Inteligencia Artificial

Estado del Arte: Examination Timetabling Problem

Gonzalo Fernández

January 10, 2021

Evaluación

Resumen (5%):	_____
Introducción (5%):	_____
Definición del Problema (10%):	_____
Estado del Arte (35%):	_____
Modelo Matemático (20%):	_____
Conclusiones (20%):	_____
Bibliografía (5%):	_____
Nota Final (100%):	_____

Abstract

El Examination Timetabling Problem (ETP) es un problema que consta de asignar horarios a un conjunto de exámenes bajo ciertas restricciones, la calidad de esta asignación tiene un gran impacto en el desempeño de los estudiantes a la hora de rendir sus exámenes, lo que hace de este un problema sumamente importante y estudiado a lo largo de la historia, pero que a su vez, se han identificado variantes según las necesidades de cada institución. Este documento plantea el problema desde sus orígenes, sus distintas soluciones propuestas desde métodos exactos hasta hyperheurísticas, así como también un modelo matemático que busca solucionarlo y algunas conclusiones respecto al trabajo actual y futuro sobre el cual se deba guiar el resto de su investigación.

1 Introducción

Una de las herramientas más importantes para medir el conocimiento adquirido de un estudiante en cualquier institución educativa son los exámenes, de estos depende la aprobación de un curso y por lo tanto, mientras mejores condiciones se brinden para su rendición, mejores resultados pueden esperarse [22]. Estas condiciones se dan por diversos factores, entre ellos están los horarios en que los exámenes son asignados pues un estudiante no puede rendir 2 exámenes al mismo tiempo y a su vez, necesita un tiempo de preparación entre exámenes. Asignar estos horarios es una tarea recurrente que consume mucho tiempo y se realiza comúnmente ajustando a la actualidad alguna asignación de exámenes previa, o bien con apoyo de un sistema de administración que no automatiza del todo el proceso [23], no obstante ninguno de estos enfoques se sostiene

ante la constante variación de cursos, estudiantes y otros factores a lo largo del tiempo, lo que hace de este un problema complejo y con mucha importancia dentro de una institución educativa.

A este problema se le conoce como Examination Timetabling Problem (ETP), el cual se define como "la calendarización de exámenes de un conjunto de cursos universitarios, evitando traslapes de cursos con estudiantes en común, y distribuyendo los exámenes de la mejor forma posible" [31]. El ETP forma parte de la familia de problemas NP-completos [37], lo cual indica que no es posible resolver el ETP en un tiempo razonable. Por otro lado, se suma a esto que cada institución educativa cuenta con distintos requerimientos a la hora de asignar los horarios de sus exámenes, de estos requerimientos se distinguen 2 tipos de restricciones al problema [27], llamadas restricciones blandas (que deben satisfacerse lo mayor posible) y restricciones duras (que siempre deben ser satisfechas). Las restricciones duras deben ser satisfechas siempre, las blandas en cambio, deben satisfacerse lo mayor posible con el fin de lograr una mejor solución. En este documento se trabajará una variación donde se cuenta con un conjunto de exámenes y un conjunto de estudiantes que debe rendirlo, el objetivo será buscar una asignación de horarios tal que ningún estudiante tenga 2 exámenes al mismo tiempo y que a su vez, se minimice la cantidad de horarios necesarios para llevar a cabo los exámenes, un segundo objetivo menos importante será distribuir los exámenes lo mejor posible, de manera tal que los estudiantes cuenten con tiempo para estudiar entre cada examen, no obstante, éste último no es un objetivo más importante que el de disminuir la cantidad de horarios.

La NP-completitud del problema, sus variadas restricciones duras (ver Tabla 1) y blandas (ver Tabla 2) y su enorme impacto tanto en el área de la educación como en problemas de horarios en general, hacen de este un problema sumamente interesante e importante de resolver.

El presente documento se estructura de la siguiente manera: primero se definirá el problema del ETP, se hablará sobre sus variantes y se detallará la variante a trabajar en la sección (2); luego se dará una reseña histórica del problema a través de su estado del arte en la sección (3); luego se planteará un modelo matemático que permita resolver la variante del ETP que se está estudiando en la sección (4); finalmente, se entregan algunas conclusiones sobre los contenidos tratados e ideas de trabajo a futuro sobre este problema en la sección (9).

2 Definición del Problema

El ETP es un Timetabling Problem, y como tal se compone de 4 parámetros principales: un conjunto finito de tiempos, un conjunto finito de recursos, un conjunto finito de reuniones y un conjunto finito de restricciones [27], donde en su forma más general los tiempos son los horarios en que se toman los exámenes, los recursos serían los estudiantes y las salas en donde estos se llevan a cabo, las reuniones serían los exámenes que cuentan con estudiantes que deben rendirlos y el conjunto de restricciones sería que un estudiante no puede tener asignado 2 o más exámenes al mismo tiempo, entre otras, buscando así generar una asignación de horarios a los exámenes tal que se satisfagan lo mejor posible las restricciones. Ésta definición general permite abordar el ETP como un problema de búsqueda ya que bastaría con satisfacer las restricciones duras (ver Tabla 1) del problema como lo sería que un estudiante no tenga 2 exámenes al mismo tiempo, no obstante, las necesidades de las instituciones educacionales van requieren considerar un conjunto de restricciones blandas(ver Tabla 2), las cuales hacen de este problema uno de optimización por sobre uno de búsqueda.

