“PROYECTO: PROGRAMACIÓN DE HORARIOS PARA CURSOS UNIVERSITARIOS”

Autor/Autores: Fiorela Lizarraga, Frank Ygnacio, Ricardo Llanos, Yulian Cama

Resumen-El presente trabajo tiene como objetivo resolver la problemática de la programación de horarios universitarios de forma eficiente dadas ciertas restricciones y basándonos en la necesidad de reducir el tiempo libre entre clases. Nuestro planteamiento se enfocó en el uso de algoritmos genéticos multi-objetivo y mono-objetivo para la solución de este problema. En este planteamiento definimos a nuestro individuo como un horario de tipo arreglo donde cada gen representa la combinación grupo-materia-día-hora-profesor-aula. Se utilizaron 3 modelos de archivos origen distintos con cantidades de materias y relaciones profesor-materia variables para verificar las soluciones. En los 3 casos logramos satisfacer el objetivo principal cumpliendo las restricciones obligatorias e inclusive cumpliendo con algunas restricciones adicionales. Los algoritmos genéticos multi y mono objetivo logran resolver sin inconvenientes el problema, la mutación de genes ayuda en sobremanera a determinar los mejores resultados con las restricciones que se necesiten.

1. Introducción

El problema de la asignación de horarios es un problema ampliamente conocido, que busca asignar materias en horarios disponibles. La complejidad surge cuando se tienen diversas restricciones que cumplir, múltiples cursos a asignar, cursos que solo pueden dictarse en algunas aulas y por ciertos profesores, profesores con cierta disponibilidad horaria, etc.

A continuación se presentan algunas propuestas de otros autores:

Nevena Dresevic, estudiante máster en Computer Science, de la Escuela Politécnica Federal de Lausanne, nos presentó el problema y las restricciones principales del mismo, ella representa al individuo base como una matriz de día-hora por clases en la cual cada elemento de la matriz es representado por el código de orden de la clase. Utiliza el algoritmo genético **(1+1) evolutionary strategy** y **simulated hardening** para resolver el problema.

Branimir Sigl, Marin Golub y Vedran Mornar, en su artículo científico **Solving Timetable Scheduling Problem by Using Genetic Algorithms** proponen un algoritmo mono-objetivo en el que cada individuo de la población representa un horario y cada gen de un individuo representa un aula donde el valor del gen será el ordinal de una variable binaria perteneciente a esta aula. El algoritmo utiliza selección elitista y por ruleta y uniform crossover. Para mejorar el desempeño se aplica selección por torneo.

Achini Kumari Herath (2017) en su paper **Genetic Algorithm For University Course Timetabling Problem** propone un algoritmo genético con selección por torneo, uniform crossover y mutación uniforme, representa el cromosoma como una matriz donde cada fila corresponde a un profesor y cada columna a una hora.

Samuel Lukas, Arnold Aribowo y Milyandreana Muchri (2012) en su paper **Solving Timetable Problem by Genetic Algorithm and Heuristic Search Case Study: Universitas Pelita Harapan Timetable** plantean un algoritmo genético con cycle crossover, swap mutation y truncation selection, se utiliza codificación por permutación donde cada gen representa un curso que se imparte en un aula y el orden de genes en el cromosoma representa el orden de programación de los cursos.

Hipótesis del problema

Nuestra solución plantea aplicar algoritmos genéticos mono y multi-objetivo para brindar horarios que cumplan las restricciones obligatorias dadas y en lo posible se cumplan los criterios adicionales. Se espera que el algoritmo multi-objetivo tenga un mejor desempeño.

