 A 18:00				EE445 M(T) FIGUEROA		
18:00 A 19:00	EE526M (T) ALVAREZ MONTOYA	EE535M (T) A. PATTRONI				EE545M (T) ALVAREZ MONTOYA
19:00 A 20:00						
20:00 A 21:00	EE526M (P) ALVAREZ MONTOYA	EE535M (P) A. PATTRONI	CI105M (T) GOMEZ SAENZ		CI105M (T) GOMEZ SAENZ	EE545M (P) ALVREZ MONTOYA
21:00 A 22:00						
	*EE515M (L) NARVAJA 18:00 - 21:00					

Figura 23 Horario hecho manualmente para la sección de electrónica del noveno ciclo del semestre 2016-2

CICLO:	IX	SECCION:	M			
HORA	LUNES	MARTES	MIERCOLES	JUEVES	VIERNES	SÁBADO
08:00 A 8:00						EE525M (T) GARCIA L. Q1-105
08:00 A 10:00	EE435M (T) SAL Y ROSAS Q1-104	EE641M (L) VALENZUELA/ CABEZAS Lab. L2	EE515M (T) RODRIGUEZ Q1-102	EE515M (P) RODRIGUEZ Q1-103		
10:00 A 11:00						
11:00 A 12:00						EE525M (P) GARCIA L. Q1-105
12:00 A 13:00						
13:00 A 14:00				EE445M (P) FIGUEROA Q1-107	EE435M(P) SAL Y ROSAS Q1-202	
14:00 A 15:00	EE625M (T) MACHUCA Q1-105	EE435M2 (L) LAZO Lab. L2	EE445 M(T) FIGUEROA Q1-104			
15:00 A 16:00				EE625M (P) MACHUCA Q1-107	EE435M1 (L) LAZO Lab. L2	CI106M (T) MONTES B. Q1-201
16:00 A 17:00						
17:00 A 18:00						
18:00 A 19:00	EE545M (T) ALVAREZ M. Q1-108	EE535M (T) PATTRONI Q1-208	EE545M (L) CAJAHUARINGA Lab. L3	EE545M (L) CAJAHUARINGA Lab. L3		EE526M (T) ALVAREZ M. Q1-106
19:00 A 20:00					EE515M (L) NARVAJA Lab. L3	
20:00 A 21:00	EE545M (P) ALVREZ M. Q1-108	EE535M (P) PATTRONI Q1-208	CI106M (T) GOMEZ S. Q1-207	CI106M (T) GOMEZ S. Q1-207		EE526M (P) ALVAREZ M. Q1-106
21:00 A 22:00						

Figura 24 Horario generado por el algoritmo genético del noveno ciclo del semestre 2016-2 (en color gris profesores tiempo completo y color ciano a tiempo parcial)

Lo mismo se puede observar en el noveno ciclo de L2, que hay una mejor distribución de las horas teniendo claramente definido el horario para el almuerzo de los docentes y alumnos, así con la mayoría de horarios en la tarde, los de la mañana están así porque

esos profesores expresamente han dado su franja horaria en horas de la mañana y está desarrollado de lunes a sábado por los profesores a tiempo parcial.

4.3 Contrastación de la hipótesis

En el caso de la hipótesis general, se ha verificado que el empleo de una metodología permite mejorar la eficiencia de la asignación de la carga académica de una facultad en una institución universitaria.

Sobre las hipótesis secundarias se ha verificado que:

1. El empleo de la metodología basada en la Inteligencia Artificial permite reducir el tiempo de la elaboración de los horarios, de tres o más semanas a menos de una hora.
2. La aplicación de la metodología evita tener horarios con cruces y saturados.
3. La aplicación de los modelos permiten verificar que se cumplan las restricciones

CONCLUSIONES

1. El modelo desarrollado con Algoritmos Genéticos del tipo NP-Hard ha demostrado ser el más adecuado para la elaboración del horario universitario.

2. Se puede concluir que a partir de los modelos provistos, el sistema es capaz de generar resultados que, a pesar de ser imperfectos, siguen siendo válidos y aceptables dado el número de restricciones que se le imponen.