Las variaciones a este problema corresponden a la realidad de cada institución educacional en la que el autor se concentra a la hora de estudiarlo, entre estas se encuentran la implementación de diversas versiones para el mismo examen, de modo tal que un estudiante no tenga

2 exámenes al mismo tiempo y así, mejorar la distribución de los mismos [41], otra variación considera un subconjunto de salas para cada examen donde éste puede ser rendido y consultar a los estudiantes sus preferencias para calendarizar evaluaciones, sin que ésto les garantice que serán calendarizadas en esos horarios [16]. Todos estos ajustes hacen de este un problema lleno de variaciones en la práctica (no así en los benchmarks donde se ocupan datasets específicos como por ejemplo el ITC2007 [13]), más no hay una variación particular en la que la literatura se concentre.

Al ser un Timetabling Problem, todos los problemas de esta familia se parecen en cierta medida al ETP, pero los que más parecidos son el Timetabling Course Problem que consta de asignar un conjunto de cursos a un conjunto finito de horarios y salas; y el School Timetabling Problem que consta de asignar un conjunto de distintos recursos (horarios, profesores, estudiantes, salas, etc) a un conjunto de eventos [35], ambos han sido ampliamente estudiados y sus descubrimientos han aportado mútuamente tanto al ETP como a otros Timetabling Problems.

Para efectos de este documento, se trabajará la siguiente variante: Se cuenta con un conjunto de exámenes y un conjunto de estudiantes que debe rendirlos, el objetivo será asignar la menor cantidad posible de horarios de exámenes tal que ningún estudiante tenga 2 exámenes al mismo tiempo y, en el camino, intentar distribuir los exámenes lo más posible para así aumentar el tiempo entre exámenes de cada estudiante, no obstante, el objetivo principal será disminuir la cantidad de horarios con los cuales llevar a cabo los exámenes.

3 Estado del Arte

Las primeras apariciones del ETP en la literatura datan del 1964, donde Broder [2] da una formalización del problema debido a la necesidad de asignar los horarios de los exámenes finales de los cursos de una universidad y propone un modelo matemático para el mismo, también propone un algoritmo aleatorizado de tipo Monte Carlo para encontrar una solución que minimizara la cantidad de conflictos entre estudiantes y sus exámenes, esta solución encontrada si bien no garantizaba ser la mejor, usualmente sería óptima. Ese mismo año Cole [8] propone una matriz de conflictos entre exámenes y una serie de restricciones particulares, junto con un algoritmo "largest degree first: fill from top" para generar una solución que también sería cercana a la óptima, además su método puede ejecutarse en computadores con poca capacidad de almacenamiento.

En 1966, Peck [26] propone un algoritmo de particionamiento y coloreo de grafos el cual sólo se encarga de asignar horarios lo mejor posible sin garantizar que un estudiante no tendrá 2 exámenes al mismo tiempo ni considerar otras restricciones. Ya en 1968, Wood [40] ideó un algoritmo que fue implementado en su universidad, éste también se basó en la minimización de una matriz de conflictos como la de Cole [8], no obstante, la calidad de la solución que éste entregaba se medía según los criterios de aquella universidad, por otro lado, en caso de que 2 cursos empatasen en su asignación, Wood implementó una estrategia "look ahead" para elegir qué examen sería asignado en ese horario, de modo tal que esta decisión no generara problemas en las próximas asignaciones y en caso de no poder entregar una asignación completa, bastaba con asignar manualmente los cursos que no quedaron asignados y echar a andar de nuevo el algoritmo, no obstante, dado el crecimiento exponencial de memoria que sufre al aumentar el tamaño de la entrada, éste algoritmo puede usarse con todos los exámenes de una universidad al mismo tiempo, requiere separar su uso en unidades lógicas como por ejemplo las carreras o departamentos.

Más adelante, Desroches et. al. [12] presentan en 1978 el algoritmo HOREX, el cual es descrito en sus distintas etapas donde usa desde algoritmos de coloreo de grafos hasta progra-

mación entera para resolver el problema, resultando ser un algoritmo superior a los anteriores para una cantidad pequeña de horarios disponibles para realizar la asignación, pero no estando a la altura para escenarios más grandes. Un año más tarde White y Chan [38] desarrollaron un algoritmo derivado de HOREX para aplicarlo en su universidad, buscando mejorar la satisfacción de restricciones blandas, no obstante, su buen desempeño también se veía restringido a cantidades pequeñas de horarios para realizar la asignación.

