Inteligencia Artificial

Informe 2: Examination Timetabling Problem

Gabriel Araya G.

10 de Enero del 2021

Evaluación

Mejoras 1ra Entrega (10 %):	_____
Código Fuente (10 %):	_____
Representación (15 %):	_____
Descripción del algoritmo (20 %):	_____
Experimentos (10 %):	_____
Resultados (10 %):	_____
Conclusiones (20 %):	_____
Bibliografía (5 %):	_____
Nota Final (100):	_____

Resumen

Las universidades requieren, durante sus periodos de evaluaciones, calendarizar los exámenes en diferentes bloques horarios sin conflictos entre ellos. Tal problema se le conoce como Examination Timetabling Problem (ETP) y en este informe se mostrará la definición formal del ETP y sus variantes, se analizarán distintos papers de las últimas dos décadas que aborden el ETP para determinar su estado del arte actual, se confeccionará un modelo matemático, y se mostrarán los resultados de resolver el ETP utilizando Simulated Annealing bajo distintas condiciones de experimentación.

Keywords— Examination Timetabling Problem, Simulated Annealing, ETP.

1. Introducción

Las universidades y otros centros de educación con asistencia masiva de estudiantes requieren, durante sus periodos de evaluaciones, calendarizar los exámenes en diferentes bloques horarios teniendo el cuidado de que un alumno no deba rendir dos exámenes al mismo tiempo. Este problema ha sido denominado como Examination Timetabling Problem (ETP) el cual es una particularización del Timetabling Problem (TP)

y la importancia y motivación de resolver adecuadamente este problema es que cada alumno pueda rendir sus exámenes sin la preocupación de tener topes, disminuir la carga académica del estudiando cuando debe rendir más de un examen y, finalmente, distribuir eficientemente los recursos (por ejemplo bloques y salas) que dispone la universidad en cuestión para realizar la jornada de exámenes.

Este primer documento del proyecto buscará establecer la definición formal del problema y luego establecer el modelo matemático que permitirá resolverlo con la técnica de solución asignada para el segundo informe. Para ello se hará un recorrido por las principales técnicas y autores influyentes que se han enfrentado al ETP en las últimas dos décadas.

Este informe primero mostrará una descripción y definición actual del ETP como un problema de la familia de TPs, se revisarán sus variantes y corroborarán los supuestos necesarios para este proyecto. Luego se estudiará el estado del arte desmenuzando los papers más influyentes que han salido desde los años 2000 y, de los cuales, se escogerá un modelo matemático el cual será levemente modificado para cumplir con los propósitos de este proyecto en particular. Luego se hará una **Descripción del Algoritmo** para solucionar el ETP, se propondrán **Experimentos** con ese algoritmo y se mostrarán los **Resultados**. Finalmente en las **Conclusiones** se hará un análisis de los resultados y se propondrá una propuesta de heurística conforme a la evidencia empírica obtenida.

2. Definición del Problema

Como se explicó anteriormente, se puede definir el ETP como una variación del clásico Timetabling Problem (TP). Según la reciente publicación de **Tan et al.** [1], todo TP está compuesto de cuatro parámetros: un conjunto finito de tiempos o bloques horarios (T), un conjunto finito de recursos (R), un conjunto finito de encuentros o meetings (M) y un conjunto finito de restricciones (C). La meta principal de estos problemas consiste en asignarle bloques y recursos a estos encuentros minimizando los conflictos con las restricciones. Lamentablemente minimizar estos conflictos se vuelve cada vez más difícil ya que los tipos de restricciones existentes para estos problemas aumentan con el día a día, volviéndose el problema cada vez más complejo a medida que se adecúa al la realidad y por no decirlo también, lo caótico, del día a día en una sociedad moderna.

Entendiendo lo anterior, según **Turabieh y Abdullah** [2], se puede pensar un ETP como el problema en que un conjunto de exámenes académicos deben ser calendarizados dentro de cierto periodo de tiempo bajo ciertas restricciones. Los elementos en común en un ETP normalmente incluyen:

- un conjunto de exámenes a calendarizar;
- una largo fijo o variable de timeslots o bloques; y
- un conjunto de estudiantes inscritos a rendir uno o más exámenes, y por ende conflictos entre exámenes.

Dado que un estudiante puede dar varios exámenes, se definen los conflictos como aquellos exámenes que tienen alumnos en común.

Hay dos categorías de restricciones, las cuales pueden ser duras (hard) o blandas (soft). Las restricciones «duras» o «de conflicto» son aquellas que deben ser satisfechas y que no pueden ser violadas bajo ninguna circunstancia. Al ser satisfechas, se puede comenzar a hablar de que la solución es factible. Por el contrario, las restricciones blandas son consideradas como menos esenciales y que cuya violación es aceptable pero no deseable, por lo que suelen estar acompañadas con algún tipo de penalización en la factibilidad de la solución en caso de que se viole la restricción blanda.

Como se puede notar respecto a la descripción del ETP versus un TP, falta definir todavía los recursos. El ETP puede tener tantas variantes como nuevas restricciones y recursos se le añadan. Algunos ejemplos:

- Algunos exámenes pueden tener precedencia sobre otros.
- Algunos exámenes deben ser rendidos en alguna sala en específico.
- Exámenes deben cumplir con los recursos de sillas en una sala.
- Exámenes deben cumplir con la presencia de profesores en específico.
- Estudiantes no deben dar más de un examen al día.
- un conjunto de estudiantes inscritos a rendir uno o más exámenes, y por ende conflictos entre exámenes.

Sobre el ETP en particular tratado en este documento, se puede decir que es una versión simplificada de lo anteriormente hablado, compartiendo los elementos comunes que se encuentran en todo ETP, pero con los siguientes supuestos respecto a los tiempos y recursos:

- El problema debe ser resuelto en una sola jornada, asumiendo que ese día tiene infinitos bloques horarios para desarrollar los exámenes.
- Hay una cantidad infinita de salas que pueden ser utilizadas en cada bloque, por lo que se podrían agendar en paralelo el máximo número de exámenes de no haber conflictos.
- Las salas tienen capacidad para recibir un infinito número de estudiantes, por lo que los exámenes no tienen una restricción dura respecto al tamaño de las salas o número de sillas.
- Como restricción dura, no pueden haber dos exámenes en conflicto en un mismo timeslot.
- Como restricción blanda, habrá penalización inversamente proporcional de acuerdo a qué tan contiguos sean los exámenes en conflicto.

Habrán dos objetivos para el ETP de este proyecto listados en orden de prioridad:

1. Que un timeslot tenga la mayor cantidad de exámenes posibles desde el más temprano hasta el último de la jornada y, como consecuencia de lo anterior, se buscará minimizar la cantidad de timeslot que tendrá la jornada en total.
2. Minimizar la penalización promedio para cada estudiante es esencial para no afectar negativamente la calidad de vida estudiantil durante esa jornada.

La dificultad de este problema radicará en la técnica de solución que sea asignada a este proyecto, ya que por ejemplo al haber restricciones blandas podría ser difícil determinar en qué momento realizar un salto inteligente para las técnicas de búsqueda completa.

