UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

FACULTAD DE CONTADURÍA, ADMINISTRACIÓN E INFORMÁTICA

MAESTRÍA EN OPTIMIZACIÓN Y CÓMPUTO APLICADO

Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios

T E S I S

Que para obtener el Grado de

Maestría en Optimización y Cómputo Aplicado

Presenta

Lorenzo Antonio Cardoso Contreras

Director de Tesis

Dr. Federico Alonso Pecina

Co-Director

Dr. Marco Antonio Cruz Chávez

Revisores:

Dra. Irma Yazmin Hernández Báez

Dr. Federico Alonso Pecina

Dr. Martín Heriberto Cruz Rosales

Dr. Marco Antonio Cruz Chávez

Dr. José Crispín Zavala Díaz

CUERNAVACA, MORELOS MARZO,2022

CARTA CERTIFICADA DE NO PLAGIO

A QUIEN CORRESPONDA,

Siendo las 10:00 horas del día 30 de marzo de 2022 en las instalaciones de la Facultad de Contaduría, Administración e Informática de la Universidad Autónoma del Estado de Morelos, declaramos los abajo firmantes, que el trabajo de investigación Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios, desarrollado por el estudiante Lorenzo Antonio Cardoso Contreras, de la generación 2019-2021 de la Maestría en Optimización y Computo Aplicado.

1. No tiene plagio.

2. Las citas de otros autores han sido debidamente identificadas en el trabajo, por lo que no se ha asumido como propias las ideas vertidas por terceros, ya sea de fuentes encontradas en medios escritos como en Internet.

3. No ha sido presentado anteriormente para obtener algún grado académico o título.

Sabemos que este compromiso de autenticidad y no plagio puede tener connotaciones éticas y legales. Por ello, en caso de incumplimiento de esta declaración, el Director de Tesis, Co - director y el estudiante se someten a lo dispuesto en las normas académicas que dictamine el Comité de Ética de la Universidad Autónoma del Estado de Morelos.

Se firma la presente carta por:

Dr. Federico Alonso Pecina	Dr. Marco Antonio Cruz Chávez	Lorenzo Antonio Cardoso Contreras
Director de Tesis	Co- Director de Tesis	Estudiante

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

FEDERICO ALONSO PECINA | Fecha:2022-03-30 00:17:35 | Firmante

vqUZr8v1QU9zb9lZg7GdiNmfgTbcccnIhhmOPeYOUI+0U4LOQ0owQoP2b9quuT0FMEXAJevSg+Gl23MHPJvb0X0wMmFodwnC5yap5TwqNbxYWC43+loijdfNB2urXR/Jz5t7JcJ
8DevcXzNhC0buOEhHuhSAFYWsE1Yjihw0ARlK7es87jQCQF4dN2O/W+DVZDGH9C2rExK5jlcA3c9HQzXSKl4qSOEw0K3Ahl0soYetNNrTCjIC7AuNfja542WREutNquOm5ikwfI
+vfEJ7jhGJbRx5gk3VeDGO1nCGBNNtwb+A9QrHCDskTDYIsZ/EZVi3KpguBLQlWpqO+DJiug==

MARCO ANTONIO CRUZ CHAVEZ | Fecha:2022-03-30 15:09:54 | Firmante

tc39gUAYtjiupf/+qSKO7B8ROu0okFgT/pZNXt1PnM2ToeXV1yEB+RK3lfnazaF77cmFzXU/RKZBDWpbJK4GnDQ3lS+Bl6hfk50GdMJr7u90MUTbIIuzLVQ/FLQ6WSKco1mCf0Cp
UF9OCGYSL5+RTqIqp/y8GUKId2TzuldTobg55P6oE6oEJnayV1IEEyO5w0eHlM8a/N2YRj55hV8oe8jTUs8YtPSfdKHbepk26LdlAPYX7LWM5NKZlRTvxW+Z6DKaGis+F6/OB354
+VgrCjp+jGrxrHFG2WIjCEDEzgS1rOakXxUxS5ewTivWSQx7W0lbnhSE18dWQ455rTgpkA==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

