Inteligencia Artificial

Informe Final: Examination Timetabling Problem

Benjamín Araos

24 de noviembre de 2025

Evaluación

Código Fuente (20 %):	_____
Resumen (2 %):	_____
Introducción (3 %):	_____
Representación (10 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

El *Examination Timetabling Problem* (ETP) es un problema de optimización combinatoria que consiste en asignar un conjunto de exámenes a bloques horarios y salas limitadas, sujeto a restricciones de capacidad y conflictos. Este trabajo presenta una revisión exhaustiva del estado del arte, analizando su evolución desde los métodos de coloración de grafos de 1960 hasta las metaheurísticas modernas. Como aporte principal, se propone e implementa un algoritmo de Búsqueda Tabú para la variante *Room-Capacitated*, diseñando operadores de vecindario específicos (movimientos espejo y cambios de sala) y una estrategia de evaluación incremental. Finalmente, se formula un modelo matemático de programación lineal entera corregido para representar formalmente el problema y su espacio de búsqueda de orden exponencial $2^{n \cdot p \cdot m + t \cdot n \cdot 5}$.

1. Introducción

El *Examination Timetabling Problem* (ETP) representa uno de los desafíos logísticos más críticos en la gestión académica. En este documento se abordan los detalles de la variante *Room-Capacitated*, la cual exige no sólo evitar conflictos horarios entre estudiantes, sino también respetar rigurosamente las capacidades físicas de las salas asignadas. El problema consiste en determinar el momento y lugar exacto para un conjunto de evaluaciones, minimizando la duración total del periodo de exámenes.

1.1. Propósito e Innovación

El propósito de este trabajo es desarrollar una solución robusta basada en la metaheurística de Búsqueda Tabú que sea capaz de generar horarios factibles y compactos en tiempos de

cómputo razonables. La principal innovación de la propuesta radica en el diseño del vecindario de búsqueda: se introducen movimientos no convencionales, como el "movimiento espejo" (invertir el bloque horario de un examen), combinados con una estrategia de evaluación incremental (Delta Evaluation). Esto permite explorar el espacio de soluciones escapando eficientemente de óptimos locales, una limitación común en los enfoques constructivos tradicionales.

1.2. Estructura del Documento

El documento se encuentra organizado de la siguiente manera: En la Sección 2, se define formalmente el problema, detallando sus variables y restricciones. La Sección 3 presenta el estado del arte, revisando desde los orígenes históricos hasta las tendencias actuales en algoritmos de resolución. En la Sección 4 se formula el modelo matemático de programación lineal entera propuesto. Posteriormente, las Secciones 5 y 6 describen la representación de la solución y la implementación detallada del algoritmo de Búsqueda Tabú, respectivamente. Finalmente, la Sección 7 expone el diseño experimental, la Sección 8 analiza los resultados obtenidos y la Sección 9 presenta las conclusiones y trabajos futuros.

2. Definición del Problema

El **Examination Timetabling Problem (ETP)** se define como la asignación de un conjunto de exámenes a un número finito de bloques horarios y salas, satisfaciendo un conjunto de restricciones duras y blandas.

2.1. Componentes y Variables

La complejidad del problema surge de la interacción entre sus cuatro componentes principales:

- **Exámenes (E):** Conjunto de pruebas a planificar. Cada examen tiene una lista de estudiantes inscritos.
- **Estudiantes (S):** Conjunto de alumnos que generan las restricciones de conflicto (un alumno no puede estar en dos lugares a la vez).
- **Bloques Horarios (T):** Periodos de tiempo discretos disponibles.
- **Salas (R):** Espacios físicos con una capacidad de aforo (asientos) definida.

2.2. Dificultades y Complejidad

El ETP pertenece a la clase de problemas **NP-Duro** (NP-Hard) [7]. Esto implica que no existe un algoritmo conocido que pueda encontrar la solución óptima en tiempo polinomial para todas las instancias. Las principales dificultades incluyen:

- **Densidad de Conflictos:** En instancias reales, la matriz de conflictos suele ser densa, lo que hace que encontrar siquiera una solución factible sea un desafío similar al problema de coloración de grafos.
- **Restricciones de Capacidad:** A diferencia del problema clásico, la variante con capacidad limita severamente el número de exámenes simultáneos, desconectando el espacio de búsqueda y dificultando la navegación entre soluciones vecinas.

2.3. Variantes del Problema

Además de la variante estudiada, la literatura identifica otras formulaciones relevantes [13]:

- **Uncapacitated ETP:** Se asume que las salas tienen capacidad infinita. El problema se reduce a minimizar los bloques horarios respetando sólo conflictos de alumnos.
- **Room-Capacitated ETP:** (Variante de este proyecto) Las salas tienen capacidades fijas y limitadas. Es la variante más realista para la mayoría de las instituciones.
- **Slot-Capacitated ETP:** En lugar de salas específicas, se define una capacidad global máxima de alumnos por bloque horario.
- **Split-Exams ETP:** Permite que un mismo examen se divida en múltiples salas si supera la capacidad de una sola, añadiendo una capa extra de complejidad a la asignación.