3. Estado del Arte

El ETP ha sido estudiado por la comunidad científica desde la década de 1960 pero fueron los estudios de **Carter y Laporte** [3] y **Schaerf** [4] los que proveyeron de un acercamiento temprano para resolver el ETP, lo que conlleva a que las principales técnicas para resolver ETP aparecieran desde los años 2000 en adelante.

El estado del arte del ETP para este informe será basado en el espectacular trabajo de **Rong Qu et al.** [5] en el cual clasifican todas las técnicas de resolución habidas hasta el 2009, y además se agregarán las técnicas post-2010 encontradas durante la investigación para este proyecto . Cabe señalarse que este informe sólo mencionará las técnicas más actuales y/o contribuyentes, ya que se entiende que éstas representan el estado del arte actual.

3.1. Técnicas Secuenciales Basadas en Grafos

Welsh y Powell [6] en 1967 realizaron la importante contribución de demostrar que un TP puede ser representado como un problema de coloreo de grafos, lo cual conllevó a futuro a un montón de investigaciones sobre estos grafos y sus heurísticas para resolverlos.

Burke y Newall [7] investigaron en 2004 una estrategia de orden adaptativo en el cual los exámenes se reordenaban dinámicamente en cada iteración cuando se resolvía el ETP. Se observó que utilizar heurísticas fijas y predefinidas no siempre se comportaban bien para resolver el ETP y que por tanto era importante tener una heurística que modificara el ordenamiento de los exámenes. Este acercamiento demostró ser simple y efectivo.

Asmuni et al. [8] en 2004 ordenó los exámenes basados en heurísticas de coloreo de grafos pero con la idea de añadir el parámetro de dificultad en los exámenes y gestionar el orden dependiendo de la dificultad. El objetivo fue mejorar la calidad del ETP respecto a la frustración que pueda percibir el estudiante.

3.2. Técnicas Basadas en Restricciones

Brailsford, Potts y Smith [9] demostraron en 1999 que las técnicas de satisfacción de restricciones podían ser aplicadas en problemas de optimización como por ejemplo el ETP. Lamentablemente estas técnicas son computacionalmente costosas debido al crecimiento exponencial del espacio de búsqueda de las soluciones conforme crecen las variables. Es por lo anterior que diferentes heurísticas y técnicas han sido integradas a este tipo de resoluciones para disminuir los tiempos de computo para problemas prácticos.

Merlot et al. [10] emplearon técnicas de programación de restricciones usando **OPL** [11], una optimización de lenguaje de programación, para producir soluciones iniciales y, luego, con técnicas de Recocido Simulado (Simulated Annealing o SA) y Hill Climbing (HC) se mejoraron las soluciones. Utilizando este método híbrido se obtuvieron los mejores resultados para los datasets de las universidades de Toronto y Nottingham hasta ese entonces.

3.3. Técnicas Basadas en Búsqueda Local

Esta familia de técnicas resuelven los problemas buscando en los vecindarios de una solución inicial por mejores resultados y su ventaja radica en que entregan muy buenas soluciones con costo computacional bajo o aceptable.

Cabe señalarse que estas técnicas son las que marcan la tendencia actualmente y son utilizadas «híbridamente» en casi todos los estudios sobre el ETP mezclándose con otras técnicas o algoritmos, como los genéticos o heurísticos por dar un ejemplo.

3.3.1. Hill Climbing

La escalada, ascenso de colinas o Hill Climbing (HC) es un algoritmo greedy e iterativo que comienza de una solución inicial arbitraria hasta encontrar una mejor solución variando incrementalmente de a un único elemento de la solución. Este algoritmo suele encontrar óptimos locales pero no garantiza encontrar óptimos globales dentro del espacio de búsqueda. Se suele remediar la situación anterior usando reiniciaciones.

En 2013, **Abdul-Rahim, Bargiela y Qu** [12] buscaron comparar Hill Climbing contra los algoritmos genéticos en el ETP. Su algoritmo consistió en un HC greedy con permutaciones, shifts y swappings en los timeslots asociados a cada examen, todo en un proceso de doble orden de optimización. Los resultados de su HC greedy sobrepasaron en calidad al algoritmo genético.

Bykov y Petrovic [13] presentaron en 2016 una heurística de búsqueda local que denominaron como «Step Counting» y la utilizaron en un Hill Climbing (SCHC). El «Step Counting» es un método en que el costo actual de recuentos sirve como un límite de aceptación para una serie de pasos consecutivos. El recuento de pasos se puede organizar de diferentes formas y, por lo tanto, el método propuesto puede generar una gran cantidad de variantes y extensiones. El SCHC mostró un mejor rendimiento en comparación al HC, al Simmulated Annealing y al «Great Deluge Algorithm»,

3.3.2. Tabu Search

Tabu Search explora un espacio de búsqueda teniendo el cuidado de no realizar los mismos movimientos de búsqueda, recordando estos movimientos en una lista tabú, excepto si estos movimientos pueden entregar una mejor solución escapando de un óptimo local.

Di Gaspero y Schaerf [14] estudiaron estrategias de selección exhaustivas y sesgadas para una familia de técnicas basadas en Tabu Search, cuyos movimientos dependían si se violaban restricciones duras o blandas.

Luego en 2002, **Di Gaspero** [15] mejoró el enfoque anterior implementando múltiples vecindades y empleando recolor(cambiar un sólo examen), shake (intercambiar grupos de exámenes) y kickers (cambiar la secuencia de exámenes únicos).

También en 2002, **Paquete y Stutzle** [16] desarrollaron su técnica de Tabu Search basándose en un orden de prioridad para las restricciones. Utilizaron dos enfoques: resolver de una en una las restricciones, o resolver todas al mismo tiempo. El segundo enfoque resultó tener mejores resultados, pero el primer enfoque era más consistente.

3.3.3. Recocido Simulado

El Recocido Simulado o Simulated Annealing (SA) busca escapar de la zona de óptimo local durante la exploración del espacio de búsqueda realizando algún «mal movimiento» con cierta probabilidad. Esa probabilidad al inicio es alta pero va decreciendo con el tiempo, y, de ahí el término de «recocido» (ya que su temperatura disminuye).

Como se mencionó anteriormente, **Merlot et al.** [10] utilizaron una técnica híbrida de SA y OPL [11].

Duong y Lam [17] determinaron que dado un tiempo de búsqueda finito, es crucial escoger los hiperparámetros correctos. Para ello desarrollaron diversos mecanismos y algoritmos para el enfriamiento de la temperatura en el SA.

3.3.4. Otras Técnicas de Búsqueda Local

Burke et al. [18] estudiaron una variante del SA llamada «Great Deluge Algorithm» en el cual el «mal movimiento» sólo se aceptaba si no bajaba más de cierto nivel la calidad la solución, pero donde el nivel de calidad va decreciendo con la iteración (de forma parecida al enfriamiento de la temperatura). Este algoritmo resultó ser mejor que SA y mostró los mejores resultados en los datasets de las universidades de Toronto y Nottingham con respecto a las demás técnicas, superando los resultados de **Merlot et al.** [10].

Abdullah et al. [19] en 2007 mejoraron el «Great Deluge Algorithm» de **Burke et al.** [18] realizando una gran búsqueda en los vecindarios y aplicando la metodología de mejoras en la construcción del grafo. En lugar de simplemente considerar los operadores tradicionales basados en intercambios por pares, diseñaron una estructura de vecindario basada en árboles para realizar intercambios cíclicos entre todos los

intervalos de tiempo.