3. El sistema puede proporcionar soluciones válidas que se pueden usar. Sin embargo, no proporciona una automatización completa. Todavía hay escenarios que requerirían que el operador ajuste algunas entradas para mejorar la solución.

4. Se puede encontrar una buena solución si la aplicación tiene suficiente tiempo y poder de cómputo. La evaluación completa del sistema seguirá siendo difícil de resolver ya que la libertad para la configuración del algoritmo proporciona una gran cantidad de combinaciones.

5. El algoritmo reconfigurable permite a los usuarios usar y experimentar fácilmente con la aplicación hasta que encuentren el ajuste adecuado para su caso particular.

6. Cuando el número de docentes y cursos es mediano (menor a 150 c/u, que es el caso de una facultad promedio) no es necesario usar una base de datos, es suficiente archivos tipo CSV.

RECOMENDACIONES

1. Esta solución no significa que deba usarse para todos los casos, sobre todo si las restricciones aumentan o son muy complejas, lo cual implicará modificación del algoritmo.
2. Sin embargo, el uso de esta solución es conveniente, ya que es más fácil de usar que otras soluciones.
3. Se puede usar este enfoque para resolver problemas de optimización o incluso crear versiones más eficientes. Este trabajo de investigación incluye datos útiles para comprender el proceso y que serán de ayuda a futuras investigaciones

GLOSARIO

Algoritmo genético: una meta heurística inspirada en el proceso de selección natural que pertenece a la clase más grande de algoritmos evolutivos (AE). Los algoritmos genéticos se utilizan comúnmente para generar soluciones de alta calidad para la optimización y los problemas de búsqueda al depender de operadores de inspiración biológica, como la mutación, el cruce y la selección.

Aptitud (*fitness*): es la que permite valorar la aptitud de los individuos y debe tomar siempre valores positivos.

Aprendizaje automático: un campo de la informática que da a las computadoras la capacidad de aprender sin estar programado explícitamente.

Aprendizaje profundo: forma parte de una familia más amplia de métodos de aprendizaje automático basados en representaciones de datos de aprendizaje, a diferencia de los algoritmos específicos de tareas. El aprendizaje puede ser supervisado, parcialmente supervisado o no supervisado.

Cromosoma: conjunto de parámetros que define una solución propuesta que el algoritmo genético intenta resolver.

Crossover: un operador genético utilizado para variar la programación de un cromosoma o cromosomas de una generación a otra. Es análogo a la reproducción y al cruce biológico, sobre el cual se basan los algoritmos genéticos.

Fenotipo: la representación del mundo real de la solución.

Gen: La unidad básica de un cromosoma. Representa como parte de una solución.

Genotipo: la representación genética de la solución, en una solución básica, podría representarse como una cadena de caracteres binarios.

NP: *Nondeterministic Polynomial time*, en español tiempo polinomial no determinista. Es el conjunto de problemas que pueden ser resueltos en tiempo polinómico por una máquina de Turing no determinista.

NP-Complete: Una clase de complejidad donde las soluciones se pueden verificar rápidamente y si hay un algoritmo rápido para resolver este tipo de problema, también se puede usar para resolver otros problemas de NP.

NP-Hard: Conjunto de los problemas de decisión que contiene los problemas H tales que todo problema L en NP puede ser transformado polinomialmente en H.

Mutación: un operador genético utilizado para mantener la diversidad genética de una generación de cromosomas de algoritmos genéticos a la siguiente generación. Es análogo a la mutación biológica.

Población: un conjunto de soluciones propuestas.

Restricciones duras: (*Hard Constraints*) un conjunto de reglas que produciría una solución no válida si se rompe.

Restricciones medias: (*Medium Constraints*) conjunto de reglas que se pueden romper con un efecto sobre la validez del resultado. Sin embargo, esto solo se puede romper si el escenario es lógicamente inválido o imposible.

Restricciones suaves (*Soft Constraints)*: conjunto de reglas que se pueden romper sin afectar la validez del resultado.

Selección - Etapa de AG que selecciona el cromosoma para futuras combinaciones.

