Implementación del modelo de restricciones del problema de calendarización de horarios universitarios con PYTHON

Lluvia Carolina Morales Reynaga*, José Figueroa Martínez**

Resumen

En este documento se describe el modelo de restricciones del problema de calendarización de horarios de clases de la Universidad Tecnológica de la Mixteca (UTM) que se ha desarrollado utilizando el lenguaje de programación Python, de tal manera que se pueda generar una función objetivo que permita medir la calidad de una solución del problema.

Al proceso de seleccionar y asignar recursos y tiempo al conjunto de actividades de una planificación, y teniendo en cuenta que la asignación debe cumplir con un conjunto de restricciones que reflejan la relación temporal entre las actividades y la capacidad limitada de los recursos compartidos se le llama Problema de Programación o Problema de Calendarización [1].

Las restricciones son los requerimientos del problema que pueden ser cuantificados para generar una medida de calidad de la solución/horario propuesto, por ejemplo, si tenemos las restricciones siguientes:

Un profesor no puede estar asignado a dos salones simultáneamente y

Un lugar (salón de clases) no puede ser utilizado por dos grupos diferentes al mismo tiempo (a la misma hora, el mismo día).

Entonces si no se cumple con alguna de las restricciones, se deberá considerar una penalización cuantificable por cada restricción no satisfecha.

La función objetivo se utiliza en problemas de optimización. En este caso se utilizará en la implementación de meta-heurísticas, específicamente se implementarán cuatro, las cuales son: Algoritmo Genético, Algoritmo de Búsqueda Tabú, Algoritmo de Búsqueda Armónica y Algoritmo de Recocido Simulado. Todo esto para comparar el desempeño de cada una de éstas y encontrar una solución factible y buena en términos de tiempo y función objetivo al problema de calendarización de la UTM.

1.Restricciones

Las restricciones se dividen en dos categorías:

* lluviamorales@mixteco.utm.mx. Universidad Tecnológica de la Mixteca.

** jfigueroa@mixteco.utm.mx. Universidad Tecnológica de la Mixteca.

Colaboración de: Juan Pablo Gómez Martínez, juanpablo.gomezmartinez89@gmail.com. Alumno de Maestría en Tecnologías de Cómputo Aplicado.

Restricción fuerte: Son las que deben cumplirse de manera obligatoria. En la figura 1 se muestra un ejemplo de esta restricción; y

Restricción débil: Si no se puede cumplir, no afecta a la factibilidad de la solución. Esto significa que esta restricción puede no satisfacerse [2].

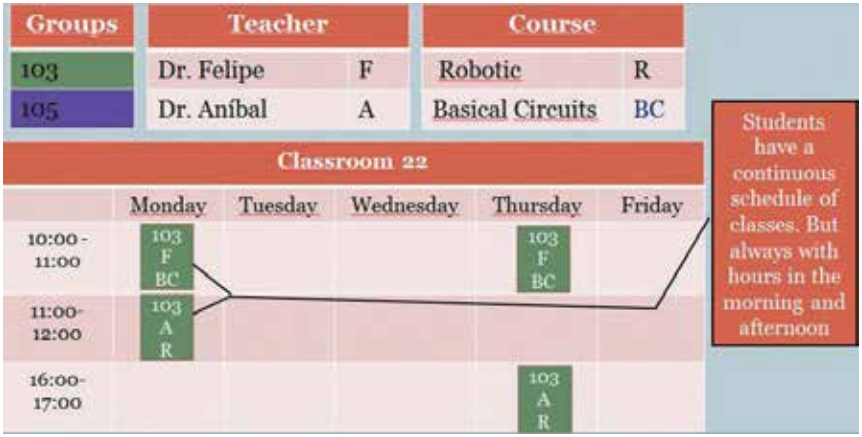

Figura 1. Ejemplo de una restricción fuerte.

A continuación se enlistan las restricciones con las que se trabaja actualmente en la UTM.

Restricciones fuertes:

1. Las clases de teoría y práctica no deben ser a la misma hora, el mismo día.
2. Las clases no deben impartirse durante algún día y hora restringida para ellas.
3. Un profesor no puede estar en dos lugares distintos a la misma hora, el mismo día.
4. Dos o más profesores o grupos no pueden estar en el mismo lugar a la misma hora, el mismo día; a menos que sea una multiasignación de grupos y/o profesores.
5. Si un profesor da clases a las 8:00 a.m., no puede dar clases a las 1:00 p.m.
6. Un espacio debe permitir sólo 9 horas de clase al día ya sea del mismo o diferentes grupos.
7. Un grupo debe tener dos horas de clases libres al día para que en alguna de ellas se les pueda asignar clases de inglés.
8. Un grupo no debe tomar clases en la hora en que un alumno toma clases de recursamiento junto a otro grupo.

Restricciones débiles:

1. Que los profesores tengan un horario continuo de clases. A menos que el profesor especifique lo contrario. Y que tengan su horario en las horas que ellos prefieran.
2. Que los alumnos tengan un horario continuo de clases. Pero siempre con horas en la tarde y en la mañana.
3. Que no existan clases de práctica fuera del horario habitual de clases, a menos que se especifique directamente en los datos de entrada.
4. Que los profesores impartan las mismas materias que en semestres anteriores afines.