Turabieh y Abdullah [2] mejoraron el proceso anterior realizando un híbrido entre el algoritmo que desarrollaron **Abdullah et al.** [19] en el 2007 y heurísticas de los algoritmos de mecanismos electromagnéticos. En un primer paso utilizaban heurísticas de coloreo de grafos, generaban soluciones iniciales a partir de ahí y a cada una le calculaban su "fuerza electromagnéticaz luego con el «Great Deluge Algorithm» encontraban las soluciones finales.

3.4. Algoritmos Basados en la Población

3.4.1. Algoritmos Genéticos

Desde un conjunto de soluciones iniciales, un algoritmo genérico se encarga de .ªparear.ésas soluciones y verificar si la descendencia entrega mejores resultados. La forma en que se aparean las soluciones es que la solución hijo hereda cierto porcentaje de fragmentos de la soluciones en las variables de ambos padres. A demás, existe una cierta probabilidad de que ocurran mutaciones al final de los apareamientos en alguna variable de la solución hijo.

Sheibani [20] en 2002 construyó un modelo matemático especial y desarrolló un algoritmo genético estándar cuyo objetivo era maximizar los intérvalos de tiempos entre exámenes. Para ello desarolló una red de actividades dirigidas para estimar la cercanía entre exámenes.

También en 2002, **Wong, Côte y Gely** [21] utilizaron un torneo de selección para escoger a las soluciones padres e incorporaron estrategias de re-apareamiento con mutaciones.

En 2004, **Côte, Wong y Sabourin** [22] estudiaron un algoritmo evolutivo con dos funciones objetivos para minimizar los timeslots del ETP y espaciar los exámenes en conflicto. En vez de recombinar operadores, se utilizaron dos operadores de búsqueda local para lidear con las restricciones blandas y duras.

Ülker et al. [23] desarrollaron en 2006 un algoritmo genético que utilizaba "Linear Linkage Encodingçomo método de representación. Diferentes operadores de crossover fueron utilizandos en conjunto con coloreo de grafos.

3.4.2. Algoritmos Meméticos

Son una variante de los algoritmos genéticos, donde una solución padre puede mejorarse, por ejemplo con algún algoritmo de búsqueda local, antes de aparearse.

Burke, Newall y Weare [24] desarrollaron un algoritmo memético en el que se emplearon operadores de mutación ligera y pesada para reasignar exámenes individuales y conjuntos de exámenes con precedencia. Luego utilizaron Hill-Climbing en las soluciones y mejorar la calidad de los horarios, aunque ello aumentó el costo computacional.

3.4.3. Algoritmos de Hormiga

También conocidos como Colonia de Hormigas, estos algoritmos buscan simular el cómo las hormigas escogen las rutas más cortas hacia la comida a través de caminos de feromonas. Cada hormiga construye una solución y la información obtenida se deja como feromona la cual es usada posteriormente para generar más soluciones en la siguiente iteración.

Naji-Azimi [25] implementó en 2004 un Sistema de Colonia de Hormigas y lo comparó con respecto a

Simulated Annealing, Tabu Search y Algoritmos Genéticos. Las soluciones iniciales de la colonia de hormigas se generaron a través de heurísticas y se mejoraron con técnicas de búsqueda local, de las cuales Tabu Search resultó ser la mejor.

En 2005, **Dowsland y Thompson** [26] desarrollaron un algoritmo de hormigas basado en coloreo de grafos sin considerar las restricciones blandas.

Eley [27] en 2006 comparó dos Algoritmos de Hormiga: Sistema de Hormiga Max-Min para el ETP, y el algoritmo ANTCOL para coloreo de grafos. Se concluyó que ANTCOL fue el con mejor rendimiento cuando ambos algoritmos eran modificados con Hill-Climbing.

3.5. Técnicas Multi-Criterio

Se han estudiado Técnicas Multi-Criterio en el ETP con el objetivo de manejar diferentes restricciones considerando un vector de restricciones en lugar de una única suma ponderada. En las Técnicas Multi-criterio, se puede considerar que cada criterio corresponde a una restricción, que tiene un cierto nivel de importancia y se trata de forma individual.

En el 2000, **Burke, Bykov y Petrovic** [28] desarrollaron un enfoque Multi-Criterio de dos etapas con nueve criterios de aceptación para el ETP.

Luego **Petrovic y Bykov** [29] para el 2002 mejoraron el enfoque anterior desarrollando un algoritmo híbrido ocupando el «Great Deluge Algorithm» para mejorar las soluciones iniciales.

3.6. Hiper-heurísticas

Ajustar los parámetros para adaptarse a nuevos problemas puede considerarse tan difícil como el de desarrollar nuevas técnicas de solución. Este tema bien conocido ha llevado a varios investigadores a desarrollar nuevas tecnologías destinadas a operar a un nivel superior de generalidad. El término Hiper-heurística puede verse como aquellas heurísticas que eligen heurísticas, es decir, un espacio de búsqueda de heurísticas es el foco de atención en lugar de un espacio de búsqueda de soluciones.

En 2006, **Bilgin et al.** [30] analizaron 7 métodos de selección de heurísticas y 5 criterios de aceptación para el ETP. Concluyeron que diferentes combinaciones de métodos y criterios de aceptación tenían buen rendimiento para diferentes instancias de ETP, aunque algunas combinaciones trabajaban ligeramente mejor que otras.

Burke et al. [31] investigaron las heurísticas en 2007 usando Tabu Search para encontrar el orden de secuencias de heurísticas óptimo para coloreo del grafo del ETP. Se concluyó que a mayor sea el número de heurísticas de bajo nivel, mejor era el resultado, sin embargo, el espacio de búsqueda crecía significativamente.

Durante el 2008, **Qu y Burke** [32] investigaron más a fondo el efecto de emplear diferentes algoritmos de búsqueda de alto nivel respecto al grafo unificado de hiper-heurísticas para el ETP. Los resultados experimentales demostraron que el método hiper-heurístico empleado en el espacio de búsqueda del grafo de heurística no era crucial.

3.7. Técnicas de Descomposición/Clustering

Son técnicas basadas en el principio de Dividir y Conquistar, ya que buscan tomar un problema y dividirlo en un conjunto de subproblemas más pequeños. No son muy populares ya que, una vez juntadas todas

las soluciones de los subproblemas, es fácil obtener soluciones globales infactibles o que las restricciones blandas las castigan demasiado.

Qu y Burke [33] en 2007 estudiaron un enfoque en el que los exámenes eran divididos en dos conjuntos (fáciles y difíciles) respecto a qué tan difícil eran de calendarizar durante las iteraciones previas mientras se construían las soluciones. El rendimiento se vio altamente mejorado gracias a la decisión de trabajar los exámenes difíciles por separado. Este enfoque tuvo uno de los mejores resultados hasta ese entonces en una de las instancias del dataset de la universidad de Toronto.

4. Modelo Matemático

El siguiente modelo estará basado en la versión de **Turabieh y Abdullah** [2], y modificado para que cumpla con los requerimientos asignados para este proyecto.