BIBLIOGRAFÍA

[1] **Álvarez P.** (2014) Procedimiento para la elaboración de los horarios de clase y los calendarios de exámenes en la Facultad de Ciencias de la UEx, [en línea], Universidad de Extremadura, Colombia, consultado el 10/05/2018, URL disponible en: https://www.unex.es/conoce-la-uex/centros/ciencias/archivos/ficheros/sgic/manual-de-calidad-procesos-y-procedimientos/PR_CL004_PEHYC_aprobadoJF040714faltadiagrama.pdf

[2] **Ansari A., Bojewar S.** (2014) Genetic Algorithm to Generate the Automatic Time-Table – An Over View, [en línea], International Journal on Recent and Innovation Trends in Computing and Communication, Volume: 2 Issue: 11, URL disponible en http://www.ijritcc.org

[3] Asmuni H. B. (2008) Fuzzy Methodologies for Automated University Timetabling Solution Construction and Evaluation, [en línea], PhD thesis, University of Nottingham, Inglaterra, consultado el 18/01/2018, URL disponible en http://delta.cs.cinvestav.mx/~ccoello/EMOO/thesis-asmuni.pdf.gz

[4] Blakesley J., Murray K., Wolf F., Murray D. (1998) Space Management and Academic Scheduling. [en línea], consultado el 14/01/2018, URL disponible en: https://www.unitime.org/papers/patat98.pdf.

[5] Braak M. T., (2012) A Hyperheuristic for generating Timetables in the XHSTT Format, [en línea], MSc Thesis, University of Twente, consultado el 14/01/2018, URL disponible en: https://essay.utwente.nl/62055/1/MSc_M_ter_Braak.pdf.

[6] Burke E., Elliman D., Weare R. (1995) A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems, [en línea], consultado el 24/01/2018, URL disponible en: https://pdfs.semanticscholar.org/c566/9a6e00fa1fa66ac8da870f3a38ae13819be2.pdf

[7] Chaudhuri A., De K., (2010) Fuzzy Genetic Heuristic for University Course Timetable Problem, Int. J. Advance. Soft Comput. Appl., Vol. 2, No. 1, [en línea], consultado el

23/01/2018, URL disponible en: https://i-csrs.org/Volumes/ijasca/vol.2/vol.2.1.7.March.10.pdf.

[8] Chia Lih B., San Nah S., Alamshah Bolhassan N. (2015) A study on heuristic timetabling method for faculty course timetable problem, [en línea], consultado el 19/02/2018, URL disponible en: http://ieeexplore.ieee.org/abstract/document/7349832/

[9] Colorni A., Dorigo M. Maniezzo V. (1997), A genetic algorithm to solve the timetable problem, Computational Optimization and Applications Journal, [en línea], consultado 3/02/2018, URL disponible en https://pdfs.semanticscholar.org/da83/bc782bffff20d5a8b829808c62727556c597.pdf.

[10] Datta D., Deb K., Fonseca C. (2005) Solving Class Timetabling Problem of IIT Kanpur using Multi-Objective Evolutionary Algorithm, KanGAL Report Number 200600, [en línea], consultado el 14/01/2018, URL disponible en https://pdfs.semanticscholar.org/335f/0790f16214cb29a61e3583553be746129604.pdf

[11] Elmohamed, M., Coddington P., Fox G. (1997) A Comparison of Annealing Techniques for Academic Course Scheduling [en línea], consultado el 20/01/2018 URL disponible en: https://www.ics.uci.edu/~dechter/courses/ics-175a/spring-2011/papers/paper1.pdf

[12] El-Sakka T. (2015) University Course Timetable using Constraint Satisfaction and Optimization. International Journal of Computing Academic Research Volume 4, Number 3, pp.83-9, [en línea], consultado el 16/02/2018, URL disponible en: https://www.academia.edu/22839216/University_Course_Timetable_using_Constraint_Satisfaction_and_Optimization.

[13] Elsaka T. (2017) Autonomous generation of conflict-free examination timetable using constraint satisfaction modelling, [en línea], consultado el 22/02/2018, URL disponible en: https://ieeexplore.ieee.org/document/8090236/.