Elementos:

Clase: Cada clase es representada como un bloque que en nuestro caso sería un gen. Este gen va a tener los siguientes elementos:

Grupo: Grupo al que se le va a dictar la clase. Cada grupo de alumnos va a tener su propia lista de clases. Para este caso estamos asumiendo que los diferentes grupos están llevando las mismas clases a la vez (como en el colegio o las universidades cuando dividen por secciones).
Materias: Cursos a dictar, contienen las siguientes características:
- Nombre de la Materia
- Duración
- Lista de aulas donde se puede dictar la materia
- Profesores disponibles para esta materia

Día: Día de la semana donde se va a dictar la materia
Hora: Hora del día donde comienza la clase
Profesor: Profesor que va a dictar la materia. Cada profesor tiene una lista de horas en las que prefiere dictar clases.
Aula: Aula donde se va a dictar la materia

Especificaciones: Para este problema se tienen las siguientes especificaciones:

- Las horas de dictado de clases pueden ser ingresadas.
- Los días de dictado de clases pueden ser ingresados.
- Los cantidad de grupos de alumnos puede ser ingresado.
- Las horas de dictado preferencial pueden ser ingresadas.

Restricciones (implementadas en código)
Se han implementado las siguientes restricciones tanto al crear el horario o como al realizar la mutación (de tal forma que los cursos en el cronograma siempre van a cumplir con estas restricciones) :

- Los cursos se dictan de corrido en un mismo día. No es posible que un curso inicie un día y termine otro.
- El profesor elegido solo podrá dictar clases a las cuales esté relacionado.
- El aula elegida para una clase deberá ser parte de las aulas disponibles para dicha clase, y esta solo puede albergar un grupo de alumnos a la vez.
- Cada grupo de alumnos llevan las mismas materias.
- En la semana solo se puede dictar una materia por grupo.
  * Para controlar esto al crearse los genes se itera y se hace una combinación de materias y grupos. Al realizarse una mutación en un gen puede mutar la hora, el día , el profesor , el aula, pero no la materia ni el grupo.

Restricciones (mediante fitness)

Restricciones Nivel 1:

Estas restricciones deberían cumplirse si o si para tener un horario funcional. Por cada una de las siguientes restricciones que se incumplan se tendrá un puntaje negativo de **-100 en el fitness**. Se espera que con este puntaje negativo se descarte rápidamente las soluciones que incumplan estas restricciones.

1) Ningún profesor puede impartir dos clases al mismo tiempo.
2) Ningún grupo puede escuchar dos clases al mismo tiempo.
3) Ningún aula puede recibir dos clases al mismo tiempo.

Restricciones Nivel 2:

Estas restricciones son los objetivos que se quieren minimizar para el algoritmo genético. Los horarios que pasaron el nivel 1 de restricciones son horarios factibles, sin embargo pueden tener muchas horas de inactividad o "huecos" tanto para los profesores como para los grupos.

Con las siguientes restricciones se minimiza la cantidad de "huecos" en el día optimizando los horarios. Se ha considerado un puntaje de **-30 por cada hora de pausa**. Las restricciones son:

1) Minimizar el "tiempo de inactividad" por cada grupo.
2) Minimizar el "tiempo de inactividad" por cada profesor.
3) Penalización por cada día que no tenga clase.
4) Penalización por cada hora asignada fuera del horario de preferencia del profesor.

El punto 3 se consideró ya que después de realizar algunas pruebas se creaban horarios funcionales pero que tendían a agrupar las clases en ciertos días y dejar uno o dos días vacíos. Con esto se mejora la distribución de las horas.

El punto 4 no se restringió a nivel de código ya que aunque los horarios resultantes deberían ajustarse a la disponibilidad del profesor, siempre hay la posibilidad de que algunos profesores sean flexibles y acepten dictar una hora más allá de su horario previamente seleccionado.

Restricciones Nivel 3 (mejoras):

Cumpliendo las restricciones nivel 1 y 2 se tiene un fitness positivo lo que significa un horario funcional y optimizado para minimizar las pausas de los horarios.

Sin embargo, se podría seguir mejorando el horario. Por lo que se consideró las siguientes mejoras:

1) Horas premium: En todo instituto educativo hay horas que son preferidas por los usuarios, por ejemplo las horas de la noche pueden ser ideales para quienes trabajan. En este caso se asigna un puntaje positivo de **+1 por cada hora dentro del horario premium.**
2) Distribución de los cursos en la semana para los profesores:Para un profesor es mejor tener la carga de cursos distribuida durante la semana que tener uno o dos días llenos de cursos. Para lograr esto se está asignado un fitness positivo de **+1 por cada día que el profesor tiene clases**, Por ejemplo, si tiene clases de lunes a viernes, tendría 5 puntos, pero si todas sus clases se dictan en solo 2 días tendría 2 puntos.
3) Distribución de los cursos en la semana para los grupos: Se aplica el mismo criterio que en el punto 2.

