Inteligencia Artificial

Informe Final: Problema "Examination Timetabling Problem"

December 18, 2022

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):	_____

Abstract

La calendarización de evaluaciones es un problema cotidiano para las instituciones educativas y producto de su complejidad existen hartos estudios sobre cómo abordar estos problemas, los cuales han sido clasificados como ETP por la comunidad. Muchas técnicas han sido empleadas para resolver estos problemas a lo largo del tiempo obteniendo diversos resultados, en la actualidad, la mezcla de varias de estas técnicas junto con heurísticas ha permitido ver avances en el medio. En este documento se abordará la implementación de un algoritmo basado en Forward Checking para la resolución de 4 instancias para analizar la efectividad de esta.

1 Introducción

La calendarización de actividades es un tema complejo y en muchas ocasiones parece ser imposible, este es un problema que habitualmente sufren las instituciones educacionales al momento de planificar las evaluaciones de los estudiantes y esto es producto de la cantidad de elementos que se deben tomar en consideración al momento de realizar los calendarios, puesto que se deben cumplir una serie de compromisos con otras entidades pero también existen condiciones o políticas que cada institución considera fundamentales para un correcto desarrollo de las actividades académicas. La compatibilización de todos estos factores llevo a la formulación y el estudio de los problemas ETP, estos problemas se caracterizan por ser NP-Completo lo que ha llevado a diseñar diversos modelos para abordar y dar solución a las distintas realidades de diversas instituciones a lo largo del tiempo.

A lo largo de este documento se explorará en detalle los fundamentos detrás de este problema, modelos planteados y soluciones propuestas por el medio con el fin de comprender la relevancia que tiene en las comunidades educativas y que ha llevado a una serie de estudios a lo largo del

tiempo que en la actualidad aún son relevantes. Para ello, en la sección 2 se comenzará hablando en profundidad sobre en qué consiste y compone al problema, además de problemas similares que comparten elementos en común con el objeto de estudio, en la sección 3 se abordará la evolución del problema, los inicios, soluciones, metodologías y el estado actual con respecto a las investigaciones que se han realizado al respecto, en la sección 4 se introducirá un modelo que ha permitido la confección de este trabajo, en la sección 5 se introducirá la representación que se empleará en el algoritmo basado en Forward Checking implementado para 4 instancias de problemas ETP, en la sección 6 se entrará en detalle sobre cómo funciona dicho algoritmo y que lo conforma, en la sección 7 se planteará la metodología de trabajo para analizar la implementación, en la sección 8 se expondrán los resultados obtenido por el algoritmos, para finalmente en la sección 9 entregar las apreciaciones sobre los resultados y como estos se corresponden o no a los esperado, junto con lineamientos a mejorar.

2 Definición del Problema

ETP es un problema a lo largo del tiempo ha tenido distintas formas de ser planteado, pero todos tienen una base en común, buscar minimizar las jornadas o cantidad de bloques en la que se desarrollan los exámenes en un periodo evaluativo usualmente a finales de semestre sujeto a una serie de restricciones fundamentales y otras de menor prioridad, esto producto de la necesidad de poder realizar las labores administrativas para dar por finalizado un periodo lectivo y entregar la información a las autoridades para tramitar temas de subvenciones, becas, presupuestos entre otros compromisos internos como externos.

Para poder tener un conocimiento general del problema se empezará viendo la forma más básica del asunto, en la cual a partir de un conjunto de exámenes que deben ser agendados y un listado de alumnos con las evaluaciones para las que se encuentran habilitados a rendir, se debe calendarizar las evaluaciones evitando que los estudiantes tengan conflictos con tener más de una evaluación al mismo tiempo, se podría decir que esto es lo mínimo para poder plantear el problema, pero la realidad ha dado a conocer que hay más restricciones que acotan el espacio de búsqueda y estos pueden variar por cada lugar donde se intente replicar, sin embargo, hay puntos en común que se han podido recopilar a lo largo del tiempo, para ello se han definido 2 tipos de restricciones que el problema debe cumplir.