[14] Ferdiana R., Ridwan N., Hidayat N. (2017) A pragmatic algorithm approach to develop course timetable web application based on cloud technology, 2017 7th International Annual Engineering Seminar, [en línea], consultado el 16/08/2018, URL disponible en: https://ieeexplore.ieee.org/document/8068540.

[15] Ilyas R., Iqbal Z. (2015) Study of hybrid approaches used for university course timetable problem (UCTP), [en línea], consultado el 16/08/2018, URL disponible en: https://ieeexplore.ieee.org/document/7334198/

[16] Isabel S. (2018) ¿Por qué la asignación de horarios es tan compleja? https://www.u-planner.com/es/blog/planificacion-academica-por-que-la-asignacion-de-horarios-es-tan-compleja, [en línea], consultado 11/10/ 2018

[17] Luká˘c M. (2013) Course timetabling at Masaryk University in the UniTime system Master Thesis, [en línea], Masaryk University - Faculty of Informatics, consultado el 15/02/2018, URL disponible en: https://is.muni.cz/th/y0gxo/Master_Thesis.pdf.

[18] Marte, M. (2002) Models and Algorithms for School Timetabling – A Constraint-Programming Approach, MSc Thesis, [en línea], Universitat München - Fakultat f ¨ ur Mathematik, Informatik und Statistik, consultado el 19/02/2018, URL disponible en: https://www.pms.ifi.lmu.de/publikationen/dissertationen/PMS-DISS-2003-1/PMS-DISS-2003-1.pdf.

[19] Mittal M., Doshi H., Sunasra M., Nagpure R. (2015) Automatic Timetable Generation using Genetic Algorithm International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 2, [en línea], consultado el 18/02/2018, URL disponible en https://www.researchgate.net/publication/276771023_Automatic_Timetable_Generation_using_Genetic_Algorithm.

[20] Nanda A, Pai M., Gole A. (2012) An Algorithm to Automatically Generate Schedule for School Lectures Using a Heuristic Approach, International Journal of Machine Learning and Computing, Vol. 2, No. 4 [en línea], consultado el 18/02/2018, URL disponible en http://www.ijmlc.org/papers/174-L031.pdf.

[21] Narang B., Gupta A., Bansal R., (2013) Use of Active Rules and Genetic Algorithm to Generate the Automatic TimeTable, Special Issue: Proceedings of 2nd International Conference on Emerging Trends in Engineering and Management, , [en línea], consultado el 12/02/2018, URL disponible en:. http://www.journals.rgsociety.org/index.php/ijse/article/view/423.

[22] Pimmer M., Raidl G. (2011) A Timeslot-Filling Heuristic Approach to Construct High-School Timetables, MIC 2011: The IX Metaheuristics International Conference, [en línea],

consultado el 20/02/2018, URL disponible en:. https://www.ac.tuwien.ac.at/files/pub/pimmer-12.pdf

[23] Pupeikeine L., Strukov D., Bivanis V. (2008) Optimization Algorithms in School Scheduling Programs - Study, Analysis and Results, Informatics in Education, 2009, Vol. 8, No. 1, 69–84, [en línea], consultado el 22/02/2018, URL disponible en:. https://www.semanticscholar.org/paper/Optimization-Algorithms-in-School-Scheduling-Study%2C-Pupeikiene-Strukov/a2f52953be4491c903e1efbf54407170d4213139.

[24] Ribic S., Turcinihodzic R., Tumbul A., (2017) Modeling double-shift school timetables, [en línea], consultado el 8/07/2018, URL disponible en: https://ieeexplore.ieee.org/iel7/8125633/8171593/08171628.pdf

[25] Rudova H. & Müller T. (2011) Rapid Development of University Course Timetables, [en línea], consultado el 17/02/2018, URL disponible en https://www.unitime.org/papers/mista11.pdf.

[26] Sandhu K. (2001) Automating Class Schedule Generation in the Context of a University Timetabling Information System, [en línea], PhD Thesis, consultado el 5/3/2018, URL disponible en http://delta.cs.cinvestav.mx/~ccoello/EMOO/thesis-asmuni.pdf.gz.