Función Fitness:

En función a lo descrito anteriormente, se implementó las siguientes funciones Fitness

fitness_mono_objetivo=(dias_ocupados_profesores)+(dias_ocupados_grupos)+(horas_premium)-(dias_no_usados)-(profesor_extras)-(horas_libres_profesores*30) - (horas_libres_grupo*30) - (100*conflictos)

fitness_multi_objetivo[0]=(dias_ocupados_profesores)+(dias_ocupados_grupos)+(horas_premium)-(dias_no_usados)-(profesor_extras)-(horas_libres_profesores*30)- (100*conflictos)

fitness_multi_objetivo[1]=(dias_ocupados_profesores)+(dias_ocupados_grupos)+(horas_premium)-(dias_no_usados)-(profesor_extras)-(horas_libres_grupo*30) - (100*conflictos)

Datos de Salida:

El resultado de la implementación es el horario semanal con todas las clases asignadas en los días y rango de horas brindados.

2. Metodología

o Técnicas base (baselines):

- El algoritmo base a utilizar en esta experiencia es el algoritmo genético mono-objetivo, esto debido a que su foco hace que se descuiden las múltiples restricciones que puede tener este tipo de problemas y que hace que tienda a una solución sub-óptima sin considerar todas las variables involucradas.

- Este algoritmo está siguiendo de igual manera la misma secuencia de pruebas que el algoritmo multi-objetivo; es decir, se le aplica de forma iterativa la variación de los hiper parámetros para obtener su mejor fitness y poder comparar su rendimiento con el fitness provisto por su par más óptimo.

o Enfoque(s) propuestos:

- Describir la representación de estados/individuos elegida

Individuo: Cada individuo es un horario compuesto de GxM genes, donde G es el número de grupos y M es el número de materias. Por ejemplo si tenemos 3 grupos y 5 materias distintas tendríamos 15 clases para distribuir en el horario.

Los individuos se crean mediante un for anidado de Grupos y Materias por lo que el orden de los genes del individuo es importante para que no se repitan las clases. Por ello solo estamos usando "crossover one point" y "multipoint" pero no el "crossover order permutation".

Gen: Los genes tienen la siguiente estructura:

Código del grupo	Código de la Materia	Día de la Semana	Hora de Inicio de la Materia	Código del Profesor	Código del Salón
G1	MA	LU	11	JU	C1

Figura 1. Representación del gen.

Ejemplo: 'G1MALU11JUC1'

'G1'= Código del grupo, en este caso el Grupo 1
'MA'=Código de la materia, ejemplo: Matemáticas
'LU'=Dia de la semana (Lunes)
'11' =Hora del día (11 am)
'JU'=Código del profesor (Juan)
'C1'= Código del salon

La mutación de los genes se realiza con Oneflip o Multiflip.Como se tiene un número de clases exacto a ordenar en el horario y cada clase se identifica con un grupo y materia , solo se realizaría la mutación del día, la hora, el profesor y el salón.

- Describir el comportamiento entrada/salida del (los) enfoques.

El algoritmo recibe como entrada lo siguiente:

Un archivo con las materias que se van a dictar. Cada línea de la materia contiene el nombre de la misma , el código, la duración y la lista de aulas donde se puede dictar.

Un archivo con los profesores. Cada línea del profesor contiene el nombre del profesor, el código, la lista de materias y las horas con disponibilidad para dictar.

Por último el usuario ingresa: Hora de inicio y fin de las clases, hora de inicio y fin del horario premium, días de las semana que se va a dictar clases y número de grupos de alumnos que van a llevar las clases.