2fD0xYQWv

https://efirma.uaem.mx/noRepudio/bBPIIGavd6Kl8WApxL6hi3k2E7OS6fXG

Cuernavaca, Morelos, a 23 de marzo de 2022

CERTIFICACIÓN DE AUTORÍA

Yo, Lorenzo Antonio Cardoso Contreras, certifico que la Disertación titulada, **Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios,** la cual presento como requisito para optar por el grado de Maestro en Optimización y Cómputo Aplicado en la Facultad de Contaduría, Administración e Informática de la Universidad Autónoma del Estado de Morelos en el país México, es el producto de mi labor investigativa.

Asimismo, doy fe de que este trabajo ha observado las normas establecidas en el Reglamento de la Universidad y del posgrado de la Universidad Autónoma del Estado de Morelos para su realización.

Con lo anterior, deslindo a la institución de toda acción que genere plagios y asumo toda responsabilidad.

Lorenzo Antonio Cardoso Contreras

Matricula:10034118

RESUMEN

En esta tesis se aborda el problema de los horarios universitarios (UCTP). En el contexto de una universidad, el cual se encuentra dentro de los problemas NP-Completos. El problema consiste en la asignación de una serie de eventos (conferencias, exámenes, tutorías, sesiones de laboratorio, etc) a un número limitado de intervalos de tiempo y salones, de modo que se cumplan una serie de restricciones. El problema que se aborda en esta tesis contempla tres conjuntos de restricciones duras, las cuales deberán cumplirse para obtener un horario factible.

Se trabajó con el benchmark propuesto por Rhyan Lewis: 60 instancias que tienen una complejidad mayor a las utilizadas por la competencia de Practice and Theory of Automated Timetabling (PATAT, 2002), la cual está enfocada en el estudio del problema de Timetabling (el problema de programación de horarios).

Para encontrar Factibilidad en dichas instancias, se abordan 3 algoritmos, el primero es un enfoque Heurístico que ayuda crear la solución inicial y también logra hallar factibilidad en algunas instancias de tipo “Small”, posterior se desarrollaron dos enfoques Metaheurísticos: Aceptación por Umbral y Recocido Simulado, en los cuales se implementaron los vecindarios “Mover evento” e “Intercambio de eventos”.

Se logro encontrar 55 soluciones factibles solo por debajo de un algoritmo que ha encontrado 58. Con la implementación de las metaheurísticas se logró aumentar la factibilidad en las soluciones encontradas.

El porcentaje de factibilidad aumento considerablemente entre la heurística del “evento más restringido” y las Metaheurísticas aceptación por Umbral y Recocido Simulado.

La heurística obtuvo 8.33% de factibilidad, posteriormente al emplear Aceptación por Umbral se obtuvo el 70% de factibilidad, finalmente con la implementación de Recocido Simulado se obtuvo un 91.66% de factibilidad.

ABSTRACT

This tesis address the university Course Timetabling problem (UCTP), In the context of a university, which is within NP-Complete problems. The problem consists of assigning a series of events (conferences, exams, tutorials, lab sessions, etc.). to a limited number of time slots and rooms, so that a set of restrictions are met. The problem tackle in this thesis contemplates 3 hard restrictions which must be met to obtain a feasible schedule.

We worked with the benchmark proposed by Rhyan Lewis: 60 instances that have a greater complexity than those used by the Practice and Theory of Automated Timetabling competition (PATAT 2002), which is focused on the study of the Timetabling problem (the problem of scheduling university events).

To find Feasibility in these instances, 3 algorithms are approached, the first is a Heuristic approach that helps to create the initial solution and also manages to find Feasibility in some instances of type "Small". After we developed two metaheuristic approaches: Threshold Accepting and Simulated Annealing, in which the neighborhoods "Move event" and "Exchange events" were implemented.