Sean M, N, B los números total de alumnos, exámenes y timeslots (bloques) destinados a cierto periodo de exámenes en alguna universidad o establecimiento educacional. Sean $E = \{e_1, e_2, ..., e_N\}$ el conjunto de exámenes que serán rendidos, $A = \{a_1, a_2, ..., a_M\}$ el conjunto de alumnos que rendirán el exámen y $T = \{t_1, t_2, ..., t_B\}$ el conjunto de timeslots que destinó la universidad para el periodo de exámenes. Por tanto, se define el vector de soluciones para este problema como $\vec{x} = (t_1, t_2, ..., t_N)$ y que para el cual se describe lo siguiente:

Variables de decisión:

1. $|T_{final}|$: Total de timeslots mínimos a utilizar
2. t_i: Timeslot en el cual se rendirá el examen i, con $i = \{1, 2, ..., B\}$

Dado que el problema propuesto no indican una cantidad B definida de timeslots iniciales, se considerará un hiper-parámetro, ya que impacta directamente en el espacio de búsqueda, por lo que se puede experimentar y verificar qué tan eficiente es el algoritmo a medida que se «tunea» ese parámetro. Entonces, el espacio de búsqueda queda definido por B^N.

Parámetros adicionales:

1. $C_{ij} = \begin{cases} K & \text{si los exámenes i y j tienen K alumnos en común} \\ 0 & \text{si no} \end{cases}$

Función Objetivo:

La Función Objetivo (1) será la propuesta por **Turabieh y Abdullah** [2]. Está función representa el la penalización promedio para cada estudiante.

$$\min \frac{1}{M} \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{ij} \cdot w(t_i, t_j) \tag{1}$$

donde

$$w(t_i, t_j) = \begin{cases} 2^{5-|t_i - t_j|} & \text{si } 1 \leq |t_i - t_j| \leq 5 \\ 0 & \text{si no} \end{cases} \tag{2}$$

Maximizar la cantidad de exámenes que se rindan en un timeslot, priorizando los primeros timeslots, es equivalente a minimizar los timeslots que sean asignados a cada examen. Las minimizaciones (1) y (2) cumple con el objetivo de minimizar la penalización promedio de cada estudiante conflicto. Entre esas dos minimizaciones, (2) buscará priorizar siempre que se minimicen los timeslots por sobre los conflictos, incluso

si se da la situación de que no haya conflictos, es decir, si $C_{ij} = 0$ siempre.

La función de penalización (3) indica los puntajes de penalización que ocurren según la proximidad entre exámenes.

Restricciones:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{ij} \cdot \lambda(t_i, t_j) = 0 \tag{3}$$

donde

$$\lambda(t_i, t_j) = \begin{cases} 1 & \text{si } t_i = t_j \\ 0 & \text{si no} \end{cases} \tag{4}$$

La restricción dura (4) permite asegurar que dos exámenes en conflicto nunca sean rendidos en el mismo timeslot. La función (5) sirve para diferenciar cuando los exámenes tienen un mismo timeslot, la restricción (4) se cumpla sólo si los exámenes no están en conflicto.

5. Representación

En la literatura se han encontrado dos formas para representar las soluciones: vectorial y matricial. Se decidió optar por la representación vectorial ya que, primero, consume significativamente menos memoria en el ordenador, y segundo, es equivalente a la definición de E en el modelo matemático por lo que es innecesario realizar algún tipo de interpretación o decodificación, obteniéndose los resultados de cada examen directamente simplemente observando el vector solución.

En el vector solución, la posición representa el id del examen y lo que está contenido en esa posición representa el timeslot asociado a ese examen. Por ejemplo, en la **Figura 1**, leyendo de izquierda a derecha, se observa que en la primera posición está el número tres, representado que el primer examen será calendarizado en el tercer timeslot. Así mismo, en la segunda posición está el número uno, indicando el que segundo examen será calendarizado en el primer timeslot.

Si el número máximo de timeslots es $|T|$ y el número de exámenes es $|E|$, con la representación anterior podemos calcular el espacio de búsqueda como $EB = |T|^{|E|}$. El espacio de búsqueda contiene tanto soluciones factibles como infactibles, así que será necesario que el algoritmo propuesto próximamente deba ser capaz de descartar soluciones infactibles.

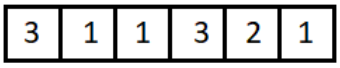

Figura 1: Ejemplo de vector solución.

6. Descripción del Algoritmo

El algoritmo a utilizar es el Simulated Annealing (SA). A grandes rasgos es un algoritmo reparador el cual parte con una temperatura alta $Temp = 100$ y se va enfriando a una razón *alpha* a medida que se itera. El algoritmo dejará de iterar cuando se haya enfriado el sistema hasta una temperatura T_{min}, o si durante $iter\_max2 = 5$ iteraciones consecutivas no ha mostrado mejoras la calidad de la solución.

Existe otra iteración,interna a la iteración de temperatura, la cual se encarga de actualizar la solución candidata en la que se escoge estocásticamente una solución nueva en el vecindarios de la solución candidata actual. Esta solución nueva será la próxima solución candidata del algoritmo si muestra tener una mejor calidad que la solución inicial, o en caso contrario, si cumple con el requisito probabilístico en función de la temperatura actual el cual será explicado en detalle más adelante. El proceso anterior se repite una cantidad constante de $iter\_max1$ iteraciones, dando paso después a que se enfríe el sistema y se vuelva a repetir todo el proceso con una nueva temperatura.

A continuación se explicaran detalladamente las componentes esenciales del algoritmo, por lo que es necesario aclarar que aquellas funciones utilizadas para la lectura de los dataset, la obtención de la cantidad S de estudiantes (M en el modelo matemático), E de exámenes, el cálculo de la matriz de conflictos C, y las funciones que escriben los archivos de solución serán obviadas. Las funciones esenciales son:

1. Función *checkeo*: Esta función verifica si la restricción dura de la ecuación (3) se cumple para un vector solución v. Para ello realiza una doble sumatoria con los mismos rangos de la ecuación (3) hasta que la suma de un valor distinto de cero. Si el valor de la suma es mayor a cero se rechaza v, caso contrario, se aprueba.
2. Función *calidad*: Esta función calcula la calidad de un vector solución v el cual ya ha sido aprobado anteriormente por la función *checkeo*. Esta función calcula la siguiente ecuación:

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} C_{ij} \cdot penalizacion(t_i, t_j) \tag{5}$$

