Universidad Nacional Mayor de San Marcos
Facultad de Ingeniería de Sistemas e Informática
Escuela Profesional de Ingeniería de Software

Optimización de la generación de Horarios Académicos mediante la utilización de algoritmos metaheuristos mixtos en una escuela de la UNMSM

Tesis
Para optar el Bachiller Profesional de Ingeniero de Software

Autor: Nicolas Alonso ROJAS GALA
Asesor: José Alfredo HERRERA QUISPE

Lima, Perú

2025

Capítulo 1

Resumen

La Superintendencia Nacional de Educación Superior Universitaria (SUNEDU), entidad encargada de supervisar y licenciar a las universidades del Perú, informó en su III Reporte Bianual de 2022 que 94 universidades contaban con licencia de funcionamiento y se encontraban distribuidas en todas las regiones del país. Estas instituciones ofrecían estudios superiores a aproximadamente el 60 % de la población en edad universitaria, lo que representa cerca de 1.4 millones de estudiantes [6]

En la Facultad de Ingeniería de Sistemas e Informática (a partir de ahora, FISI) de la Universidad Nacional Mayor de San Marcos (a partir de ahora, UNMSM), se registraron alrededor de 1,800 estudiantes matriculados al cierre del semestre 2023-I, según el último ranking académico disponible [8]. Cada año académico se compone de dos periodos regulares y un ciclo vacacional conocido como çiclo verano". Para cada uno de estos ciclos es necesario llevar a cabo un proceso de matrícula, precedido por la elaboración de horarios académicos, responsabilidad de los directores de escuela. Este proceso de planificación puede tardar entre dos y tres días, y suele implicar ajustes de último minuto debido a la disponibilidad de docentes y la carga académica asignada.

Diversos estudios y experiencias previas han demostrado que este tiempo de planificación puede reducirse drásticamente —incluso a 30 minutos— mediante el uso de herramientas automatizadas. Esta mejora permitiría anticipar con mayor margen los posibles problemas relacionados con la disponibilidad horaria de docentes, la distribución equitativa de carga académica y los conflictos entre horarios, beneficiando directamente a estudiantes y personal administrativo.

Ante esta problemática, se propone el uso de algoritmos metaheurísticos, capaces de abordar problemas complejos de optimización pertenecientes a la clase NP. Basándonos en estudios previos, esta investigación adopta una estrategia híbrida que combina algoritmos genéticos con búsqueda tabú, con el fin de diseñar un sistema capaz de generar horarios académicos de forma automática, respetando las restricciones establecidas y reduciendo significativamente el tiempo de planificación en comparación con los métodos manuales actualmente utilizados.

Palabras clave: horarios académicos, algoritmos genéticos, búsqueda tabú, resolución de problemas de horario, optimización, metaheurísticas.

Capítulo 2

Abstract

The National Superintendence of Higher University Education (SUNEDU), the entity responsible for supervising and licensing universities in Peru, reported in its Third Biannual Report of 2022 that 94 universities held operating licenses and were distributed across all regions of the country. These institutions served approximately 60

At the Faculty of Systems and Informatics Engineering (from now, FISI) of the National University of San Marcos (from now, UNMSM), approximately 1,800 students were enrolled at the end of the 2023-I semester, according to the latest available academic ranking [8]. Each academic year consists of two regular semesters and a vacation period known as the "summer cycle." For each cycle, a course scheduling process must be completed prior to student registration. This process, managed by school directors, typically takes two to three days and often requires last-minute adjustments due to faculty availability and workload distribution.

Several studies and previous experiences have shown that this planning time can be drastically reduced—even to just 30 minutes— through the use of automated tools. Such improvement would enable earlier detection and resolution of potential issues related to faculty availability, fair distribution of teaching loads, and scheduling conflicts, directly benefiting both students and administrative staff.

To address this challenge, this research proposes the use of metaheuristic algorithms, which are capable of solving complex NP-class optimization problems. Based on previous studies, this work adopts a hybrid strategy combining genetic algorithms and tabu search, aiming to develop a system that can automatically generate academic schedules while respecting institutional constraints and significantly reducing the planning time compared to current manual methods.

Keywords: academic scheduling, genetic algorithms, tabu search, time tabling solving, optimization, metaheuristics.

Capítulo 3

Introducción

3.1. Antecedentes del problema

La Superintendencia Nacional de Educación Superior Universitaria (SUNEDU), como entidad encargada de supervisar y licenciar a las universidades del Perú, reportó en su III Informe Bianual del año 2022 que un total de 94 universidades contaban con licencia de funcionamiento, distribuidas en todas las regiones del país. Estas instituciones ofrecían educación superior a aproximadamente el 60 % de la población en edad universitaria, lo que representa alrededor de 1.4 millones de estudiantes [6].

En la Facultad de Ingeniería de Sistemas e Informática (FISI) de la Universidad Nacional Mayor de San Marcos (UNMSM), el proceso de matrícula involucra alrededor de 1,800 estudiantes distribuidos en dos periodos académicos regulares y un ciclo intensivo de verano. Cada uno de estos periodos requiere un proceso de planificación y asignación de horarios académicos, actividad que actualmente se realiza de forma manual por los directores de escuela. Esta tarea puede tomar entre tres y cinco días, y con frecuencia presenta errores como cruces entre cursos, asignaciones fuera del horario disponible para los docentes y sobrecarga en la gestión administrativa.

Entre los principales problemas identificados en el proceso actual de elaboración de horarios se encuentran:

- **Sobrecarga administrativa:** Debido a la acumulación de tareas en el área de gestión académica y al tiempo prolongado que toma la planificación.
- **Conflictos en la asignación:** Tales como cruces de horarios entre cursos del mismo ciclo o cursos asignados a un mismo docente simultáneamente.
- **Identificación tardía de conflictos de horarios**: Esto puede desembocar en cambios repentinos de horario o aumentar el volumen de rectificaciones de matrícula debido al cambio de horario de la asignatura

Esta problemática no se limita únicamente a la Facultad de Ingeniería de Sistemas e Informática (FISI), sino que también afecta a otras facultades dentro de la misma Universidad Nacional Mayor de San Marcos (UNMSM), así como a diversas instituciones de educación superior a nivel nacional e internacional. Aunque cada universidad posee características organizativas y académicas particulares —como estructuras curriculares, número de estudiantes, capacidad de infraestructura y disponibilidad docente—, muchas comparten restricciones y necesidades comunes al momento de planificar sus horarios académicos.

Estas coincidencias permiten proponer soluciones escalables, capaces de adaptarse a diferentes contextos institucionales sin perder efectividad. Una herramienta automatizada, diseñada

con flexibilidad y capacidad de configuración, no solo mejoraría la eficiencia en la generación de horarios, sino que también podría reducir errores, aliviar la carga operativa de los equipos administrativos y mejorar la experiencia general de estudiantes y docentes.

Implementar un sistema de este tipo en la FISI serviría como caso piloto, con potencial de replicarse en otras facultades de la UNMSM, universidades públicas del país e incluso instituciones extranjeras con problemáticas similares. Esto posicionaría la solución no solo como una propuesta técnica, sino como un aporte estratégico a la modernización de la gestión académica universitaria.

3.2. Formulación del problema

El proceso de generación de horarios académicos en la Facultad de Ingeniería de Sistemas e Informática (FISI) de la Universidad Nacional Mayor de San Marcos presenta múltiples desafíos que impactan negativamente en la eficiencia operativa y la calidad del servicio educativo. Este proceso, realizado manualmente por los directores de escuela, suele extenderse entre tres y cinco días y con frecuencia genera errores, sobrecarga administrativa y malestar entre los estudiantes.

En este contexto, surge la siguiente pregunta de investigación:

> ¿Es posible optimizar el proceso de generación de horarios académicos en la Facultad de Ingeniería de Sistemas e Informática de la Universidad Nacional Mayor de San Marcos mediante el uso de algoritmos metaheurísticos?

3.2.1. Problemas específicos

- ¿Puede una solución automatizada reducir la sobrecarga administrativa durante la planificación de horarios?
- ¿Es posible disminuir los conflictos y cambios de último minuto que afectan a los estudiantes?
- ¿Se puede mejorar la distribución horaria de los cursos, evitando solapamientos y considerando las preferencias docentes?

3.3. Justificación

La planificación de horarios académicos es una función estratégica dentro de la gestión universitaria, ya que afecta directamente la calidad del proceso formativo, el aprovechamiento de los recursos institucionales y la experiencia de estudiantes y docentes. Sin embargo, en muchas instituciones, este proceso se sigue ejecutando de forma manual, con herramientas básicas y sin apoyo tecnológico especializado.

En el caso particular de la FISI, este enfoque manual conlleva múltiples desventajas: tiempos prolongados de elaboración, errores frecuentes, ajustes de último momento y una distribución poco equitativa de la carga académica. Estos problemas generan insatisfacción entre los estudiantes, sobrecargan al personal administrativo y dificultan la implementación de mejoras curriculares o ajustes logísticos.

Desde el punto de vista computacional, la generación de horarios académicos es un problema de optimización combinatoria de alta complejidad (NP-difícil), caracterizado por una gran cantidad de restricciones y un espacio de búsqueda inmenso. Por ello, el uso de algoritmos metaheurísticos —como los algoritmos genéticos y la búsqueda tabú— resulta adecuado, ya que

permiten encontrar soluciones viables en tiempos razonables, incluso en problemas altamente restringidos.

Además, esta problemática no se limita a la FISI. Otras facultades de la UNMSM y universidades a nivel nacional e internacional enfrentan desafíos similares. Si bien existen particularidades en cada institución, las restricciones compartidas permiten desarrollar soluciones escalables y adaptables a distintos contextos. Un sistema automatizado y flexible tendría un impacto significativo al profesionalizar y modernizar la gestión académica.

3.4. Objetivos

3.4.1. Objetivo general

Explorar y determinar una solución algoritmica para la generación de horarios académicos en la escuela de ingeniería de software de la UNMSM, utilizando algoritmos metaheurísticos, con el fin de optimizar la asignación de cursos, docentes y horarios, minimizando conflictos y mejorando la eficiencia del proceso de planificación académica.

