ESTUDIO COMPARATIVO DE ESTRATEGIAS HEURÍSTICAS DE GENERACIÓN DE SOLUCIONES PARA EL PROBLEMA DE ASIGNACIÓN DE EXÁMENES

J. Tessore[1,2,4],L. Cicerchia[1,2, 4], L. Esnaola[1, 4], H. Ramón[1, 3,4], C.Russo[1,3,4]

Resumen. Timetabling se refiere a un conjunto de problemas de optimización combinatoria, que intentan asignar recursos, sean aulas, docentes o intervalos de tiempo para distintas necesidades de estudiantes, cursos y exámenes. En el presente trabajo se aborda una de las variantes de este problema que busca agendar exámenes a distintos intervalos de tiempo, cumpliendo con las restricciones de que ningún alumno debe asistir a más de un examen en el mismo momento y, en la medida de lo posible, que tenga el mayor tiempo libre entre las evaluaciones. La cantidad de combinaciones a considerar para una instancia tamaño moderado hacen inviable la búsqueda de la solución óptima, debido al tiempo que demandaría encontrarla. En consecuencia, en este artículo se utilizan distintas estrategias para combinar heurísticas que permiten obtener una buena solución al problema en un intervalo de tiempo reducido. Las heurísticas mencionadas fueron probadas sobre un conjunto de instancias estándar de manera individual así como también combinadas de manera secuencial yjerárquica. En las pruebas realizadas se obtuvieron mejores resultados mediante el método jerárquico. Debido a lo anterior es posible afirmar la superioridad de este último método sobre los demás utilizados en el presente trabajo.

Keywords: Timetabling, Heurísticas, Secuencial, Jerárquica.

Introducción

La asignación de exámenes y cursos, también conocida como el problema de *timetabling*, es de gran interés para la comunidad educativa. Normalmente esta tarea requiere de mucho tiempo si es

[1] Instituto de Investigación y Transferencia en Tecnología (ITT), Escuela de Tecnología, Universidad Nacional del Noroeste de la Provincia de Buenos Aires (UNNOBA), Centro Asociado a la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)

[2] Becario de la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)

[3] Investigador Asociado de la Comisión de Investigaciones Científicas de la Provincia de Buenos Aires (CIC)

[4]{juanpablo.tessore, lucas.cicerchia,leonardo.esnaola, hugo.ramon, claudia.russo}@itt.unnoba.edu.ar

realizada de manera manual y es muy común que los resultados obtenidos se encuentren lejos del óptimo.

Debido a que, para instituciones de un tamaño moderado, la cantidad de variables a tener en cuenta es considerable, la obtención del óptimo no es una alternativa ya que el tiempo que demandaría encontrarlo la hace inviable. Como consecuencia, se han desarrollado un conjunto de métodos que permiten encontrar una solución lo suficientemente buena en un tiempo razonable.

Dentro de estos métodos pueden mencionarse:

- Métodos basados en programación matemática, como por ejemplo programación lineal[1]. Sin embargo, la posibilidad de utilizarlos está limitada por el tamaño de la solución.
- Métodos basados en coloreo de grafos[2], en este enfoque y para el caso de *examinationtimetabling*, los exámenes se representan con nodos y los conflictos con vértices entre los mismos. Existen diversas técnicas dentro de este enfoque que pueden ser utilizadas por sí solas o combinadas.
- Métodos de *clustering* o agrupamiento[3], estos intentan generar grupos libres de conflictos y luego intercambiarlos entre los *timeslots* disponibles para intentar cumplir mejor con algunas de las restricciones del problema.
- Métodos metaheurísticos, estos han ganado popularidad en las últimas décadas. Entre ellos se pueden destacar métodos de búsqueda local como *Hill Climbing*[4], *TabuSearch*[5],*SimulatedAnnealing*[6],Algoritmos Genéticos[7] o Algoritmos de Colonia de Abejas[8]. Estos, en general, comienzan con una o más soluciones iniciales y emplean diferentes técnicas de búsqueda para no quedarse estancados en óptimos locales.

En este trabajo se emplean distintas estrategias para combinar heurísticas de bajo nivel, con el objetivo de obtener una buena solución de manera rápida. Las estrategias adoptadas se detallan en las secciones subsiguientes.

El problema

En este trabajo se analiza una instancia particular del problema de *timetabling,* quesegún[9]consiste en asignar un conjunto de exámenes, cada uno con un grupo específico de estudiantes, a un conjunto dado de intervalos de tiempo. En dicha instancia, denominada "*uncapacitateduniversityexaminationtimetablingproblem*", no existen limitaciones en cuanto a capacidad de las aulas ni disponibilidad de los docentes.