En primer lugar, existen las restricciones ***Hard*** las cuales determinan si es posible encontrar una solución al problema:

- Los estudiantes no pueden tener más de una evaluación en el mismo periodo de tiempo.
- La asignación de estudiantes para rendir un examen en una sala determinada no exceda la capacidad de esta.
- Una sala solo puede tener un examen asignado en un mismo periodo de tiempo
- Respetar la precedencia entre exámenes. Ej: Examen 1 antes/después que Examen 2.
- Respetar asignaciones de examen-sala establecidas previamente. Ej: Examen 1 en Sala 100.
- Exámenes con bloques predefinidos

Por otra parte existen las restricciones de tipo ***Soft*** que se encarga de "penalizar" las soluciones encontradas:

- Los estudiantes no pueden tener 2 evaluaciones seguidas

- Las evaluaciones deben estar distribuidas a lo largo del periodo evaluativo
- Los estudiantes no pueden tener 2 evaluaciones en un mismo día

Esto es solo una manera general de englobar las restricciones más frecuentes que se han podido encontrar en el medio, y la inclusión de una o más de estas, incluso, restricciones no planteadas anteriormente van a depender del contexto detrás del planteamiento del problema.

Lo anterior implica que los elementos definidos inicialmente para modelar el problema no resultan ser suficientes, puesto que dependiendo del enfoque se pueden tener variantes de ETP que busquen solo satisfacer restricciones Hard, otros que busquen minimizar las penalizaciones de las restricciones Soft y finalmente planteamientos que relajan las restricciones. Para poder ampliar el repertorio se considera una matriz de conflicto entre exámenes, donde las casillas i, j representan con un 1 si existen estudiantes que tengan conflicto con la evaluación i y j, mientras que con un 0 en el caso contrario que se construye a partir del conjunto de exámenes y alumnos, variables que definen si una evaluación es parte un bloque determinado, que sala fue asignada a cierto examen, constantes como la capacidad de las salas, el tiempo en el que inicia un bloque, entre otras dependiendo como el problema ha sido planteado. Junto con ello los objetivos de estos problemas ya no se reducen a minimizar el periodo de tiempo que toma aplicar todas las evaluaciones, sino que se consideran minimizaciones de las penalizaciones asociadas a las restricciones Soft, minimizar el estrés de los estudiantes dado que no siempre es posible satisfacer todas las restricciones, distribuir eficientemente los recursos, etc.

No obstante, los problemas de tipo ETP no solo se enfrascan en ambientes académicos, existen otras aplicaciones en otros ámbitos como la medicina para la asignación de personal en operaciones, en transporte para asignación de recorridos y conductores, y cualquier otra actividad administrativa que implique asignaciones. Por lo mismo, el desarrollo de heurísticas para estos problemas implica que puedan ser adaptados a la solución de las distintas variantes de ETP debido a las similitudes entre estos problemas como ha sido el caso de Nurse Rostering[1].

3 Estado del Arte

El problema de la calendarización de exámenes está completamente arraigado a las instituciones educaciones, mas centradas en universidades dada la complejidad que yace en sus estructuras docentes, sin embargo, la transición del proceso manual de agendar a la automatización producto de los avances computacionales trajo consigo los primeros modelos que se han documentado de ETP, en la década de los 60 se comienzan a resaltar las bondades del uso computacional para poder modelar sistemas capaces agendar los periodos de exámenes tomando en cuenta restricciones físico-temporales.[2] Estas soluciones lejos de ser eficientes producto de la tecnología de la época, resultaron ser frutíferas al reducir tantos los errores humanos como los tiempos en la confección de los calendarios de final de semestre.

Al mismo tiempo se comenzaban a emplear la teoría de coloreo de grafos para determinar el límite inferior de una solución al problema de scheduling[3], lo que en un principio no parece resolver el problema, si daba información útil para instancias del problema simplificadas, pero no resultaba ser practico para caso más reales, con el paso del tiempo esta técnica se utilizará como heurística en la confección de nuevos algoritmos que buscabaran mejorar los resultados obtenidos por su predecesores.

