Un algoritmo heurístico para la Asignación de Aulas en el Campus Universitario

Gastón Carrasco*, Cristian Martínez**, and Diego Rodríguez***

Departamento de Informática - Facultad de Ciencias Exactas
Universidad Nacional de Salta - Av. Bolivia 5150 CP 4400, Salta, Argentina

Resumen A medida que las organizaciones crecen, surgen problemas cuya resolución de manera manual o artesanal ocasiona pérdidas económicas o disminución en la eficiencia de sus operaciones. Uno de ellos es el de planificación de horarios (Timetabling Problem). El presente trabajo está enfocado a la planificación de horarios en el ámbito univesitario. Para ello, proponemos un nuevo algoritmo heurístico que mediante lista tabú y exploración de vecindarios es capaz de ofrecer soluciones satisfactorias sobre un conjunto de instancias conocidas de la literatura.

Keywords: Algoritmos · Heurísticas · Timetabling Problem

1. Introducción

El Timetabling Problem (TTP) es un problema al cual se enfrenta toda Universidad para programar los horarios de clases. Incluye la asignación de eventos o clases (cursos con profesores y estudiantes) en un número limitado de salones/aulas bajo ciertas condiciones (como por ejemplo, capacidad de aulas y disponibilidad horaria de docentes). Al igual que el TTP, el University Timetabling Problem (UTP) es un problema de complejidad NP-Completo [1,3].

Se propone un algoritmo heurístico para el UTP tal que a partir del uso de memoria y exploración de diferentes vecindarios, genera soluciones diversas y de calidad. Para medir la calidad de la propuesta, se usaron los datasets del International Timetabling Competition (ITC)[1] de 2007.

El resto del trabajo se organiza de la siguiente manera: la Sección 2 introduce al problema, las características del algoritmo propuesto se describen en la Sección 3, en la Sección 4 se muestran los resultados de las pruebas computacionales, y en la Sección 5 se presentan las conclusiones y líneas futuras del trabajo.

2. University Timetabling Problem (UTP)

Según Wren [9], el Timetabling Problem está referido a la asignación de recursos sujetos a restricciones, de reuniones o eventos que se colocan en espacios

* carrasco.gaston.94@gmail.com
** cmartinez@unsa.edu.ar
*** drodriguez@di.unsa.edu.ar
[1] Link: http://www.cs.qub.ac.uk/itc2007/index.htm

de tiempo (timeslots) definidos, buscando satisfacer un conjunto de objetivos deseables. Dentro del mismo, existe una rama específicamente dedicada a entidades educativas, incluyendo a las instituciones de nivel medio y superior.

En el ambiente universitario, los estudiantes eligen las asignaturas a cursar, lo que dificulta la obtención de soluciones debido a conflictos de asignaturas de cursado común y cantidad dinámica (y desconocida) de estudiantes inscriptos en cada curso. Babaei et. al. [1] exponen el problema el cual ha sido abordado en la literatura. Es posible clasificar dos problemas de timetabling universitario:

- Programación de horarios para exámenes: presentan restricciones especiales para la toma de exámenes universitarios. Bolaji et. al. [11] presentan un algoritmo ABC para afrontar este problema.
- Programación de horarios para cursos: puede ser posterior a la inscripción (Post-Enrolment) o mediante información histórica y aplicación de currículas (Curriculum Based). Al-Betar y Khader [10] presentaron un algoritmo armónico para esta variante del problema.

En este trabajo nos enfocaremos en el timetabling de cursos universitarios. Según lo propuesto por la ITC, la calidad de la solución al problema depende de una funcíon de penalidades relacionadas relacionadas con a) Cantidad de días de la asignación de cada curso, b) Asignaciones en horarios de forma aislada, y c) Capacidad y estabilidad de las aulas asignadas para cada curso. Mientras menos se incumplan las restricciones, la penalidad aplicada a una solución es menor.

3. Algoritmo Propuesto

En el Algoritmo 1 se muestra una versión en pseudo-código del algoritmo propuesto, denominado como Heur-UTP.

Algorithm 1 Heur-UTP

```
for i ← 1, #Ejecuciones do
    InicializarSolucion(Dataset)
    (CursosAsignados, CursosNoAsignados) ← AsignacionDirecta()
    while (CursoNoAsignado ≠ ∅) do
        OperadorAula()
        OperadorCurricula()
    end while
    AjustarDias()
    CompactarCurriculas()
    ReasignarAulas()
    DevolverSolucion()
end for
```

El algoritmo se ejecuta una cantidad determinada de corridas independientes, obteniéndose una solución al problema por vez. En cada corrida, luego de

inicializar las estructuras de datos necesarias, se obtiene una solución inicial a partir de la asignación directa de una cantidad de timeslots a cursos aleatoriamente elegidos. Para cada curso, se calcula su dominio (timeslots disponibles) y se intenta asignar cada una de sus clases a aulas, satisfaciendo las restricciones del problema. Al finalizar, se obtienen 2 listas que indican cuáles cursos fueron asignados y cuáles no.