5. Que los profesores tengan horarios muy similares, si no es que el mismo que en semestres anteriores afines.
6. Que los grupos no tomen clases continuas en lugares demasiado alejados.
7. Que los grupos de diferentes carreras tomen clases en grupos de aulas "interdisciplinarios".

2. Python como lenguaje de programación para las restricciones

Se utiliza Python para la evaluación de las restricciones y para la generación de soluciones. ¿Por qué? A continuación enumeramos algunas de las justificaciones:

Python es un lenguaje de programación orientado a objetos, interpretado e interactivo.

Python combina un poder notable con una sintaxis muy clara.

Tiene módulos, clases, excepciones, tipos de datos dinámicos de muy alto nivel y escritura dinámica.

Hay interfaces para muchas llamadas de sistema y bibliotecas, así como a varios sistemas de ventanas.

Los nuevos módulos integrados se escriben fácilmente en C o C ++ (u otros lenguajes, dependiendo de la implementación elegida).

Python también es útil como un lenguaje de extensión para aplicaciones escritas en otros lenguajes que necesitan interfaces de scripting o automatización fáciles de utilizar [3].

Éstas son las principales ventajas que tiene Python, destacando su legibilidad, compatibilidad con otros lenguajes y su extensa comunidad en la red que crece constantemente, de donde podemos obtener información con respecto a las funciones o nuevos módulos durante el proceso de programación. También es necesario mencionar que los tipos de datos que se utilizan en la implementación son principalmente diccionarios, listas y conjuntos, ya que con estas funciones se facilita la evaluación de las restricciones.

Para el intercambio de datos del lenguaje de programación y la base de datos se utiliza JavaScript Object Notation (JSON). La ventaja de este formato es que es simple de generar e interpretar, además de que es legible por parte del programador, de esta manera se puede acelerar el tiempo de programación y permitir una fácil corrección y/o modificación a futuro. Python cuenta con módulos de terceros para leer e interpretar JSON. En las figuras 2 y 3 se pueden apreciar la manera en que los datos están estructurados para que pueda ser utilizado por Python.

```
1.{“metabounds”:
2. {“samehourtp”:false,
3.  “aid”:1887,
4.  “mid”:null,
5.  “gpid”:103,
6.  “gtid”:101,
7.  “assignments”:
8.  [{“assignment”:1887,
9.   “teacher”:211,
10.   “group”:399,
11.   “recursamientos”:[],
12.   “specialty”:null}],
13.  “size”:5,
14.  “recursamientos”:[],
16.  “lgs”:
17.  [{“lectures”:[{“t”:”t”, “n”:5, “d”:1}],
18.   “hours”:[9, 10, 11, 12, 13],
19.   “days”:[0, 1, 2, 3, 4],
20.   “daysinorder”:false,
21.   “hidx”:2,
22.   “pidx”:-1},
23.  {“lectures”:[{“t”:”p”, “n”:1, “d”:2}],
24.  “hours”:[8,9,10,11,12,13,16,17,18],
25.   “days”:[0, 1, 2, 3, 4],
26.   “daysinorder”:false,
27.   “hidx”:3,
28.   “pidx”:4}]}
```

Figura 2. Estructura en JSON de las restricciones y preferencias de los profesores que son utilizados para formular las restricciones.

```
1.{“bounds”:
#Espacio de teoría (se elige Aleatorio)
2. [2,25,15,30,33,19,6,40,12,9,44,11,18,
   26,41,14,4,72,32,43,10,13,8,5,34,37,
   16,7,2  9,3,2139,42,17],
#Espacio de práctica (se elige Aleatorio)
3. [64],
#Hora de Teoría (se elige Aleatorio)
4. [8, 9, 10, 11, 12, 13, 16, 17, 18],
#Hora de Práctica (se elige Aleatorio)
5. [8, 9, 10, 11, 12, 13, 16, 17, 18],
#Día de Práctica (se hace una Permutación y se
elige el primero de la lista, véase en la figura 2 que “n”
es el número de días)
6. [0, 1, 4, 3, 2]}
```

Figura 3. Estructura en JSON de los espacios, horas y días que pueden ser seleccionados para generar una solución.

En la figura 4 se muestra una parte de código y cómo es utilizado la estructura JSON para obtener los datos.

```
for j in range(lenlgs):
    lect = lgs[j]["lectures"]

    if(lgs[j]["hidx"] != -1):
        hora = solucion[ctrl+lgs[j]["hidx"]]
    if(lgs[j]["pidx"] != -1):
        diaVect = solucion[ctrl+lgs[j]["pidx"]]
    else:
        diaVect = lgs[j]["days"]
```

Figura 4. Parte de código implementado en python donde se observa el uso de datos obtenidos con JSON.

Para la implementación de las restricciones en el código json de la figura 1 se describen los elementos implicados que hacen posible una asignación, los cuales son utilizados de manera implícita en cada restricción.

