ASIGNACIÓN DE HORARIOS CON ALGORITMOS EVOLUTIVOS - LIBRERÍA DEAP PYTHON

M.D. Quilindo
Estudiante Pregrado
Ingeniería de Sistemas
Universidad del Cauca,
mdquilindo@ucauca.edu.co

J.S. Parra,
Estudiante Pregrado
Ingeniería de Sistemas
Universidad del Cauca,
juanparra@ucauca.edu.co

Resumen

Este artículo presenta un algoritmo evolutivo para la resolución del problema de asignación de horarios, para las asignaturas del programa de Ingeniería de Sistemas de la Universidad del Cauca dictadas en el Edificio de Ingenierías (aulario), cumpliendo con ciertos requerimientos y condiciones.

Palabras clave

Horarios, gestión, computación evolutiva, estrategia evolutiva.

1. Introducción

El Edificio de Ingenierías de la Universidad del Cauca, comparte sus aulas con tres programas afines para los distintos cursos correspondientes[4].

Actualmente ya se tiene una distribución de los horarios de clases, que no han tenido modificaciones durante más de cuatro semestres.

En este artículo se propone una aplicación de un algoritmo evolutivo para la resolución de la asignación de horarios de las asignaturas del programa de Ingeniería de Sistemas dictadas en las aulas del Edificio de Ingenierías. Se toma como base la información de los cursos programados para el semestre 2019-2 y un total de 16 aulas distribuidas entre salones y laboratorios.

Este tipo de problemas son clasificados como NP-Hard; por cada restricción que adicionamos al contexto la complejidad para hallar una solución aceptable se verá afectada críticamente, así, se clasifican como restricciones fuertes y débiles[2].

2. El algoritmo

En esta sección se presenta el algoritmo evolutivo adaptado a la problemática de la asignación de los horarios; y la descripción de los operadores a utilizarse para llegar a obtener un resultado aceptable y óptimo. Se decidió implementar el algoritmo en el framework DEAP de Python.

Entendemos como algoritmo evolutivo aquel que simula la evolución natural, donde prevalece la ley del más fuerte, es decir, aquellos individuos con mejores calificaciones son los que prevalecen generación tras generación heredando sus mejores atributos a sus hijos y mutando en pequeña medida[1].

2.1. Restricciones utilizadas

Las restricciones se clasifican en fuertes (RF) y débiles (RD):

1. Dos clases o más no pueden dictarse simultáneamente en la misma aula (RF).
2. Un profesor no puede impartir dos o más clases simultáneamente (RF).
3. Una asignatura no puede tener dos o más clases en un solo dia (RF).
4. Al menos una clase de una asignatura no se repita con demás clases de otras asignaturas del mismo semestre (RF).
5. Las clases de asignaturas teorías sólo pueden dictarse en salones, en cambio, las clases de asignaturas prácticas solo se dictan en laboratorios (RF)
6. Las clases de asignaturas prácticas de séptimo semestre en adelante, solo deben dictarse en el laboratorio de la Sala 4 (por capacidad de cómputo) (RD).
7. Las clases de asignaturas de semestres adyacentes no pueden dictarse simultáneamente (RD).

2.2. Representación

Los cursos son dictados entre lunes a viernes (5 días) dentro de jornadas de dos horas, entre las 07:00 hasta las 18:00 (5 jornadas). El total de aulas son 16, 12 salones y 4 laboratorios.

Nota: cada asignatura tiene el atributo de créditos que representar la cantidad de jornadas.

Se representa a un individuo con un vector de 513 posiciones, calculado de la siguiente manera:

$$\sum créditos * tamaño\ del\ curso = 171 * 3 = 513\ unidades$$

Donde,

tamaño del curso es la representación de un sub-vector que contiene en 3 posiciones, la información del día, jornada y aula.