Otro de los avances resulto ser la programación lógica a partir de restricciones [4] obteniendo resultados favorables en lo que respecta a problemas con gran número de restricciones y que las soluciones de la época basadas en heurísticas de KP, TSP o técnicas como CG no podían

asimilar correctamente.

Justamente el uso de heurísticas ha sido fundamental para producir los métodos más reconocidos para abordar este problema, dada su gran complejidad, es común ver reducciones a modelos de grafos para obtener relaciones que ayuden a minimizar los conflictos, sin embargo, como fue mencionado anteriormente estos métodos no eran eficaces para generalizar un algoritmo capaz de aplicarse a cada institución puesto que las restricciones asociadas a cada una pueden presentar gran cantidad de variaciones lo que volvió en muchas ocasiones algunas formulaciones más ineficientes que otras.[5] Dentro de este universo se puede encontrar los algoritmos basados en metaheurísticas como Tabu Search, Ant Colony Optimization, Evolutionary Algorithms, Simulated Annealing, Local Search y Great Deluge[6].

Sin embargo, estas no son las únicas formulaciones que se han encontrado durante los años y la existencia de algoritmos que se encargan de encontrar soluciones en múltiples pasos han sido más relevantes en la actualidad, para ello utilizan técnicas anteriormente descritas para obtener candidatos a soluciones las cuales a partir de un proceso iterativo refinan la búsqueda en el espacio para obtener resultados que consideren también a las restricciones Soft, de esta manera los algoritmos híbridos o basados en hiperheurísticas son los casos de estudio más comunes en la actualidad, dada su capacidad de obtener soluciones localmente y a su proceso iterativo a partir de la elección de la heurística correcta lo que ha permitido resolver una mayor cantidad de variante de ETP acercándose a la ansiada generalidad y obteniendo mejores resultado en casos de estudio. [7][8][9]

Las bondades de los métodos basados en hiperheurísticas han demostrado ser ideales para ser considerados los mejores métodos en la actualidad y aun es discutible cual es la técnica definitiva, sin embargo, la que mejores resultados ha obtenido han resultado ser aquellas basadas en Great Deluge, lo que ha llevado a variaciones o extensiones de esta que han logrado superar registros con respecto a algoritmos de la misma clasificación [10][11][12], entregando mejores soluciones reduciendo la penalización al mismo tiempo, sin embargo, aún se encuentran limitadas por el tiempo empleado en la convergencia de soluciones locales optimas mientras que disciernen entre soluciones infactibles.

4 Modelo Matemático

Dada la complejidad del problema, existe una infinidad de modelos, algunos con características similares y otros que han sido extensiones de otros ya existentes, en este trabajo se optará por estudiar un modelo empleado en un caso real, más precisamente en la Universidad Kebangsaan, Malaysia[13]

- Función Objetivo

$$\min \text{F} = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} * Penalty(t_i, t_j)}{M}$$

$$Penalty(t_i, t_j) : \begin{cases} 2^{(5-\Delta t)(2-\Delta d)} & \text{if } \Delta t \leq 5 \text{ or } \Delta d \leq 2 \\ 0 & \text{else} \end{cases}$$

- Variables

$$E_i : i\text{-esimo examen a agendar}$$

$$i \in \{1, ..., N\}$$

$$t_i : \text{Timeslot en el cual fue agendado el examen} E_i,\ t_i \in \{1, ..., T\}$$

d_i : Día en el que fue agendado el examen E_i, $d_i \in \{1, ..., D\}$

$$\lambda_{is} = \begin{cases} 1 & \text{if } E_i \text{ ha sido asignado al timeslot s} \\ 0 & \text{else} \end{cases}$$

$$x(t_i, t_j) = \begin{cases} 1 & \text{if } t_i = t_j \\ 0 & \text{else} \end{cases}$$

S_t : Cantidad de alumnos a rendir exámenes en el timeslot t, $S_t \in \{1, ..., T\}$