En 1981 Mehta [19] hizo un método (también para su universidad) cuyo fin era minimizar los conflictos y los casos en que 3 exámenes seguidos eran asignados, utilizando un enfoque de coloreo de grafos y una rutina de "compresión" para computar los conflictos que podía generar una asignación y escoger qué horario eliminar de ésta para volver a distribuir los exámenes, sus resultados fueron un gran aporte para demostrar que los "saturation degree methods" son las mejores técnicas de coloreo de grafos, respaldando a su vez las técnicas previas y marcando un punto en que encontrar soluciones libres de conflictos para este problema ya era una realidad.

Ya en 1984, Laporte et. al. [16] presentaron el sistema HORHEC donde por primera vez se planteó la idea del "costo de aversión" basado en una lista de preferencias de los estudiantes y el "costo de proximidad" basado en la cantidad de horarios entre los cuales son asignados 2 exámenes, éstas restricciones blandas se sumarían a la restricción dura de no asignar más de 1 examen al mismo tiempo para 1 mismo estudiante (entre otras). La estrategia que plantearon se encargaba de probar la generación de asignaciones de horarios buscando minimizar la función objetivo planteada, aumentando la cantidad máxima de horarios a usar según un mínimo y máximo indicados por el usuario, ésta estrategia podría verse como un un polynomially bounded backtracking que permite "corregir errores anteriores" de soluciones previas [6], siendo así de los primeros algoritmos en no presentar una heurística de coloreo de grafos por detrás como los anteriores para resolver el problema.

Pese a estos avances, la mayoría de las soluciones propuestas se concentraban en el contexto de la institución educacional particular de los autores, sin que éstas llegasen a ser usadas en otras instituciones con realidades distintas, tampoco se realizaron comparaciones entre éstas y otros enfoques alternativos, en el fondo, la mayoría de autores no estaban al tanto de la existencia de otras soluciones publicadas debido a un muy desorganizado estado del arte [6] hasta que en 1996, Carter et. al. [7] estudiaron 5 de las estrategias previamente mencionadas concluyendo que ninguna de ellas era sobresaliente para todos los casos en los que fueron probadas, no obstante, introdujeron al estudio de este problema un conjunto de 13 casos de prueba para así realizar benchmarks y dar un primer paso a estandarizar las comparaciones [27]. Desde entonces, es posible comenzar a organizar de mejor manera las soluciones que se han propuesto.

Técnicas Basadas en Satisfacción de Restricciones

Debido a la flexibilidad con la que estas técnicas se podían aplicar al ETP, se propusieron métodos de satisfacción de restricciones para resolver el problema, no obstante, debido al crecimiento exponencial de memoria que requieren, en su mayoría sólo podían generar soluciones parciales, que luego requerían ser completadas o mejoradas por otros procedimientos [11, 20, 1], aunque también habían casos en que se ajustaban perfectamente a las necesidades de la universidad donde se implementaban [34]. Estas técnicas fueron respaldadas por grandes avances en lenguajes de programación de restricciones, pero no lograron ser un enfoque superior a la hora de aplicarse al problema de manera general, si no más bien, se ajustaban a las realidades de la universidad correspondiente, aunque es importante destacar que para ciertos casos de prueba, logran los mejores resultados [27].

Técnicas de Búsqueda Local

Estas técnicas se guiaban por el uso de una función objetivo que permitía evaluar la calidad de las soluciones generadas y permitían manejar fácilmente las diferentes restricciones de las variantes del ETP. Entre estas podemos encontrar Tabu Search, donde se implementaron funciones objetivo adaptativas [14], búsquedas por fases (en cada fase de búsqueda de la solución se añaden restricciones no consideradas antes) y asignación de prioridades a las restricciones [25].

Otra técnica usada es Simulated Annealing, la cual por sí sola no generaba soluciones mejores que otros métodos, no obstante, entonces se investigó el usarlo de forma híbrida con backtracking [36], obteniendo resultados mucho mejores e incentivando así el uso de enfoques que usaran esta técnica de forma híbrida. Merlot [20] reforzó esta idea y realizó un híbrido con técnicas de programación de restricciones, logrando así los mejores resultados conocidos hasta la época y marcando un precedente para los enfoques híbridos a la hora de atacar el ETP.

Otros autores enfocaron su investigación en modificar la estructura de los vecindarios de búsqueda del espacio de soluciones a la hora de abordar el ETP , no obstante, el enorme cómputo que requieren las deja muy por detrás de las otras soluciones [27]

Algoritmos Basados en Población

Entre los algoritmos basados en población se trabajaron algoritmos de hormigas para resolver el modelo del ETP planteado como un problema de coloreo de grafos [10], sin considerar restricciones blandas ni obtener resultados significativos en comparación con los demás. También se trabajaron algoritmos genéticos y meméticos con el fin de determinar las mejores heurísticas para resolver el problema [15, 30, 33], así como también, encontrar la mejor representación del problema con el fin de mejorar los resultados, no obstante, las soluciones por éste ámbito no destacan tanto como las otras. También se trabajó con Artificial Immune Algorithms [18], los cuales presentaron los mejores resultados en ciertos casos de prueba, demostrando así el potencial de estos algoritmos para trabajos futuros con ellos [27].