Espacios: Representan salones de clases o aulas especiales (cubículos, salas de cómputo) en la UTM que cuentan con los elementos necesarios para que sea posible impartir una clase dentro de ella. En la universidad hay 95 espacios.

Días: Son los días hábiles de la semana en las que es posible impartir clases. Se toma en cuenta 5 días de la semana (Lunes-Viernes).

Horas: Son el tiempo del día en las que es posible asignar una clase. Se utilizan 9 horas de trabajo, las cuales son de 8:00 horas hasta las 14:00 horas y de 16:00 horas hasta las 19:00 horas.

Profesores: Son los catedráticos de tiempo completo que laboran en la Universidad y que se les puede asignar alguna materia a impartir (de su línea de investigación). Se toman en cuenta 200 profesores.

Materias: Se refieren al plan de estudios que se dictan en la Universidad Tecnológica de la Mixteca (Ej. Cálculo Diferencial, Programación orientada a objetos, Comunicación Visual, etc.). En total hay 246 materias en la Universidad.

Grupos: Lo conforman un conjunto de alumnos que cursan una misma carrera. Actualmente se cuenta con 50 grupos.

También se especifican los elementos que definirán de qué manera es la asignación.

Duración: Cuánto tiempo tendrá vigencia una clase. Puede ser de 1 hora hasta 5 horas.

Hora de Inicio: Se refiere al tiempo en que se inicia la clase. El intervalo es de 8:00 a 13:00 y 16:00 a 18:00.

Hora de Fin: Se refiere al tiempo en que culminará una clase. Está entre los intervalos 9:00 a 14:00 y 17:00 a 19:00.

Clase de Teoría: Se refiere a si existe una clase teórica (puede ser de práctica si el profesor lo requiere).

Clase de Práctica: Se refiere a si existe una clase de práctica.

Día restringido: Es un día de la semana laboral en la que no es posible tener una asignación (por cargos u otros impedimentos por parte del profesor).

Hora restringida: Es una hora del día en la que no puede haber una asignación (por cargos u otros impedimentos por parte del profesor o los alumnos).

Hora Preferida: Es una hora en la que un profesor puede dar clases de acuerdo a su conveniencia.

A partir de las definiciones mencionadas anteriormente, se generan los datos que se utilizan en la lógica del código fuente en Python para evaluar cada una de las restricciones. Esto se lleva a cabo utilizando el intercambio de datos con JSON para facilitar la programación. Además, con estos datos se define una función de factibilidad y una función heurística; la función de factibilidad representa el conjunto de restricciones fuertes y la función heurística el conjunto de restricciones débiles que unidas permiten definir la función objetivo.

La función de factibilidad se puede representar como una suma de los costos de cada restricción fuerte i multiplicado por un peso i que se le da a cada restricción dependiendo de la importancia que tenga éste con respecto a la solución final; esto es:

$$f = \sum_{i=1}^{9} (peso_i * costo_i)$$

En donde el costo es calculado a partir del cumplimiento o incumplimiento de cada elemento de una solución, por ejemplo, contabilizando cada insatisfacción. Esta función definirá la factibilidad de la solución que se está evaluando de tal manera que pueda ser comparada con otras soluciones y de esa manera encontrar una solución factible al problema de horarios.

Por otra parte la función heurística se encarga de verificar la bondad de una solución, por consiguiente es necesario tener una solución factible para verificar si la solución es buena, ya que sólo de esta manera se puede asegurar que la solución sea factible y buena a la vez; esto no siempre es posible pero siempre se pretende obtener una solución lo más cercano a lo óptimo.

Conclusiones

La implementación en el lenguaje de programación Python facilita la comprensión del código para futuras modificaciones, además de que es rápido y eficiente ya que dispone de diferentes módulos y funciones que ayudan en la implementación, y no es necesario preocuparse por el tipo de datos en casos donde no se utilizan.

Por el hecho de que el código en Python es legible y fácil de implementar, es preferible escribir en código las restricciones para después hacer una formalización, de esta manera se puede tener una idea de la estructura de ésta de manera más clara y concisa en caso de que se tengan problemas.

Por último, es importante mencionar que este modelo permite agregar restricciones evaluables por medio del código en Python ya que cada una de ellas se ha implementado en un módulo independiente que permite agregarlas a la función de factibilidad o heurística sin necesidad de cambiar todo el código o, dependiendo de las circunstancias se puede modificar parte del código, pero se pretende que sean mínimas las modificaciones a realizar.

Python es un buen lenguaje para los que empiezan a programar y se puede convertir en un lenguaje potente y robusto para los conocedores de mismo, implementando técnicas de programación más avanzadas como la programación funcional que ya incluyen las nuevas versiones de Python.

Referencias

[1] D. de Werra. Some combinatorial models for course scheduling. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[2] Hamed Babaei, Jaber Karimpour, and Amin Hadidi. A survey of approaches for university course timetabling problem. Computers and Industrial Engineering, 86 (April):43-59, 2014.

[3] https://www.python.org citado en Abril de 2017.