3. Estado del Arte

3.1. Orígenes Históricos

Los orígenes del Examination Timetabling Problem (ETP) se remontan a la década de 1960, siendo uno de los primeros registros el trabajo de Gotlieb [8] que, aunque enfocado en horarios de clases, sentó las bases para los problemas de planificación educativa. Poco después, Cole [4] presentó uno de los primeros trabajos en abordar específicamente la programación de exámenes usando métodos computacionales.

Un avance significativo llegó con Wood [19], quien desarrolló uno de los primeros algoritmos prácticos para resolver problemas de gran tamaño utilizando el sistema Atlas. La base teórica se consolidó con Welsh y Powell [17], estableciendo la conexión fundamental entre la coloración de grafos y los problemas de planificación. Finalmente, Laporte y Desroches [9] en 1984 realizaron una revisión extensa y describieron un procedimiento automático completo para la elaboración de horarios de exámenes universitarios.

3.2. Evolución de Métodos de Solución

3.2.1. Métodos Tradicionales (1960-1990)

Los primeros enfoques para resolver el ETP se basaron en métodos exactos como técnicas de coloración de grafos [17], que establecieron la conexión fundamental entre ambos problemas. Sin embargo, la naturaleza NP-duro del ETP [7] limitó severamente la aplicabilidad de estos métodos a instancias pequeñas, ya que para instancias grandes era imposible sacar buenos resultados en tiempo, no era escalable.

En la década de 1980, las heurísticas constructivas (construyen solución desde cero) ganaron popularidad, destacándose algoritmos secuenciales [11] que asignaban exámenes uno por uno según criterios de dificultad, y técnicas basadas en grafos más sofisticadas [5]. Estos métodos, aunque no garantizaban que sean los más óptimos, permitían resolver instancias de tamaño realista en tiempos computacionales razonables, mucho mejor que antes.

3.2.2. Era de las Metaheurísticas (1990-2010)

La década de 1990 marcó un punto de inflexión con la adopción de metaheurísticas (sirve como guía para el proceso de búsqueda de soluciones) que demostraron superior efectividad en la exploración de espacios de búsqueda complejos:

- **Algoritmos Genéticos** [6]: Operan sobre una población de horarios, donde las instancias más aptas son seleccionados para seguir avanzando. Utilizan operadores de cruce especializados que mantienen la factibilidad de los horarios y mutaciones que intercambian períodos de exámenes conflictivos, evolucionando iterativamente hacia soluciones de mayor calidad.
- **Recocido Simulado** [16]: Explora el espacio de horarios mediante movimientos que reubican exámenes, aceptando ocasionalmente soluciones peores según un criterio probabilístico que disminuye con el tiempo. Esta característica le permite escapar de óptimos locales en la búsqueda de una distribución globalmente mejor de los exámenes.
- **Búsqueda Tabú** [18]: Realiza una exploración local inteligente manteniendo una memoria de movimientos recientes (estas se guardan en una lista tabú) para evitar ciclos, como reasignar repetidamente el mismo examen a períodos conflictivos. Incorpora estrategias de diversificación que reestructuran horarios con muchos conflictos estudiantiles, buscando mejoras en la distribución temporal de los exámenes.
- **Algoritmos Híbridos** [2]: Combinando las ventajas de múltiples técnicas, como la rapidez de métodos constructivos con el refinamiento de búsqueda local.

3.3. International Timetabling Competition (ITC)

Las International Timetabling Competitions han sido fundamentales para estandarizar la investigación en el área y demostrar el progreso en el desarrollo de algoritmos, en esta se analizan y realizan tres problemas distintos entre ellos el Examination Timetabling Problem. La **ITC 2007** [10] estableció puntos de referencia estandarizados que permitieron la comparación de diferentes enfoques algorítmicos para el mismo problema.

Los métodos ganadores en estas competencias compartieron características como lo son:

- **Búsqueda local inteligente**: Utiliza estructuras de vecindario especializadas y estrategias con memoria, como la búsqueda tabú, para explorar eficientemente el espacio de soluciones evitando ciclos y escapes de óptimos locales.
- **Mecanismos adaptativos**: Incorporan estrategias que ajustan automáticamente sus parámetros durante la ejecución en función del comportamiento de la búsqueda, mejorando el rendimiento en diferentes instancias del problema.
- **Enfoques híbridos**: Combinan múltiples metaheurísticas, aprovechando las fortalezas de cada una para lograr un balance óptimo.

3.4. Tendencias Actuales y Mejores Resultados

Las tendencias actuales reflejan la evolución hacia métodos más sofisticados y automatizados:

- **Optimización Multi-objetivo:** Tan et al. [14] destacan enfoques que optimizan simultáneamente múltiples criterios (ej. uso de salas y preferencia de alumnos) sin combinarlos en una sola función de costo lineal.
- **Búsqueda Local Adaptativa:** Ceschia et al. [3] describen algoritmos que modifican dinámicamente sus operadores de vecindario según el progreso de la búsqueda, eliminando la necesidad de ajuste manual de parámetros.
- **Hiperheurísticas:** Burke et al. [1] proponen métodos que "eligen qué heurística aplicar" en cada momento, elevando el nivel de abstracción de la solución.