3.4.2. Objetivos específicos

- Disminuir la sobrecarga administrativa en el proceso de planificación horaria
- Identificar oportunamente y diminuir la cantidad de conflictos y cambios de último minuto que afecten a los estudiantes
- Distribuir equitativamente las clases en turnos preferidos según el ciclo académico (mañana para ciclos iniciales, noche para ciclos avanzados).

3.5. Alcances

La presente investigación propone el desarrollo de un sistema capaz de generar automáticamente horarios académicos completos, considerando una variedad de restricciones realistas y relevantes para el contexto universitario. El enfoque se basa en la construcción de un modelo híbrido con algoritmos genéticos y búsqueda tabú, enfocado inicialmente en los datos de la FISI, con potencial de ser adaptado a otras facultades.

El sistema deberá cumplir como mínimo con los siguientes desafíos:

- Eliminar conflictos entre cursos del mismo ciclo.
- Asignar docentes según preferencias mientras sea posible.
- Minimizar la mayor cantidad de espacios entre franjas horarias, tanto para docentes como para estudiantes.
- Priorizar la distribución de cursos en franjas horarias coherentes con las características del ciclo (se explicará a más detalle a posterior)
- Limitar el uso del día sábado, salvo cuando sea estrictamente necesario. Esto especialmente para ciclos académicos con más de 6 cursos.

3.5.1. Restricciones duras (Hard constraints)

- Un docente no puede dictar más de un curso al mismo tiempo.
- Se debe respetar el máximo de horas lectivas por docente
- Cada docente puede dictar como máximo 2 o 3 materias distintas
- Debe existir al menos 1 horario que contenga todos los cursos de los cursos que deberían aperturarse según la etapa del semestre en la que nos encontremos,de tal manera que, la asginatura se pueda cursar sin cruces entre franjas horarias
- Cursos de 1 a 4 ciclo deben llevarse en el horario entre las 8am y 1 pm y en lo mínimo posible en los horarios entre las 2pm a 6pm. Mientras que, los cursos desde 5to ciclo, deberán llevarse entre los horarios de 6pm a 10pm con algunos casos que se lleven entre las 2pm y 5pm
- Si estamos en la primera mitad del ciclo, todos los cursos impares deben tener un docente asignado o al menos un horario asignado. En caso estemos en la segunda mitad del ciclo, sucederá lo mismo, pero con los cursos impares

3.5.2. Restricciones blandas (Soft constraints)

- Reducir al mínimo los cruces entre cursos que no tengan relación de prerrequisitos.
- Minimizar el uso de clases en sábado, siempre que sea posible.
- Se buscará evitar cruces entre distintos ciclos académicos

3.6. Propuesta

Esta investigación propone el desarrollo de un sistema automatizado para la generación de horarios académicos, basado en un enfoque híbrido que combine algoritmos genéticos y búsqueda tabú. La solución será implementada en Python, permitiendo flexibilidad, modularidad y posibilidad de integración futura con sistemas académicos existentes.

El algoritmo genético se encargará de generar soluciones iniciales mediante una codificación adecuada del problema, usando operadores de selección, cruce y mutación adaptados a las restricciones del dominio. Luego, la búsqueda tabú se utilizará para refinar las mejores soluciones encontradas, evitando estancamientos y explorando nuevas combinaciones más eficientes.

Cada solución será evaluada mediante una función de penalización que integre tanto las restricciones duras como las blandas. Esta función permitirá comparar soluciones, guiar el proceso de evolución y asegurar que las propuestas finales cumplan los requisitos académicos y administrativos.

Con este enfoque, se espera reducir significativamente el tiempo y esfuerzo requeridos para la planificación de horarios, minimizar errores y ofrecer una herramienta práctica, escalable y adaptable, que mejore la experiencia de los usuarios y modernice la gestión académica en entornos universitarios.

3.7. Marco Teórico

3.7.1. Generación de Horarios Académicos como Problema de Optimización Combinatoria

La generación de horarios académicos constituye uno de los problemas más relevantes y complejos dentro del ámbito de la optimización combinatoria, debido a la gran cantidad de variables y restricciones que deben ser consideradas simultáneamente. Este tipo de problema pertenece a la clase de problemas **NP-difíciles**, lo que implica que no existe un algoritmo determinista y de tiempo polinómico conocido capaz de garantizar una solución óptima para todas las instancias del problema en un tiempo computacionalmente razonable. En la práctica, esto significa que conforme aumenta el número de cursos, docentes, aulas y franjas horarias involucradas, la cantidad de combinaciones posibles crece de manera exponencial, haciendo inviable el uso de enfoques de fuerza bruta o algorítmos exactos para casos de tama no realista.

Formalmente, el problema puede modelarse como un **problema de asignación restringida**, en el cual se busca encontrar la mejor manera de asignar un conjunto de cursos $C = c_1, c_2, ..., c_n$ a un conjunto de docentes $D = d_1, d_2, ..., d_m$ y aulas $A = a_1, a_2, ..., a_p$ dentro de un marco temporal definido por un conjunto de franjas horarias $H = h_1, h_2, ..., h_q$. Esta asignación debe cumplir una serie de restricciones duras, que no pueden violarse bajo ninguna circunstancia, y restricciones blandas, cuya satisfacción es deseable pero no obligatoria, y cuya violación introduce penalizaciones.

Hard Constrains

Son condiciones que deben cumplirse obligatoriamente para que una solución sea considerada válida. El incumplimiento de una restricción dura implica una violación crítica que invalida total o parcialmente la solución propuesta. Por ejemplo, asignar a un docente dos clases en la misma franja horaria o exceder su carga horaria máxima son restricciones duras, ya que atentan contra la factibilidad operativa del horario

Soft constrains

Son condiciones deseables pero no obligatorias, cuyo cumplimiento mejora la calidad de la solución sin ser estrictamente necesario. La violación de una restricción blanda no invalida la solución, pero puede generar penalizaciones que afectan su evaluación. Ejemplos comunes incluyen respetar las preferencias horarias de los docentes, evitar largas esperas entre clases o mantener los cursos del mismo ciclo dentro de franjas horarias similares.

3.7.2. Programación Académica

Para la problemática observada, la programación académica es entendida como el proceso de asignar, de manera organizada y eficiente, un conjunto de horarios, asignaturas y secciones a los docentes disponibles dentro de una unidad académica. Este proceso debe considerar diversas restricciones y criterios, tanto administrativos como pedagógicos, tales como la disponibilidad horaria del personal docente, la carga académica permitida por normativa, el número mínimo de secciones requeridas por curso, y la coherencia con los ciclos académicos establecidos.

Una programación académica adecuada busca maximizar el aprovechamiento de los recursos disponibles (aulas, docentes, tiempos) y minimizar conflictos como solapamientos de horarios, distribución desigual de carga horaria o ausencia de docentes para determinadas asignaturas. En contextos donde este proceso aún se realiza manualmente, como ocurre en varias facultades del país, se vuelve propenso a errores y requiere una cantidad significativa de tiempo y esfuerzo.

Por tanto, automatizar este proceso mediante técnicas de optimización puede representar una mejora sustancial en términos de eficiencia, equidad y calidad en la planificación educativa.

3.7.3. Problemas Metaheurísticos y su Aplicación a la Generación de Horarios

Los problemas de generación de horarios académicos se clasifican como **problemas de optimización combinatoria**, pertenecientes a la clase de problemas **NP-difíciles**, lo que implica que no existe un algoritmo polinómico conocido para resolverlos de manera óptima en un tiempo razonable cuando la instancia del problema crece en dimensiones. Estos problemas suelen presentar una gran cantidad de restricciones duras (que deben cumplirse) y blandas (preferencias deseables), lo cual amplía considerablemente el espacio de búsqueda.

Ante esta complejidad, se recurre a **metaheurísticas**, un conjunto de estrategias algorítmicas generales que permiten encontrar soluciones aceptables en tiempos razonables. Estas técnicas no garantizan la óptima global, pero son capaces de entregar soluciones de alta calidad mediante búsqueda inteligente y adaptativa. Ejemplos comunes incluyen *algoritmos genéticos*, *búsqueda tabú*, *simulated annealing*, y *enjambre de partículas*.

3.7.4. Algoritmos Genéticos (AG)

Los algoritmos genéticos, propuestos por Holland (1992)[10], están inspirados en los mecanismos de la evolución biológica y trabajan con una **población de soluciones** que evolucionan con el tiempo. Cada solución se representa como un **individuo** o **cromosoma**, compuesto por una secuencia de **genes**, que codifican una posible configuración del problema.

El proceso evolutivo de un AG incluye:

- **Selección**: se eligen los individuos más aptos según una función de evaluación o *fitness*. Los principales métodos incluyen:
  - **Selección por torneo**: selecciona aleatoriamente un subconjunto de la población y elige el mejor individuo entre ellos. Controla la presión selectiva y es robusto frente al ruido.
  - **Selección por ruleta**: asigna probabilidades proporcionales al valor de aptitud, similar a una ruleta ponderada. Puede tener problemas si hay grandes diferencias de aptitud.
  - **Selección por ranking**: ordena la población por aptitud y asigna probabilidades en función de la posición. Evita la dominancia de super-individuos.
  - **Selección por truncamiento**: escoge directamente el mejor porcentaje de la población. Simple pero propenso a perder diversidad.
- **Cruzamiento (crossover)**: se intercambian segmentos de genes entre padres para crear hijos.
- **Mutación**: se alteran aleatoriamente algunos genes para mantener la diversidad genética.

El AG termina cuando se alcanza un número máximo de generaciones o se obtiene una solución satisfactoria.

3.7.5. Representación de una asignación como Cromosoma

En el contexto de la generación de horarios, un **cromosoma** representa una asignación completa de cursos, docentes, días y franjas horarias. Cada **gen** podría codificar una asignación del tipo:

```
(asignatura, asignacion, docente, dia, hora_inicio, hora_fin, ciclo)
```

Por ejemplo, si tenemos 15 cursos, el cromosoma consistirá en 15 genes, cada uno representando la programación de un curso.

3.7.6. Búsqueda Tabú

La **búsqueda tabú** (Glover, 1986)[9] es una técnica de optimización que explora el espacio de soluciones vecinas y evita ciclos al mantener una memoria temporal de movimientos "prohibidos"(tabú). Se utiliza comúnmente para refinar soluciones generadas por otras metaheurísticas, como los algoritmos genéticos. Esta hibridación ha demostrado ser eficaz en problemas complejos de horarios universitarios, como se evidencia en Umam et al. (2022)[20].