En general, para evaluar la solución obtenida existen dos tipos de restricciones, restricciones duras, que deben cumplirse obligatoriamente, y restricciones blandas, que tienen que cumplirse en la medida de lo posible. Una solución con mayor grado de cumplimiento de estas últimas será de mayor calidad.

Las restricciones duras consisten en que ningún alumno puede asistir a más de un examen en un mismo intervalo de tiempo y, además, que ningún examen puede asignarse a más de un intervalo de tiempo. Una solución que cumpla con estas restricciones será

considerada factible y será considerada infactible o no factible, en caso contrario.

Las restricciones blandas, por su parte, consisten en maximizar el tiempo medio entre exámenes de los alumnos. Una solución que satisfaga mejor este tipo de restricciones, será considerada de mayor calidad.

Para definir formalmente el problema se recurre a las siguientes fórmulas:

$$\min Z = \sum_{n=1}^{N} \sum_{p=1}^{P} C_{np} t_{np} \tag{2.1}$$

Sujeto a:

$$\sum_{p=1}^{P} t_{np} = 1 \, , donde \; n \; \in \{1, \dots, N\} \tag{2.2}$$

$$\sum_{n=1}^{N-1} \sum_{m=n+1}^{N} \sum_{p=1}^{P} t_{np} t_{mp} Y_{nm} = 0 \tag{2.3}$$

Siendo:

- N es el número de exámenes.
- P es el número de timeslots.
- t_{np} es 1 si el examen ‘n’es asignado *timeslot* ‘p’.
- C_{np} es el costo de asignar el examen ‘n’al *timeslot*‘p’.
- Y_{nm} = 1 si el examen ‘n’ colisiona con el examen ‘m’ (es decir, si los exámenes ‘n’ y ‘m’ comparten alumnos) y 0 en caso contrario.

A su vez la función que determina el costo de una solución y permite comparar la calidad entre distintas soluciones, está basada en la siguiente ecuación:

$$F_c = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \; prox(i,j)}{M}$$

donde, (2.4)

$$prox(i,j) = \begin{cases} 2^{5-|t_i - t_j|}, & si \; 1 < |t_i - t_j| < 4 \\ 0, & de \; otra \; manera. \end{cases}$$

Siendo:

- N es el número de exámenes.
- C_{ij} es la cantidad de alumnos en común entre el examen ‘i’ y el examen ‘j’.
- M es el número de estudiantes.
- t_i es el número de *timeslot* al que está asignado el examen ‘i’.
- t_j es el número de *timeslot* al que está asignado el examen ‘j’.

Solución propuesta

Para la representación de la solución propuesta se utiliza un vector de tamaño N, donde N es la cantidad de exámenes para la instancia a resolver. Cada una de las posiciones de dicho vector tiene un valor entre 0 y P-1, donde P es la cantidad de *timeslots* disponibles en el problema.

La cantidad de combinaciones posibles para cada instancia de este problema es igual a N^P, de lo cual se desprende que salvo para el caso de valores de N y P muy pequeños, la búsqueda de la solución óptima se vuelve inviable debido a la cantidad de posibilidades a analizar. En consecuencia, para abordar este problema se consideraran distintas estrategias heurísticas, ya sea individualmente o combinadas, que si bien no garantizan encontrar el óptimo, permiten hallar una solución aceptable en un tiempo razonable.

A continuación se definen las heurísticas de bajo nivel consideradas para este problema y se explican las estrategias utilizadas para combinarlas.

Construcción de la solución inicial

Según algunas teorías de asignación secuencial basadas en grafos [10], existen diversos criterios para determinar el orden en el cual se asignan los exámenes a los *timeslots*. Estos ordenamientos permiten construir soluciones iniciales factibles y de menor costo. Los mismos se detallan a continuación:

- *Largest Enrollment* (LE): los exámenes se ordenan de manera decreciente según la cantidad de alumnos.
- *Largest Weighted Degree* (LWD): se asignan primero los exámenes con mayor cantidad de estudiantes que tienen conflictos con otras evaluaciones.
- *Largest Degree* (LD): los exámenes se ordenan de manera decreciente según los que más conflictos tengan con otros exámenes.
- *Saturation Degree* (SD): los exámenes se ordenan de manera ascendente según la cantidad de timeslots disponibles, es decir, los timeslots que no producen colisiones.
- *Highest cost* (HC): se elijen primero los exámenes con mayor costo de asignación.