[27]Shatnawi A., Fraiwan M., Al-Qahtani H. (2017) Exam scheduling - A case study 2017 Ninth International Conference on Advanced Computational Intelligence, [en línea], consultado el 15/02/2018, URL disponible en: https://ieeexplore.ieee.org/document/7974498

[28] Valdivia N., Moreno A., Carranza D. (2010) Sistema de Horarios SIHO, [en línea], Proyecto Profesional, Universidad Peruana de Ciencias Aplicadas (UPC), Lima – Perú, consultado el 04/07/2018, URL disponible en http://hdl.handle.net/10757/273597

[29] Wall B., (1996) A Genetic Algorithm for Resource-Constrained Scheduling, [en línea] MSc Thesis. Massachusetts Institute of Technology, consultado el 04/02/2018, URL disponible en: https://www.researchgate.net/publication/37605064_A_genetic_algorithm_for_resource-constrained_scheduling.

[30] Zhang Y., Meng Z., Ralescu A. (2017) Dynamic timetable scheduling with reverse-flow technique in fuzzy environment, [en línea], consultado el 15/06/2018, URL disponible en: https://ieeexplore.ieee.org/document/8023312

ANEXOS

A 1. Código en Python

main.py

```
from PyQt5 import QtWidgets
from components import Database as db
from containers import Main
import sys

# Entry point for application
if __name__ == '__main__':
    if not db.checkSetup():
        db.setup()
    app = QtWidgets.QApplication(sys.argv)
    parent = QtWidgets.QMainWindow()
    Main.MainWindow(parent)
    parent.show()
    sys.exit(app.exec_())
```

database.py

```
import sqlite3


def checkSetup():
    conn = sqlite3.connect('gas.db')
    cursor = conn.cursor()
    cursor.execute("SELECT name FROM sqlite_master WHERE type='table' AND name='
    result = cursor.fetchone()
    conn.close()
    if result is None:
        return False
    return True


def setup():
    conn = sqlite3.connect('gas.db')
    cursor = conn.cursor()
    create_instructors_table = """
        CREATE TABLE IF NOT EXISTS instructors(
          id INTEGER PRIMARY KEY,
          name TEXT NOT NULL,
          hours INTEGER NOT NULL,
          schedule TEXT NOT NULL,
          active BOOLEAN NOT NULL DEFAULT 1 CHECK(
            active IN(0,1)
          )
        );
    """
```

```
create_rooms_table = """
    CREATE TABLE IF NOT EXISTS rooms (
      id INTEGER PRIMARY KEY,
      name TEXT NOT NULL,
      type TEXT NOT NULL,
      schedule TEXT NOT NULL,
      active BOOLEAN NOT NULL DEFAULT 1 CHECK (
        active IN (0, 1)
      )
    );
"""
create_subjects_table = """
    CREATE TABLE IF NOT EXISTS subjects (
      id INTEGER PRIMARY KEY,
      name TEXT NOT NULL,
      hours REAL NOT NULL,
      code TEXT NOT NULL,
      description TEXT NOT NULL,
      instructors TEXT NOT NULL,
      divisible BOOLEAN NOT NULL DEFAULT 1 CHECK (
        divisible IN (0, 1)
      ),
      type TEXT NOT NULL
    );
"""
create_sections_table = """
    CREATE TABLE IF NOT EXISTS sections
      id INTEGER PRIMARY KEY,
      name TEXT NOT NULL,
      schedule TEXT NOT NULL,
      subjects TEXT NOT NULL,
      active BOOLEAN NOT NULL DEFAULT 1 CHECK (
        active IN (0, 1)
      ),
      stay BOOLEAN NOT NULL DEFAULT 0 CHECK (
        active IN (0, 1)

    );
"""
create_sharing_table = """
    CREATE TABLE IF NOT EXISTS sharings
      id INTEGER PRIMARY KEY,
      subjectId INTEGER NOT NULL,
      sections TEXT NOT NULL,
      final BOOLEAN NOT NULL DEFAULT 0 CHECK (
        final IN (0, 1)

    );
"""
create_results_table = """
    CREATE TABLE IF NOT EXISTS results (
      id INTEGER PRIMARY KEY,
      content BLOB NOT NULL,
      timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
    );
"""
```

```
    cursor.execute(create_instructors_table)
    cursor.execute(create_rooms_table)
    cursor.execute(create_subjects_table)
    cursor.execute(create_sections_table)
    cursor.execute(create_sharing_table)
    cursor.execute(create_results_table)
    conn.commit()
    conn.close()


def getConnection():
    return sqlite3.connect('gas.db')
```