Los **algoritmos más exitosos** en la literatura reciente suelen seguir una estructura híbrida que consiste en:

- **Fase constructiva** basada en reglas heurísticas informadas por el dominio, incorporando conocimiento específico sobre conflictos estudiantiles.
- **Fase de mejora** mediante búsqueda local inteligente con operadores adaptativos que seleccionan automáticamente los movimientos más prometedores.
- **Mecanismos de diversificación** avanzados [15] para evitar estancamiento en óptimos locales, incluyendo reinicios adaptativos y estrategias de aceptación por umbrales dinámicos.

La competencia ITC 2019 [12] ha continuado esta tradición, enfocándose en problemas de programación de horarios universitarios del mundo real, impulsando el desarrollo de algoritmos más robustos y adaptativos capaces de manejar las complejidades crecientes de las instituciones educativas modernas.

La **tendencia actual** favorece algoritmos adaptativos que seleccionan automáticamente los operadores más prometedores durante la búsqueda, demostrando superior rendimiento en instancias complejas y variadas del ETP, con un creciente enfoque en la integración de aprendizaje automático y meta-aprendizaje para recomendar las mejores metaheurísticas y configuraciones de parámetros para instancias específicas.

4. Modelo Matemático

A continuación se presenta el modelo de programación lineal entera para la variante *Room-Capacitated*.

4.1. Parámetros

- p: Número de exámenes a realizar.
- m: Número de salas disponibles.
- t: Número de alumnos.
- C_k: Capacidad de la sala k, para $k = 1, \ldots, m$.
- $P_{j,a}$: 1 si el alumno a debe realizar el examen j, 0 si no.
- $T_j = \sum_{a=1}^{t} P_{j,a}$: Número de alumnos que realizan el examen j.
- w_d: Penalización por tener exámenes con separación $d - 1$ bloques.

4.2. Variables de Decisión

- $X_{i,j,k} \in \{0, 1\}$: 1 si el examen j se asigna al bloque i en la sala k, 0 si no
- $Y_{a,i,d} \in \{0, 1\}$: 1 si el alumno a tiene dos exámenes con separación d empezando en el bloque i, 0 si no

4.3. Objetivos

4.3.1. Objetivo Principal: Minimización de Bloques

El objetivo es compactar el horario. Matemáticamente, esto se modela minimizando la suma ponderada de las asignaciones por su posición temporal, lo que empuja los exámenes hacia los primeros bloques:

$$\text{mín } z_1 = \sum_{i=1}^{n} \sum_{j=1}^{p} \sum_{k=1}^{m} i \cdot X_{i,j,k}$$

4.3.2. Objetivo Secundario: Penalización por Proximidad

Minimizar la incomodidad para los estudiantes debido a exámenes consecutivos o muy cercanos, se usa como desempate:

$$\text{mín } z_2 = \sum_{a=1}^{t} \sum_{i=1}^{n-1} \sum_{d=1}^{5} w_d \cdot Y_{a,i,d}$$

Donde $w_d \in \{16, 8, 4, 2, 1\}$ son los pesos estándar para separaciones de 0 a 4 bloques libres.

4.4. Restricciones

4.4.1. Asignación Única (Hard)

$$\sum_{i=1}^{n} \sum_{k=1}^{m} X_{i,j,k} = 1 \quad \forall j \in E$$

4.4.2. Capacidad de Sala (Hard)

$$T_j \cdot X_{i,j,k} \leq C_k \quad \forall i, j, k$$

Donde T_j es el número de alumnos inscritos en el examen j y C_k la capacidad de la sala k.

4.4.3. Conflicto de Estudiantes (Hard)

$$\sum_{j=1}^{p} \sum_{k=1}^{m} P_{j,a} \cdot X_{i,j,k} \leq 1 \quad \forall i \in T, \forall a \in S$$

Un estudiante a no puede tener más de un examen en el mismo bloque i.

4.5. Espacio de Búsqueda

El tamaño del espacio de búsqueda se determina por las combinaciones posibles de asignación. Para la variable principal $X_{i,j,k}$, el espacio teórico es $2^{n \cdot p \cdot m}$. Al considerar las variables auxiliares Y, el espacio total explota a $2^{n \cdot p \cdot m + t \cdot n \cdot 5}$. Esta magnitud exponencial justifica plenamente el uso de metaheurísticas como Tabu Search en lugar de solucionadores exactos (como Simplex o Branch & Bound) para instancias reales.

5. Representación

Para el diseño de la solución, se descartó el uso de las matrices tridimensionales dispersas ($X_{i,j,k}$) propuestas en el modelo matemático, optando por una representación directa y compacta que reduce significativamente el consumo de memoria y facilita la manipulación de soluciones.