3.7.7. Programación Lineal Entera

La **programación lineal entera** (PLE) consiste en optimizar una función lineal sujeta a restricciones lineales, donde algunas o todas las variables deben ser enteras. Es apropiada para problemas bien estructurados, pero su aplicación puede ser limitada en entornos altamente restrictivos o con criterios cualitativos, como preferencias horarias o equidad percibida. Estudios como el de Castro (2016)[14] muestran mejoras al aplicar PLE en contextos acotados, pero sufre escalabilidad frente a espacios de búsqueda muy grandes.

3.7.8. Comparación de Enfoques: Algoritmos genéticos y Búsqueda Tabú

Tan et al. (2021)[19] y Abdipoor et al. (2023)[1] concluyen que las metaheurísticas como AG y búsqueda tabú superan en flexibilidad y adaptabilidad a métodos clásicos como la programación lineal cuando se aplican a problemas con restricciones blandas, como la generación de horarios universitarios. Además, estas estrategias permiten incorporar preferencias y condiciones institucionales cambiantes con menor esfuerzo de reconfiguración.

Capítulo 4

Estado del Arte

4.1. Antecedentes Internacionales

4.1.1. Generación automática de horarios universitarios usando algoritmos evolutivos

Díaz y Espinosa-Arango [5] proponen un enfoque basado en algoritmos genéticos para la generación de horarios universitarios, considerando restricciones duras y blandas. Su investigación demuestra que estas técnicas pueden reducir los tiempos de planificación y aumentar la calidad de los horarios generados. El modelo presentado logra una asignación eficiente de cursos y docentes, adaptándose a las necesidades específicas de cada institución.

4.1.2. Optimización de horarios considerando eficiencia energética en edificios

Sun et al. [17] introducen un enfoque que, además de asignar horarios óptimos, minimiza el consumo energético al evitar abrir aulas con baja ocupación. Utilizan algoritmos genéticos considerando la cantidad de estudiantes por sección como restricción, lo que permite una asignación eficiente de recursos y una reducción significativa en el consumo energético de los edificios universitarios.

4.1.3. Revisión de metodologías de optimización en problemas de horarios escolares

Tan et al. [19] presentan un análisis comparativo entre algoritmos genéticos, programación lineal y búsqueda tabú, destacando la eficiencia de estas 'ultimas frente a métodos clásicos en problemas con gran cantidad de restricciones. Su estudio proporciona una guía valiosa para la selección de métodos de optimización adecuados según las características específicas del problema de horarios.

4.1.4. Hibridación de algoritmos genéticos y búsqueda tabú

Umam et al. [20] proponen una solución híbrida para la programación de producción, aplicable también a horarios académicos. Utilizan un algoritmo genético para generar soluciones base, afinadas posteriormente mediante búsqueda tabú. Esta combinación mejora la calidad de las soluciones y reduce el tiempo de cómputo necesario para encontrar horarios óptimos.

4.1.5. Problemas de horario escolar en contexto de reformas educativas

Sun y Wu [18] analizan cómo nuevas condiciones, como la elección de materias optativas, introducen incertidumbre en la demanda de secciones. Resaltan la necesidad de soluciones escalables y adaptables a cambios curriculares, proponiendo algoritmos que pueden ajustarse dinámicamente a las variaciones en las preferencias estudiantiles y en las políticas educativas.

4.1.6. Sincronización de calendarios académicos

Mallari et al. [12] enfatizan la importancia de contar con mecanismos que prioricen restricciones cuando no se pueden cumplir todas. Proponen una función de penalización flexible como herramienta clave, permitiendo una asignación de horarios que equilibra las necesidades institucionales con las preferencias individuales de docentes y estudiantes.

4.1.7. Enfoques metaheurísticos para problemas de horarios universitarios

Abdipoor et al. [1] dise nan e implementan un sistema basado en algoritmos genéticos que cumple con restricciones duras y blandas, mostrando su aplicabilidad en escenarios reales. Su investigación destaca la eficacia de los enfoques metaheurísticos en la resolución de problemas complejos de asignación de horarios en entornos universitarios.

4.2. Antecedentes Nacionales

4.2.1. Asignación de horarios con algoritmos genéticos

Castro [14] aplica un algoritmo genético a la planificación académica considerando disponibilidad docente y capacidad de aulas. Demuestra una mejora significativa frente a métodos manuales, alcanzando una mejora porcentual del 73.7

4.2.2. Sistema con interfaz gráfica para la generación de horarios

Blaz [2] desarrolla un sistema con interfaz gráfica para facilitar la asignación y edición de horarios en universidades peruanas. Incluye pseudocódigo replicable, lo que permite su adaptación y mejora por parte de otras instituciones educativas.

4.2.3. Propuesta de sistema para la FISI

Naupari y Rosales [13] plantean un sistema basado en algoritmos genéticos para la Facultad de Ingeniería de Sistemas e Informática (FISI), pero sin acotar adecuadamente las restricciones, lo cual podría generar soluciones ineficientes si la función de aptitud no está bien definida. Su trabajo resalta la importancia de una correcta definición de restricciones y funciones objetivo en la generación de horarios.

4.2.4. Importancia de una función de aptitud robusta

Holland [10] y Blaz [2] destacan que una función de aptitud mal dise nada puede comprometer la calidad de los resultados, incluso con un buen algoritmo base. Subrayan la necesidad de

una función de aptitud que refleje adecuadamente las prioridades y restricciones del problema de asignación de horarios.

4.2.5. Modelo basado en búsqueda tabú para la FISI

Ramos [15] aplica búsqueda tabú considerando restricciones específicas de la FISI. Su modelado lógico es base para soluciones adaptables, permitiendo una asignación de horarios que se ajusta a las particularidades de la facultad y mejora la eficiencia en la planificación académica.

4.2.6. Optimización considerando preferencias docentes

Basurto [3] propone incorporar preferencias horarias docentes como restricciones blandas. Argumenta que esto mejora la calidad de ense nanza y la aceptación del horario, al considerar las necesidades y disponibilidades de los docentes en el proceso de asignación de horarios.

Capítulo 5

Metodología

5.1. Enfoque metodológico

La presente investigación adopta un enfoque de **Ciencia del Diseño** (*Design Science Research*), cuyo objetivo es la construcción y evaluación de un artefacto computacional: un sistema generador de horarios académicos optimizados. Este artefacto se apoya en un enfoque **experimental-computacional**, donde se simula el comportamiento del algoritmo sobre instancias reales del problema y se miden los resultados obtenidos.

Complementariamente, se aplica extracción por medio de lectura de pdf mediante un flujo semiautomatizado de limpieza y ordenamiento

5.2. Recolección de datos

Se utilizaron tres técnicas principales:

- **Pdf scanning**: Se extrajeron datos desde páginas oficiales de la FISI (como el portal de horarios y docentes) utilizando scripts en Python.
- **Entrevistas semiestructuradas**: Dirigidas a responsables académicos (directores de escuela y personal administrativo), para identificar restricciones duras y blandas aplicadas en el proceso manual.
- **Observación directa**: Se documentaron los pasos del proceso actual de elaboración de horarios, incluyendo duración, tareas críticas y criterios informales de decisión.

5.3. Evaluación del modelo y validación

Dado que el producto final de este sistema es un archivo Excel con horarios académicos generados automáticamente, la evaluación del modelo se enfoca en la calidad de las soluciones y su adecuación a las restricciones institucionales, específicamente en la calidad de la información que se disponibilizará.

5.3.1. Criticidad de los constrains

Cuadro 5.1: Clasificación de restricciones según su criticidad

Categoría	**Valor**
Obligatoriedad	6
Muy crítico	5
Crítico	4
Preferible	3
Estético / de comodidad	2
No restrictivo	1

Se establecen distintos niveles de criticidad con el objetivo de jerarquizar las restricciones del sistema, permitiendo una evaluación más precisa y diferenciada de su impacto. Consideremos a las restricciones de criticidad de nivel 4 a 6 como un hard constrain mientras que las restricciones de 1 a 3 como soft constrains.

5.3.2. Constrains utilizadas

Cuadro 5.2: Restricciones consideradas y su nivel de criticidad

Índice	**Restricción**	**Categoría**
R1	Los cursos del mismo ciclo no deben compartir franja horaria	6
R2	Los cursos deben cumplir la totalidad de sus horas	6
R3	Los cursos estarán en determinadas franjas horarias (entre 8 a 1 pm y 2 pm a 10 pm)	6
R4	Un docente no puede dictar más de 4 cursos	5
R5	Que no considere menos docentes que la cantidad mínima de docentes recomendados por curso	5
R6	Los cursos desde el primer ciclo hasta el quinto ciclo se llevarán preferiblemente entre las 8 am y 1 pm, con un descanso entre la 1 pm y 2 pm, continuando a las 2 pm hasta las 6 pm	4
R7	Las preferencias horarias de los docentes disponibles pueden ser consideradas al momento de la asignación de horarios	3
R8	Para cursos del mismo ciclo se evitará agrupar 3 franjas horarias distintas en el mismo día	3
R9	Para un mismo docente, en un mismo día, se evitará tener espacios de tiempo mayores a 3 horas entre franjas horarias	3
R10	Para cursos del mismo ciclo, en un mismo día, se evitará tener espacios de tiempo vacíos mayores a 3 horas entre franjas horarias	3
R11	Los cursos que no sean prerequisitos de horario deben evitar compartir franjas horarias con los cursos de ciclos inmediatamente superiores o inferiores	2
R12	Los cursos se llevarán de lunes a viernes de preferencia	1

Se utilizaron las restricciones presentadas en el Cuadro 5.2, cada una asociada a un grado de criticidad que servirá como factor de ponderación en el cálculo de penalizaciones.

5.3.3. Validez interna

Se mide comparando el desempeño del sistema automático frente al proceso manual, utilizando los mismos datos de entrada. Se evalúa si el modelo genera horarios factibles, sin conflictos, respetando disponibilidad docente y distribuciones esperadas.