GeneticAlgorithm.py

```
from PyQt5 import QtCore
from components import Settings
from operator import itemgetter
from collections import Counter
import copy
import itertools
import numpy as np


class GeneticAlgorithm(QtCore.QThread):
    # Current phase of the algorithm
    statusSignal = QtCore.pyqtSignal(str)
    # Genetic algorithm variable details
    detailsSignal = QtCore.pyqtSignal(list) #
    Running process type
    operationSignal = QtCore.pyqtSignal(int) #
    List of chromosomes for preview
    dataSignal = QtCore.pyqtSignal(list)

    def __init__(self,data):
        self.averageFitness = 0
        self.pastAverageFitness = 0
        self.running = True
        self.chromosomes = []
        self.data =
                {
            'rooms':[],
            'instructors':[],
            'sections':[],
            'sharings':[],
            'subjects':[]
        }
        self.stayInRoomAssignments = {}
        self.tournamentSize = .04
        self.elitePercent = .05
        self.mutationRate = .10
        self.lowVariety = 55
        self.highestFitness = 0
        self.lowestFitness = 100
        self.elites = []
        self.matingPool = []
        self.offsprings = []
        self.tempChromosome = None
        self.tempSections = None
        self.data = data
        self.settings = Settings.getSettings()
        self.stopWhenMaxFitnessAt = self.settings['maximum_fitness']
        super().__init__()
```

```
def initialization(self):
    # Generate population based on minimum population
    self.generateChromosome(self.settings['minimum_population'])

def generateChromosome(self, quantity):
    for i in range(quantity):
        self.statusSignal.emit('Creating #{} of {} Chromosomes'.format(i, quantity))
        self.tempChromosome = Chromosome(self.data)
        # {id: [[subjectIds](, stay|roomId = False)]}
        self.tempSections = sections = {key: [value[2], value[3]] for (key, value) in
                                        copy.deepcopy(self.data['sections']).items()}
        # {id: [subjectId, [sections]]}
        self.tempSharings = sharings = copy.deepcopy(self.data['sharings'])
        # [roomIds]
        self.rooms = rooms = list(self.data['rooms'].keys())
        # Distributed Room selection for staying sections
        if not len(self.stayInRoomAssignments):
            selectedRooms = []
            for section in sections:
                if sections[section][1]:
                    room = False
                    attempts = 0
                    while not room:
                        attempts += 1
                        candidate = np.random.choice(rooms)
                        if attempts == self.settings['generation_tolerance']:
                            room = candidate
                        if self.data['rooms'][candidate][1] == 'lec':
                            if candidate not in selectedRooms:
                                selectedRooms.append(copy.deepcopy(candidate))
                                room = candidate
                    sections[section][1] = room
                    self.stayInRoomAssignments[section] = room
        else:
            for section, room in self.stayInRoomAssignments.items():
                sections[section][1] = room
        # Remove subjects from sections that are already in
        sharing for sharing in sharings.values():
            for section in sharing[1]:
                sections[section][0].remove(sharing[0])
        self.generateSubjectPlacementsForSharings(sharings)
        self.generateSubjectPlacementsForSections(sections
        ) self.chromosomes. append(self.tempChromosome)

def generateSubjectPlacementsForSharings(self, sharings):
    sharingOrder = list(sharings.keys())
    np.random.shuffle(sharingOrder)
    for sharing in sharingOrder:
        result = self.generateSubjectPlacement(sharings[sharing][1], sharings[sharing][0]
        if not result:
            self.tempChromosome.data['unplaced']['sharings'].append(sharing)

# {id: [[subjectIds](, stay|roomId = False)]}
def generateSubjectPlacementsForSections(self, sections):
    # Maximum length of section subjects