Técnicas Multi-Criterio

Con el fin de cuantificar la calidad de una solución bajo un criterio distinto al de calcular una función objetivo, se plantearon técnicas multi-criterio las cuales [8, 5], si bien entregaron buenos resultados en ciertos casos, su aporte al problema fue brindar flexibilidad para determinar la importancia individual de cada restricción con el fin de obtener soluciones deseadas más allá de las que minimizaban una función objetivo particular.

Técnicas Basadas en Grafos

Distintos autores introdujeron mejoras en las heurísticas de coloreo de grafos para atacar el problema [4, 32] e integrando técnicas de redes neuronales para generar mejores soluciones [9], estas técnicas híbridas generaron buenos resultados usando los datasets actuales, por lo que más adelante dan pie al uso de hyperheurísticas más adelante.

Técnicas de Descomposición y Clustering

También se trabajó la idea de descomponer el problema en pequeños subproblemas [17, 3] para encontrar soluciones óptimas, no obstante, el gran problema de este enfoque es que no se pueden evaluar restricciones blandas cuando un problema es descompuesto, por lo que no generó mucho impacto pese a lograr uno de los mejores resultados para 1 sólo caso de prueba [28]

	Principales restricciones duras
1.	Los exámenes no deben compartir recursos (por ejemplo, estudiantes) de manera simultánea
2.	Los recursos deben ser suficientes (es decir, el número de estudiantes asignados a una sala debe ser menor o igual a la capacidad de la misma, así como también, deben haber suficientes salas para todos los exámenes)

Table 1: Principales restricciones duras en ETP [27]

	Principales restricciones blandas
1.	Distribuir los exámenes que entran en conflicto de la manera más uniforme posible, o bien, no en x horarios consecutivos o días
2.	Grupos de exámenes que requieren ser tomados al mismo tienpo, en el mismo día o en el mismo lugar.
3.	Exámenes que deben ser consecutivos.
4.	Calendarizar todos los exámenes, o los más largos, lo antes posible.
5.	Un ordenamiento (precedencia) de exámenes necesita ser satisfecha.
6.	Número limitado de estudiantes y / ó exámenes en cierto horario.
7.	Requerimientos de tiempo (por ejemplo, realizar (o no) exámenes en ciertos horarios).
8.	Exámenes que entran en conflicto en el mismo día deben ser asignados en horarios cercanos.
9.	Exámenes que pueden separarse sobre distintos lugares.
10.	Sólo exámenes del mismo largo pueden ser asignados en la misma sala.
11.	Requerimientos de recursos (por ejemplo, salas con facilidades específicas para el examen).

Table 2: Principales restricciones blandas en ETP [27]

Hyperheurísticas

Dada la dificultad de encontrar meta-heurísticas que se ajusten a las variaciones que puede un ETP particular tanto en sus restricciones blandas 2 y duras 1, por lo que se desarrollaron hyperheurísticas con resultados prometedores para fomentar futuras investigaciones usando esta técnica [29], dando como resultado que para el 2019 se planteara una hyperheurística basada en el algoritmo Great Deluge, cuyos resultados fueron sobresalientes en todos los casos de prueba usados en comparación a otras técnicas [22], posicionándose como una de las mejores técnicas hasta el momento y sentando las hyperheurísticas como las bases para trabajos futuros.

4 Modelo Matemático

Con el fin de resolver la variante del ETP a trabajar, se propone un modelo matemático que considera un costo de proximidad entre exámenes [16] y que busca minimizar la cantidad de horarios [39]

Parámetros:

- E: Cantidad de exámenes, con $E \geq 1$
- L: Cantidad máxima de horarios, con $L \geq 1$
- e_i: Cantidad de estudiantes que deben rendir el examen i, con $1 \leq i \leq E$
- C_{ij}: Cantidad de estudiantes que deben rendir el examen i y j, con $(1 \leq i < j \leq E)$

- w_s: Costo de proximidad entre exámenes separados por s horarios

  Se define según Laporte [16] como: $w_1 = 16, w_2 = 8, w_3 = 4, w_4 = 2, w_5 = 1$

Variables:

- $x_{il} = \begin{cases} 1 & \text{si el examen } i \text{ es asignado en el horario } l, \text{ con , con } 1 \leq i \leq E \text{ y } 1 \leq l \leq L \\ 0 & \text{en cualquier otro caso} \end{cases}$
- $t_l = \begin{cases} 1 & \text{si el horario } l \text{ es usado en la solución, con } 1 \leq l \leq L \\ 0 & \text{en cualquier otro caso} \end{cases}$
- M: Un número arbitrariamente grande ($M > C_{ij}; i, j = 1, \ldots, E$, con $i < j$)