5.3.4. Validez externa

La validez externa se considera mediante la simulación de escenarios futuros, como la modificación de franjas horarias, incorporación de nuevos cursos o variación en la carga docente. Se busca verificar si el sistema mantiene su efectividad sin necesidad de ajustes estructurales.

5.3.5. Confiabilidad

La confiabilidad del sistema se evaluó mediante múltiples ejecuciones del algoritmo genético-tabú con el mismo conjunto de datos, midiendo la variación en los resultados y comprobando la estabilidad de las soluciones obtenidas.

5.4. Técnicas de evaluación

Para medir la calidad de las soluciones producidas por el modelo, se utilizaron los siguientes métodos:

- **Simulación comparativa**: Comparación entre el horario manual y el generado automáticamente en base a indicadores definidos.
- **Función de penalización**: Se construyó una función que asigna penalizaciones a horarios con cruces, violaciones de restricciones duras y blandas, o distribución inadecuada de cursos. Esta función guía la evaluación interna del algoritmo.
- **Medición de tiempos**: Se registró el tiempo de ejecución computacional del algoritmo y se contrastó con el tiempo promedio que toma la planificación manual en la FISI.

5.5. Limitaciones de la metodología

- La siguiente investigación se encuentra limitada a funcionalidades de optimización, información y centrada en el entregable final, un archivo consumible por excel
- La validación externa se limita a simulaciones controladas, sin pruebas aún en otras facultades o universidades.
- La posibilidad de extracción del proceso de los datos extraídos depende de la disponibilidad y estructura de los portales institucionales.

Capítulo 6

Desarrollo

6.0.1. Obtención de información

Obtención de información

Para la obtención de información se recurrió a diferentes secciones del Sistema Único de Matrícula (SUM) de la Universidad Nacional Mayor de San Marcos[1]. Desde esta plataforma institucional se extrajo la información necesaria para la construcción del modelo, la cual incluyó:

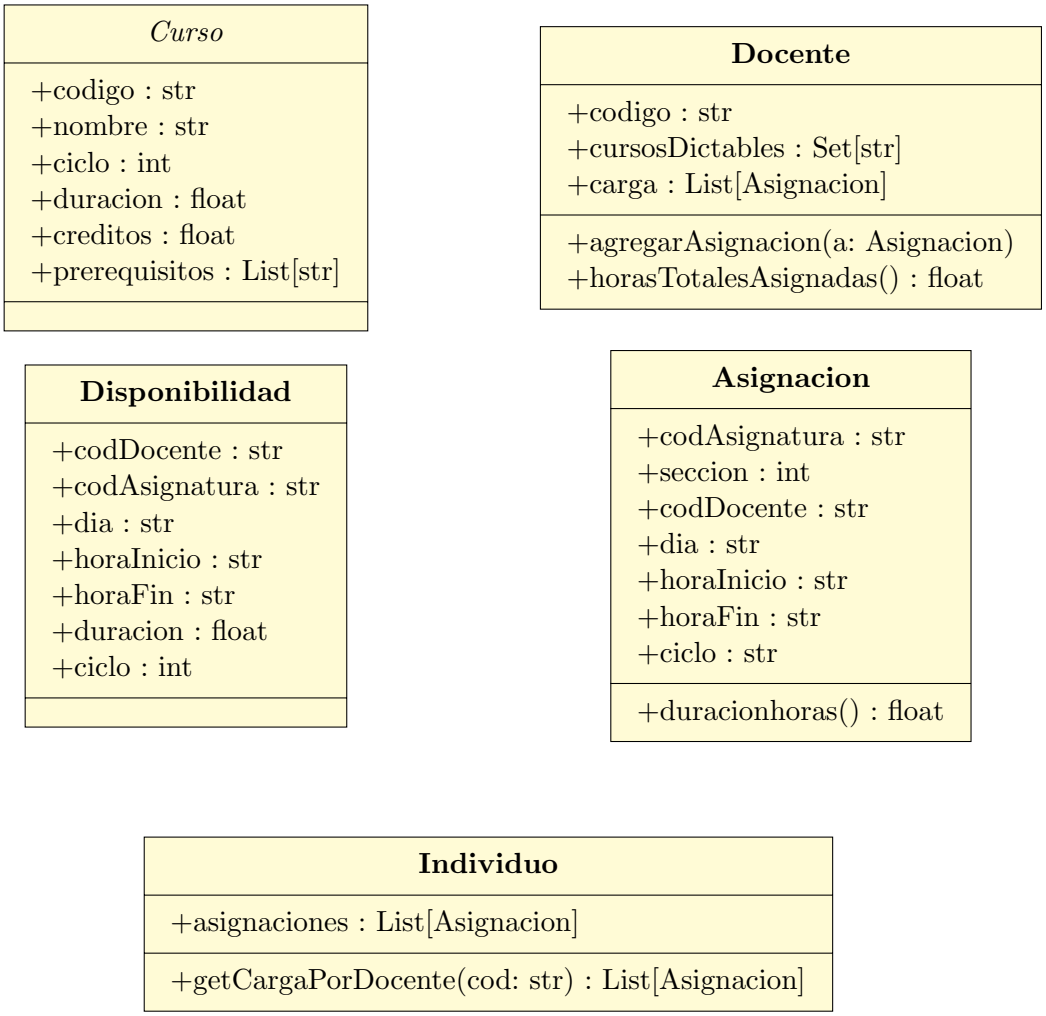
- **Horarios programados por semestre:** Se recopiló la oferta académica registrada, incluyendo cursos, horarios, docentes asignados y duración de cada sesión.
- **Volumen de estudiantes matriculados:** Se identificó la demanda estimada por curso, lo que permitió calcular el número mínimo de secciones requeridas.
- **Lista de docentes por escuela profesional:** Se obtuvo un listado de docentes disponibles, junto con sus códigos y, en algunos casos, su carga horaria máxima o preferencia.
- **Planes de estudio y dependencias:** Se recuperaron las mallas curriculares vigentes, incluyendo prerrequisitos y ciclos académicos, lo que permitió definir restricciones de consistencia entre asignaturas.

Toda esta información fue consolidada y procesada en hojas de cálculo para su posterior uso en el algoritmo de generación de horarios.

[1] `https://sum.unmsm.edu.pe`

6.0.2. Definición de clases

Figura 6.1: Diagrama de clases del modelo de horarios académicos



El presente modelo proporciona la estructura base sobre la cual se construyen los individuos candidatos en el algoritmo. Cada individuo está compuesto por una lista de asignaciones, donde se relacionan cursos, docentes y franjas horarias. A su vez, se contempla la disponibilidad del docente y las restricciones derivadas del plan curricular. Esta abstracción permite evaluar la factibilidad y calidad de cada horario generado, considerando tanto la carga docente como los requisitos curriculares y de asignación.

6.0.3. Creación del algoritmo genético

Creación del individuo

La creación de un individuo representa la generación inicial de una posible solución al problema de horarios académicos. Cada individuo está compuesto por una lista de objetos `Asignacion`, donde cada uno de ellos representa una sesión de clase asignada a una determinada franja horaria, día, docente y sección.

El proceso de creación de un individuo sigue los siguientes pasos:

1. **Definición de sesiones:** A partir de la demanda académica y la duración de cada curso, se define el número mínimo de secciones requeridas por asignatura. Para cada una de estas secciones, se crean las sesiones necesarias según la cantidad de horas lectivas estipuladas.

2. **Asignación de docente:** Para cada sesión, se selecciona aleatoriamente un docente disponible que tenga la capacidad horaria suficiente para dictar dicha clase.

3. **Asignación de franja horaria:** Se selecciona aleatoriamente una franja horaria válida entre las 08:00 a.m. y las 10:00 p.m., garantizando que el horario no se extienda fuera del rango permitido.

4. **Asignación de día:** Se elige aleatoriamente un día de la semana dentro del conjunto de días válidos: `LUNES` a `SÁBADO`.

5. **Composición del individuo:** Cada sesión se almacena como una instancia del objeto `Asignacion`, y el conjunto completo de sesiones conforma un individuo.

Este procedimiento busca generar una población inicial diversa que sirva como punto de partida para las fases posteriores de selección, cruce y mutación del algoritmo genético. Cabe resaltar que en esta etapa no se consideran restricciones duras como solapamientos o preferencia de docentes, lo cual permite una mayor exploración del espacio de búsqueda en las primeras generaciones

Calculo del fitness

Para la obtención del fitness se generaron distintas subfunciones que buscan evaluar cada una de las constrains

Restricción: Solapamiento entre cursos del mismo ciclo

Sea $I = \{a_1, a_2, \ldots, a_n\}$ el conjunto de asignaciones de un individuo. Cada asignación a_i tiene los atributos: `asignatura`,`seccion`, `docente`, ,`hora_inicio`,`hora_fin`, `ciclo`.

Definimos la penalidad para esta restricción como:

$$P_{\text{solape\_ciclo}} = \sum_{c \in C} \sum_{\substack{a_i, a_j \in I_c \\ i<j}} \delta(a_i, a_j)$$

Donde: - C es el conjunto de todos los ciclos presentes en el individuo. - I_c es el subconjunto de asignaciones tal que $a \in I_c$ si $a.\texttt{ciclo} = c$. - $\delta(a_i, a_j)$ es una función que vale 1 si existe solapamiento entre a_i y a_j en el mismo día, y 0 en caso contrario:

$$\delta(a_i, a_j) = \begin{cases} 1, & \text{si } a_i.\texttt{día} = a_j.\texttt{día} \text{ y seSolapan}(a_i, a_j) \\ 0, & \text{en otro caso} \end{cases}$$

La función `seSolapan` se define como:

$$\text{seSolapan}(a_i, a_j) = (a_i.\texttt{hora\_inicio} < a_j.\texttt{hora\_fin}) \wedge (a_j.\texttt{hora\_inicio} < a_i.\texttt{hora\_fin})$$

Así, se penaliza con 1 cada par de cursos del mismo ciclo que se solapan en el mismo día.

Restricción: Cumplimiento de horas requeridas por asignatura

Cada asignación $a_i \in I$ tiene los atributos: `cod_asignatura`, `seccion`, `hora_inicio`, `hora_fin`.

Sea $H_{\text{req}}(c, s)$ la cantidad de horas requeridas para la asignatura c y sección s, según la tabla de referencia (`df_duracion`).