Para un mejor entendimiento se representa el siguiente diagrama:

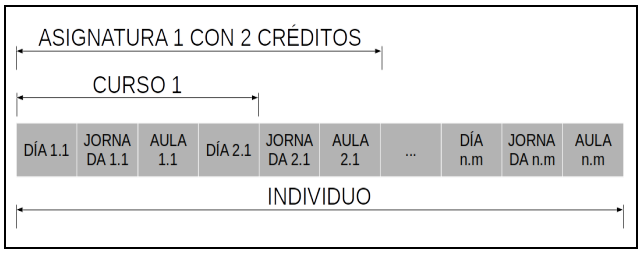


Imagen 1.0 Representación del individuo

2.3. Función de generación del individuo

Con el fin de evitar individuos mutantes los cuales contuviera índices no validados para los vectores de días, jornadas y aulas. Se definió una función que tuviera en cuenta el dominio de los índices.

2.4. Operador de cruce

Este operador está basado en la función cxOnePoint[5], pero con la diferencia de definir un total 4 puntos de corte al azar en el inicio de un curso. También se tuvo en cuenta asignar índices válidos a cada sub-vector, al igual que la función del *punto 2.3*.

2.5. Operador de mutación

La mutación se basa en un operador básico, que brinda aleatoriedad a los individuos de una población. Si bien el operador de cruce se encarga de hacer una búsqueda en el espacio de posibles soluciones, es el operador de mutación
el encargado de aumentar o reducir el espacio de búsqueda en un algoritmo y de proporcionar variabilidad al individuo[3].

Este operador está basado en la función mutFlipBit[5], pero con la diferencia de recorrer el individuo de tres en tres (tamaño del curso), con el fin de modificar el día y jornada con una probabilidad menor, y modificar el salón con una probabilidad más alta. También se tuvo en cuenta asignar índices válidos a cada sub-vector, al igual que la función del *punto 2.3*.

3. Resultados y análisis

3.1. Plataforma de desarrollo y ejecución

El algoritmo se desarrolló en el framework DEAP de Python.

El proceso de calibración y evaluación se llevaron a cabo en una laptop con procesador Intel Core i5 7th Gen 2.5 GHz, 12 Gb de RAM y S.O Ubuntu 18.04 LTS.

3.2. Ajuste paramétrico

Los algoritmos tienen la cualidad de ser un proceso estocástico, por lo tanto, se tuvieron en cuenta las siguientes recomendaciones de un experto:

- Tener un margen de error del 5%, es decir, no cumplir para nuestro caso con un total de 3 configuraciones de cursos que violen las restricciones débiles, pero simultáneamente cumplir todas condiciones fuertes.
- La solución aceptable dada por el algoritmo no debe prolongarse largos periodos de tiempo.
- La población debe ser igual a 2 veces el tamaño del individuo.

De esta manera se determinó que los siguientes parámetros tuvieran los siguiente valores:

- **indpb:** atributo de la mutación. Probabilidad independiente de que cada atributo se invierta = 0.01

- **tournsize:** atributo de selección. El número de individuos que participan en cada torneo = 3
- **population | n:** población inicial = tamaño del individuo * 2 = 513 * 2 = 1026
- **cxpb:** atributo del algoritmo evolutivo. Probabilidad de aparear dos individuos = 0.7
- **mutpb:** atributo del algoritmo evolutivo. Probabilidad de mutar a un individuo = 0.2
- **ngen:** atributo del algoritmo evolutivo. Número de generaciones = 150

4. *Evaluación experimental*

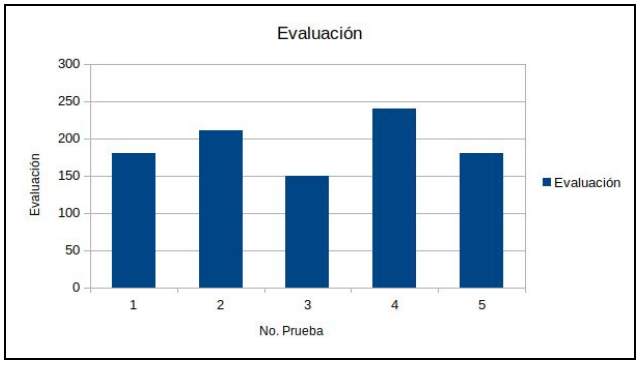
A partir de los anteriores ajustes, obtuvimos los siguientes resultados:

No. Prueba	Tiempo (m)	Evaluación
1	2.99	180
2	3.05	210
3	3.03	150
4	3.04	240
5	3.01	180
Desviación	0.02	34.20
Mediana	3.03	180

Tabla 1.0 Resultados de evaluación con 1026 individuos y 150 generaciones

De la tabla 1.0, se detalla el tiempo requerido para obtener una solución aceptable es de un máximo de 3m con 1s, además, la peor evaluación se obtiene con un valor de 240, es decir, 8 configuraciones de asignaturas las cuales no cumplen las restricciones débiles.

Nota: el número de configuraciones de asignaturas que no cumplen con las restricciones básicas se obtiene dividiendo el resultado de la evaluación entre 30 (penalización para este tipo de restricciones).



Gráfica 1.0 Evaluaciones obtenidas

A partir de la *gráfica 1.0* se puede observar que las evaluaciones no tuvieron valores tan dispersos, y esto se corrobora con la desviación estándar con un valor de 34.20 y una mediana de 180, lo que se traduce a, obtener en la mayoría de casos 6±1 configuraciones de asignaturas que no cumplen las restricciones débiles.

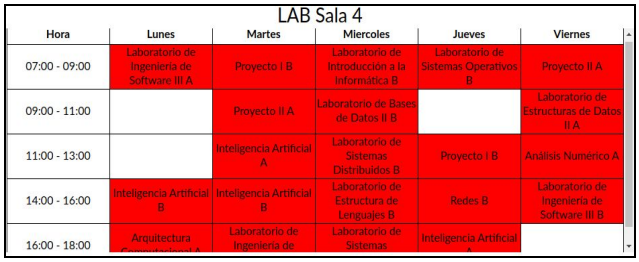


Imagen 2.0 Horario de la Sala 4 LAB

La restricción débil más particular de nuestro contexto:

- Las clases de asignaturas prácticas de séptimo semestre en adelante, solo deben dictarse en el laboratorio de la Sala 4 (por capacidad de cómputo) (RD). **Restricción número 6.**

Esto afectó significativamente la calidad de la solución, como se observa en la imagen 2.0 que la restricción no se cumple en 4 de las 5 configuraciones de asignaturas arrojadas en la prueba número 3 de la *tabla 1.0*.

A pesar del mínimo de configuraciones de asignaturas que no cumplen con el número propuesto por el experto asesorador, de que deben haber 3 penalizaciones en uso de las restricciones débiles, para una evaluación máxima aceptable de 90. Se debe principalmente a la alta demanda de asignaturas (20 cursos) para un solo salón ofrece capacidad para un total de 25 cursos, se podría deducir a simple vista la cantidad de 5 slot más los cuales pueden ser ocupados,

pero se debe tener en cuenta las restricciones fuertes pueden interferir en la asignación correcta.

5. Conclusiones

- Se obtuvo un algoritmo acorde a las necesidades del problema.
- Las restricciones débiles muy ligadas a las limitantes propias del contexto, bajan condicionan la solución dada.
- El aumento de restricciones tanto fuertes como débiles, aumentan la complejidad del algoritmo para hallar una solución óptima.

6. Bibliografía

[1]R.M. Brady (1985). Optimization strategies gleaned from biological evolution, Nature, 317, 804-806.
[2]J. Holland (1975). Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor.
[3]M.F. Bramlette (1991). Initialization, mutation and selection methods in genetic algorithms for function optimization. Proceedings of the Fourth International Conference on Genetic Algorithms, 100-107.
[4]http://www.unicauca.edu.co/versionP/oferta-academica/programas-de-pregrado/ingenieria-de-sistemas
[5]https://deap.readthedocs.io/en/master/api/tools.html