It was possible to find 55 feasible solutions only below an algorithm that has found 58. With the implementation of metaheuristics, it was possible to increase the feasibility of the solutions found.

The percent of feasibility increases considerably between the heuristic of the "most restricted event" and the metaheuristics Threshold Acceptance and Simulated Annealing. The heuristic obtained 8.33% of feasibility, later when using Threshold Acceptance, 70% of feasibility was obtained, finally with the implementation of Simulated Annealing, a 91.66% of feasibility was obtained.

AGRADECIMIENTOS:

A Mi madre Antonia Contreras, por darme la vida, quererme mucho, creer en mí, haberme apoyado en todo momento, por sus consejos, sus valores, y por su gran amor incondicional. Mi padre Lorenzo Cardoso, Por los ejemplos de perseverancia y constancia que lo caracterizan y que me ha infundado siempre, por el valor mostrado para salir adelante.

A mis hermanos Edgar, Rodrigo y Ricardo por estar conmigo y apoyarme en mis estudios, por compartir alegrías y tristezas, por estar siempre en los malos y buenos ratos, por su confianza y permitirme estar en sus vidas.

Quiero expresar también mi más sincero agradecimiento Mi asesor el Doctor Federico Alonso pecina por su inestimable ayuda y paciencia, así como su tiempo compartido que me permitió, para la culminación y elaboración de esta tesis.

A Yasmin Guadarrama, por su apoyo y paciencia que me brinda día con día, por tantas ayudas y tantos aportes no solo para el desarrollo de mi tesis, sino también para mi vida.

A mis sinodales Dra. Irma Yasmin Hernández Báez Dr. Marco Antonio Cruz Chávez, Dr. José Crispín Zavala Díaz, Dr. Martin Heriberto Cruz: por sus valiosas aportaciones y comentarios que fortalecieron mis conocimientos a lo largo de esta estancia.

A mis amigos Eduardo, Jonathan, Erick, Urías, Luis, Carlos, Emmanuel por los buenos momentos que hemos compartido, amistad y apoyo moral han aportado en un alto porcentaje a mis ganas de seguir adelante en mi carrera profesional

GLOSARIO

Instancia dada de un problema que se utiliza como parámetro para medir el desempeño de las técnicas utilizadas para resolver el mismo.	***Benchmark***
Datos de entrada para el procesamiento de un problema determinado.	***Instancia***
No Polinomiales	***NP***
Practice and Theory of Automated Timetabling, son conferencias internacionales de Timetabling que tiene lugar cada dos años a nivel mundial.	***PATAT***
Calendarización de recursos	***Timetabling***
(TTP) Problema de Programación de Horarios	***TimeTabling Problem***
(UCTP) Problema de Programación de Cursos Universitarios	***University Course Timetabling Problem***

ÍNDICE DE TABLAS

ÍNDICE

CAPÍTULO I INTRODUCCCIÓN

1.1 INTRODUCCIÓN

El problema de programación de horarios es de suma importancia en muchas áreas de la vida diaria, como el trabajo, la educación, el transporte, el entretenimiento, etc. En muchos casos de la vida real, cuando los recursos son limitados (como personas, vehículos, espacio o tiempo), la construcción de horarios se debe intentar resolver de la mejor manera, los horarios a menudo deberán actualizarse o rehacerse por completo, por ejemplo, los horarios escolares continuamente se elaboran desde cero al comienzo de cada año académico [Lewis & Paechter 2007]. Los problemas de programación incluyen la generación de horarios para tareas definidas que se establecen para cumplir mejor las condiciones o requisitos específicos. Estos problemas son muy comunes, Se puede encontrar en diferentes tipos, por ejemplo: horarios de preparatoria, horarios universitarios (de exámenes y cursos) [André, 2018].