Los primeros tres ordenamientos pueden establecerse de antemano, mientras que en los últimos dos el orden de asignación debe calcularse paso a paso, debido a que cada asignación realizada en un paso previo afectará las asignaciones que pueden realizarse posteriormente.

Estos métodos permiten obtener soluciones factibles, sin embargo [9][10] las mismas no son de buena calidad. En general deben aplicarse criterios de mayor complejidad para obtener una buena solución inicial. La combinación de los métodos mencionados, llamados heurísticas de bajo nivel, tienden a producir mejores resultados que si se las aplica de manera individual.

Se realizarón pruebas intentando generar soluciones con cada una de estas heurísticas de manera individual. Se hicieron 10 corridas

para cada una de las heurísticas, lo cual da un total de 50 pruebas por instancia.

Combinación secuencial de heurísticas de bajo nivel

A continuación se describe unade las variantes seleccionadas para construir soluciones iniciales. En ella se combinan de manera secuencial las heurísticas de bajo nivel descriptas en el punto anterior. El objetivo es lograr soluciones factibles de mejor calidad que si se utilizaran las heurísticas de manera individual.

Según la bibliografía [11], algunas heurísticas de bajo nivel, como por ejemplo SD, tienden a privilegiar la factibilidad de la solución por sobre su calidad. Por el contrario otras, como por ejemplo LD, ponen por encima la calidad de las soluciones. Teniendo en cuenta lo anterior, se realizaron pruebas combinando todas las heurísticas de bajo nivel descriptas entre sí.

En las pruebas, se definió una heurística primaria, la cual se ejecuta en las iteraciones iniciales del algoritmo, y una heurística secundaria, utilizada al final. La cantidad total de iteraciones del proceso de construcción de la solución es igual a la cantidad de exámenes de la instancia que se está analizando.

Se probaron todas las combinaciones de las 5 heurísticas de bajo nivel tomadas de a 2 en las cuales el orden es un factor importante, esto es igual a $_5P_2$posibilidades. Para cada una de las anteriores se realizaron 5 ejecuciones asignando desde 5% hasta un 95% (con intervalos de 5 %) de las iteraciones a la heurística primaria y lo restante de cada ejecución a la secundaria. En consecuencia, el número total de pruebas por instancia asciende a 1900.

Combinación jerárquica de heurísticas de bajo nivel

En estavariante se combinan de manera jerárquica las diferentes heurísticas de bajo nivel descriptas previamente. De esta manera, se construyen soluciones asignando los exámenes según una heurística primaria, si la heurística determina que hay más de un examen posible, entonces se utiliza otra secundaria para desempatar, y en caso de persistir el empate, el examen se selecciona mediante una heurística terciaria o bien con un criterio aleatorio.

Al probar todas las combinaciones posibles, esto es $_5P_3$ heurísticas, más $_5P_2$heurísticas para el caso del criterio aleatorio como terciario. En este caso se realizaron 3 corridas para cada una de las combinaciones, esto implica un total de 240 corridas por instancia.

Resultados

Los resultados presentados a continuación fueron obtenidos realizando cuatro ejecuciones simultáneas, paralelizadas con GNU Parallel[12], de manera tal de dedicarle un procesador del equipo a cada ejecución. El equipo en cuestión tiene las siguientes características:

- Sistema operativo: Linux versión 4.8.0-1-amd64 (debian-kernel@lists.debian.org) (gcc versión 5.4.1 20161019 (Debian 5.4.1-3)) #1 SMP Debian 4.8.7-1 (2016-11-13).
- Procesadores: 4 x Intel(R) Xeon(R) CPU E5-2609 0 @ 2.40GHz.
- Memoria: 8Gb.

El código fue escrito para python 3 (ejecutado con python 3.5) y se hace uso, en la medida de lo posible, de bibliotecas como Numpy[13] para optimizar los cálculos numéricos.

Las pruebas fueron realizadas con un conjunto de instancias estándar conocido como *Toronto benchmark*[14].

La Tabla 1 presenta los mejores resultadosobtenidos en la ejecución de las heurísticas puras para cada una de las instancias. En cada fila puede verse el menor *fitness* obtenido para cada instancia y mediante que heurística se consiguió. Nótese que las filas sin datos son instancias en las que no se obtuvieron soluciones factibles mediante esta estrategia. También se informa el tiempo total de ejecución en minutos para cada instancia.

Tabla 1. Comparativo de costos de la solución encontrada para las heurísticas puras.