Como salida o resultado se genera el mejor horario del algoritmo monoobjetivo (de acuerdo al fitness) y un horario aleatorio de la lista de horarios de la frontera de pareto del algoritmo multiobjetivo.

- Describir los operadores/funciones desarrolladas para solucionar el problema planteado. No describir métodos genéricos, sino detalles de cómo se aplicó/adaptó al problema

Para solucionar el problema se creó la clase GenClase, la cual tiene las siguientes funciones:

1) Importar_datos: Con esta función se importan los datos de los profesores y de las asignaturas desde archivos .csv
2) Initialize: Esta función crea un gen. Recibe como parámetros el grupo y la materia y asigna de forma aleatoria el día, la hora, el profesor y el aula, siguiendo las restricciones para que sea un gen válido.
3) Mutar_gen: recibe un gen y primero escoge si va a mutar el día, la hora, el profesor o el aula, y a continuación selecciona aleatoriamente siguiendo las restricciones.
4) Horas_curso: Devuelve una lista de horas de un curso seleccionado. Por ejemplo si el curso dura 3 horas y empieza a las 10, devolvería [10,11,12]. Esta función se usa para validar cruces de horarios.
5) Evaluacion_cromosoma: Evalúa un cromosoma en base a las restricciones mencionadas anteriormente y devuelve los puntajes que se van a usar en la función fitness.

También se crearon las siguientes funciones:

6) Imprimir_horario: recibe una solución (horario) y lo imprime en pantalla.
7) Exportar_horario: recibe una solución (horario) y la prepara para la exportación a un archivo .csv

Para la parte del algoritmo genético se usó como base el algoritmo de la mochila mono y multi objetivo. Primero se unió ambos colab y posteriormente se realizó cambios en los siguientes puntos:

8) Mutación Flip y Multiflip: Se realizó el cambio para llamar a la función Mutar_gen pasandole el gen a mutar
9) Funcion get_fitness Mono y Multiobjetivo: Llama la función descrita anteriormente. Para el multiobjetivo se separa en 2 objetivos diferentes las horas libres de los profesores y los grupos, en la función Mono Objetivo usas ambas horas libres en la misma función
10) Función init_population: Hacer un recorrido (for anidado) por grupos y materias y crea los genes pasandoles el grupo y materia respectiva, de tal forma que se genere aleatoriamente los demás valores.

    GX=Grupo
    MX=Materia
    DI=Dia aleatorio
    HO=Hora aleatoria
    PR=Profesor aleatorio
    AU=Aula aleatoria

    Por ejemplo, si tuviéramos 2 grupos y 2 materias, el cromosoma sería así:

    G1M1DIHOPRAU-G1M2DIHOPRAU-G2M1DIHOPRAU-G2M2DIHOPRAU

3. Experimentación y Resultados

o Setup experimental:

- Describir framework/simulador de entorno (si aplica)
- No aplica.
- Describir datos usados (o método para obtenerlos) (si aplica).

Para la experimentación se generaron 3 pares de archivos de profesores y materias. Cada uno de estos archivos tenía diferentes grados de dificultad. El grado de dificultad estaba relacionado con la cantidad de cursos que se deben asignar en el horario.

- Describir las métricas de evaluación

Las métricas de evaluación son:
Fitness: Puntaje obtenido
Conflictos: No. de conflictos de horarios que hacen inviable el cronograma.
Huecos profesores: Nro. de horas de hueco que tienen los profesores al dia
Huecos Grupos: Nro. de horas de hueco que tienen los grupos al dia.
Horas Premium usadas: Horas del curso que se encuentran dentro del horario premium.
Dias ocupados x profesor: Nro de dias de la semana que el profesor dicta clases
Días no usados: días de la semana que no hubo clases
Horas extras usadas: Nro. de horas asignadas al profesor que se encuentran fuera de su horario disponible

- Describir los experimentos hechos (qué componentes/ parámetros se probaron, qué valores, qué estrategia de validación).