- Parámetros

c_{ij} : Número de alumnos inscritos para rendir el examen E_i como E_j

M : Número de estudiantes

N : Número de exámenes a agendar

T : Número de timeslot disponibles

D : Número de días disponibles

A : Asientos disponibles por timeslot

- Restricciones

$$\sum_{s=1}^{T} \lambda_{is} = 1, \ \forall i \in \{1, ..., N\} \tag{1}$$

$$\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} * x(t_i, t_j) = 0 \tag{2}$$

$$S_t \leq A, \forall t \in \{1, .., T\} \tag{3}$$

$$d_k \neq d_i, \forall i, j \in \{1, ..., N\} \tag{4}$$

con $c_{ij} \neq 0; c_{ik} \neq 0; t_i = x; i \neq j \neq k; [t_j = x + 1$ OR $t_j = x - 1]$ AND $d_i = d_j$

A partir de la función objetivo se puede evidenciar el tipo de modelo planteado, en este caso, este busca reducir la penalización producida por las restricciones Soft permitiendo distribuir las evaluaciones a lo largo de las jornadas que están planificadas para tomar exámenes, esto en cierta manera se ve representada por la función de penalización donde mientras mayor sea la distancia entre timeslot o días, la penalización es reducida considerablemente, permitiendo que los estudiantes puedan prepararse de mejor manera.

Por otra parte las restricciones fundamentales o Hard se ven reflejadas en las restricciones del modelo, la primera hace referencia a que los exámenes solo pueden ser asignados en un único timeslot, verificando que el examen no haya sido agendado en más de un timeslot, por otra parte la segunda restricción hace referencia a que los estudiantes no pueden tener 2 evaluaciones en un mismo timeslot, esto a partir de la matriz de conflictos o mediante la propia asignación de exámenes, la tercera restricción se asegura que no puedan haber más estudiantes próximos a rendir una evaluación que asientos en el establecimiento para un timeslot dado, finalmente, la última restricción se encarga de que los estudiantes no tengan más de 3 evaluaciones consecutivas por día.

5 Representación

Puesto que la técnica empleada para la resolución del problema corresponde a una técnica completa, la representación de las soluciones fue construida para poder almacenar información necesaria para analizar la factibilidad de estas.

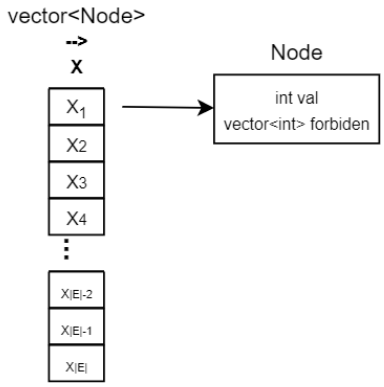

Figure 1: Representación basada en un vector de Nodes

Para ello se introduce el objeto Node, el cual es la pieza básica de la representación, esta se compone de 2 campos, un campo int valor, el cual corresponde al valor que tomara la prueba a la que el objeto representará a lo largo de las distintas iteraciones del algoritmo, es decir, los valores del dominio, además, de un vector de int denominado forbiden que contiene aquellos valores del dominio que fueron filtrados en pasos previos, es decir, almacena los valores los cuales no pueden ser asignados al atributo valor del objeto en sí, de esta forma se puede conocer de manera implícita que valores del dominio son válidos sin tener que almacenarlos reduciendo el almacenamiento necesario en casos donde los valores filtrados no son considerables.