Función Objetivo:

$$\min F : \sum_{l=1}^{L} \sum_{s=1}^{5} \sum_{i=1}^{E} \sum_{j=1}^{E} (x_{il} x_{j,l-s} + x_{il} x_{j,l+s}) C_{ij} w_s + \sum_{l=1}^{L} t_l \tag{1}$$

Restricciones:

$$x_{il} + x_{jl} \leq 2 - \frac{C_{ij}}{M} \quad \forall i, j : i < j \wedge i, j \in \{1, \ldots, E\} \wedge l \in \{1, \ldots, L\} \tag{2}$$

$$\sum_{l=1}^{L} x_{il} t_l = 1 \quad \forall i \in \{1, \ldots, E\} \tag{3}$$

$$\sum_{i=1}^{E} \sum_{l=1}^{L} x_{il} t_l \leq L \tag{4}$$

Naturaleza de las Variables:

$$x_{ij} \in \{1, 0\}$$

$$t_l \in \{1, 0\}$$

$$M \in \mathbb{N}$$

Detalles del Modelo:

- Dado que la minimización de los horarios a usar para la asignación es más importante que minimizar el costo de proximidad entre exámenes, el modelo plantea el parámetro L para asignar el máximo de horarios que la institución educacional desea usar, el valor común para este parámetro debería ser E poniéndose en el caso de que cada examen se asigne en un horario distinto y continuo.
- La ecuación (1) es la función objetivo que mide la calidad de la solución y que el modelo busca minimizar, la primera parte de la sumatoria cuantifica el costo de proximidad entre exámenes mientras que la segunda parte cuantifica el costo asociado a la cantidad de horarios que se utilizarán.
- La ecuación (2) busca que si un estudiante debe rendir los exámenes i y j, entonces esos exámenes no pueden ser asignados en el mismo horario, la variable M está para que si $C_{ij} > 0$ entonces la restricción sea equivalente a $x_{il} + x_{jl} \leq 1$ [16].
- La ecuación (3) se asegura que un examen pertenezca a sólo 1 horario de los que se usarán en la solución (por eso considera t_l).

- La ecuación (4) se asegura que no se asignen más de L horarios (el máximo indicado por la institución educacional), para así evitar que se asigne una cantidad enorme de horarios con el fin de minimizar el costo de proximidad entre exámenes.
- El espacio de búsqueda de este modelo matemático equivale a todas las combinaciones posibles de valores que se pueden realizar con las variables, lo que se expresa como $2^{(E+1)L}$ y que determina el crecimiento exponencial del espacio de búsqueda en función de la cantidad de exámenes E a asignar y la cantidad máxima de horarios L que la institución educacional determina usar.

5 Representación

Representación de **soluciones** (arreglos, matrices, etc.). En caso de técnicas completas indicar variables y dominios. Incluir justificación y ejemplos para mayor claridad.

6 Descripción del Algoritmo

Para llevar a cabo la resolución del problema, se implementó un algoritmo de búsqueda completa con Forward Checking [21]. El algoritmo consiste en fijar una cantidad de horarios inicial a asignar entre el total de exámenes y, en caso de detectar que no es posible generar una calendarización válida con la cantidad de horarios fijadas, repetir el proceso considerando un horario extra. Cada vez que se asigna un horario a un examen, se lleva a cabo un filtro en los posibles horarios a asignar de los exámenes posteriores, es decir, se aplica arco-consistencia sin propagación entre la solución parcial y los dominios de los exámenes restantes. Si durante éste proceso un examen queda sin horarios disponibles para asignarle (vale decir, su dominio queda vacío), entonces se vuelven a agregar los horarios filtrados previamente y se intenta asignar el siguiente horario disponible al examen anterior, repitiendo el proceso. Cuando la cantidad de horarios a asignar es suficiente como para encontrar una calendarización válida, el algoritmo toma esa cantidad como la mínima necesaria para resolver la instancia y, sobre ella, intenta minimizar la penalización promedio asociada al esparcimiento de los exámenes asignados a cada estudiante.

Para facilitar las cosas, el algoritmo considera que el máximo de horarios siempre será la cantidad total de exámenes, pues en el peor de los casos, siempre será posible generar una calendarización válida asignando un horario distinto a cada examen (ejemplo, una instancia donde al menos 1 estudiante debiese rendir todos los exámenes). Además, se considera que el mínimo de horarios para comenzar la búsqueda completa debe ser la cantidad máxima de exámenes a rendir por algún estudiante, pues éste requerirá como mínimo 1 horario distinto por cada uno de sus exámenes. Finalmente, para la ejecución del algoritmo, se requiere contar con los siguientes parámetros y variables:

- **Parámetros**:
  - $E \leftarrow$ Cantidad de Exámenes
  - $L \leftarrow$ Cantidad máxima de exámenes que un alumno tiene asignado en la instancia, lo que a su vez, se considera la cantidad mínima de horarios para partir
  - $cMatrix_{E \times E} \leftarrow$ Matriz de conflictos entre exámenes, $cMatrix[i][j]$ es 1 si el examen i tiene conflictos con el examen j, 0 si no
- **Variables**:
  - $D[E][E] \leftarrow$ Lista de horarios asignables (dominio) del examen i, donde $D[i][j]$ se marca con el examen del cual se realizó el FC para filtar el horario j

- $sol[E]$ ← Lista que representa la solución actual, donde $sol[i]$ almacena el horario asignado al examen i
- maxSlots ← Cantidad máxima de horarios a usar, se ajusta luego de cada BT + FC
- minSlotsFound ← Flag que indica si se encontró el mínimo de horarios necesarios para resolver la instancia
- bestSol[E] ← Mejor solución encontrada hasta el momento, misma representación que $sol[i]$
- bestPenalty ← Penalización asociada a la mejor solución encontrada hasta el momento

Teniendo esta información, el algoritmo comienza a recorrer el espacio de búsqueda desde el primer examen considerando la mínima cantidad de horarios necesaria. Cuando termina su recorrido, si no se encuentra una solución que marque el flag *minSlotsFound*, se aumenta la cantidad mínima de horarios en 1 y se reinician los dominios filtrados de todos los exámenes para así repetir el proceso, según el siguiente algoritmo:

Algoritmo 1: Resolución del ETP con BT + FC aumentando la cantidad de horarios a usar

Resultado: Encontrar una solución a una instancia del ETP con la mínima cantidad de slots

```
Procedure doMaxTimeSlotsAdjustment()
    maxSlots ← 0;
    bestPenalty ← INF; Repeat
        maxSlots ← maxSlots + 1;
        doBackTracking(1);
        resetDomains(0);
    Until minSlotsFound;
End
```

Para recorrer el espacio de búsqueda se realiza un Backtracking recursivo que asigna un horario a cada examen en la lista *sol*[examen]. Ésta asignación se lleva a cabo a partir de un examen al cual se asigna cada uno de los horarios disponibles en orden ascendente, siempre y cuando el horario no se encuentre filtrado del dominio del examen actual. Una vez asignado un horario válido al examen, se revisa si el examen trabajado es el último de la instancia, en tal caso se llegó a una solución factible con cantidad mínima de horarios necesaria para la calendarización, luego se marca el flag *minSlotsFound* como verdadero y se calcula la penalización asociada. Si la penalización es menor a la penalización de la mejor calendarización encontrada hasta el momento, se actualiza *bestSol* y *bestPenalty*, de lo contrario, se realiza arco consistencia entre la solución parcial y el resto de exámenes por asignar a través del *Algoritmo 3*, que aplica la técnica de Forward Checking. Si el *Algoritmo 3* indica que alguno de los exámenes posteriores se quedó sin horarios disponibles, entonces se procede a trabajar con el siguiente

Cómo fue implementada la solución. Interesa la implementación particular más que el algoritmo genérico, es decir, si se tiene que implementar SA, lo que se espera es que se explique en pseudocódigo la estructura general y en párrafos explicativos cómo fue implementada cada parte para su problema particular. Si se utilizan operadores/movimientos se debe justificar por qué se utilizaron dichos operadores/movimientos. En caso de una técnica completa, describir detalles relevantes del proceso, si se utiliza recursión o no, explicar cómo se van construyendo soluciones, cuándo se revisan restricciones, cómo se registran conflictos, etc. En este punto no

se espera que se incluya código, eso va aparte.

Algoritmo 2: Asignar un horario a cada examen filtrando los horarios disponibles con Forward Checking

Resultado: Calendarización de exámenes para la cantidad de horarios actual, o bien, determinar que no es posible una calendarización con la cantidad de horarios disponible

```
Procedure doBackTracking(exam)
    slot ← 1;
    mientras slot ≤ maxSlots hacer
        si D[exam][slot] entonces
            continue
        fin
        sol[exam] ← slot;
        si exam es el último examen por asignar entonces
            minSlotsFound ← true;
            penalty = calculatePenalty(sol);
            si bestPenalty < penalty entonces
                bestSol = sol;
                bestPenalty = penalty;
            fin
        en otro caso
            si doForwardChecking() retorna 0 entonces
                resetDomains(exam);
                continue;
            fin
            doBackTracking(exam + 1);
            resetDomains(exam);
        fin
        slot ← slot + 1;
    fin
End
```

Algoritmo 3: Reiniciar dominios filtrados por un examen

Resultado: Reiniciar (marcar con 0) los dominios filtrados por la arco-consistencia aplicada por el Forward Checking desde el examen indicado

Procedure `resetDomains(`*actualExam*`)`

```
exam ← actualExam + 1;
mientras exam ≤ E hacer
    horario ← 1;
    mientras horario ≤ E hacer
        si si D[exam][horario] está marcado por actualExam entonces
            D[exam][horario] = 0;
        fin
        horario ← horario + 1;
    fin
    exam ← exam + 1;
fin
```