Definimos la cantidad de horas dictadas por clave (c, s) como:

$$H_{\text{dict}}(c,s) = \sum_{a_i \in I_{(c,s)}} (a_i.\texttt{hora\_fin} - a_i.\texttt{hora\_inicio})$$

Donde $I_{(c,s)}$ es el conjunto de asignaciones de la asignatura c y sección s.
La penalidad total se define como:

$$P_{\text{horas}} = \sum_{(c,s)} \begin{cases} 1, & \text{si } H_{\text{req}}(c,s) \text{ no existe en la base de datos} \\ |H_{\text{dict}}(c,s) - H_{\text{req}}(c,s)|, & \text{en otro caso} \end{cases}$$

Es decir: - Si no se encuentra el curso en la tabla `df_duracion`, se penaliza con 1 unidad. - Si existe diferencia entre horas dictadas y requeridas, se penaliza proporcionalmente a la magnitud de la diferencia.

Restricción: Franjas horarias válidas

Cada asignación $a_i \in I$ tiene atributos `hora_inicio` y `hora_fin`, ambos en formato `HH:MM`. Se define el rango horario permitido como:

$$T_{\min} = 08:00 \quad (480 \text{ minutos}) \quad \text{y} \quad T_{\max} = 22:00 \quad (1320 \text{ minutos})$$

Convertimos las horas de cada asignación a minutos:

$$\text{inicio}_i = 60 \times h_{\text{ini}} + m_{\text{ini}} \quad ; \quad \text{fin}_i = 60 \times h_{\text{fin}} + m_{\text{fin}}$$

La penalidad total por franjas inválidas se define como:

$$P_{\text{franja}} = \sum_{a_i \in I} \begin{cases} 1, & \text{si inicio}_i < T_{\min} \text{ o fin}_i > T_{\max} \\ 0, & \text{en otro caso} \end{cases}$$

Es decir, se penaliza con 1 cada clase cuya franja horaria inicie antes de las 08:00 o termine después de las 22:00.

Restricción: Máximo de 4 cursos por docente

Sea D el conjunto de docentes, y para cada docente $d_j \in D$, definimos C_j como el conjunto de cursos únicos que le han sido asignados en el individuo:

$$C_j = \{c_i \mid \text{asignación } a_i \in I, \text{ con } a_i.\texttt{cod\_docente} = d_j\}$$

La cantidad de cursos asignados al docente d_j es:

$$n_j = |C_j|$$

Se permite un máximo de 4 cursos por docente. La penalidad total es:

$$P_{\text{cursos\_docente}} = \sum_{d_j \in D} \text{máx}(0, n_j - 4)$$

Es decir, por cada curso extra que exceda el límite de 4, se suma 1 punto de penalización.

Restricción: Mínimo de docentes por curso

Sea C el conjunto de cursos, y para cada curso $c_i \in C$, definimos D_i como el conjunto de docentes únicos que imparten dicho curso en el individuo:

$$D_i = \{d_j \mid \text{asignación } a_j \in I,\ a_j.\texttt{cod\_asignatura} = c_i\}$$

Se define m_i como el número mínimo de docentes requeridos para el curso c_i, según el conjunto de datos externos ('df$_d ocentes_m in$').

La penalidad para cada curso se calcula como:

$$P_i = \begin{cases} 1, & \text{si } c_i \notin \text{df\_docentes\_min} \\ \max(0, m_i - |D_i|), & \text{si } c_i \in \text{df\_docentes\_min} \end{cases}$$

La penalidad total es la suma sobre todos los cursos:

$$P_{\text{min\_docentes}} = \sum_{c_i \in C} P_i$$

Esta restricción penaliza con:

- 1 punto si el curso no se encuentra en la lista de requerimientos mínimos.
- 1 punto adicional por cada docente faltante con respecto al mínimo requerido.

Restricción: Horarios no preferidos para ciclos bajos

Definimos el conjunto de ciclos considerados bajos como:

$$C_{\text{bajo}} = \{1, 2, 3, 4, 5\}$$

Y los intervalos horarios penalizados:

- Hora de almuerzo: entre las 13:00 y las 14:00.
- Noche: a partir de las 18:00.

Para cada asignación a_i en el individuo, si el ciclo de a_i pertenece a C_{bajo} y la hora de inicio de clase h_i se encuentra en los intervalos penalizados, se suma una penalización.

Formalmente, para cada $a_i \in I$ con $a_i.\texttt{ciclo} \in C_{\text{bajo}}$, definimos:

$$P_i = \begin{cases} 1, & \text{si } 13{:}00 \leq h_i < 14{:}00 \\ 1, & \text{si } h_i \geq 18{:}00 \\ 0, & \text{en otro caso} \end{cases}$$

La penalidad total se calcula como:

$$P_{\text{horarios\_no\_preferidos}} = \sum_{a_i \in I,\, a_i.\texttt{ciclo} \in C_{\text{bajo}}} P_i$$

Esta restricción busca proteger los horarios de mejor desempeño para estudiantes de los primeros ciclos, evitando asignaciones durante la hora de almuerzo o a partir de las 6:00 p.m.

Restricción: Preferencias horarias del docente

Cada docente d_j declara su disponibilidad como un conjunto de intervalos válidos $[h_{ini}, h_{fin})$ para cada día de la semana. Estas preferencias están definidas en un diccionario de disponibilidad:

$$\text{Disponibilidad}[d_j][\text{día}] = \{[h_{ini}^{(1)}, h_{fin}^{(1)}], [h_{ini}^{(2)}, h_{fin}^{(2)}], \dots\}$$

Para cada asignación $a_i \in I$ con docente d_j, día d y horario $[h_{ini}^i, h_{fin}^i)$, se considera válida si existe algún intervalo en la disponibilidad declarada del docente que contenga completamente el horario asignado:

$$\exists [h_{ini}, h_{fin}] \in \text{Disponibilidad}[d_j][d] \text{ tal que } h_{ini} \leq h_{ini}^i \wedge h_{fin}^i \leq h_{fin}$$

Si no se cumple esta condición, se penaliza la asignación:

$$P_i = \begin{cases} 1, & \text{si el horario de clase no está dentro de la disponibilidad declarada} \\ 0, & \text{en otro caso} \end{cases}$$

La penalidad total asociada a esta restricción es:

$$P_{\text{preferencias\_docente}} = \sum_{a_i \in I} P_i$$

Esta restricción busca respetar las preferencias horarias declaradas por cada docente, promoviendo un horario más realista y aceptado.

Restricción: Máximo de 2 franjas por día para un mismo ciclo

Para evitar la sobrecarga académica en un mismo día, se establece que los estudiantes de un mismo ciclo no deben tener más de dos franjas horarias de clases por día.

Definimos:

- I: conjunto de asignaciones.
- c_i: ciclo de la asignación $a_i \in I$.
- d_i: día de la semana de la asignación a_i.

Se agrupan las asignaciones por par (c_i, d_i) y se contabiliza el número total de franjas (asignaciones) en ese día para ese ciclo:

$$\text{franjas}(c, d) = |\{a_i \in I \mid c_i = c \wedge d_i = d\}|$$

Se penaliza por cada franja adicional más allá de 2:

$$P_{c,d} = \begin{cases} \text{franjas}(c, d) - 2, & \text{si franjas}(c, d) > 2 \\ 0, & \text{en otro caso} \end{cases}$$

La penalidad total asociada es:

$$P_{\text{3\_franjas\_mismo\_día}} = \sum_{\substack{c \in \text{Ciclos} \\ d \in \text{Días}}} P_{c,d}$$

Esta restricción promueve una distribución más equilibrada de la carga horaria semanal para los estudiantes, evitando días excesivamente cargados para un mismo ciclo.

Restricción: Espacios mayores a 3 horas entre clases del mismo docente

Se penalizan los casos en los que un docente tiene intervalos de tiempo mayores a 3 horas entre dos sesiones de clase en el mismo día. Esta medida busca promover la eficiencia del tiempo docente y evitar huecos excesivos en su jornada laboral.

Definimos:

- I: conjunto de asignaciones.
- d_i: día de la semana de la asignación $a_i \in I$.
- t_i^{ini} y t_i^{fin}: hora de inicio y fin de la sesión a_i.
- doc_i: código del docente asignado a a_i.

Para cada docente d y cada día j, se agrupan y ordenan sus sesiones por hora de inicio. Para cada par de sesiones consecutivas (a_i, a_{i+1}), se calcula el tiempo entre la hora de fin de la sesión a_i y la hora de inicio de la siguiente sesión a_{i+1}:

$$\Delta t = t_{i+1}^{ini} - t_i^{fin}$$

Se penaliza si este espacio supera las 3 horas:

$$P_{docente,día} = |\{(a_i, a_{i+1}) | \Delta t > 3 \text{ horas}\}|$$

La penalización total por esta restricción se calcula como:

$$P_{\text{espacios\_mayores\_3h}} = \sum_{\substack{d \in \text{Docentes} \\ j \in \text{Días}}} P_{docente,día}$$

Esta restricción promueve una asignación más compacta y eficiente de los horarios del profesorado.

Restricción: Espacios mayores a 3 horas entre clases del mismo ciclo

Esta restricción penaliza los casos en los que los estudiantes de un mismo ciclo académico tienen huecos mayores a 3 horas entre clases programadas el mismo día. El objetivo es mejorar la continuidad del aprendizaje durante el día y reducir tiempos muertos innecesarios para los estudiantes.

Definimos:

- I: conjunto de asignaciones.
- c_i: ciclo al que pertenece la asignación $a_i \in I$.
- d_i: día de la semana de la asignación a_i.
- t_i^{ini} y t_i^{fin}: hora de inicio y fin de la asignación a_i.

Para cada ciclo c y día j, se agrupan las sesiones y se ordenan por hora de inicio. Se evalúan los espacios entre clases consecutivas (a_i, a_{i+1}):

$$\Delta t = t_{i+1}^{ini} - t_i^{fin}$$

Se considera penalizable si $\Delta t > 3$ horas:

$$P_{ciclo,día=|\{(a_i,a_{i+1})|\Delta t>3 \text{ horas}\}|}$$

La penalización total por esta restricción se calcula como:

$$P_{\text{espacios\_mayores\_3h\_ciclo}} = \sum_{\substack{c\in\text{Ciclos}\\ j\in\text{Días}}} P_{ciclo,día}$$

Esta penalización fomenta una programación académica más fluida para cada ciclo, reduciendo tiempos muertos prolongados entre sesiones.