El problema de programación de horarios escolares tiene como objetivo encontrar un horario que cumpla con todos los requisitos establecidos, por lo general se optimizan, dividiéndolo en dos tipos de restricciones, restricciones duras y restricciones suaves, cuyo objetivo es encontrar un horario que cumpla con todas las restricciones duras y minimice las restricciones suaves [Avella et al, 2007]. Entre los problemas de programación de horarios escolares, se encuentran la construcción de cursos y horarios de exámenes universitarios los cuales se catalogan por ser más difíciles entre los horarios escolares. [Bardadym, 1995].

Los problemas de programación de horarios escolares se encuentran el de cursos universitarios donde el objetivo es asignar un conjunto de entidades (como cursos, exámenes, personas) a un número limitado de recursos a lo largo del tiempo, de tal manera que cumpla con un conjunto de requisitos de programación [Lewis, 2008]. Estos problemas se pueden definir de forma general, como la asignación de un conjunto de eventos a un espacio y tiempo cumpliendo una serie de restricciones que implican un conjunto limitado de recursos [Henry, 2010], la programación de horarios de cursos en una universidad tiene como objetivo el garantizar que todos los estudiantes tomen sus materias requeridas apegándose a los recursos que están disponibles.

El problema de programación de cursos universitarios ha sido de gran interés por parte de los investigadores, donde existe una organización llamada Practice and Theory of Automated Timetabling (PATAT) que está enfocada en el estudio del problema de Timetabling (el

problema de programación de horarios). Las conferencias del PATAT, se llevan a cabo cada dos años y sirve como foro para una comunidad internacional de investigadores, profesionales y proveedores sobre todos los aspectos de la generación de horarios asistida por computadora.

1.2 ANTECEDENTES

Se puede hacer una importante generalización del problema de horarios de universidad, es decir, el problema es NP-completo en casi todas las variantes. De hecho, desde varios aspectos, equivale a la coloración gráfica, el embalaje de envases y la combinación tridimensional [Cooper, 1995.], existen algunos casos en los que se puede resolver en tiempo razonable, algunas de estas métricas son: lineal, logarítmica, cuadrática, polinomial, exponencial [Bradley, 1997].

Muchos problemas de optimización matemática de aplicaciones son NP-hard y, por lo tanto, no se ha encontrado un algoritmo que los resuelva en un tiempo polinomial. Por esta razón, las metaheurísticas son una manera de abordar estos problemas y encontrar soluciones de calidad en tiempos razonables. [Limota et al, 2021].

La primera definición del problema de horarios fue introducida por [Gotlieb, 1963], dividida en tres conjuntos: estudiantes, salones e intervalos de tiempo para las instituciones educativas, la construcción de horarios es un problema combinatorio clásico que abarca la programación de encuentros entre profesores y alumnos para asegurar algunos requisitos dados [Schaerf & Di Gaspero, 2001]. Generalmente, el proceso de optimización se centra en asignar una serie de eventos o conferencias a un período de tiempo prefijado a lo largo de los días laborables de la semana [Harrabi & Siala, 2020]. Dentro del ámbito académico se encuentra la calendarización de horarios, generalmente el problema en el TTP (TimeTabling Problem por sus siglas en inglés) se consideran dos tipos de restricciones de las cuales desprenden duras y suaves, la primeras son indispensables satisfacer para que el horario resulte factible, mientras que las suaves son deseables de cumplir, más no indispensables para que el horario sea factible [Schaerf, 1999]; En lo particular, en el caso de educación, se divide el problema en tres tipos: programación de horarios en escuelas, programación de exámenes y programación de cursos universitarios [Schaerf, 2001], en esta tesis nos enfocamos en los cursos universitarios. El problema de horarios de cursos universitarios (UCTP por sus siglas en inglés University Course Timetabling Problem) es un problema clásico y famoso en el campo de los problemas de optimización, es común en instituciones académicas como escuelas, colegios o universidades [Chen et al, 2021]. El objetivo del problema de horarios de cursos universitarios es encontrar

un método para asignar eventos a intervalos de tiempo predefinidos y salones, donde todas las restricciones dentro del problema deben estar satisfechas. Los eventos incluyen estudiantes, profesores y cursos, dónde los recursos abarcan las instalaciones y equipos de los salones. El UCTP es diferente de otros horarios escolares, dado que los estudiantes pueden elegir entre muchos cursos diferentes. Una solución al problema es una combinación de salones y eventos asignadas a intervalos de tiempo [Bardadym, 1995].