    maxSubjects = max(len(subjects[0]) for subjects in
    sections.values()) # Put one random section subject per turn
    for i in range(maxSubjects):
        for section in sections:
```

```
            subjectList = sections[section]
            [0] if not len(subjectList):
                continue
            subjectToPlace = np.random.randint(0, len(subjectList))
            result = self.generateSubjectPlacement([section], subjectList[subjectToPlace]
            if not result:
                self.tempChromosome.data['unplaced']['sections'][section].append
            (subjectL sections[section][0].pop(subjectToPlace)

# Section = [id], Subject = int (id)
def generateSubjectPlacement(self, section, subject, sharing=False):
    generating = True
    generationAttempt = 0
    error = None

    stayInRoom = False if section[0] not in self.stayInRoomAssignments.keys() else self.s
        section[0]]
    subjectDetails = self.data['subjects'][subject]

    room = stayInRoom if stayInRoom else None
    # [[day/s], startingTimeSlot, length]
    timeDetails = []
    instructor = None

    while generating:
        # Control generation to avoid impossible combinations
        generationAttempt += 1
        if generationAttempt > self.settings['generation_tolerance']:
            generating = False
            return False
        # Allow random meeting patterns if generation is taking long
        forceRandomMeeting = True if generationAttempt > self.settings['generation_tolera
        # First time generation
        if not error:
            if not stayInRoom or (stayInRoom and subjectDetails[6] == 'lab'):
                room = self.selectRoom(subject)
            if len(subjectDetails[4]) > 1:
                instructor = self.selectInstructor(subject)
            elif len(subjectDetails[4]):
                instructor = subjectDetails[4][0]
            else:
                instructor = False
        timeDetails = self.selectTimeDetails(subject, forceRandomMeeting)
        else:
            # Randomly select if choosing new entry or replacing subject time
            detailsif error == 1 or error == 2:
                if np.random.randint(0, 2):
                        error = 3
                 elif error == 1:
                    if not stayInRoom or (stayInRoom and subjectDetails[6] == 'lab'):
                        room = self.selectRoom(subject)
                    else:
                        error = 3
                else:
                    if len(subjectDetails[4]) > 1:
                        instructor = self.selectInstructor(subject)
                    else:
                        error = 3
            # Select subject time details
            elif error == 3:
```

```
                elif error == 1:
                    if not stayInRoom or (stayInRoom and subjectDetails[6] == 'lab'):
                        room = self.selectRoom(subject)
                    else:
                        error = 3
                else:
                    if len(subjectDetails[4]) > 1:
                        instructor = self.selectI
                    nstructor(subject)else:
                        error = 3
            # Select subject time details
            elif error == 3:
                timeDetails = self.selectTimeDetails(subject, forceRandomMeeting)

        # [roomId, [sectionId], subjectId, instructorID, [day / s], startingTS, length(,
        scheduleToInsert = [room, section, subject, instructor, *timeDetails]
        if sharing:
            scheduleToInsert.append(sharing)
        error = self.tempChromosome.insertSchedule(scheduleToInsert)
        if error is False:
            return True

def selectRoom(self, subject):
    room = None
    while not room:
        candidate = np.random.choice(self.rooms)
        if self.data['subjects'][subject][6] == self.data['rooms'][candidate]
        [1]:room = candidate
    return room

def selectInstructor(self, subject):
    instructor = None
    subjectInstructors = self.data['subjects'][subject][4]
    while not instructor and len(subjectInstructors):
        instructor = np.random.choice(subjectInstructors)