Restricción: Cursos programados antes que sus prerrequisitos

Esta restricción penaliza los casos en que un curso es programado en un horario anterior (en el transcurso de la semana) que su prerrequisito directo, lo cual contradice la lógica pedagógica y de progresión curricular.

Para cada par (c, p) tal que p es prerrequisito de c, se evalúan los siguientes elementos:

- d_a: día de la semana en que se dicta una asignación a.
- t_a: hora de inicio de la asignación a.
- Se considera la sesión más temprana de cada curso según el par ordenado (d_a, t_a).

Sea:

$$\min_p = \min_{a\in\text{Sesiones de } p}(\text{orden}(d_a), t_a), \quad \min_c = \min_{a\in\text{Sesiones de } c}(\text{orden}(d_a), t_a)$$

$$P_{c\to p} = \begin{cases} 1, & \text{si } \min_c < \min_p \\ 0, & \text{en otro caso} \end{cases}$$

La penalización total se define como:

$$P_{\text{prerrequisitos}} = \sum_{(c,p)\in\text{Prerrequisitos}} P_{c\to p}$$

Esta penalización asegura que los cursos avanzados no sean dictados antes que sus prerrequisitos, respetando la secuencia curricular.

6.0.4. Función de evaluación (fitness)

La función de evaluación, también conocida como *fitness*, tiene como propósito cuantificar qué tan adecuada es una solución (individuo) dentro del contexto del problema de generación de horarios académicos. Esta función considera múltiples restricciones agrupadas en pedagógicas, administrativas y operativas, cada una con una penalización proporcional a su nivel de incumplimiento.

Cada restricción c_i se evalúa sobre el individuo, generando una penalidad p_i. A cada restricción se le asigna un peso w_i que representa su importancia relativa. La función total de penalización se define como:

$$\text{fitness}(x) = \sum_{i=1}^{n} w_i \cdot p_i(x)$$

donde:

- x: Individuo (solución candidata).
- $p_i(x)$: Número de violaciones a la restricción i en el individuo x.
- w_i: Peso asignado a la restricción i, que determina su impacto en la evaluación.

El valor de penalización total obtenido por esta fórmula se utiliza como función objetivo a minimizar. Cuanto menor sea el valor de `fitness`, mayor será la calidad de la solución.

Asignación de pesos

La asignación de pesos w_i fue realizada de forma empírica en base a la criticidad de cada restricción. Por ejemplo:

- Se asignó un peso de 6 a restricciones críticas como:
  - Cumplimiento de la totalidad de horas por curso.
  - Que los cursos de un mismo ciclo no compartan franja horaria.
  - Asignación en franjas horarias válidas (8 a.m. a 1 p.m., 2 p.m. a 10 p.m.).
- Se usaron pesos intermedios (5) para restricciones como:
  - Un docente no debe dictar más de 4 cursos.
  - Debe cumplirse con la cantidad mínima de docentes por curso.
- Se asignaron pesos bajos (2–3) a restricciones deseables pero no estrictas:
  - Respetar preferencias horarias de los docentes.
  - Evitar huecos mayores a 3 horas para un mismo docente o ciclo.
  - Evitar que un mismo ciclo tenga más de 2 franjas distintas en un día.

Resultado detallado

Además del valor total de fitness, la función retorna un desglose por tipo de restricción y una descripción detallada de los errores encontrados, lo cual permite identificar oportunidades de mejora a través de algoritmos de búsqueda local como Tabú o evolutivos como el genético.

Operador de cruce inteligente

El operador de cruce es una de las fases clave del algoritmo genético, ya que permite combinar soluciones existentes para explorar nuevas regiones del espacio de búsqueda. En este trabajo, se diseñó un **cruce inteligente guiado por penalidades**, que prioriza la herencia de componentes válidos de los padres, reduciendo la aparición de violaciones a las restricciones.

Selección de padres

Se utiliza selección por torneo para elegir a los dos padres:

- Se seleccionan aleatoriamente dos torneos independientes de k individuos cada uno (donde $k = 5$).
- En cada torneo, se elige como padre al individuo con menor penalización total según la función `calcular_fitness`.

Cruce basado en penalidades

Dado un par de padres P_1 y P_2, se evalúan sus penalidades específicas por asignación (*gene-wise*) usando la salida detallada de la función de evaluación.

- Para cada posición i del individuo (que representa una asignación de curso):
  - Si la penalidad de ambos padres en esa posición es cero, se escoge aleatoriamente entre los dos.
  - Si solo uno tiene penalidad cero, se elige ese.
  - Si ambos tienen penalidad, se escoge el que tiene menor penalidad en esa asignación.
- De esta forma se construye un nuevo individuo (**hijo**) que hereda las mejores decisiones de sus padres respecto a cada sección o sesión de clase.

Probabilidad de cruce

El cruce se aplica con una probabilidad $p_c = 0{,}9$. Si no se realiza cruce, ambos padres se copian directamente a la nueva población, preservando la diversidad genética.

Ventajas del cruce inteligente

- Permite una búsqueda más dirigida al evitar la propagación de errores.
- Aumenta la probabilidad de obtener hijos con menor penalización total.
- Favorece la convergencia hacia soluciones válidas más rápidamente.

Mutación basada en penalidad

Dada una lista de sesiones que representan un individuo (cada una correspondiente a una clase programada), se aplica una función de mutación cuyo grado de alteración depende directamente de la penalidad total del individuo.

- Sea penalidad $\in R^+$ la penalización actual del individuo, y penalidad_max la penalidad máxima esperada.
- Se define una probabilidad de mutación como:

$$p_{\text{mut}} = \text{mín}\left(1{,}0, \frac{\text{penalidad}}{\text{penalidad\_max}}\right)$$

- Para cada sesión del individuo:
  - Con probabilidad p_{mut}, se elige un nuevo día diferente al actual. Si un día específico ha sido pasado como argumento, se asigna ese día directamente. En caso contrario, se selecciona aleatoriamente de entre los días válidos excepto el día actual.
  - El nuevo día se asigna a la sesión.
- El individuo original no se modifica; en su lugar, se retorna una copia mutada del mismo.

Los días válidos son: `LUNES`, `MARTES`, `MIÉRCOLES`, `JUEVES`, `VIERNES`, `SÁBADO`.

Esta estrategia busca diversificar la población de soluciones con mayor penalidad, otorgando mayor presión evolutiva sobre los individuos menos aptos.

6.0.5. Creación del algoritmo tabú

Definición del vecindario

La generación de vecinos consiste en producir soluciones candidatas cercanas a la solución actual mediante la modificación de ciertas características del individuo. En este caso, el individuo I está representado como una lista de asignaciones A_i, donde cada A_i representa una sesión de clase definida por su asignatura, docente, día, hora de inicio y fin, y ciclo.

El conjunto de vecinos $\mathcal{V}(I)$ se construye aplicando dos mecanismos principales de exploración local:

1. **Resolución de solapamientos:** Se identifica cada par de asignaciones A_i y A_j dentro del mismo ciclo y día, y se verifica si hay solapamiento entre sus franjas horarias, es decir, si se cumple:

   $$A_i.\text{hora\_inicio} < A_j.\text{hora\_fin} \land A_j.\text{hora\_inicio} < A_i.\text{hora\_fin}$$

   Cuando se detecta un solapamiento, se crea una copia del individuo y se intenta reubicar A_i en una nueva franja no solapada utilizando una función auxiliar:

   $$(h'_{\text{inicio}}, h'_{\text{fin}}) = \texttt{mover\_franja\_no\_solapada}(A_i)$$

   Si existe una franja válida $(h'_{\text{inicio}}, h'_{\text{fin}})$, se asigna a A_i y se añade la nueva solución al conjunto de vecinos:

   $$\mathcal{V}(I) \leftarrow \mathcal{V}(I) \cup \{I'\}$$

2. **Apertura de nuevas secciones:** Para cada curso con número insuficiente de secciones según los requerimientos mínimos de cobertura, se genera una nueva asignación válida A_k:

   $$A_k = \texttt{generar\_asignacion\_valida}(c)$$

   Donde c es el código de asignatura que requiere mayor oferta. Esta asignación se añade a una copia del individuo original:

   $$I' \leftarrow I + A_k \quad \Rightarrow \quad \mathcal{V}(I) \leftarrow \mathcal{V}(I) \cup \{I'\}$$

Este procedimiento permite que el algoritmo de búsqueda tabú explore el espacio de soluciones intentando mejorar la solución actual, tanto reduciendo penalidades por solapamientos como incrementando la cobertura de demanda estudiantil. Cabe resaltar que esta generación se realiza sin aplicar restricciones duras adicionales, permitiendo una exploración más amplia del espacio factible en las primeras fases del algoritmo.

Algoritmo Tabú para minimizar solapamientos y garantizar secciones mínimas

El algoritmo Tabú propuesto tiene como objetivo mejorar progresivamente un individuo (i.e., una solución representando un horario académico) minimizando penalizaciones asociadas a solapamientos y a la falta de secciones necesarias por curso.

Inicialmente se parte de un individuo I_0 considerado como solución actual y también como la mejor solución encontrada hasta el momento. En cada iteración se generan vecinos aplicando dos estrategias: la reasignación de franjas horarias para eliminar solapamientos entre clases del mismo ciclo y día, y la adición de nuevas asignaciones en caso de que no se cumpla con la cantidad mínima de secciones requeridas para un curso.

Sea $V(I)$ el conjunto de vecinos generados a partir de un individuo I. Se define una lista tabú T de tamaño fijo que impide volver a visitar soluciones recientemente evaluadas. De este modo, se evita caer en ciclos locales.

En cada iteración del algoritmo:

1. Se genera el conjunto de vecinos filtrados por la lista tabú:

$$V'(I) = \{v \in V(I) \mid v \notin T\}$$

2. Se evalúa el fitness de cada vecino $v \in V'(I)$ mediante una función de penalización $f(v)$ que cuantifica cuán válida es la solución en base a restricciones establecidas (e.g., solapamientos, secciones mínimas, etc.).