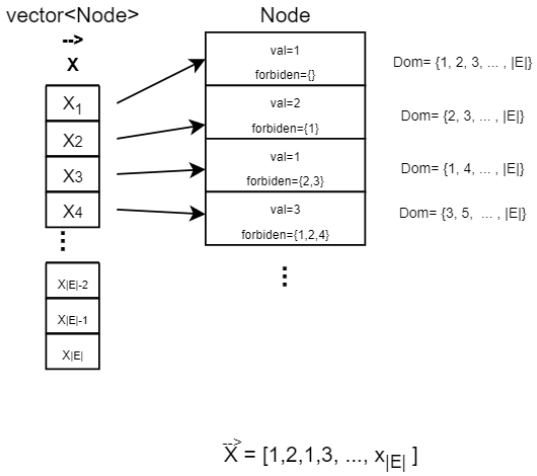

Figure 2: Ejemplo de la representación

Ahora bien, ya definido el objeto Node, la representación de las soluciones está dada por un vector de tipo Node, de tamaño E, el cual corresponde a la cardinalidad de exámenes a agendar,

donde la posición i-esima corresponde al i-esimo+1 examen, dado que los identificadores de examen comienzan desde 1, en otras palabras, la posición 0 del vector corresponde al Node asociado al examen 1 y así para el resto de exámenes. Este vector de Node cumple la función del vector de planificación, donde cada Node representa a la variable x_i abstrayéndose de la existencia del vector forbiden, puesto que su existencia es con fines de manejar la restricción de que ningún estudiante puede tener 2 exámenes en el mismo timeslot.

6 Descripción del algoritmo

La implementación de la técnica Forward Checking para este problema está basada en principios de recursión para maneja los niveles buscando simular el comportamiento de un árbol, pero además a segmentar los elementos de la implementación general en diversas funciones para tener control sobre el debugging, principalmente por el uso de punteros sobre las estructuras que esta sujetas a modificación constante como es el vector de Nodes, así como también poder controlar los puntos donde se pueden producir errores con el fin de separar lo que funciona sin problemas de lo defectuoso.

```
main:
    alumnos <-- init_alumnos(file_alumnos)
    timeslot <-- Cantidad de examenes
    size <-- Cantidad de examenes
    matriz_conflictos <-- conflictos(alumnos, size)
    solucion <-- Vector de Node de tamano size
    FC(solucion, alumnos, matriz_conflictos, 0, size)
```

El main del algoritmo es bastante sencillo, sin embargo, gran parte de lo que hace funcionar a la implementación de FC es debido a lo que se produce en este punto, puesto que aquí es donde se inicializan todas las estructuras que emplea la función FC encargada de aplicar el algoritmo en sí, sin embargo, para tener un mayor contexto se van a describir brevemente sin entrar en demasiados detalles, salvo que lo ameriten.

```
init_alumnos(file):
    alumnos <-- Vector de tipo Alumno
    alumno <-- Objeto Alumno
    f <-- open(file)
    aux, test <-- read(f)
    Anadir test a alumno
    while(f no este vacio):
        repeat:
            alu, test <-- read(f)
            if (aux no es igual a alu):
                Anadir alumno a alumnos
                alumno <-- Otro objeto Alumno
                Anadir test a alumno
                aux <-- alu
            else:
                Anadir test a alumno
    Anadir alumno a alumnos
    return alumnos
```

La función init_alumnos se encarga principalmente de inicializar un vector de tipo Alumno, que es un objeto similar a Node, pero que solo contiene un vector con el id de las pruebas que

debe rendir el alumno i y que se inicializa a partir de un archivo que contiene dicha información. Este verifica que siempre se esté trabajando con el mismo alumno y en caso contrario añade lo documentado al vector y continua con otro objeto repitiendo el mismo proceso hasta que el archivo se encuentre vacío.

```
conflict(alumnos, size):
    conflictos <-- matriz de 0 de tamano size
    for(i in range(size)):
        for(alumno in alumnos):
            aux <-- Obtener las pruebas de alumno
            if(prueba_i en aux):
                Cambiar por 1 la casilla que representan las otras pruebas
                del alumno con respecto a la prueba i
    return conflictos
```