La estructura principal se define como un vector de pares ordenados de longitud N, donde N corresponde a la cantidad total de exámenes a planificar. En esta representación, el índice j del vector identifica de forma única al examen j (para $j = 0, \ldots, N-1$).

Cada elemento del vector contiene un par de valores enteros (b, s), interpretados de la siguiente manera:

- **Primer componente (b):** Representa el bloque horario asignado al examen.
- **Segundo componente (s):** Representa la sala asignada al examen.

De esta forma, existe una relación uno a uno entre la estructura de datos y la solución física: si la posición j contiene el par $(3, 5)$, implica que el examen j se rendirá en el bloque 3 dentro de la sala 5.

Índice (Examen ID)	0	1	2	...
Valor (Bloque, Sala)	(3, 1)	(0, 2)	(5, 1)	...

Figura 1: Esquema del vector de solución: El Índice representa el examen y el contenido su asignación.

5.1. Justificación de la Representación

Esta representación es apropiada para el problema por las siguientes razones:

1. **Eficiencia Espacial:** Reduce la complejidad espacial de $O(N \cdot \text{Bloques} \cdot \text{Salas})$ (del modelo binario) a $O(N)$, eliminando el almacenamiento de ceros innecesarios.
2. **Acceso Directo:** Permite consultar o modificar la asignación de cualquier examen en tiempo constante $O(1)$, lo cual es crucial para la eficiencia de los movimientos en la Búsqueda Tabú.
3. **Manejo de Restricciones:** Se complementa con estructuras auxiliares (matrices de ocupación de salas y listas de adyacencia de alumnos) que permiten verificar la factibilidad de un movimiento sin necesidad de recalcular toda la solución, agilizando la evaluación del vecindario.

6. Descripción del algoritmo

La solución implementada se basa en la metaheurística de Búsqueda Tabú (Tabu Search). Este método de búsqueda local explora el espacio de soluciones moviéndose iterativamente desde una solución actual a la mejor solución de su vecindario, utilizando una memoria de corto plazo (lista tabú) para evitar ciclos y escapar de óptimos locales.

6.1. Estructura General y Pseudocódigo

El algoritmo sigue un esquema clásico de optimización iterativa. En cada paso, para cada examen no tabú, se evalúan específicamente movimientos de avance, retroceso y espejo en el tiempo, considerando variaciones en la sala asignada.

A continuación se presenta el pseudocódigo que resume la lógica implementada en la función principal `Resolver`:

```
Entrada: Instancia del problema (Examenes, Salas, Alumnos)
Salida: Mejor asignacion encontrada

1. SolucionActual <- GenerarSolucionInicial(EstrategiaSeleccionada)
2. MejorSolucion <- SolucionActual
```

```
3. ListaTabu <- Vacia()
4. Iteracion <- 0

5. Mientras Iteracion < MaxIteraciones hacer:
6.    MejorVecino <- Nulo
7.    Para cada examen e en Examenes hacer:
8.        Si e no esta en ListaTabu entonces:
9.            // Generar vecinos especificos (Avance, Retroceso, Espejo)
10.           Vecinos <- CalcularVecinosFactibles(e, SolucionActual)
11.           Para cada v en Vecinos hacer:
12.               Delta <- CalcularDeltaPenalizacion(v, SolucionActual)
13.               Si v es mejor que MejorVecino entonces:
14.                   MejorVecino <- v
15.                   MovimientoRealizado <- e
16.
17.    Si MejorVecino existe entonces:
18.        SolucionActual <- MejorVecino
19.        ActualizarListaTabu(MovimientoRealizado)
20.        Si SolucionActual es mejor que MejorSolucion entonces:
21.            MejorSolucion <- SolucionActual
22.
23.    Iteracion <- Iteracion + 1

24. Retornar MejorSolucion
```

6.2. Inicialización

Para evaluar el impacto de la solución de partida en la convergencia del algoritmo, se implementaron y compararon tres mecanismos de generación de soluciones iniciales. Todas las estrategias garantizan la factibilidad de las restricciones duras (capacidad y conflictos) desde el primer momento.

1. **Inicialización Secuencial Simple (Naive):** Esta estrategia asigna cada examen a un bloque horario distinto de forma secuencial (el i-ésimo examen al bloque i), buscando luego la primera sala capaz de albergarlo. Aunque genera soluciones de muy baja calidad con una excesiva cantidad de bloques, esta técnica es fundamental como línea base para validar la capacidad de compactación de algoritmos más complejos [13]. Esta técnica genera soluciones iniciales de muy baja calidad (con una cantidad de bloques igual a la cantidad de exámenes), pero sirve como línea base para verificar la capacidad del algoritmo de "comprimir" el horario drásticamente.

2. **Inicialización Voraz (Greedy):** Busca generar una solución compacta desde el inicio. Recorre los exámenes secuencialmente y, para cada uno, intenta asignarlo en el primer bloque posible (comenzando desde el bloque 0) y la primera sala válida encontrada. Esta estrategia produce soluciones con una función objetivo inicial mucho mejor que la secuencial, reduciendo el trabajo necesario en la fase de optimización.