3. Se selecciona el vecino con menor penalización:

$$I_{\text{vecino}} = \arg \min_{v \in V'(I)} f(v)$$

4. Se actualiza la solución actual $I_{\text{actual}} \leftarrow I_{\text{vecino}}$, y se añade a la lista tabú. En caso de superar el tamaño máximo permitido, se elimina el elemento más antiguo.

5. Si la penalización del vecino seleccionado mejora la mejor solución encontrada:

$$f(I_{\text{vecino}}) < f(I_{\text{mejor}})$$

   entonces se actualiza $I_{\text{mejor}} \leftarrow I_{\text{vecino}}$.

Este proceso se repite hasta completar un número máximo de iteraciones k o hasta que no existan más vecinos válidos fuera de la lista tabú.

Finalmente, el algoritmo retorna la mejor solución encontrada I_{mejor} y el historial de penalizaciones registradas a lo largo del proceso.

[H] Individuo inicial I_0, función de penalización f, iteraciones máximas k, tamaño de lista tabú t Mejor solución I_{mejor}, historial de penalizaciones $actual \leftarrow I_0$, $mejor \leftarrow I_0$,
$T \leftarrow \emptyset$
$i = 1$ k $V \leftarrow$ vecinos de $actual$ que no están en T
V está vacío**break**
$v^* \leftarrow \arg\min_{v \in V} f(v)$
$actual \leftarrow v^*$
Agregar v^* a T, eliminar el más antiguo si $|T| > t$
$f(v^*) < f(mejor) mejor \leftarrow v^*$ $mejor$, historial de penalizaciones

Capítulo 7

Análisis de resultados

7.0.1. Análisis de resultados Ejecución Genética

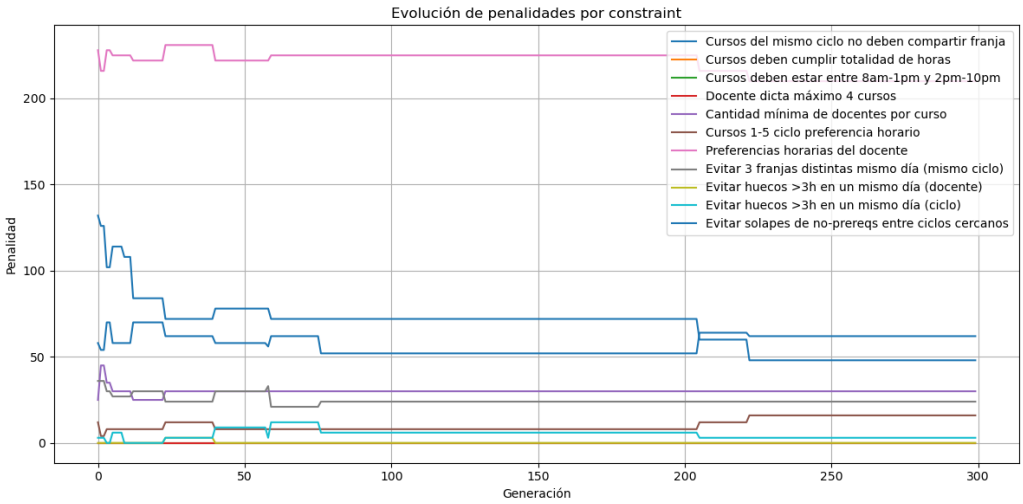


Figura 7.1: Descripción de la imagen.

Como se observa en la Figura 7.1, a medida que avanzan las generaciones del algoritmo, la cantidad total de penalidades tiende a disminuir progresivamente. Las penalidades más frecuentes en las primeras generaciones corresponden principalmente a las preferencias horarias de los docentes. Sin embargo, esto puede considerarse razonable, dado que el modelo parte del supuesto de que los horarios de clase son definidos previamente por la facultad, y la asignación de docentes se realiza en función de dicha estructura, priorizando el cumplimiento de restricciones académicas por encima de las preferencias individuales.

Asimismo, se evidencia que en etapas más avanzadas del proceso evolutivo, las penalidades predominantes corresponden a restricciones de tipo *soft*, lo cual indica una mejora significativa en el cumplimiento de las restricciones *hard* como la no superposición de clases, la apertura mínima de secciones y el respeto por las cargas máximas de los docentes. Esto sugiere una convergencia positiva del modelo hacia soluciones más viables desde un punto de vista institucional.

7.0.2. Análisis de resultados Búsqueda Tabú

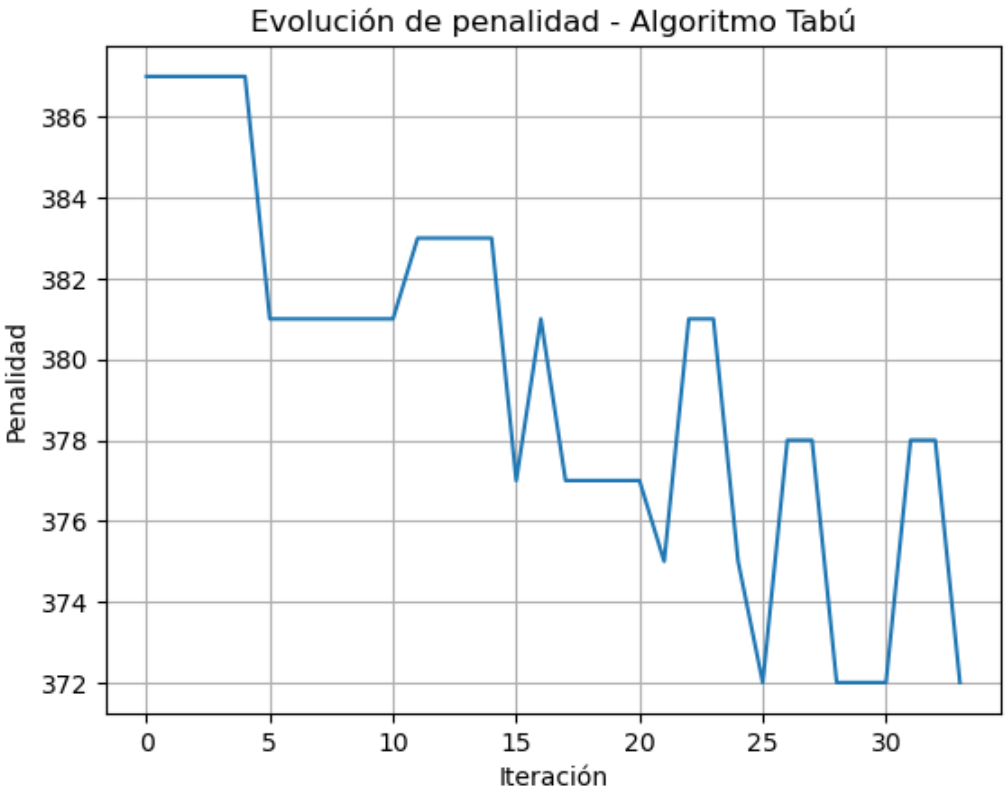


Figura 7.2: Descripción de la imagen.

Asimismo, se comprueba la efectividad del algoritmo Tabú como técnica de mejora local, ya que logra reducir significativamente las penalidades obtenidas inicialmente por el algoritmo genético. Esta reducción evidencia que el modelo híbrido propuesto permite refinar las soluciones generadas evolutivamente, ajustándolas aún más a las restricciones definidas.

Sin embargo, se observa que aún persisten algunas penalidades residuales, las cuales corresponden mayormente a restricciones de tipo *soft*, como las preferencias horarias o distribución óptima de sesiones semanales. Estas penalidades, si bien no comprometen la viabilidad de la solución, representan oportunidades de mejora y delimitan un espacio claro para trabajos futuros que busquen una optimización más exhaustiva y personalizada del horario académico.

Capítulo 8

Conclusiones

8.0.1. Conclusiones

Viabilidad de la metaheurística híbrida.
La combinación del algoritmo genético con el algoritmo Tabú ha demostrado ser una estrategia viable y eficaz para abordar el problema de la generación de horarios académicos, especialmente en un entorno con múltiples restricciones y dependencias como el de la Facultad de Ingeniería de Sistemas e Informática.

Convergencia progresiva hacia soluciones más viables.
A medida que avanzan las generaciones, se evidencia una clara tendencia decreciente en la cantidad de penalidades totales. Esto indica una convergencia del modelo hacia soluciones que respetan de mejor forma las restricciones *hard*, como la no superposición de clases, la carga horaria máxima de los docentes y la apertura mínima de secciones por curso.

Efectividad del algoritmo Tabú como refinamiento.
El algoritmo Tabú ha sido clave para reducir penalidades residuales tras la ejecución del algoritmo genético. Este proceso de mejora local permitió refinar asignaciones sin modificar sustancialmente la estructura general del individuo, reforzando la calidad y coherencia del horario generado.

Se observa, además, un rendimiento positivo del enfoque híbrido respecto a la disminución de penalidades por restricciones, lo cual respalda su efectividad general. No obstante, se advierte que, ante la posible incorporación de nuevas *constrains* en el sistema, será necesario realizar ajustes adicionales al modelo, tanto en su representación como en los operadores evolutivos y de búsqueda local utilizados.

Capítulo 9

Trabajo futuro

9.0.1. Errores detectados en los datos oficiales

Durante el análisis y procesamiento de los datos extraídos de los archivos PDF oficiales del SUM (Sistema Universitario de Matrícula), se identificaron inconsistencias que podrían afectar directamente la calidad del modelo de automatización propuesto. Entre los errores más relevantes se encontraron:

- **Duplicidad de horarios para un mismo docente:** Se hallaron casos donde un mismo docente aparece asignado a más de una clase en franjas horarias coincidentes, lo cual representa un solapamiento inviable en la realidad operativa.
- **Asignaturas con horarios no válidos o inconsistentes:** Algunas materias registraban horarios fuera del rango habitual (e.g., fuera del horario lectivo entre 08:00 y 22:00) o con duraciones incoherentes respecto a su carga horaria oficial.