1.3 PLANTAMIENTO DEL PROBLEMA

En 2002, la primera edición del concurso internacional de horarios (ITC, por sus siglas en inglés). Organizado por Red de Metaheurísticas (Metaheuristics Network) este un proyecto de la Comisión Europea emprendido por cinco institutos europeos para comparar y analizar el rendimiento de varios algoritmos metaheurísticos para diferentes problemas de optimización combinatoria (incluidos los horarios). El objetivo principal es diseñar un problema de optimización combinatoria para que los investigadores intercambien ideas y se propongan nuevas soluciones [ITC, 2002]. En este caso, dentro del problema se propusieron tres restricciones duras y tres restricciones suaves. Junto con un conjunto de 20 instancias todo con el objetivo de que investigadores y profesionales aplican diferentes métodos para abordar el problema. En años posteriores, este problema y sus instancias se utilizaron como referencia para que más investigadores probaran la eficacia de sus propias técnicas de solución.

Lewis y paetcher Generaron 60 instancias de prueba tiene una mayor complejidad que las utilizadas en el Competencia UCTP 2002 PATAT [Lewis, 2007A], en el sentido de que las soluciones factibles eran más difíciles de obtener. Los autores informaron que algunas técnicas secuenciales tradicionales solo podían programar alrededor del 80% de los eventos. Hasta ahora, no hay algoritmos que puedan resolver las 60 instancias de prueba. El problema de horarios que se aborda en este trabajo son las instancias propuestas por Lewis en 2005, divididas en 3 partes de 20, small, médium y big, las cuales van creciendo de complejidad en cada parte.

1.4 JUSTIFICACIÓN

La programación de horarios escolares en lo particular es un problema que se cataloga dentro del problema general de asignación de recursos, comúnmente se conoce científicamente como University Course Timetabling [Banczyk et al, 2006].

En el campo de la educación, es necesario crear y planificar horarios. De la forma más eficiente en términos de asignar personal, horarios, salones y equipos para lograr mejores usos de

recursos y tendencia, para desempeñarse mejor a los estudiantes y personal académico. La gran cantidad de eventos a programar y la gran variedad de restricciones impuestas en un problema de horarios, hace que el conjunto de posibles soluciones también crezca en complejidad computacional [Burke & Petrovic, 2002].

Algunos problemas prácticos involucran la calendarización de cientos de cursos, donde la capacidad de los salones de estos cursos es limitada y la disponibilidad de periodos de tiempo varía. Establecer un horario puede ser una tarea muy difícil, y resolverlo manualmente puede requerir mucho esfuerzo. La resolución manual de problemas de horarios suele requerir varios días de trabajo, debido al gran tiempo de trabajo requerido, es necesario desarrollar métodos automatizados para programar los horarios.

1.5 HIPOTESIS

Es posible encontrar al menos un 80% soluciones factibles en el benchmark propuesto por Lewis [Lewis, 2007B] para el problema de Timetabling utilizando los algoritmos de Aceptación por Umbral y Recocido Simulado.

1.6 OBJETIVOS GENERAL Y OBJETIVOS ESPECIFICOS

A continuación, se presenta el objetivo general y específicos que se abordan dentro del presente trabajo.

1.6.1 OBJETIVO GENERAL

Desarrollar e implementar un algoritmo para encontrar soluciones factibles en el problema de asignación de cursos universitarios, utilizando el benchmark propuestas por Lewis [Lewis, 2005].