Para la prueba se consideró los siguientes parámetros:

Minimo Pop Size [100,300]
Maximo Pop Size [100,300]
Número de generaciones: [300,500,1000]
Porcentaje de mutación [10%,50%,80%]
Método de crossover ["uniform", "onepoint"]
Método de mutación ["flip", "multiflip"]
Metodo de seleccion de padres ["Torneo"] (*)
Metodo de seleccion de sobrevivientes ["Ranking"] (*)

Considerando todas las variaciones de los parámetros se tuvo que ejecutar el algoritmo 144 veces, una por cada combinación de parametos.
(*) No se seleccionó “Ruleta” ya que generaba problemas

Detallar características del computador y lenguaje utilizado.

Se ejecutó en un Google Colab (sin GPU) y en lenguaje de programación Python.

Los experimentos deben ser planificados para poder caracterizar/compararlos enfoques desarrollados. Aquí algunas preguntas que deben responder los experimentos

- ¿Los enfoques desarrollados resuelven siempre el problema?
- Depende de la data y las restricciones. En el caso del horario, si la cantidad de grupos de alumnos y de materias a dictar es alta y la cantidad de profesores y de aulas es baja no se va a poder lograr un resultado óptimo no importa cuantas veces se ejecute, simplemente porque la cantidad de horas disponibles por hora y salón es superada por la demanda de horas de grupos por materias..

- ¿Qué tan eficientemente lo resuelven?
- Si la cantidad de horas disponibles es suficiente y las restricciones de horarios o disponibilidad de profesores y aulas no son tan ajustadas , en la mayoría de los casos logra un puntaje positivo por lo que se superan todas las restricciones críticas.

- ¿Cuál es el desempeño relativo entre los diferentes modelos?

- En las pruebas realizadas ha funcionado mejor el mono objetivo. El multi objetivo incumplia más restricciones críticas que el mono objetivo.

- ¿Cómo se comportan los enfoques en diferentes instancias del problema (alto/bajo número de datos, alta/baja complejidad del entorno, más/menos tiempo permitido de respuesta, etc.)?
- Cuando la complejidad es alta (mucha demanda de horas, muchas restricciones en los horarios) logra eliminar las restricciones más importantes pero no obtiene un horario perfecto. Con una complejidad más baja obtiene horarios muy buenos. La performance mejora cuanto más generaciones se ejecuta el algoritmo y cuando consideramos una mutación de 0.5

- ¿Cómo influyen los parámetros del enfoque en su desempeño?
- Uno de los parámetros que más influye es la tasa de mutación en todos los casos la tasa de mutación ideal es del 50%. Para entornos de complejidad media o alta influye el tamaño de la población y el número de generaciones, cuanto más altos mejor.

o Resultados y Discusión:
- Presentar resultados numéricos generados en los experimentos por los baselines y el (los) enfoque(s) planteado(s). Hacer un análisis de dichos resultados

	NIVEL_1		NIVEL_2		NIVEL_3	
DESCRIPCIÓN	MONO OBJETIVO	MULTI OBJETIVO	MONO OBJETIVO	MULTI OBJETIVO	MONO OBJETIVO	MULTI OBJETIVO
BEST FITNESS	74	148 (74, 74)	101	146(88,58)	-30	(-64,-184)
CONFLICTOS	0	0	0	0	0	1
HUECOS PROFESORES	0	0	0	0	0	3
HUECOS GRUPOS	0	0	0	1	4	7
HORAS PREMIUM USADAS	65	65	125	125	170	189
DÍAS OCUPADOS x PROFESOR	5	5	4	5	4	5
DÍAS OCUPADOS x GRUPOS	4	4	5	5	5	5
DÍAS NO USADOS	0	0	0	0	0	0
HORAS EXTRAS USADAS	0	0	33	47	89	73
TAMAÑO MÍN. POBLACIÓN	100	100	300	300	300	300
TAMAÑO MÁX. POBLACIÓN	100	100	300	300	300	300
NÚMERO DE GENERACIONES	300	1000	1000	1000	1000	1000
PORCENTAJE DE MUTACIÓN	0.5	0.5	0.5	0.8	0.5	0.5
TIPO DE CROSSOVER	"onepoint"	"onepoint"	"uniform"	"uniform"	"onepoint"	"onepoint"
TIPO DE MUTACIÓN	"flip"	"flip"	"multiflip"	"multiflip"	"flip"	"flip"
MÉTODO SELECT PADRES	"tournament"	"tournament"	"tournament"	"tournament"	"tournament"	"tournament"
MÉTODO SELECT SOBREV.	"ranking"	"ranking"	"ranking"	"ranking"	"ranking"	"ranking"