   Es decir, es la función objetivo (1), pero sin dividirla por la cantidad de estudiantes. La razón de escoger esta función como la función de evaluación es porque, dado que la cantidad de estudiantes es siempre constante, es irrelevante durante las iteraciones. Además, que esta función al ser calculadas muchas veces durante las iteraciones, disminuirle una operación elemental puede ayudar, aunque sea un poco, a disminuir el tiempo de ejecución del algoritmo completo.
3. Función *penalizacion*: calcula la penalización entre los timeslot de los exámenes E_i y E_j con forme a la ecuación (2).
4. Función *simulatedAnnealing*: Esta función (ver Algorithm 1) se encarga de crear la solución inicial con la función *solucionInicial* y luego repararla con las iteraciones externa, asociada a la temperatura actual del sistema, y la iteración interna, asociada a repetir $iter\_max1$ veces la generación del vecindario de la solución actual y elección de una de esas soluciones a través de la funciones de $movimiento_i$ y *evaluacion*. Este algoritmo se detiene cuando se haya enfriado el sistema hasta una temperatura $T_{min} <= 5$, o si durante $iter\_max2 = 5$ iteraciones consecutivas no ha mostrado mejoras la calidad de la solución.
5. Función *evaluacion*: Determina si la solución nueva reemplazará a la solución actual. Para ello calcula la diferencia Δ entre la calidad de la solución nueva y la calidad de la solución actual respectivamente. Si $\Delta < 0$, entonces la calidad de la solución nueva es menor que la de la solución actual y se aprueba que la solución nueva reemplace a la actual. Caso contrario, si se escoge un numero entre 0 y 1 *aleatorio* y se cumple que $e^{\frac{-\Delta}{Temp}} > aleatorio$ entonces se aprueba que la solución nueva reemplace a la actual.
6. Función *solucionInicial*: Esta función es la encargada de generar la primera solución con la que se empezarán las iteraciones del SA. Esta solución está basada en el algorimo greedy propuesto por **Kusumawardani et al.** [34] de asignarle el menor valor de timeslot posible al examen con más conflictos, luego lo mismo con el segundo examen con más conflictos, y así sucesivamente hasta llenar el vector solución $|E|$ veces.

7. Función $movimiento_i$: Se implementaron 4 funciones de movimientos con las cuales se realizarán experimentos. Estas funciones son:

   - Función $movimiento1$: Esta función, cuyo input principal es la solución actual, y mediante un ciclo for, busca el timeslot T_i en la posición i de la solución actual e intenta generar dos vectores nuevos: uno igual a la solución actual pero con el elemento i con un timeslot $T_i + 1$; y otro similar pero con un timeslot $T_i - 1$. Estos vectores son verificados con la función *checkeo* y si aprueban son añadidos a la lista de "soluciones vecinas", por lo que existe el potencial de generar un vecindario con $2 * E$ soluciones. Luego, de entre todas estas soluciones, se escoge una al azar y se retorna.
   - Función $movimiento2$: Esta función, cuyo input principal es la solución actual, se calcula el mínimo timeslot min y el máximo timeslot max de la solución actual y, mediante un ciclo for, busca el timeslot T_i en la posición i de la solución actual y le asigna un valor al azar entre min y max y genera un nuevo vector igual a la solución actual a excepción del nuevo elemento i, luego se verifica la solución con la función *checkeo* y si aprueba se agrega a la lista de "soluciones vecinas", por lo que existe el potencial de generar un vecindario con E soluciones. Luego, de entre todas estas soluciones, se escoge una al azar y se retorna.
   - Función $movimiento3$: Esta función, cuyo input principal es la solución actual, y mediante un ciclo for, se intercambia el elemento i con otro elemento escogido al azar y se genera un vector similar a la solución actual a excepción por las dos posiciones intercambiadas, luego se verifica la solución con la función *checkeo* y si aprueba se agrega a la lista de "soluciones vecinas", por lo que existe el potencial de generar un vecindario con E soluciones. Luego, de entre todas estas soluciones, se escoge una al azar y se retorna.
   - Función $movimiento4$: Esta función es una combinación entre las funciones $movimiento2$ y $movimiento3$. El input principal es la solución actual, se calcula el mínimo timeslot min y el máximo timeslot max de la solución actual y, mediante un ciclo for, se intercambia el elemento i con otro elemento escogido al azar, pero además, el elemento i se reemplaza por un timeslot determinado al azar entre min y max y se genera un vector similar a la solución actual a excepción por las dos posiciones intercambiadas, luego se verifica la solución con la función *checkeo* y si aprueba se agrega a la lista de "soluciones vecinas", por lo que existe el potencial de generar un vecindario con E soluciones. Luego, de entre todas estas soluciones, se escoge una al azar y se retorna.

Algorithm 1 simulatedAnnealing

Result: sb

$contador2 = 0$

$Temp = 100$

$sc = solucionInicial(E, C)$

$sb = sc$

while $contador2 < 5$ *and* $Temp > 5$ **do**

  $contador1 = 0$

  $aux = sb$ **while** $contador1 < iter\_max1$ **do**

    $sn = movimiento_i(sc, C, E)$

    if $evaluacion(sn, Temp, C, E, S)$ **then**

      $sc = sn$

    else

    end

    if $calidad(sc, C, E, S) < calidad(sb, C, E, S)$ **then**

      $sb = sc$

    else

    end

    $contador1++$

  end

  $Temp = alpha * Temp$

  if $calidad(sb, C, E, S) == calidad(aux, C, E, S)$ **then**

    $contador2++$

  else

  end

end

7. Experimentos

Los experimentos realizados tienen el objetivo de encontrar el mejor de todos los *movimientos* y el mejor valor de los parámetros *alpha* e $iter\_max1$. La temperatura inicial se dejó constante durante todos los experimentos con un valor $Temp = 100$.

Los experimentos se llevaron a cabo en un sistema operativo Ubuntu Desktop 20.04.1 LTS, con un procesador AMD FX-8350, el cual posee 8 núcleos y un clock de 4.2 Ghz. Se realizaron 3 experimentos para cada elemento a testear y se sacó el promedio y la desviación estándar de éstos. EL promedio entrega un valor aproximado de la calidad que se esperaría obtener al repetir un experimento, y la desviación estándar informa que tanto intervalo de fallo hay con respecto al promedio, por lo que entre menos desviación estándar uno esperaría que un experimento sea similar con respecto a cualquier otra repetición bajo las mismas condiciones.

Para decidir qué instancia utilizar, inicialmente se utilizó un $alpha = 0{,}85$ y un $iter\_max1 = 3500$ según lo sugerido por **Kusumawardani et al.** [34]. Finalmente se escogió la instancia St.Andrews83 ya que posee

la suficiente cantidad de exámenes para que no fuera muy rápida (entre 5 a 15 minutos) para realizar comparaciones, pero tampoco era excesivamente lenta cada prueba a diferencia de otras instancias que llegaban a demorar tiempos de 30 minutos (YorkMills83) hasta 14 horas (Carleton91), y de esta forma realizar varias iteraciones de experimentos.

El dataset St.Andrews83 es una recopilación de exámenes y estudiantes que los deben rendir del año 83 en la universidad Saint Andrews del Reino Unido. Este dataset cuenta con 611 estudiantes quienes deben rendir 139 exámenes. La distribución de estos exámenes se encuentra en el archivo $St.Andrews83.stu$.

Luego de realizar todos los experimentos y haber decidido los mejores movimientos y parámetros, se obtendrán los resultados para algunos de los datasets propuestos para este problema en el ramo.

8. Resultados

Se realizaron tres experimentos para cada movimiento. Utilizando un $alpha = 0{,}85$ y un $iter\_max1 = 3500$ se obtuvo los siguientes resultados:

Penalizaciones Promedio de:	**#1**	**#2**	**#3**	**Promedio**	**Desv. Estándar**
Movimiento1	174.87	174.81	174.85	174.84	0.03
Movimiento2	179.78	178.73	169.41	175.97	5.70
Movimiento3	199.32	199.32	199.32	199.32	0.00
Movimiento4	166.32	166.24	165.20	165.92	0.62

Cuadro 1: Penalizaciones promedios para los tres experimentos de cada movimiento.