3. **Inicialización Aleatoria Factible:** Asigna cada examen a un bloque y una sala seleccionados al azar. Si la asignación aleatoria viola alguna restricción (conflicto o capacidad), se repite el intento hasta encontrar una posición válida. Este método es fundamental para explorar diversas regiones del espacio de búsqueda y evitar que el algoritmo determinista quede atrapado siempre en el mismo óptimo local al partir siempre de la misma configuración.

6.3. Definición de Movimientos y Vecindario

El núcleo de la búsqueda radica en cómo se generan los vecinos. En cada iteración, el algoritmo intenta modificar la asignación de un examen e aplicando uno de los siguientes operadores de movimiento. Para cada examen, se intentan hasta 6 variantes (3 movimientos temporales $\times$ 2 opciones de sala):

1. **Avance de Bloque ($t+1$):** Se intenta mover el examen e del bloque actual t al bloque siguiente $t+1$. Este movimiento busca compactar el horario llenando huecos vacíos posteriores.

2. **Retroceso de Bloque ($t-1$):** Se intenta mover el examen e al bloque anterior $t-1$. Este movimiento es crucial para reducir la duración total del periodo de exámenes, tratando de traer exámenes tardíos a bloques más tempranos.

3. **Movimiento Espejo ($N_{bloques}-1-t$):** Se traslada el examen e a su posición "invertida" en el horizonte temporal. Este operador fomenta la diversificación (saltos grandes) para sacar exámenes de zonas localmente congestionadas.

Para cada uno de estos movimientos temporales, se evalúan dos posibilidades espaciales: mantener la sala actual o asignar una nueva sala aleatoria. Esto permite resolver conflictos de capacidad que de otra forma impedirían el cambio de bloque.

6.4. Función de Evaluación

La calidad de una solución se mide mediante una función de costo jerárquica que guía la búsqueda hacia los objetivos del problema. El operador de comparación prioriza estrictamente la cantidad de bloques utilizados y, en caso de empate, minimiza la penalización secundaria.

$$F(S) = \langle |B|, P_{total} \rangle \tag{1}$$

Donde $|B|$ es el número de bloques. La penalización secundaria P_{total} se compone de dos factores:

1. **Factor de Posición:** Se suma el índice del bloque asignado a cada examen ($\sum bloque_j$). Este término actúa como un gradiente suave que empuja constantemente a todos los exámenes hacia el tiempo $t=0$, favoreciendo la compactación.

2. **Factor de Proximidad:** Se penaliza la cercanía de exámenes para un mismo alumno con pesos decrecientes $\{16, 8, 4, 2, 1\}$ para separaciones de 0 a 4 bloques respectivamente.

6.5. Optimización por Evaluación Incremental

Uno de los desafíos computacionales de la Búsqueda Tabú es el alto costo de evaluar todo el vecindario en cada iteración. Si se calculara la Función de Evaluación completa ($O(N)$) para cada movimiento candidato, el tiempo de ejecución crecería linealmente con el tamaño de la instancia, volviendo al algoritmo ineficiente.

Para mitigar esto, se implementó una estrategia de **Evaluación Incremental** (Delta Evaluation). Esta técnica se basa en la premisa de que el movimiento de un solo examen e no afecta la penalización de los exámenes que no están relacionados con él ni temporalmente ni por conflictos de alumnos.

En lugar de recalcular la penalización total desde cero, el algoritmo calcula exclusivamente el costo local asociado al examen e antes y después del movimiento, obteniendo la variación Δ. De esta manera, la nueva evaluación global se estima como $FE_{global} + \Delta$. Esta optimización reduce drásticamente la complejidad computacional de la exploración del vecindario.

6.6. Manejo de la Lista Tabú

La lista tabú se implementa como una cola FIFO (`deque`) de tamaño dinámico, proporcional a la instancia ($N/2$). Cuando un examen es movido, su identificador ingresa a la lista y se le "prohíbe" ser seleccionado para un nuevo movimiento durante un número de iteraciones igual al tamaño de la lista. Esto previene que el algoritmo revierta inmediatamente un cambio (ciclado), forzándolo a explorar nuevas configuraciones.

7. Experimentos

7.1. Objetivos de la Experimentación

El diseño experimental tiene como objetivo evaluar el desempeño y la robustez del algoritmo de Búsqueda Tabú bajo diferentes condiciones iniciales y horizontes de ejecución. Se busca responder a dos preguntas fundamentales:

1. **Impacto de la Inicialización:** ¿Cuánto influye la calidad de la solución inicial (Greedy vs. Aleatoria vs. Naive) en la calidad final tras aplicar la metaheurística?
2. **Análisis de Convergencia:** ¿Es suficiente un número bajo de iteraciones (60) para encontrar óptimos locales de calidad, o se requiere una exploración profunda (1.000 iteraciones) para salir de estancamientos en instancias complejas?

7.2. Instancias de Prueba

Se utilizaron 8 instancias estandarizadas (`i1.in` a `i8.in`). Estas varían en densidad de conflictos, número de alumnos y disponibilidad de salas, ofreciendo un espectro representativo de dificultad.