La función conflict busca inicializar la matriz de conflictos a partir de la información que contiene el vector de alumnos, lo cual mediante una iteración sobre este, busca las pruebas que tengan conflictos para representarlas en la matriz, por ejemplo, si un alumno tiene inscrita las pruebas 1 y 3, la casilla (1,3) y (3, 1) de la matriz van a contener un 1 indicando el conflicto.

```
FC(nodes, alumnos, matriz_conflicto, indice, maximo):
    for (i=1 to maximo):
        if (i es un valor valido en el dominio):
            Asignar i al nodo en la posicion indice
        else:
            continue
        if (indice == max-1)
            Evaluar Solucion
        else:
            Filtrar dominios del resto de nodos
            Guardar sobre que nodos se hizo filtro
            FC(nodes, alumnos, matriz_conflicto, indice+1, maximo)
            Deshacer filtro
```

La función FC, la cual contiene la lógica detrás de la implementación de la tecnica, no es un calco, más bien es una adaptación al contexto del problema ETP, la idea de que sea una función recursiva es aprovechar lo similar que es FC con Backtracking en algunos aspectos, pero además, por la propia naturaleza de la técnica, puesto que se maneja filtros que deben ser deshechos, el trabajar con recursión, permite tener dichos filtros en el contexto en el cual fueron generados, de manera que estos no tengan conflictos con otros filtros realizados en otros niveles del árbol, para ello se realiza una asignación implícita del nivel en cada llamado de manera que se pueda referencia el nivel a un node particular, por ejemplo la primera ejecución realizada en el main, vincula la primera prueba con el primer nivel y así sucesivamente para cada node, de manera que cuando se llegue al último node sea este quien haga el llamado a evaluar la solución puesto que de él depende que la solución este completa. Ahora bien, lo primero que se realiza es definir el intervalo de evaluación del dominio, el cual por definición corresponde a la cardinalidad de exámenes, dado que el algoritmo se construyó con el supuesto de que existen tantos timeslot como exámenes, por lo mismo no existe un mecanismo de retorno anticipado como dicta la técnica, ahora bien, a pesar de que recorre todo el dominio, esto no implica que se considera la totalidad de este como correcto, puesto que el dominio valido es aquel el cual fue filtrado.

Para ello, lo primero que se realiza es verificar la restriccion de que los alumnos no pueden tener 2 pruebas al mismo tiempo, para ello se revisa dentro de los valores filtrados si el valor de la iteracion es un valor valido del dominio, en caso de serlo se asigna, cumpliendo con el principio de que los dominios son filtrados, en caso contrario, se sigue buscando el valor del dominio que corresponde a asignar. Cabe destacar que la unica variable que obligatoriamente recorre todo el dominio asignandolo es la correspondiente a la primera prueba, puesto, que es el punto del cual se inician las soluciones, mientras que el resto de variables dependen de la cantidad de filtros a la que fueron sometidos.

Una vez definido el timeslot asignado a la prueba, se produce a verificar el punto sobre el cual se encuentra la recursion, si esta al nivel de las hojas, se comprueba la solucion, si la solucion encontrada resulta disminuir el valor de la funcion objetivo, es decir, disminuir la cantidad de timeslots, se obtiene el penalty asociado y se registra, en cambio, si no logra disminuir la cantidad de timeslots, pero si lo iguala y a la vez resuelve de mejor manera la restriccion soft, en otras palabras, disminuye el penalty, tambien es registrada, en cualquiera de los casos, los anteriores optimos son actualizados para seguir buscando soluciones. En cambio, si la recursion esta en niveles superiores, se realizan los filtros sobre los dominios que lo ameriten a partir de la matriz de conflictos, se deja constancia sobre que dominios se realizo un filtro, para posteriormente ser desechos cuando el nivel posterior haga return, de manera que se pueda asignar un nuevo valor del dominio y continuar explorando nuevas soluciones.

7 Experimentos

Con el fin de analizar los alcances de la implementacion, se realizaron 4 experimentos con instancias de distintos tamaños para determinar el comportamiento del algoritmo en la busqueda de nuevos optimos, para ello, cada una de las pruebas fue realizada en una maquina virtual Ubuntu 20.04 LTS con 4GBs de RAM y el acceso a un solo nucleo de un procesador Intel i5-8300H.