Tiempo [s] de:	**#1**	**#2**	**#3**	**Promedio**	**Desv. Estándar**
Movimiento1	351.40	1299.88	416.97	689.42	529.69
Movimiento2	694.45	491.49	734.71	640.22	130.36
Movimiento3	169.75	172.65	172.48	171.63	1.63
Movimiento4	475.58	550.84	449.85	492.09	52.48

Cuadro 2: Tiempo de ejecución en segundos para los tres experimentos de cada movimiento.

Timeslots totales de:	**#1**	**#2**	**#3**	**Promedio**
Movimiento1	13	13	13	13
Movimiento2	13	13	13	13
Movimiento3	13	13	13	13
Movimiento4	13	13	13	13

Cuadro 3: Timeslots totales para los tres experimentos de cada movimiento.

De lo anterior, la función *movimiento*4 resultó ser la mejor para continuar con la siguiente fase de la experimentación, ya que fue la que mejor minimizó las penalizaciones promedio en tiempo de $492[s]$ aproximadamente y con desviaciones estándar del 9 % de ese valor. El *movimiento*1 podría ser un buen candidato pero, a diferencia del *movimiento*4, en promedio consume cerca de un 40 % más de tiempo de ejecución y

a la vez tiene una desviación estándar demasiado alta, por lo que hay riesgo de que demore hasta un 150 % más de tiempo que el *movimiento*1 en ejecutarse.

Luego se experimentó con el parámetro *alpha*, manteniendo constante $iter\_max1 = 3500$ y utilizando *movimiento*4 para generar el vecindario de la solución actual. Los resultados fueron los siguientes:

Penalizaciones Promedio de:	**#1**	**#2**	**#3**	**Promedio**	**Desv. Estándar**
0.95	168.23	166.97	165.69	166.96	1.27
0.90	167.38	166.63	165.47	166.49	0.96
0.85	166.10	165.25	166.51	165.95	0.64
0.80	165.18	165.22	166.22	165.54	0.59

Cuadro 4: Penalizaciones promedios para los tres experimentos de cada *alpha*.

Tiempo [s] de:	**#1**	**#2**	**#3**	**Promedio**	**Desv. Estándar**
0.95	564.19	803.21	394.94	587.45	205.12
0.90	657.52	297.52	468.14	474.39	180.08
0.85	435.18	439.64	435.66	436.83	2.45
0.80	392.49	395.47	386.02	391.33	4.83

Cuadro 5: Tiempo de ejecución en segundos para los tres experimentos de cada *alpha*.

Timeslots totales de:	**#1**	**#2**	**#3**	**Promedio**
0.95	13	13	13	13
0.90	13	13	13	13
0.85	13	13	13	13
0.80	13	13	13	13

Cuadro 6: Timeslots totales para los tres experimentos de cada *alpha*.

Como se puede observar, y contrario a lo que se pensaría, $alpha = 0{,}80$ obtuvo los mejores resultados tanto en penalización promedio como en tiempo de ejecución. Tener una factor de enfriamiento mayor implicaría una mayor cantidad de iteraciones y más oportunidades para encontrar el óptimo. El autor de este documento cree que debido la naturaleza estocástica de el algoritmo junto con la eficiencia del *movimiento*4 para generar vecindarios que diversifiquen jugó un papel relevante en estos resultados.

De igual forma se experimentó con el parámetro $iter\_max1$, manteniendo constante $alpha = 0{,}80$ y utilizando *movimiento*4 para generar el vecindario de la solución actual. Los resultados fueron los siguientes:

Penalizaciones Promedio de:	**#1**	**#2**	**#3**	**Promedio**	**Desv. Estándar**
1000	165.67	165.84	169.07	166.86	1.92
2000	167.97	166.93	165.35	166.75	1.32
3000	165.79	167.56	166.13	166.49	0.94
4000	165.49	165.92	165.63	165.68	0.22

Cuadro 7: Penalizaciones promedios para los tres experimentos de cada $iter\_max1$.

Tiempo [s] de:	#1	#2	#3	Promedio	Desv. Estándar
1000	125.58	125.30	129.06	126.65	2.10
2000	260.04	271.03	276.95	269.34	8.58
3000	409.17	405.42	394.52	403.04	7.61
4000	519.45	535.71	537.91	531.02	10.08

Cuadro 8: Tiempo de ejecución en segundos para los tres experimentos de cada $iter\_max1$.

Timeslots totales de:	#1	#2	#3	Promedio
1000	13	13	13	13
2000	13	13	13	13
3000	13	13	13	13
4000	13	13	13	13

Cuadro 9: Timeslots totales para los tres experimentos de cada $iter\_max1$.

En este caso, $iter\_max1 = 4000$ tuvo la mejor penalización promedio, pero, observando los **Cuadros 4 y 5** cuando se experimento para $alpha = 0{,}80$ con $iter\_max1 = 3500$ se obtuvo un mejor desempeño tanto en penalización promedio como en tiempo de ejecución. Es de esperarse que mientras más iteraciones internas se obtengan mejores resultados, pero claramente hay un óptimo en eficiencia temporal para esta cantidad de iteraciones.

Finalmente, utilizando la función $movimiento4$, y los parámetros $iter\_max1 = 3500$ y $alpha = 0{,}80$ se obtuvo lo siguientes resultados para estos datasets:

Datasets:	Penalización Promedio	Timeslots totales	Tiempo [s] de ejecución
Carleton91	7.218	36	33685.450
EdHEC92	8.833	20	140.322
LSE91	12.886	20	12112.900
St.Andrews83	165.234	13	394.563
TorontoE92	23.475	11	1599.350
Trent92	11.939	22	3296.262
YorkMills83	32.337	24	1083.377

Cuadro 10: Resultados finales para distintos datasets.

9. Conclusiones

9.1. Estado del Arte:

Las técnicas vistas en el estado del arte, en general, tratan de resolver los mismos datasets de las universidades de Toronto y Nottingham y «competir» entre qué técnica lo hace mejor, por lo que todas intentan resolver el mismo problema, con sus mismos niveles de complejidades. Esto se debe, probablemente, a que los mismos autores están involucrados en varios estudios, representando en la últimas dos décadas a la vanguardia de autores interesados en llegar a la máxima eficiencia posible.

El gran parecido de esas técnicas con el problema dado en este informe y proyecto, es que se intentará resolver el dataset de la universidad de Toronto, por lo que se puede asegurar que se está trabajando con la misma definición del problema que aquellos estudios que hayan hecho benchmark para esa universidad, y la diferencia principal radicaría en la función objetivo (2) del modelo matemático propuesta en este informe es original del autor de este informe.

Otra diferencia notable entre estas técnicas y la propuesta en el modelo de este informe es que todos los estudios tienen una cota máxima de timeslots a trabajar (ya sea obtenidas con hiper-heurísticas u otra manera), en cambio la propuesta en este informe es una cantidad infinita de timeslots (que claramente no puede superar a la cantidad de exámenes).

Hay estudios que no abordan los datasets de Toronto y Nottingham, y resuelven problemas más prácticos que teóricos, donde abordan la existencia de salas de clases limitadas, con un número de sillas limitadas, profesores encargados del examen, etc. Por lo que la definición del problema es más realista y los modelos matemáticos de aquellos estudios son por tanto mucho más complejos y restringidos.