Las instancias se encuentran disponibles en el siguiente repositorio:
`https://github.com/benjacha/ETP-Tabu-Search/tree/main/codigo/Instancias`

- `i1.in`: Número de exámenes: 607, Número de Salas: 7
- `i2.in`: Número de exámenes: 870, Número de Salas: 49
- `i3.in`: Número de exámenes: 934, Número de Salas: 48
- `i4.in`: Número de exámenes: 273, Número de Salas: 1
- `i5.in`: Número de exámenes: 1017, Número de Salas: 3
- `i6.in`: Número de exámenes: 242, Número de Salas: 8
- `i7.in`: Número de exámenes: 1096, Número de Salas: 15
- `i8.in`: Número de exámenes: 587, Número de Salas: 8

7.3. Entorno de Hardware y Software

Los experimentos fueron ejecutados en un entorno controlado con las siguientes características:

- **Lenguaje:** C++ (Estándar C++11), utilizando la librería `chrono` para mediciones de tiempo de alta precisión.
- **Compilador:** G++ sin banderas de optimización agresivas para asegurar la estabilidad de la medición.
- **Equipo:** AMD Ryzen 7 9700X 8-Core, 16GB RAM.

7.4. Configuración de Parámetros

Para asegurar la reproducibilidad, se definieron los siguientes parámetros fijos y variables:

- **Función Objetivo:** Pesos de penalización $w_d = \{16, 8, 4, 2, 1\}$ para distancias de 0 a 4 bloques.
- **Lista Tabú:** Dinámica, calculada como máx$(1, NE/2)$.
- **Variables de Control (Iteraciones):** Se definieron tres escenarios de profundidad de búsqueda para evaluar la convergencia:
  - **Corto Plazo:** 60 iteraciones.
  - **Mediano Plazo:** 500 iteraciones.
  - **Largo Plazo:** 1.000 iteraciones.

donde cada una de las 8 instancias fue resuelta utilizando las 3 estrategias de inicialización (Greedy, Naive, Aleatoria) y sometida a los 3 niveles de profundidad de iteración. Se registraron métricas de Tiempo de Inicialización (T_{init}), Tiempo de Búsqueda (T_{search}), Penalización Final y Cantidad de Bloques utilizados. Los resultados se almacenaron en archivos de registro diferenciados (`Greedy.txt`, `Naive.txt`, `Aleatorio.txt`) para su posterior análisis.

8. Resultados

Esta sección presenta el análisis de rendimiento del algoritmo de Búsqueda Tabú, evaluando la evolución de la calidad de la solución y el uso de recursos a través de tres horizontes de ejecución: corto plazo (60 iteraciones), mediano plazo (500 iteraciones) y largo plazo (1000 iteraciones).

8.1. Evolución de la Minimización de Recursos (Bloques)

El primer desafío del algoritmo es compactar el horario para utilizar la menor cantidad de bloques posibles.

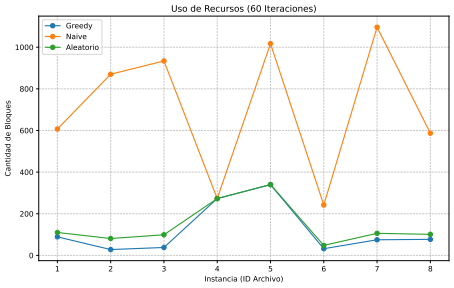

Figura 2: Bloques a 60 Iteraciones.

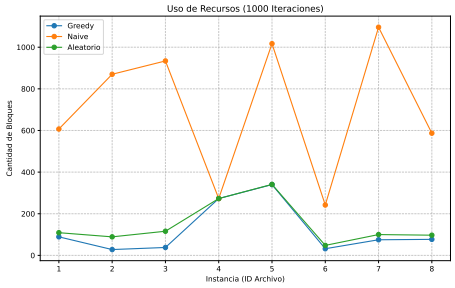

Figura 3: Bloques a 1000 Iteraciones.

La comparativa entre la Figura 2 y la Figura 3 revela una dinámica interesante:

- **Convergencia Rápida de Greedy y Aleatorio:** Ya a las 60 iteraciones, las estrategias Greedy (azul) y Aleatoria (verde) logran niveles de compactación muy altos. El aumento a 1000 iteraciones sólo refina marginalmente este aspecto en las instancias más difíciles.

- **Estancamiento de Naive:** La estrategia Naive (naranja) comienza con un exceso masivo de bloques. Aunque logra reducir su uso al pasar de 60 a 1000 iteraciones (note la pendiente descendente en las instancias centrales), **no logra converger** a la calidad de las otras estrategias, quedándose atrapada en óptimos locales con cientos de bloques extra.

8.2. Evolución de la Calidad de Solución (Penalización)

Una vez estabilizado el número de bloques, el algoritmo se enfoca en minimizar la penalización por proximidad de exámenes.