Según los resultados obtenidos se observa que para horarios menos complejos (nivel 1) no se necesita un número muy alto de generaciones (300) ni de población (100). Sin embargo para niveles más altos si es favorable un número mayor de generaciones (1000) y población (300).

En todos los casos un porcentaje de mutación intermedio (0.5) seria la mejor elección, por lo que la mutación favorece para tener mejores resultados.

Respecto al tipo de crossover o de mutación no hay una tendencia clara por lo que cualquiera de las opciones puede funcionar adecuadamente.

Se observa que en el nivel 2 y el 3 se ha obtenido un mejor resultado con el algoritmo mono objetivo el cual genera menos conflictos y menos horas libres de los profesores y los grupos que el multiobjetivo.

4. Conclusión

Dar las conclusiones principales con base a los resultados obtenidos y a lo que fue planteado en su hipótesis. ¿qué se puede decir del o los enfoques desarrollados y/o del problema abordado?

1) El algoritmo genético multi-objetivo planteado en este trabajo tiene un mejor desempeño en relación al mono-objetivo, debido a que el primero busca minimizar el tiempo libre tanto de docentes como de clases de los grupos.
2) A pesar de ser un caso de estudio, queda demostrado que haciendo uso de algoritmos genéticos se mejora la eficiencia del proceso de elaboración de horarios en tiempo.
3) El algoritmo genético mono-objetivo funciona mejor para entornos planteados cuya complejidad es menor. En un nivel de mayor complejidad del problema de optimización de horarios, en los niveles 2 y 3 el algoritmo multi-objetivo incumple más restricciones hard.
4) Los mejores parámetros del algoritmo mono-objetivo son replicados en el algoritmo multi-objetivo para entornos no complejos, logrando una maximización absoluta del fitness para ambos objetivos. En cambio, en entornos complejos, existe una variación en los parámetros de solución entre el multiobjetivo y el mono objetivo, así como en sus valores fitness obtenidos.
5) La optimalidad del resultado obtenido es dependiente de los parámetros condicionantes iniciales que definen el entorno, dígase: “día de inicio”, “día de fin”, “hora de inicio”, etc.

5. Sugerencias de trabajos futuros

Indicar por ejemplo: ¿qué cosas se pueden mejorar del enfoque?, ¿qué posibles otros problemas podría abordarse con el enfoque?

Posibles Mejoras

1) Se podría explorar otra representaciones, por ejemplo en vez de explorar grupos por materias, podría explorarse una representación del cromosoma de horas por salones.
2) Se podría configurar una interfaz en web o en un app móvil para ingresar los datos de los profesores y las materias.
3) Dar más fitness a los horarios que distribuyan de forma más equitativa sus horas a través de los días tanto para los profesores y los grupos (para que las horas no tiendan a acumularse en pocos días)
4) Se ha reducido la cantidad de espacios vacíos para los profesores y grupos. Se podría considerar en el fitness un ajuste para los espacios vacíos en general y premiar a los horarios más compactos.
5) Dar más fitness a los horarios que distribuyen las horas entre los salones de forma más equitativa en el mismo día.
6) Implementar restricción de hora libre
7) Dar puntaje a los profesores e incrementar el fitness a los horarios que usan más horas de esos profesores bien calificados. Otra opción sería distribuir los horarios de forma más equitativa entre los profesores, por lo que se podría dar más fitness a los horarios que de una cantidad similar de horas entre los profesores de un mismo curso.
8) Colocar restricciones de horario por salón o restricciones especiales; por ejemplo:, el laboratorio de biología o química necesitaría una hora después de clase para hacer limpieza, debido a los experimentos.
9) Dar la posibilidad de que cada grupo tenga diferentes cursos.
10) Dar la posibilidad de que las horas de las materias se dividan y repartan en días diferentes.