Estas inconsistencias deben ser corregidas o filtradas previamente en caso se desee implementar un proceso automatizado de asignación de horarios, ya que podrían afectar tanto la factibilidad como la validez de las soluciones generadas. Se recomienda establecer un procedimiento de validación de datos previo al entrenamiento de los algoritmos, así como una limpieza sistemática que permita garantizar la integridad y coherencia de la información usada como insumo

9.0.2. Dificultades para automatización

El proceso de generación automatizada de horarios académicos enfrentó diversas limitaciones, principalmente asociadas a la calidad y disponibilidad de los datos fuente. Aunque se diseñó un modelo capaz de generar asignaciones válidas a partir de los archivos oficiales del SUM (Sistema Universitario de Matrícula), su implementación completa no fue viable debido a los siguientes factores:

- **Necesidad de limpieza manual de datos:** Se detectaron múltiples inconsistencias en los archivos PDF extraídos del SUM, incluyendo duplicidad de horarios para docentes, asignaturas con franjas horarias inválidas y registros con información incompleta. Estas anomalías impiden una ejecución confiable sin intervención manual.
- **Falta de conexión directa con el API del SUM:** A pesar de que el sistema SUM permite la transmisión de información en formato `.json`, no se logró establecer una conexión automatizada con su API oficial. Esto impide validar de forma programática la veracidad o integridad de los datos obtenidos, y limita la capacidad del modelo para adaptarse a cambios en tiempo real.

- **Retrabajo requerido:** La naturaleza de los errores detectados implica que incluso si se automatizara parcialmente el proceso, aún sería necesario realizar validaciones y correcciones posteriores, lo cual contradice el objetivo de reducción de carga manual en la generación de horarios.

En consecuencia, se concluye que para alcanzar una automatización efectiva se debe primero establecer un canal de comunicación estable con el sistema institucional (preferiblemente mediante API) y asegurar que los datos entregados estén estructurados, normalizados y validados desde el origen.

9.0.3. Redefinir y ampliar *constraints*

Durante la implementación del modelo se ha identificado la necesidad de redefinir y ampliar el conjunto de *constraints* consideradas en el algoritmo, con el fin de representar de manera más precisa las condiciones reales del proceso de asignación de horarios en toda la Facultad de Ingeniería de Sistemas e Informática.

Actualmente, algunos criterios clave no han sido abordados de forma integral, siendo los más relevantes los siguientes:

- **Disponibilidad docente:** Si bien el modelo contempla franjas horarias válidas, aún no se ha incorporado una restricción que verifique explícitamente la disponibilidad declarada por cada docente. Esto puede generar asignaciones en horarios en los que el docente no se encuentra disponible, afectando directamente la viabilidad del horario.
- **Cobertura completa de programas:** La versión actual del modelo trabaja con un subconjunto representativo de cursos. Sin embargo, para una implementación a nivel facultad, es necesario incluir las secciones correspondientes a todas las escuelas profesionales, con sus respectivos ciclos, asignaturas y demandas estudiantiles.
- **Restricciones adicionales específicas:** Se requieren nuevas restricciones que consideren, por ejemplo, la secuencia adecuada de cursos por prerequisitos, la distribución equilibrada de carga docente por semana, la minimización de huecos entre clases de un mismo grupo, entre otros aspectos relevantes para una solución realista y óptima.

Por tanto, se plantea como trabajo pendiente el diseño y la integración de estas restricciones adicionales, así como una revisión detallada de los pesos asignados a cada una dentro de la función de *fitness*, para asegurar que el modelo refleje de manera fidedigna las prioridades y requerimientos académicos de la facultad.

Bibliografía

[1] Sina Abdipoor et al. "Meta-heuristic approaches for the University Course Timetabling Problem". En: *Engineering Applications of Artificial Intelligence* 19 (2023), pág. 200253. DOI: 10.1016/j.engappai.2023.200253. URL: https://www.sciencedirect.com/science/article/pii/S2667305323000789.

[2] Sherly Patricia Blaz Aristo. "Un sistema de generación de horarios para la enseñanza de pregrado en universidades peruanas mediante algoritmos genéticos". Consultado el 3 de marzo de 2025. Universidad Nacional Mayor de San Marcos, 2016. URL: https://cybertesis.unmsm.edu.pe/item/85655874-a56c-45a2-b8d4-39a58832cfc6.

[3] Daniel Elkin Basurto Rodríguez. "Asignación de horario minimizando las horas libres de los profesores mediante la metaheurística búsqueda tabú". 2022. URL: https://cybertesis.unmsm.edu.pe/handle/20.500.12672/19495.

[4] Sherly Patricia Blaz Aristo. "Un sistema de generación de horarios para la enseñanza de pregrado en universidades peruanas mediante algoritmos genéticos". 2016. URL: https://cybertesis.unmsm.edu.pe/handle/20.500.12672/4943.

[5] Juan Carlos Díaz-Ramírez y Jairo Espinosa-Arango. "Generación automática de horarios universitarios usando algoritmos evolutivos". En: *Revista DYNA* 87.214 (2020), págs. 89-96. DOI: 10.15446/dyna.v87n214.86554. URL: http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0012-73532020000400047&lang=es.

[6] Superintendencia Nacional de Educación Superior Universitaria (SUNEDU). *III Informe Bienal sobre la Realidad Universitaria en el Perú*. Consultado el 3 de marzo de 2025. 2022. URL: https://cdn.www.gob.pe/uploads/document/file/3018068/III%20Informe%20Bienal.pdf?v=1649883911.

[7] Instituto Nacional de Estadística e Informática (INEI). *Perú: Compendio Estadístico 2021*. Consultado el 3 de marzo de 2025. 2021. URL: https://www.inei.gob.pe/media/MenuRecursivo/publicaciones_digitales/Est/Lib1743/Libro.pdf.

[8] UNMSM Facultad de Ingeniería de Sistemas e Informática. *Ranking Ponderado Semestre 2023-I*. Consultado el 3 de marzo de 2025. 2023. URL: https://sistemas.unmsm.edu.pe/site/images/pdf/noticias/RANKING-PONDERADO-SEMESTRE_2023-1.pdf.

[9] Fred Glover. "Future paths for integer programming and links to artificial intelligence". En: *Computers & Operations Research* 13 (1986), págs. 533-549. DOI: 10.1016/0305-0548(86)90048-1. URL: https://www.sciencedirect.com/science/article/abs/pii/0305054886900481.

[10] John H. Holland. "The General Setting". En: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. MIT Press, 1992. URL: https://ieeexplore.ieee.org/book/6267401.

[11] Yalin Hou et al. “High capacity NP-Complete problems solver based on dual-comb asynchronous optical sampling”. En: *Optics Communications* 550 (2023), pág. 130021. DOI: 10.1016/j.optcom.2023.130021. URL: https://www.sciencedirect.com/science/article/pii/S0030401823007691.

[12] Cristina Beatrice Mallari, Jayne Lois San Juan y Richard Li. “The university coursework timetabling problem: An optimization approach to synchronizing course calendars”. En: *Computers & Operations Research* 184 (2023), pág. 109561. DOI: 10.1016/j.cor.2023.109561. URL: https://www.sciencedirect.com/science/article/abs/pii/S0360835223005855.

[13] Raúl Esteban Naupari Quiroz y Gissela Katheryn Rosales Gerónimo. “Aplicación de algoritmos genéticos para el diseño de un sistema de apoyo a la generación de horarios de clases para la Facultad de Ingeniería de Sistemas e Informática de la UNMSM”. 2010. URL: https://cybertesis.unmsm.edu.pe/handle/20.500.12672/15141.

[14] Henry Castro Pocco. “Algoritmo Genético Aplicado en la Programación Académica de una Escuela Profesional en una Facultad Universitaria”. Consultado el 3 de marzo de 2025. Universidad Nacional del Callao, 2016. URL: https://repositorio.unac.edu.pe/bitstream/handle/20.500.12952/1589/Henry_Tesis_tituloprofesional_2016.pdf?sequence=1&isAllowed=y.

[15] Frank Roger Ramos Milla. “Sistema para la generación de horarios académicos en instituciones universitarias usando algoritmo Tabú”. 2012. URL: https://cybertesis.unmsm.edu.pe/handle/20.500.12672/4913.

[16] Hugo Sandelius y Simon Forsell. “Comparison of algorithms for automated university scheduling”. Tesis de mtría. KTH Royal Institute of Technology, 2014. URL: https://kth.diva-portal.org/smash/record.jsf?pid=diva2%3A771160&dswid=-9988.

[17] Yongkai Sun, Xi Luo y Xiaojun Liu. “Optimization of a university timetable considering building energy efficiency: An approach based on the building controls virtual test bed platform using a genetic algorithm”. En: *Energy Reports* 35 (2021), pág. 102095. DOI: 10.1016/j.egyr.2021.107747. URL: https://www.sciencedirect.com/science/article/abs/pii/S235271022033727X.

[18] Zhe Sun y Qinghua Wu. “Two-phase tabu search algorithm for solving Chinese high school timetabling problems under the new college entrance examination reform”. En: *Engineering Science and Technology* 6 (2023), págs. 55-63. DOI: 10.1016/j.jestch.2022.12.001. URL: https://www.sciencedirect.com/science/article/pii/S2666764923000073.

[19] Joo Siang Tan et al. “A survey of the state-of-the-art of optimisation methodologies in school timetabling problems”. En: *Expert Systems with Applications* 165 (2021), pág. 113943. DOI: 10.1016/j.eswa.2020.113859. URL: https://www.sciencedirect.com/science/article/abs/pii/S0957417420307314.

[20] Moch Saiful Umam, Mustafid Mustafid y Suryono Suryono. “A hybrid genetic algorithm and tabu search for minimizing makespan in flow shop scheduling problem”. En: *Materials Today: Proceedings* 34 (2022), págs. 7459-7467. DOI: 10.1016/j.matpr.2021.11.686. URL: https://www.sciencedirect.com/science/article/pii/S1319157821002287.

[21] Universidad Nacional Mayor de San Marcos. *Sistema Universitario de Matrícula (SUM)*. Consultado el 3 de marzo de 2025. 2025. URL: https://sum.unmsm.edu.pe/.

[22] Xin-She Yang. *Nature-Inspired Optimization Algorithms*. Second. Scopus, 2021. URL: https://www.sciencedirect.com/topics/engineering/genetic-algorithm.