Luego, se activan dos operadores que corrigen la solución construida de manera de asignar todos los cursos a timeslots. El procedimiento *OperadorAula* elige un curso no asignado de manera aleatoria y calcula su dominio en función de las aulas. Si existen timeslots disponibles, se elige un curso completo que pueda moverse a otro horario y de esta forma determinar los intercambios de timeslots y aulas posibles. El procedimiento *OperadorCurricula* consiste en identificar el dominio de un curso incompleto sin tener en cuenta conflictos curriculares. Luego, con la selección de uno de los timeslots se intentará realizar movimientos de todos los cursos que presentan conflictos y asignar el curso incompleto en dicho timeslot liberado. Durante este proceso de reparación, se usa una lista tabú común [2] la cual mantiene durante un cierto número de iteraciones, aquellos cursos que no se han podido completar (asignar a un timeslot).

Una vez obtenida una solución inicial completa, se intenta mejorar su calidad (es decir, reducir la penalidad asignada a la solución) progresivamente mediante la aplicación de 3 operadores. En *AjustarDias* se itera reajustando las asignaciones de los cursos para aumentar o reducir la cantidad de días en los que están distribuidos. Por su parte, *CompactarCurriculas* agrupa y varía repetidamente los timeslots ocupados por cada curso sin modificar el ajuste de los días. Finalmente, en *ReasignarAulas* se cicla buscando la mejor combinación de aulas y cursos respecto a la capacidad del aula, cantidad de alumnos del curso y la cantidad de diferentes aulas utilizadas, manteniendo los timeslots ocupados.

4. Pruebas Computacionales

La Tabla 1 presenta los mejores resultados de Müller [4], Rocha et al.[7], Wahid et al.[8], Patrick et al.[5] y nuestro algoritmo, sobre los conjuntos de datasets Early, Late y Hidden del ITC 2007. Nuestros resultados corresponden a la mejor solución (menor función de penalidad obtenida) asociada sobre 100 corridas independientes por cada instancia. Las pruebas se realizaron en una PC con procesador Core I7 3.4Ghz 8 GB de RAM sobre Ubuntu 14.06 LTS.

De las pruebas realizadas, nuestros resultados son mejores a los de Wahid et al.[8] en las 21 instancias, en 16 (sobre 21) respecto a los de Patrick et al.[5], en 13 (sobre 21) sobre los de Rocha et al.[7], y en 5 (sobre 21) respecto a los de [4].

5. Conclusiones y Trabajo Futuro

Propusimos un nuevo algoritmo para el problema de horarios en el ámbito universitario. De acuerdo a pruebas computacionales sobre instancias del ITC, se obtuvieron soluciones factibles, competitivas y robustas.

Cuadro 1: Resultados sobre Datasets Early, Late y Hidden

Inst.	Müller [4]	Rocha et al.[7]	Wahid et al.[8]	Patrick et al.[5]	Heur-UTP
1	5	6	322	10	65
2	51	131	732	176	73
3	84	141	665	222	78
4	37	78	577	100	101
5	330	552	1297	606	154
6	48	123	879	178	124
7	20	120	930	123	183
8	41	87	645	112	109
9	109	164	685	172	58
10	16	78	816	125	105
11	0	0	179	1	80
12	333	517	1398	622	55
13	66	120	694	136	103
14	59	98	702	141	90
15	84	138	665	189	90
16	34	104	827	155	143
17	83	156	830	148	143
18	83	115	510	132	54
19	62	139	608	156	102
20	27	149	950	147	147
21	103	188	835	246	156

A futuro, adaptaremos técnicas basadas en GRASP [6] y otros mecanismos de exploración de Tabu Search (Path Relinking y Ejection Chains) para obtener mejores soluciones (con menor valor de penalidad) con menores tiempos de CPU.

Referencias

1. Babaei, H. , Karimpour, J., Hadidi, A.: A survey of approaches for university course timetabling problem. Computers & Industrial Engineering **86**, 43–59 (2015)
2. Glover, F., Laguna, M.: Tabu Search. 1ra edn. Springer, Boston (1997)
3. Hahn-Goldberg, S.: Defining, Modeling, and Solving a Real University Course Timetabling Problem. Master thesis. University of Toronto (2007)
4. Müller, T.: ITC2007 solver description: A hybrid approach. Annals of Operations Research, **172**(1), 429–438 (2007)
5. Patrick, K., Godswill, Z.: Greedy ACO Strategy for Solving the Curriculum Based University Course Timetabling Problem. British Journal of Mathematics & Computer Science **14**(2), 1–10 (2016)
6. Resende, M., Ribeiro, C.: Optimization by GRASP. 1ra edn. Springer, NY (2016)
7. Rocha, W., Boeres, M., Rangel, M., Ferreira, L.: Aplicacao das Meta-heurísticas GRASP, SA e AG para o Problema de Tabela-horario para Universidad. En: XLIV SOBRAPO, 3226–3236. Sobrapo, RJ-Brasil (2012)
8. Wahid, J., Hussin, N.: HSA for Curriculum-Based Course Timetabling Problem. Int. J. of Soft Computing and Software Engineering **3**(3), 365–371 (2013)
9. Wren, A.: Scheduling, timetabling and rostering, a special relationship?. En: Int. Conf. on the Practice and Theory of Timetabling, 46–75. Springer, London (1995)
10. Al-Betar, M., Khader, A.: A harmony search algorithm for university course timetabling. Annals of Operations Research.**194**(1), 3–31 (2012)
11. Bolaji, A., Khader, A., Al-Betar, M., Awadallah, M.: An improved artificial bee colony for course timetabling. Sixth International Conference on BIC-TA, 9–14 (2011)