End

Algoritmo 4: Forward Checking a los horarios de los exámenes

Resultado: Establecer arco-consistencia entre los dominios de los exámenes por asignar y las asignaciones ya realizadas, marcando los valores incompatibles con el examen que inició el FC, retorna 0 si algún examen se queda sin dominio disponible y 1 en otro caso

Procedure `doForwardChecking(`*actualExam*`)`

```
exam ← actualExam + 1;
mientras exam ≤ E hacer
    si D[exam][sol[actualExam]] no está marcado entonces
        si cMatrix[actualExam][exam] tiene un conflicto entonces
            D[exam][sol[actualExam]] = actualExam;
        fin
    fin
    horario ← 1;
    disponibles ← 0;
    mientras horario ≤ maxSlots hacer
        si D[exam][horario] no está marcado entonces
            disponibles ← disponibles + 1;
        fin
        horario ← horario + 1;
    fin
    si disponibles igual a 0 entonces
        return 0;
    fin
    exam ← exam + 1;
fin
return 1;
```

End

7 Experimentos

Se necesita saber cómo experimentaron, cómo definieron parámetros, cómo los fueron modificando, cuáles problemas/instancias se estudiaron y por qué, etc. Recuerde que las técnicas

completas son deterministas y las técnicas incompletas son estocásticas.

8 Resultados

Qué fue lo que se logró con la experimentación, incluir tablas y gráficos (lo más explicativos posible). Los resultados deben ser comentados y justificados en detalle en esta sección.

9 Conclusiones

El ETP es un problema arduamente estudiado gracias a su importancia e impacto en las instituciones educacionales, los avances en su resolución se traducen en mejores condiciones para los estudiantes a la hora de rendir sus exámenes y mejores planificaciones para la institución, no obstante, las distintas realidades de éstas hacen que los investigadores se concentren en resolver distintas variantes generando así soluciones no aplicables a todas las instituciones. Pese a esto, las distintas técnicas implementadas han permitido ahondar en cómo cada una se puede ajustar para obtener mejores resultados en la institución particular donde el autor se centra, pero al mismo tiempo no suelen ser tan buenas cuando se aplican de manera general. Esto se visualiza en como los parámetros de de la mayoría de técnicas se pueden ajustar para ser excelentes en ciertos casos o en como las técnicas exactas funcionan bien para instituciones que requieren calendarizar pocos exámenes, sin lograr ser las mejores a la hora de hacer benchmarks estandarizados, ante este escenario, las técnicas basadas en híbridos con hyperheurísticas se posicionan como las técnicas más prometedoras para atacar este problema. A futuro se plantea ahondar en la investigación basada en técnicas híbridas con hyperheurísticas que se ejecuten en paralelo utilizando CUDA [24], para así estudiar espacios de soluciones en paralelo con el fin de acelerar el proceso.

Bibliografía

[1] Duong Tuan Anh and Kim-Hoa Lam. Combining constraint programming and simulated annealing on university exam timetabling. pages 205–210, 01 2004.

[2] Sol Broder. Final examination scheduling. *Commun. ACM*, 7(8):494–498, aug 1964.

[3] E. K. Burke and J. P. Newall. A multistage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, 3(1):63–74, 1999.

[4] E. K. Burke, J. P. Newall, R. F. Weare, and Nottingham Ng Rd. A simple heuristically guided search for the timetable problem. In *In: Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, Univ. of La Laguna*, pages 574–579. Academic Press, 1998.

[5] Edmund Burke, Yuri Bykov, and Sanja Petrovic. A multicriteria approach to examination timetabling. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, pages 118–131, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[6] Michael W. Carter. A survey of practical applications of examination timetabling algorithms. *Operations Research*, 34(2):193–202, 1986.

[7] Michael W. Carter, Gilbert Laporte, and Sau Yan Lee. Examination timetabling: Algorithmic strategies and applications. *The Journal of the Operational Research Society*, 47(3):373–383, 1996.

[8] A. J. Cole. The preparation of examination time-tables using a small-store computer. *The Computer Journal*, 7(2):117–121, 01 1964.

[9] P. H. Corr, B. McCollum, M. A. J. McGreevy, and P. McMullan. A new neural network based construction heuristic for the examination timetabling problem. In Thomas Philip Runarsson, Hans-Georg Beyer, Edmund Burke, Juan J. Merelo-Guervós, L. Darrell Whitley, and Xin Yao, editors, *Parallel Problem Solving from Nature - PPSN IX*, pages 392–401, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[10] Costa D and Alain Hertz. Ants can colour graphs. *Journal of the Operational Research Society*, 48:295–305, 03 1997.

[11] Philippe David. A constraint-based approach for examination timetabling using local repair techniques. In Edmund Burke and Michael Carter, editors, *Practice and Theory of Automated Timetabling II*, pages 169–186, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg.