Instancia	Fitness	Heurística	Tiempo
car-f-92	4,97	SD	< 1
car-s-91	5,64	SD	< 1
ear-f-83	47,74	SD	< 1
hec-s-92	N/A	N/A	< 1
kfu-s-92	N/A	N/A	< 1
lse-f-91	14,01	SD	< 1
sta-f-83	169,94	SD	< 1
tre-s-92	9,96	SD	< 1
ute-s-92	N/A	N/A	< 1
yor-f-83	N/A	N/A	< 1

La Tabla 2 presenta los resultados obtenidos mediante el método de combinación secuencial de heurísticas de bajo nivel, en cada fila se muestra el mejor resultado para cada instancia, junto con los parámetros mediante los cuales se obtuvo. También se presenta el tiempo de ejecución total para la instancia en minutos.

Tabla 2. Mejores resultados obtenidos parámetros y tiempo de ejecución en minutos para el método de combinación secuencial de heurísticas de bajo nivel.

Instancia	Fitness	H1	Iteraciones H1	H2	Iteraciones H2	Tiempo
car-f-92	4,70	SD	55%	LD	45%	741
car-s-91	5,35	HC	15%	SD	85%	1234
ear-f-83	40,64	LD	95%	LWD	5%	187
hec-s-92	12,92	HC	40%	LD	60%	54
kfu-s-92	15,91	SD	55%	HC	45%	356
lse-f-91	12,27	LD	5%	LWD	95%	214
sta-f-83	163,24	HC	70%	LWD	30%	83
tre-s-92	8,94	SD	45%	LWD	55%	252
ute-s-92	29,60	LWD	15%	LD	85%	88
yor-f-83	43,13	HC	5%	SD	95%	190

La Tabla 3 presenta los resultados obtenidos mediante el método de combinación jerárquica de heurísticas de bajo nivel, en cada final se muestra el mejor resultado obtenido para cada instancia, junto con los parámetros mediante los cuales se obtuvo. También se presenta el tiempo de ejecución total para la instancia en minutos.

Tabla 3. Mejores resultados obtenidos parámetros y tiempo de ejecución en minutos para el método de combinación jerárquico de heurísticas de bajo nivel.

Instancia	Fitness	H1	H2	H3	Tiempo
car-f-92	4,48	SD	HC	LWD	2
car-s-91	5,35	SD	HC	LWD	4
ear-f-83	40,79	SD	LWD	N/A	< 1
hec-s-92	13,26	SD	HC	LWD	< 1
kfu-s-92	15,46	SD	LWD	LD	1
lse-f-91	12,26	SD	LWD	LE	< 1
sta-f-83	159,00	SD	HC	LWD	< 1
tre-s-92	8,90	SD	HC	LWD	< 1
ute-s-92	30,47	SD	LE	LD	< 1
yor-f-83	41,69	SD	HC	N/A	< 1

Finalmente, la Figura 1 presenta un gráfico comparativo del desempeño de las tres estrategias empleadas.

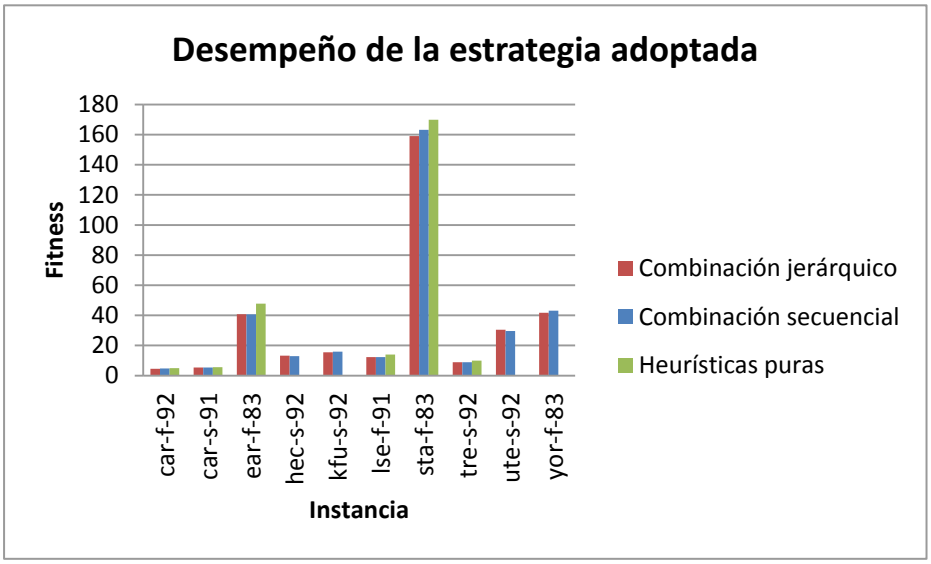

Figura 1: Gráfico comparativo del desempeño de cada estrategia.