Cabe señalarse que la gran mayoría de las técnicas vistas han resultado ser híbridos con algún algoritmo de búsqueda local, y que por tanto, las técnicas de búsqueda local han sido las más estudiadas y con mayores niveles de éxito. En consecuencia por lo anterior, se ha llevado a que esta última década los estudios, más que preocuparse del mejor algoritmo, aborden más las heurísticas del problema o los parámetros que afectan a los movimientos correctos dentro del espacio de búsqueda, lo que ha conllevado a que los estudios se enfoquen cada vez más en las hiper-heurísticas y determinar aquellas heurísticas de bajo nivel que sean las mejores para enfrentar el ETP.

Actualmente las técnicas de vanguardia, al ser híbridos de búsqueda local, tienen el limitante de «tunear» los hiper-parámetros de cada técnica como por ejemplo el número de iteraciones o tiempo de ejecución, temperatura en SA, movimientos de HC, etc. ya que estos afectan directamente a la eficiencia computacional como al resultado de las soluciones.

9.2. Algoritmo, Experimentos y Resultados

La dificultad en la implementación de todo algoritmo reparador o de búsqueda local radica en la toma de decisiones sobre cómo representar la solución de un problema, la elección de una función de evaluación (calidad) que exprese correctamente si una solución es mejor que otra y sobretodo la elección de los parámetros y movimientos en el espacio de búsqueda. El algoritmo SA tiene una ventaja por sobre los demás en que al tener mayores probabilidades de diversificar con temperaturas altas, suele encontrar soluciones de buena calidad rápidamente, y luego se dedica a intensificarlas a medida que decrece la temperatura. Lo anterior no asegura encontrar un óptimo global, pero sí es muy probable que la solución encontrada sea de buena calidad y obtenida en un tiempo de ejecución mucho menor que el de una técnica completa.

La conclusión más importante, como se pudo observar en **Cuadro 1** de los **Resultados**, es que la elección de un movimiento adecuado en el espacio de búsqueda determina qué tan rápido y con qué calidad minimiza. Los movimientos 1 y 4 fueron los mejores, pero el *movimiento*1 se mueve a pasos tan pequeños a lo largo del espacio de búsqueda que no permite aprovechar la diversificación que ofrece el algoritmo SA para las altas temperaturas iniciales, en cambio el *movimiento*4 aprovecha sus circunstancias estocásticas de decisión para lograr diversificaciones con saltos significativos en el espacio de búsqueda. Cabe señalarse que lo anterior se justifica en que el espacio de búsqueda el cual queda determinado por $EB = |T|^{|E|}$ tiene el potencial de volverse gigantesco a medida que aumentan los exámenes, incluso para $|T|$ pequeños, y dada

la función de evaluación correspondiente a la suma de las penalizaciones sobre ese espacio de búsqueda, no es difícil concluir que la topología de ese espacio está plagada de mínimos locales como en el ejemplo de la **Figura 2**:

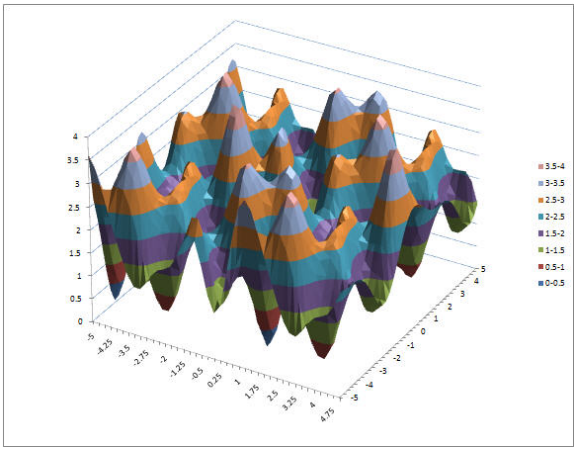

Figura 2: Ejemplo topología del EB con muchos óptimos locales.

Es por lo anterior que el ETP utilizando el algoritmo SA se ve beneficiado al utilizar movimientos que faciliten el trabajo de la diversificación. El *movimiento*4 sólo modifica dos elementos del vector solución, manteniendo intactos todos los demás, por lo que de todas formas intensifica cuando encuentra una mejor solución como cualquier otro algoritmo de búsqueda local y, a la vez, ofrece posibilidades de diversificar escapando a zonas con mejor calidad..

Parecido a lo que proponen **Kusumawardani et al.** [34] ($alpha = 0{,}85$), se encontró que el mejor ratio de enfriamiento fue $alpha = 0{,}8$. Con ratios más altos el algoritmo realizaría más iteraciones y existe una tendencia a pensar que, por ende, se encontrarían soluciones de mejor calidad, pero dada la naturaleza estocástica de *movimiento*4 es posible encontrar soluciones de buena calidad en etapas tempranas del proceso iterativo. Lo que sí se pude asegurar, gracias al **Cuadro 7**, es que a más iteraciones es más probable que un mismo algoritmo ejecutado varias veces converja a soluciones similares. **Kusumawardani et al.** [34] propusieron una cantidad $iter\_max1 = 3500$ como la mejor en términos de calidad-tiempo, y el autor de este documento está de acuerdo a esa propuesta por lo que se explicó en los Resultados de los **Cuadros 7 y 8**.

Con todo lo anterior explicado, se puede concluir de este documento que la elección de los parámetros, movimientos y representación del ETP influyen enormemente en la eficiencia del algoritmo Simulated Annealing. Ejemplo de lo anterior es cómo se decidió crear la solución inicial del problema a través de la decisión greedy de asignarle el timeslot más pequeño al examen con más conflictos, y así sucesivamente con el resto de exámenes. Recordando que el máximo teórico de la cantidad de timeslots es el número de exámenes, haber creado la solución inicial de la forma greedy permite acotar la cantidad de timeslot a una cantidad máxima abordable en términos computacionales (pensando en la explosión del EB) obteniéndose una solución que el autor de este documento ha denominado como «la peor de las buenas soluciones» y que luego el algoritmo SA se encarga de reparar.

En definitiva y conforme a la evidencia empírica, el autor de este documento propone como heurísitica del ETP ocupar el *movimiento*4 en cualquier algoritmo reparador, ya que no sólo SA se vería beneficiado de este movimiento.

Referencias

[1] Joo Tan y col. "A survey of the state-of-the-art of optimisation methodologies in school timetabling problems". En: *Expert Systems with Applications* (sep. de 2020), pág. 113943. DOI: 10.1016/j.eswa.2020.113943.

[2] Hamza Turabieh y Salwani Abdullah. "An integrated hybrid approach to the examination timetabling problem". En: *Omega* 39 (dic. de 2011), págs. 598-607. DOI: 10.1016/j.omega.2010.12.005.

[3] Michael W. Carter y Gilbert Laporte. "Recent developments in practical examination timetabling". En: *Practice and Theory of Automated Timetabling*. Ed. por Edmund Burke y Peter Ross. Berlin, Heidelberg: Springer Berlin Heidelberg, 1996, págs. 1-21. ISBN: 978-3-540-70682-3.