[12] Sylvain Desroches, Gilbert Laporte, and Jean-Marc Rousseau. Horex: A computer program for the construction of examination schedules. *INFOR: Information Systems and Operational Research*, 16(3):294–298, 1978.

[13] Luca Di Gaspero, Barry Mccollum, and Andrea Schaerf. The second international timetabling competition (itc-2007): Curriculum-based course timetabling (track 3). 01 2007.

[14] Luca Di Gaspero and Andrea Schaerf. Tabu search techniques for examination timetabling. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, pages 104–117, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[15] Wilhelm Erben. A grouping genetic algorithm for graph colouring and exam timetabling. In Edmund Burke and Wilhelm Erben, editors, *Practice and Theory of Automated Timetabling III*, pages 132–156, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

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

[17] S.L.M. Lin. A broker algorithm for timetabling problem. In *2002). Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling. 21st-23rd August*, pages 372–386. Citeseer, 2002.

[18] Muhammad Rozi Malim, Ahamad Tajudin Khader, and Adli Mustafa. Artificial immune algorithms for university timetabling. In *Proceedings of the 6th international conference on practice and theory of automated timetabling*, pages 234–245. Brno, Czech Republic, 2006.

[19] Nirbhay K. Mehta. The application of a graph coloring method to an examination scheduling problem. *Interfaces*, 11(5):57–65, 1981.

[20] Liam T. G. Merlot, Natashia Boland, Barry D. Hughes, and Peter J. Stuckey. A hybrid algorithm for the examination timetabling problem. In Edmund Burke and Patrick De Causmaecker, editors, *Practice and Theory of Automated Timetabling IV*, pages 207–231, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[21] Achref Mouelhi, Philippe Jegou, Cyril Terrioux, and Bruno Zanuttini. On the efficiency of backtracking algorithms for binary constraint satisfaction problems. 01 2012.

[22] Ahmad Muklason, Gusti Bagus Syahrani, and Ahsanul Marom. Great deluge based hyper-heuristics for solving real-world university examination timetabling problem: New data set and approach. *Procedia Computer Science*, 161:647 – 655, 2019. The Fifth Information Systems International Conference, 23-24 July 2019, Surabaya, Indonesia.

[23] Zahra Naji Azimi. Hybrid heuristics for examination timetabling problem. *Applied Mathematics and Computation*, 163(2):705 – 733, 2005.

[24] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020.

[25] Luis Paquete and Thomas Sttitzle. Empirical analysis of tabu search for the lexicographic optimization of the examination timetabling problem. 03 2003.

[26] J. E. L. Peck and M. R. Williams. Algorithm 286: Examination scheduling. *Commun. ACM*, 9(6):433–434, jun 1966.

[27] R. Qu, E. K. Burke, B. McCollum, L. T. G. Merlot, and S. Y. Lee. A survey of search methodologies and automated system development for examination timetabling. *Journal of Scheduling*, 12(1):55–89, Feb 2009.

[28] Rong Qu and E Burke. Adaptive decomposition and construction for examination timetabling problems. pages 418–425, 01 2007.

[29] P. Ross, J. G. Marin-Blazquez, and E. Hart. Hyper-heuristics applied to class and exam timetabling problems. In *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No.04TH8753)*, volume 2, pages 1691–1698 Vol.2, 2004.

[30] Peter Ross, David Corne, and Hugo Terashima-Marín. *The phase-transition niche for evolutionary algorithms in timetabling*, pages 309–324. 01 2006.

[31] A. Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, Apr 1999.

[32] Automated Scheduling and Planning Group. Solving examination timetabling problems through adaptation of heuristic orderings e.k.burke * j.p.newall +.

[33] Kaveh Sheibani, EK Burke, and P de Causmaecker. An evolutionary approach for the examination timetabling problems. In *2002). Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling. 21st-23rd August*, pages 387–396. Citeseer, 2002.

[34] Grace Tacadao. A constraint logic programming approach to the course timetabling problem using eclipse. 03 2010.

[35] Joo Siang Tan, Say Leng Goh, Graham Kendall, and Nasser R. Sabar. A survey of the state-of-the-art of optimisation methodologies in school timetabling problems. *Expert Systems with Applications*, 165:113943, 2021.

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

[37] 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, 01 1967.

[38] George White and Pak–Wah Chan. Towards the construction of optimal examination schedules. *INFOR: Information Systems and Operational Research*, 17(3):219–229, 1979.

[39] R Wijgers and J.A. Hoogeveen. Solving the examination timetabling problem. 10 2020.

[40] D. C. Wood. A system for computing university examination timetables. *The Computer Journal*, 11(1):41–47, 01 1968.

[41] Gert Woumans, Liesje De Boeck, Jeroen Beliën, and Stefan Creemers. A column generation approach for solving the examination-timetabling problem. *European Journal of Operational Research*, 253(1):178 – 194, 2016.