Mediante los experimentos, se busca identificar la cantidad de veces que la solucion fue mejorada con respecto a la inicial en un lapsus de 10 minutos, puesto que la tecnica en cuestion es completa, se quiere verificar la suceptibilidad del algoritmo a no realizar mejoras en la medida que el tamaño de las restricciones aumenta cuando la cantidad de pruebas es variable, para ello, la cantidad de optimizaciones se va a dividir en 2 ambitos, cuando se mejora con respecto a la disminucion de los timeslots y por otra parte con respecto a la satisfaccion de la restriccion soft, es decir, cuando se disminuye el penalty.

Para ello se han elegido las siguientes instancias:

- St.Andrews83: 139 Examenes, 5751 inscritos repartidos en 622
- EdHEC92: 81 Examenes, 10632 inscritos repartidos en 2823 estudiantes
- TorontoE92: 184 Examenes, 11793 inscritos repartidos en 2750 estudiantes
- EarlHaig83: 190 Examenes, 8109 inscritos repartidos en 1125 estudiantes

8 Resultados

A partir del proceso experimental llevado a cabo, se pudieron recopiar los siguientes resultados, ademas, se adjunta el optimo encontrado en la literatura para cada uno de las instancias de manera de poder constrastar el funcionamiento del algoritmo en el periodo maximo de ejecucion.

Instancia	Timeslots	Mejor Timeslot	Penalizaciones	Mejor penalizacion	Timeslot optimo
St.Andrews83	1	13	2	2.357	13
EdHEC92	1	22	0	0.023	18
TorontoE92	1	13	1	0	10
EarlHaig83	1	29	1	0.447	24

Como se puede contrastar, los resultados resultan ser bastante dispares, sin embargo, todos tiene algunos elementos en comun, en particular, el caso mas curioso es del St.Andrews, el cual obtiene la cantidad de timeslots optima en su primera solucion, sin embargo, esto se puede deber a muchas razones, como por ejemplo, que el optimo en termino de timeslots se encuentre en mas de una ocasion, el cual en dicho caso se tiende a diferenciar entre la mejor solucion y una no optima mediante el calculo del penalty como justamente se puede interpretar de los resultados obtenidos, de las 3 soluciones optimas encontradas, en al menos 2 ocasiones se mejoro el penalty, por lo cual se logran identificar al menos 3 soluciones con 13 timeslots, es posible que exista otra solucion que logre minimizar aun mas el penalty que este en algun otro punto del arbol el cual no se puede alcanzar en un tiempo prudente dado como funciona la tecnica.

En el caso de EdHEC, no se logro en ninguna ocasion encontar una optimizacion a la primera solucion, sin embargo, esto se debe al valor particularmente pequeño del penalty obtenido, por lo cual, el camino mas favorable para encontrar una optimizacion residiria en minimizar la cantidad de timeslots, que dada la cantidad de examenes y la cantidad de inscripciones se podria haber encontrado en un tiempo mas prudente, sin embargo, bastante alejado del tiempo maximo definido para el experimento dado que la cantidad de iteraciones necesarias para producir un cambio considerable en el vector de planificacion es bastante alto producto de la cantidad de examenes y la interpretacion como estos de niveles en el arbol.

La instancia de Toronto tiene algunas similitudes con la instancia de EdHEC, puedo que el nuevo penalty encontrado fue 0, por lo cual el unico camino para producir una nueva optimizacion es minimizar la cantidad de timeslots, sin embargo, la cantidad de examenes en esta ocasion es mucho mas considerable, siendo mas del doble que el anterior, lo cual es bastante improbable encontrar una optimizacion en el corto plazo, sin embargo, dada la cantidad de restricciones que puede encontrar en el gran numero de inscripciones, es notable que la primera solucion no este tan alejada del optimo, siendo la mas proxima sin contar a St.Andrews de todas las instancias estudiadas.