[4] Copyright Stichting, Mathematisch Centrum y Andrea Schaerf. "A Survey of Automated Timetabling". En: *Artificial Intelligence Review* 13 (ene. de 1996). DOI: 10.1023/A:1006576209967.

[5] Rong Qu y col. "A survey of search methodologies and automated system development for examination timetabling". En: *J. Scheduling* 12 (feb. de 2009), págs. 55-89. DOI: 10.1007/s10951-008-0077-5.

[6] D. Welsh y M. Powell. "An upper Bound to the chromatic number of a graph and its application to timetabling problems". En: *Computer J.* 10 (ene. de 1967). DOI: 10.1093/comjnl/10.1.85.

[7] E.K. Burke y J.P. Newall. "Solving Examination Timetabling Problems through Adaption of Heuristic Orderings: Models and Algorithms for Planning and Scheduling Problems (Editors: Philippe Baptiste, Jacques Carlier, Alix Munier, Andreas S. Schulz)". En: *Annals of Operations Research* 129 (jul. de 2004). DOI: 10.1023/B:ANOR.0000030684.30824.08.

[8] Hishammuddin Asmuni, Edmund Burke y Jonathan Garibaldi. "Fuzzy Multiple Ordering Criteria". En: (ago. de 2004).

[9] Sally Brailsford, Chris Potts y Barbara Smith. "Constraint Satisfaction Problems: Algorithms and Applications". En: *European Journal of Operational Research* 119 (dic. de 1999), págs. 557-581. DOI: 10.1016/S0377-2217(98)00364-6.

[10] Liam Merlot y col. "A Hybrid Algorithm for the Examination Timetabling Problem". En: vol. 2740. Ago. de 2003, págs. 207-231. DOI: 10.1007/978-3-540-45157-0_14.

[11] A. Tuson y Pascal Van Hentenryck. "The OPL Optimization Programming Language". En: *The Journal of the Operational Research Society* 51 (mayo de 2000), pág. 649. DOI: 10.2307/254202.

[12] Siti Khatijah Nor Abdul-Rahim, Andrzej Bargiela y Rong Qu. "Hill Climbing versus genetic algorithm optimization in solving the examination timetabling problem". En: *ICORES 2013 - Proceedings of the 2nd International Conference on Operations Research and Enterprise Systems* (ene. de 2013), págs. 43-52.

[13] Yuri Bykov y Sanja Petrovic. "A Step Counting Hill Climbing Algorithm applied to University Examination Timetabling". En: *Journal of Scheduling* 19 (abr. de 2016). DOI: 10.1007/s10951-016-0469-x.

[14] Luca Di Gaspero y Andrea Schaerf. "Tabu Search Techniques for Examination Timetabling". En: sep. de 2001, págs. 104-117. DOI: 10.1007/3-540-44629-X_7.

[15] Luca Di Gaspero. "Recolour, Shake and Kick: a recipe for the Examination Timetabling Problem". En: ene. de 2002.

[16] Luis Paquete y Thomas Sttitzle. "Empirical Analysis of Tabu Search for the Lexicographic Optimization of the Examination Timetabling Problem". En: (mar. de 2003).

[17] Duong Tuan Anh y Kim-Hoa Lam. "Combining Constraint Programming and Simulated Annealing on University Exam Timetabling." En: ene. de 2004, págs. 205-210.

[18] E. Burke y col. "A time-predefined local search approach to exam timetabling problem". En: *IIE Transactions* 36 (jun. de 2004), págs. 509-528. DOI: 10.1080/07408170490438410.

[19] Salwani Abdullah y col. "Investigating Ahuja–Orlin's large neighbourhood search approach for examination timetabling". En: *OR Spectrum* 29 (abr. de 2007), págs. 351-372. DOI: 10.1007/s00291-006-0034-7.

[20] K Sheibani. "An evolutionary approach for the examination timetabling problems". En: ene. de 2002, págs. 387-396.

[21] Tony Wong, Pascal L y Paul Gely. "Final Exam Timetabling: A Practical Approach". En: *Canadian Conference on Electrical and Computer Engineering* 2 (nov. de 2002). DOI: 10.1109/CCECE.2002.1013031.

[22] Pascal Côté, Tony Wong y Robert Sabourin. "A Hybrid Multi-objective Evolutionary Algorithm for the Uncapacitated Exam Proximity Problem". En: jul. de 2004, págs. 294-312. DOI: 10.1007/11593577_17.

[23] Özgür Ülker, Ender Özcan y Emin Korkmaz. "Linear Linkage Encoding in Grouping Problems: Applications on Graph Coloring and Timetabling". En: vol. 3867. Ago. de 2006, págs. 347-363. DOI: 10.1007/978-3-540-77345-0_22.

[24] E. Burke, J. Newall y R. Weare. "A memetic algorithm for university exam timetabling". En: vol. 1153. Ene. de 2006, págs. 241-250. ISBN: 978-3-540-61794-5. DOI: 10.1007/3-540-61794-9_63.

[25] Zahra Naji-Azimi. "Comparison of metaheuristic algorithms for Examination Timetabling Problem". En: *Journal of Applied Mathematics and Computing* 16 (mar. de 2004), págs. 337-354. DOI: 10.1007/BF02936173.

[26] Kathryn Dowsland y Jonathan Thompson. "Ant colony optimization for the examination scheduling problem". En: *Journal of the Operational Research Society* 56 (abr. de 2005). DOI: 10.1057/palgrave.jors.2601830.

[27] Michael Eley. "Ant Algorithms for the Exam Timetabling Problem". En: ago. de 2006, págs. 364-382. DOI: 10.1007/978-3-540-77345-0_23.

[28] Edmund Burke, Yuri Bykov y Sanja Petrovic. "A Multicriteria Approach to Examination Timetabling". En: ago. de 2000, págs. 118-131.

[29] Sanja Petrovic y Yuri Bykov. "A Multiobjective Optimisation Technique for Exam Timetabling Based on Trajectories". En: ago. de 2002, págs. 181-194. DOI: 10.1007/978-3-540-45157-0_12.

[30] Burak Bilgin, Ender Özcan y Emin Korkmaz. "An Experimental Study on Hyper-heuristics and Exam Timetabling". En: vol. 3867. Ago. de 2006, págs. 394-412. DOI: 10.1007/978-3-540-77345-0_25.

[31] Edmund Burke y col. “A Graph-Based Hyper Heuristic for Timetabling Problems”. En: *European Journal of Operational Research* 176 (jun. de 2007), págs. 177-192.

[32] Rong Qu y Edmund Burke. “Hybridizations within a graph based hyper-heuristic framework for university timetabling problems”. En: *Journal of the Operational Research Society* 60 (oct. de 2008).

[33] Rong Qu y E.K. Burke. “Adaptive decomposition and construction for examination timetabling problems”. En: *Multidisciplinary International Scheduling: Theory and Applications (MISTA'07)* (ene. de 2007), págs. 418-425.

[34] D. Kusumawardani, A. Muklason y V. A. Supoyo. “Examination Timetabling Automation and Optimization using Greedy-Simulated Annealing Hyper-heuristics Algorithm”. En: *2019 12th International Conference on Information Communication Technology and System (ICTS)*. 2019, págs. 1-6. DOI: 10.1109/ICTS.2019.8850932.