Posibles problemas: Con este enfoque se podría solucionar los siguientes problemas (haciendo breves ajustes en el código):

- Asignación de horarios para conferencias
- Asignación de horarios para clases virtuales
- Distribución de citas para clínicas
- Distribución de habitaciones para hoteles

6. Link del repositorio del trabajo

- https://github.com/LJFiorela/Optimizacion
- (acceso público)

7. Declaración de contribución de cada integrante

Describir los aportes de cada integrante al proyecto

- Desarrollo Clase GenClase: Ricardo y Yulian
- Algoritmo Mono Objetivo: Ricardo , Yulian y Frank
- Algoritmo Multi Objetivo: Fiorela y Frank
- Código para importar datos: Fiorela y Yulian

- Codigo para imprimir horario (en pantalla): Ricardo
- Codigo para exportar horario (CSV): Fiorela
- Código para probar los hiperparametros: Frank
- Powerpoint (Todos)
- Informe Final (Todos)

8. Referencias

[1]. Nevena Drešević "Timetable Generator", Github, 2019. Dirección electrónica: https://github.com/LJFiorela/Optimizacion

[2]. Samuel Lukas, Arnold Aribowo, Milyandreana Muchri, "Genetic algorithm and heuristic search for solving timetable problem case study: Universitas Pelita Harapan timetable", IEEE, 2009. Dirección electrónica: https://www.researchgate.net/publication/221927228_Solving_Timetable_Problem_by_Genetic_Algorithm_and_Heuristic_Search_Case_Study_Universitas_Pelita_Harapan_Timetable

[3]. Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, "A Genetic Algorithm To Solve The Timetable Problem", Politecnico di Milano,, 1994. Dirección electrónica: https://www.researchgate.net/publication/2253354_A_Genetic_Algorithm_To_Solve_The_Timetable_Problem

[4]. Achini Kumari Herath, "Genetic Algorithm For University Course Timetabling Problem", University of Mississippi, 2017. Dirección electrónica: https://egrove.olemiss.edu/cgi/viewcontent.cgi?article=1442&context=etd

[5]. Branimir Sigl, Marin Golub, Vedran Mornar, "Solving Timetable Scheduling Problem by Using Genetic Algorithms ", University of Zagreb,. Dirección electrónica: http://www.zemris.fer.hr/~golub/clanci/iti2003.pdf

[6]. Esraa A.Abdelhalim, Ghada A.El Khayat, "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem (UGA) ", Alexandria University 2016,. Dirección electrónica: https://www.sciencedirect.com/science/article/pii/S1110016816000703

[7]. Andrew Reid East "Timetable Scheduling via Genetic Algorithm", National University of Ireland, Galway 2019,. Dirección electrónica: https://andreweast.net/wp-content/uploads/2019/06/Timetable-Scheduling-via-Genetic-Algorithm-Andrew-Reid-East.pdf

[8]. Ashish Jain, Dr. Suresh Jain and Dr. P.K. Chande "Formulation of Genetic Algorithm to Generate Good Quality Course Timetable", International Journal of Innovation 2010,. Dirección electrónica: http://www.ijimt.org/papers/46-M431.pdf

[9]. Vijini Mallawaarachchi, "Using Genetic Algorithms to Schedule Timetables", Towards data science 2020,. Dirección electrónica: https://towardsdatascience.com/using-genetic-algorithms-to-schedule-timetables-27f132c9e280

[10]. Norris Lee Bravo Hermitaño,, "Algoritmo genetico para la elaboración de horarios de clases en universidades con recursos compartidos, caso de estudio: Universidad Nacional Agraria de la Selva" Universidad Nacional Agraria de la Selva 2019,. Dirección electrónica: http://repositorio.unas.edu.pe/bitstream/handle/UNAS/1660/TS_BHN_2019.pdf?sequence=1&isAllowed=y