1.6.2 OBJETIVOS ESPECIFICOS

- Implementar una Heurística basada en el evento “más restringido” para obtener una solución inicial
- Implementar las Metaheurísticas de Aceptación por umbral y Recocido Simulado para el problema de horarios universitarios.

1.7 ALCANCES Y LIMITACIÓNES

En el desarrollo del presente trabajo se han identificado los siguientes alcances y limitaciones:

1.7.1 ALCANCES:

- Desarrollar un algoritmo para obtener soluciones iniciales no factibles.
- Implementar Aceptación por umbral y Recocido Simulado para obtener soluciones factibles.
- Sintonización de parámetros de las metaheurísticas utilizadas.

1.7.2 LIMITACIONES

- Se trabajará con las instancias de Rhydian Lewis [Lewis, 2005A]
- Se utilizará el lenguaje C++

1.8 ORGANIZACION DE LA TESIS

En el capítulo II se muestra una revisión del estado del arte de distintos enfoques que han abordado el problema. El capítulo III muestra el procedimiento que se llevó para la solución del problema e implementación. El capítulo IV se muestra la sintonización de los algoritmos y los resultados obtenidos al aplicarlos. El capítulo V se encuentra las conclusiones y trabajos a futuro.

CAPÍTULO II MARCO TEÓRICO

2.1 COMPLEJIDAD COMPUTACIONAL

En busca de soluciones a un gran número de problemas en el área de la computación, se establece que hay algunos más difíciles para resolver que otros, teniendo en cuenta principalmente el tiempo de procesamiento y la cantidad de espacio en la memoria necesario para resolver el problema, abordando esto la complejidad del problema puede ser clasificar en 3 tipos principales: P, NP y NP-COMPLETO. [Naupari & Rosales, 2010]

Problemas P son los que se pueden resolver en el tiempo polinómico, es decir, problemas simples que se puede resolver fácilmente de una manera práctica como la multiplicación, funciones lineales cuadráticas, etc. Todos los problemas encontrados en P es parte de los problemas localizados en NP. [Cook, 2006].

Problemas de NP tienen un concepto similar al de P problemas a medida que se resuelven en un Tiempo polinomial, la diferencia es que no son problemas determinísticos, es decir, normalmente se resuelven utilizando el uso de una máquina de Turing no determinista, que no se sabe cuál es el resultado a dar y el tiempo de procesamiento depende de la cantidad de datos de entrada. Ese tipo de problema incluye los problemas que también pertenecen en las otras clases (P, NP-C). Se dice que contiene los problemas P, porque es posible la aplicación de un algoritmo polinómico que compruebe que la solución dada es válida o no, Los problemas se resuelven en tiempo polinómico y en NP los problemas se comprueban en tiempo polinómico. Principalmente esta clase abarca problemas de búsqueda y optimización, como la utilización de grafos. [Cook, 2006].

Los problemas NP-COMPLETOS, son también problemas NP, los problemas NP pueden ser reducidos a problemas NP-COMPLETOS, y el tiempo computacional requerido aumenta exponencialmente con el tamaño que tenga el problema [Cook, 2006].

Entonces se puede decir que los problemas NP-COMPLETOS son los más difíciles de resolver dentro del conjunto NP, y no están presentes dentro de los problemas P.

También se puede decir que este tipo de problemas son equivalentes entre sí. Si existe una solución para un problema NP-COMPLETO, entonces existe para cualquier problema de este tipo, y si por el contrario se comprobara que un problema NP-COMPLETO no tiene solución, entonces ninguno la tendría, en específico del problema de programación de horarios, que está ubicado en la clase de problemas NP-COMPLETOS [Cook, 2006].