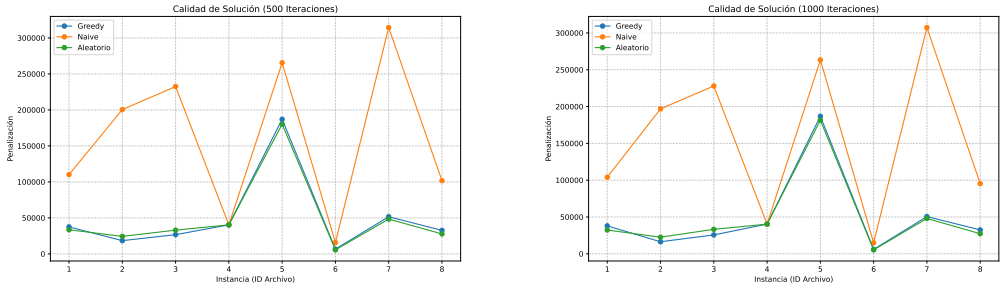

Figura 4: Comparativa de Penalización: Mediano Plazo (500 iters) vs Largo Plazo (1000 iters).

Las Figuras 4 muestran el refinamiento fino de la solución:

- **Greedy (Azul):** Se mantiene como la línea base de mejor calidad (más baja) consistentemente. Su estabilidad entre 500 y 1000 iteraciones sugiere que encuentra buenas soluciones muy rápido.
- **Recuperación de Aleatorio (Verde):** Es notable cómo la estrategia Aleatoria, que a las 60 iteraciones tenía un desempeño variable, para las 500 iteraciones ya ha logrado *casi* empatar con Greedy en varias instancias (ej. instancias 1, 4, 6). Esto valida la capacidad de la Búsqueda Tabú para corregir una mala inicialización si se le da suficiente tiempo.
- **Naive (Naranja):** Aunque mejora, sigue siendo la peor opción por un margen amplio, validando que una mala estructura inicial impone un "techo.ª la calidad máxima alcanzable.

8.3. Análisis de Costo Computacional

Finalmente, se contrasta el costo temporal de las estrategias para determinar si la complejidad de la inicialización Greedy justifica su uso.

La Figura 5 ilustra los tiempos totales de ejecución. Para un análisis más detallado, la Tabla 1 desglosa el impacto porcentual de la fase de inicialización sobre el tiempo total promedio.

Método	T. Inicial (ms)	T. Total (ms)	% del Total
Greedy	1769.66	115052.24	1.54 %
Naive	0.26	218522.14	0.00 %
Aleatorio	164.78	188582.51	0.09 %

Cuadro 1: Comparativa de tiempos promedio a 1000 iteraciones.

Los resultados son contundentes y revelan un fenómeno interesante:

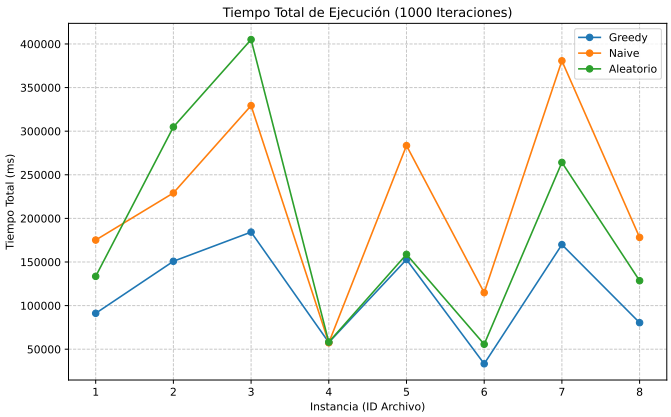

Figura 5: Tiempo Total de Ejecución (Inicialización + 1000 Iteraciones).

1. **Costo despreciable:** Aunque la inicialización **Greedy** es la más lenta en términos absolutos ($\approx$ 1,77 segundos), este costo representa apenas un **1.54 %** del tiempo total de ejecución, lo cual es marginal.

2. **Eficiencia Global:** Sorprendentemente, la estrategia Greedy resultó tener el **menor tiempo total** de ejecución promedio ($\approx$ 115 segundos), superando ampliamente a la estrategia Naive ($\approx$ 218 segundos) y a la Aleatoria ($\approx$ 188 segundos).

Esto sugiere que partir de una buena solución no sólo mejora la calidad final, sino que también **acelera el proceso de búsqueda**, posiblemente porque el algoritmo pierde menos tiempo evaluando movimientos en vecindarios de baja calidad o reparando infactibilidades complejas. En conclusión, la estrategia Greedy es dominante en todos los aspectos: calidad, uso de recursos y eficiencia temporal.

9. Conclusiones

El presente estudio abordó la resolución del *Examination Timetabling Problem* en su variante *Room-Capacitated* mediante una metaheurística de Búsqueda Tabú. Tras analizar el comportamiento del algoritmo bajo tres estrategias de inicialización y diferentes horizontes de búsqueda, se presentan las siguientes conclusiones:

9.1. Adecuación de la Propuesta

La implementación de la Búsqueda Tabú con representación vectorial y operadores de vecindario especializados (movimientos espejo y evaluación incremental) demostró ser altamente efectiva para este problema NP-Duro. El algoritmo logró no solo encontrar soluciones factibles para todas las instancias, sino también reducir drásticamente el número de bloques horarios utilizados, validando la hipótesis de que una exploración local inteligente puede superar las limitaciones de los métodos constructivos puros.