Finalmente para EarlHaig, que en terminos de contexto es bastante similar a Toronto, presenta los resultados con el peor rendimiento considerando la optimizacion de los timeslots como criterio principal, siendo el que esta mas alejado del optimo, en esta ocasion el panorama es un tanto mixto, puesto que su volumen de examenes es el mayor de las instancias, va a tener mayor dificultad en realizar una optimizacion, sumado a su baja penalizacion, es probablemente que se necesite una gran numero de iteraciones, las cuales escapan a probar la eficiencia del metodo, pero que al mismo dado que su distancia con el optimo es mayor, tiene mayor probabilidad de encontrar un nuevo optimo prontamente, sin embargo, el funcionamiento del algoritmo inclina la balanza en terminos negativos para el tiempo que podria necesitar.

9 Conclusiones

En general, la implementacion de un algoritmo basado en FC para resolver problemas de tipo ETP puede resultar una buena alternativa para encontrar una solucion inicial sobre la cual se quiera trabajar para realizar optimizaciones utilizando otro metodo, esto debido al alto costo en tiempo que es necesario para que se efectuen mejoras, dado en parte por el recorrido descedente a lo largo de un arbol que considera gran parte de los valores del dominio inicial, si bien

su filtrado anticipado reduce en ocasiones considerablemente dicho conjunto, esto variara con respecto a la naturaleza de la variante del problema que se este buscando resolver.

En particular, la implementacion realizada obtuvo buenos resultados reduciendo el valor inicial de timeslots al momento de encontrar su primera solucion, pero que posterior a ello no obtuvo buenos resultados, teniendo poca variacion a lo largo de las iteraciones y que conlleva a un alto costo en tiempo si se desea encontrar optimizaciones significativas, quedando demostrado el poco impacto que tiene el tamaño del input en este proceso. Si bien el algoritmo no es demasiado complejo, aun tiene espacio para optimizaciones en termino de la eficiencia, como por ejemplo el manejo del dominio filtrado, en vez de realizar evaluaciones por cada valor para identificar si este se encuentra filtrado, se podria realizar un proceso iterativo sobre un conjunto que cruce el dominio original con el filtrado para reducir la cantidad de evaluaciones que el algoritmo realiza para efectuar una asignacion, asi como en terminos de memoria al almacenar la matriz de conflictos, puesto que esta se comporta como una matriz simetrica, se puede realizar una abstraccion y disminuir la dimensionalidad a un estructura que solo almacene los valores una sola vez.

10 Bibliografía

References

[1] E.K. Burke, Graham Kendall, and E. Soubeiga. A tabu-search hyperheuristic for timetabling and rostering. *Journal of Heuristics*, 9:451–470, 12 2003.

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

[3] 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.

[4] P Boizumault, Y Delon, and L Peridy. Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, 26(2):217 – 233, 1996.

[5] Zahra Naji-Azimi. Comparison of metaheuristic algorithms for examination timetabling problem. *Journal of Applied Mathematics and Computing*, 16:337–354, 03 2004.

[6] R. Lewis. A survey of metaheuristic-based techniques for university timetabling problems. *OR Spectrum*, 30:167–190, 2008.

[7] Hamza Turabieh and Salwani Abdullah. An integrated hybrid approach to the examination timetabling problem. *Omega*, 39(6):598 – 607, 2011.

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

[9] Edmund Burke, Moshe Dror, Sanja Petrovic, and Rong Qu. Hybrid graph heuristics within a hyper-heuristic approach to exam timetabling problems. *Operations Research/ Computer Science Interfaces Series*, 29, 01 2005.

[10] 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.

[11] A. K. Mandal and M. N. M. Kahar. Solving examination timetabling problem using partial exam assignment with great deluge algorithm. In *2015 International Conference on Computer, Communications, and Control Technology (I4CT)*, pages 530–534, 2015.

[12] B. McCollum, P. McMullan, A. Parkes, E. Burke, and S. Abdullah. An extended great deluge approach to the examination timetabling problem. 2009.

[13] Masri Ayob, Ariff Malik, Salwani Abdullah, Abdul Hamdan, Graham Kendall, and Rong Qu. Solving a practical examination timetabling problem: A case study. pages 611–624, 08 2007.