Conclusiones

Como aspectos positivos de este trabajo es posible resaltar la aplicación de diferentes heurísticas para obtener soluciones factibles y de calidad para cada una de las instancias analizadas.

A partir de las pruebas realizadas se pudo corroborar la superioridad de los métodos jerárquicos y secuenciales por sobre los heurísticos puros. En 7 de las 10 instancias analizadas el mejor *fitness* se obtuvo mediante el método jerárquico, mientras que en tres mediante el método secuencial. Cabe destacar que en todas las instancias el método jerárquico obtuvo una solución relativamente

buena, es decir comparándola contra los otros dos métodos, en un tiempo de ejecución reducido.

Entre las mejoras posibles se puede mencionar: combinar estas estrategias con otras de búsqueda local, como por ejemplo algoritmos genéticos o búsqueda tabú, de manera de mejorar los resultados iniciales obtenidos. También variar el tiempo de ejecución de los métodos de construcción de solución inicial, por ejemplo incrementando la cantidad de iteraciones que se utilizan para la búsqueda de una solución con menor costo. Otro aspecto que podría ser tenido encuenta, es el de tener un número variable de iteraciones, dependiendo del tamaño y característicasdel *dataset* sobre el cual se está aplicando el método, esto es, cantidad de exámenes y alumnos, conel fin de determinar si dicha cantidad de iteraciones influye de algún modo en el desempeño de losmétodos analizados.

Hasta el momento se llegó a la conclusión que, con las pruebas realizadas, el método jerárquico tiene un mejor desempeño que el secuencial y el heurístico puro para las instancias analizadas en este problema.

Referencias

[1] M. W. Carter, “A Lagrangian Relaxation Approach To The Classroom Assignment Problem,” *INFOR Inf. Syst. Oper. Res.*, vol. 27, no. 2, pp. 230–246, May 1989.

[2] E. Burke, Y. Bykov, J. Newall, and S. Petrovic, “A time-predefined local search approach to exam timetabling problems,” *IIE Trans.*, vol. 36, pp. 509–528, 2004.

[3] Y. Bykov, “Time-Predefined and Trajectory-Based Search: Single and Multiobjective Approaches to Exam Timetabling,” University of Nottingham, 2003.

[4] E. K. Burke and Y. Bykov, “The late acceptance Hill-Climbing heuristic,” *Eur. J. Oper. Res.*, vol. 258, no. 1, pp. 70–78, Apr. 2017.

[5] L. Di Gaspero and A. Schaerf, “Tabu Search Techniques for Examination Timetabling,” 2001, pp. 104–117.

[6] J. M. Thompson and K. A. Dowsland, “A robust simulated annealing based examination timetabling system,” *Comput. Oper. Res.*, vol. 25, no. 7–8, pp. 637–648, Jul. 1998.

[7] S. Innet, “A noval approach of genetic algorithm for solving examination timetabling problems: A case study of Thai Universities,” in *2013 13th International Symposium on Communications and Information Technologies (ISCIT)*, 2013, pp. 233–237.

[8] M. Alzaqebah and S. Abdullah, “An adaptive artificial bee colony and late-acceptance hill-climbing algorithm for examination timetabling,” *J. Sched.*, vol. 17, no. 3, pp. 249–262, Jun. 2014.

[9] N. Pillay and W. Banzhaf, “A study of heuristic combinations for hyper-heuristic systems for the uncapacitated examination timetabling problem,” *Eur. J. Oper. Res.*, vol. 197, no. 2, pp. 482–491, 2009.

[10] D. J. Welsh and M. B. Powell, “An Upper Bound for the Chromatic

Number of a Graph and its Application to Timetabling Problem," *Comput. J. 10*, pp. 85–86, 1967.

[11] E. K. Burke, R. Qu, and A. Soghier, "Adaptive Selection of Heuristics within a GRASP for Exam Timetabling Problems," *MISTA*, 2009.

[12] O. Tange, "GNU Parallel - The Command-Line Power Tool," *login: The USENIX Magazine*, 2011. [Online]. Available: http://www.gnu.org/s/parallel.

[13] "Numpy." [Online]. Available: http://www.scipy.org/scipylib/download.

[14] M.W. Carter, G. Laporte, and S.Y. Lee. Examination timetabling: Algorithmic strategies and applications. Journal of Operational Research Society, 74:373–383, 1996.