    return instructor

def selectTimeDetails(self, subject, forceRandomMeeting):
    meetingPatterns = [[0, 2, 4], [1, 3]]
    days = [0, 1, 2, 3, 4, 5]
    np.random.shuffle(days)
    hours = self.data['subjects'][subject][1]
    # Check if hours can be splitted with minimum session of 1 hour or 2 timeslot
    if hours > 1.5 and ((hours / 3) % .5 == 0 or (hours / 2) % .5 == 0) and self.data['sul
        # If hours is divisible by two and three
        if (hours / 3) % .5 == 0 and (hours / 2) % .5 == 0:
            meetingPattern =
            np.random.choice(meetingPatterns) if
            len(meetingPattern)==days[0:3] if forceRandomMeeting else
                meetingPattern hours = hours / 3
            else:
                meetingPattern = days[0:2] if forceRandomMeeting else meetingPattern
                hours = hours / 2
        elif (hours / 3) % .5 == 0:
            meetingPattern = days[0:3] if forceRandomMeeting else
            meetingPatterns[0] hours = hours / 3
        else:
            meetingPattern = days[0:2] if forceRandomMeeting else meetingPatterns[1]
            hours = hours / 2
     # Select random day slot
```

```
    else:
        meetingPattern = [np.random.randint(0, 6)]
    # To convert hours into timetable timeslots
    hours = hours / .5
    startingTimeslot = False
    # Starting slot selection
    startingTime = self.settings['starting_time']
    endingTime = self.settings['ending_time']
    while not startingTimeslot:
        candidate = np.random.randint(0, endingTime - startingTime + 1)
        # Validate if subject will not overpass operation time
        if (candidate + hours) < endingTime - startingTime:
                startingTimeslot = candidate
    return [meetingPattern, startingTimeslot, int(hours)]

def evaluate(self):
    totalChromosomeFitness = 0
    self.pastAverageFitness =
    copy.deepcopy(self.averageFitness) self.lowestFitness = 100
    self.highestFitness = 0
    for index, chromosome in enumerate(self.chromosomes):
        self.statusSignal.emit('Evaluating #{} of {} Chromosomes'.format(index + 1, len(s
        chromosome.fitness = self.evaluateAll(chromosome)
        totalChromosomeFitness += chromosome.fitness
        self.averageFitness = totalChromosomeFitness / len(self.chromosomes)
        self.highestFitness = chromosome.fitness if chromosome.fitness > self.highestFitn
        self.lowestFitness = chromosome.fitness if chromosome.fitness < self.lowestFitnes
    chromosomeFitness = sorted(enumerate(map(lambda chromosome: chromosome.fitness, self.
                               key=itemgetter(1))
    # Emit top five chromosomes
    self.dataSignal.emit(
        list(map(lambda chromosome: [self.chromosomes[chromosome[0]], chromosome[1]],
        chr
# Evaluation weight depends on
settings def evaluateAll(self,
chromosome):
    subjectPlacement = self.evaluateSubjectPlacements(chromosome)
    lunchBreak = self.evaluateLunchBreak(chromosome) if self.settings['lunchbreak'] else
    studentRest = self.evaluateStudentRest(chromosome)
    instructorRest = self.evaluateInstructorRest(chromosome)
    idleTime = self.evaluateStudentIdleTime(chromosome)
    meetingPattern = self.evaluateMeetingPattern(chromosome)
    instructorLoad = self.evaluateInstructorLoad(chromosome)
    chromosome.fitnessDetails = copy.deepcopy([subjectPlacement, lunchBreak, studentRest,
                                meetingPattern, instructorLoad])
    matrix = self.settings['evaluation_matrix']
    return round(
        (subjectPlacement * matrix['subject_placement'] / 100) +
        (lunchBreak * matrix['lunch_break'] / 100) +
        (studentRest * matrix['student_rest'] / 100) +
        (instructorRest * matrix['instructor_rest'] / 100) +
        (idleTime * matrix['idle_time'] / 100) +
        (meetingPattern * matrix['meeting_pattern'] / 100) +
        (instructorLoad * matrix['instructor_load'] / 100),
        2
    )

# = ((subjects - unplacedSubjects) / subjects) *
100 def evaluateSubjectPlacements(self, chromosome):
    sections = copy.deepcopy({key: value[2] for key, value in self.data['sections'].items
    sharings = self.data['sharings']
```