9.2. Ventajas y Desventajas

Basado en la evidencia experimental, se identifican las siguientes fortalezas y debilidades de la propuesta:

Ventajas:

- **Eficiencia de la Inicialización Greedy:** Los resultados confirman que la estrategia Greedy es dominante. A pesar de tener un costo inicial de 1.7 segundos (frente a los milisegundos de Naive), esta inversión permite reducir el tiempo total de ejecución en casi un 50 % (115s vs 218s) y alcanzar calidades de solución inalcanzables para las otras estrategias.
- **Capacidad de Recuperación:** La estrategia de inicialización Aleatoria demostró que el algoritmo tiene una robusta capacidad de mejora, logrando equiparar la calidad de Greedy en instancias medianas tras 500 iteraciones, lo que habla bien de la capacidad de diversificación de los movimientos implementados.

Desventajas:

- **Dependencia de la Estructura Inicial:** En las instancias más complejas (i7, i8), el algoritmo no logró corregir completamente las deficiencias de una mala inicialización (Naive) dentro del límite de 1000 iteraciones, sugiriendo que la Búsqueda Tabú por sí sola puede tener dificultades para reconstruirüna solución estructuralmente deficiente.
- **Escalabilidad del Vecindario:** Aunque la evaluación incremental mitiga el costo, el análisis exhaustivo del vecindario sigue siendo el cuello de botella, consumiendo el 98 % del tiempo de cómputo.

9.3. Trabajo Futuro

Para futuras investigaciones, se propone explorar mecanismos de **Búsqueda Tabú Reactiva**, donde el tamaño de la lista tabú se ajuste dinámicamente en función de la repetición de soluciones, para evitar el estancamiento observado en la estrategia Naive. Asimismo, la integración de una fase de **Reencadenamiento de Trayectorias (Path Relinking)** podría permitir combinar las mejores características de soluciones Greedy y Aleatorias, robusteciendo la búsqueda en instancias de gran escala.

Referencias

[1] E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64(12):1695–1724, 2013.

[2] E. K. Burke and J. P. Newall. Enhancing timetable solutions with local search methods. In *Practice and Theory of Automated Timetabling IV*, pages 195–206, 2003.

[3] S. Ceschia, L. Di Gaspero, and A. Schaerf. Educational timetabling: Problems, benchmarks, and state-of-the-art results. *European Journal of Operational Research*, 304(3):909–927, 2023.

[4] A. J. Cole. The preparation of examination time-tables using a small-store computer. *The Computer Journal*, 7(2):117–121, 1964.

[5] D. de Werra. Construction of school timetables by flow methods. Technical Report 14P-11, University of Waterloo, Department of Management Sciences, 1970.

[6] W. F. Erben. A grouping genetic algorithm for graph coloring and exam timetabling. In *Practice and Theory of Automated Timetabling III*, pages 132–156, 2001.

[7] M. R. Garey and D. S. Johnson. Computers and intractability a guide to the theory of np-completeness. 1979.

[8] C. C. Gotlieb. The construction of class-teacher time-tables. *Information Processing, Proceedings of IFIP Congress*, pages 73–77, 1963.

[9] G. Laporte and S. Desroches. Examination timetabling by computer. *Computers & Operations Research*, 11(4):351–360, 1984.

[10] B. McCollum, A. Schaerf, B. Paechter, P. Mcmullan, R. Lewis, A. J. Parkes, L. Di Gaspero, R. Qu, and E. K. Burke. Setting the research agenda in automated timetabling: The second international timetabling competition. *INFORMS Journal on Computing*, 22(1):120–130, 2010.

[11] N. K. Mehta. The application of a graph coloring method to an examination scheduling problem. *Interfaces*, 11(3):57–64, 1981.

[12] T. Muller, H. Rudová, and Z. Mullerová. Real-world university course timetabling at the international timetabling competition 2019. *Journal of Scheduling*, 2024. Published online: 12 April 2024.

[13] R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee. A survey of examination timetabling. *Operations Research*, 34(1):193–223, 2009.

[14] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar. A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, 149:113–135, 2021.

[15] J. S. Tan, S. L. Goh, G. Kendall, and N. R. Sabar. A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, 149:113–135, 2021.

[16] J. M. Thompson and K. A. Dowsland. A robust simulated annealing based examination timetabling system. *Computers & Operations Research*, 25(7):637–648, 1998.

[17] D. J. A. Welsh and M. B. Powell. An upper bound for the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, 10(1):85–86, 1967.

[18] G. M. White and B. S. Xie. Examination timetables and tabu search with longer-term memory. In *Practice and Theory of Automated Timetabling II*, pages 85–103, 1998.

[19] D. C. Wood. A system for computing university examination timetables. *The Computer Journal*, 11(1):41–47, 1968.