Otra generalización que podemos hacer sobre los problemas de programación en las universidades es que son NP-completos en casi todas las variantes. Lo que esto significa es que no podemos esperar encontrar un algoritmo acotado polinomialmente para resolverlos en general, a menos que P = NP, generalmente no se considera que sea el caso, [Garey et al, 1974] mientras que [Cooper et al, 1996] por ejemplo, han demostrado que la completitud NP existe para una serie de interpretaciones diferentes de problemas que pueden surgir en la práctica. Esto, lo hacen, al proporcionar transformaciones acotadas polinomialmente de varios problemas NP-COMPLETOS bien conocidos, como coloración de grafos (graph colouring, en inglés), empaquetado de contenedores (bin packing, en inglés) y coincidencia tridimensional (three dimensional matching, en inglés) con una serie de variantes diferentes de problemas de programación.

En consecuencia, el problema de programación de horarios es considerado como un problema NP–DURO [Even & Shamir, 1975] lo que quiere decir que no es posible resolver el problema por medio de algoritmos determinísticos de tiempo polinomial.

2.2 PROBLEMA DE HORARIOS

Los problemas de programación de horarios tratan con la asignación de un conjunto determinado de entidades a un conjunto limitado de recursos.

Dependiendo de las entidades, recursos y aportes relacionados, se pueden distinguir otros subproblemas, por ejemplo, en manufactura o servicios como proyecto, programación, planificación y programación en cadenas de suministro, o planificación y programación en transporte. Los problemas de horarios educativos consisten en asignar eventos como conferencias o exámenes a un conjunto de intervalos de tiempo, satisfaciendo restricciones estrictas (por ejemplo, los profesores y los estudiantes solo pueden estar en un lugar al mismo tiempo; los salones tienen una capacidad limitada) y optimizando (minimizando o maximizando) la función objetivo, que refleja la satisfacción de restricciones suaves (por ejemplo, los profesores pueden preferir ciertos intervalos de tiempo, primero y se dejan los últimos intervalos de tiempo del día).

Dependiendo el tipo de institución educativa, tiene como responsabilidad el construir el horario escolar [Schaerf, 1999] describe tres categorías de problemas de horarios escolares:

- Horario escolar: la programación semanal de clases, donde los profesores no pueden enseñar dos clases al mismo tiempo y los estudiantes generalmente son supervisados casi todas las horas del día escolar.

- Horario del curso: la programación semanal para todas las clases de un conjunto de cursos universitarios, minimizando los conflictos de los estudiantes, un estudiante no debe asistir a un evento programado para la misma hora.

- Horario de exámenes: la programación de los exámenes de un conjunto de cursos universitarios, minimizando el traslape de exámenes que tienen estudiantes.

Dada la definición de [Wren, 1996] es necesario saber si existen suficientes recursos disponibles para que el evento seleccionado tenga lugar en el momento especificado, así como los recursos que se asignan. La terminología basada en la programación de horarios es la siguiente:

Tabla 2.1 Terminología de programación de Horarios

Termino	Definición
Evento	Una actividad que se programará. Los ejemplos incluyen exámenes y cursos.
Intervalo de tiempo (período)	Un intervalo de tiempo en el que se pueden programar eventos.
Salón	Donde será asignado el evento que cumpla con las características y satisface los requerimientos.
Características	Recursos requeridos por los eventos. Los ejemplos incluyen laboratorio, computadora, proyectores, pizarrón eléctrico, etc..
Restricciones	Una restricción para programar los eventos. Los ejemplos incluyen capacidad del salón e intervalos de tiempo específico.
Estudiantes	Aquellos que tienen que asistir a los eventos
Conflicto	Dos eventos chocan entre sí, si tienen al menos un estudiante común y están programados en el mismo intervalo de tiempo.

2.3 ALGORITMOS EXACTOS

Los métodos exactos exploran de forma determinística el espacio de soluciones del problema enfocado, asegurando una solución óptima en caso de que existiera, Garantizan el valor óptimo de la solución, el problema es que normalmente tiende a ser computacionalmente elevado, por lo que resultan ser adecuados solamente para instancias pequeñas del problema. Los procedimientos exactos usan condiciones de optimalidad como condición de paro, por ejemplo, que la diferencia del valor de la función objetivo con la cota del problema es subjetivamente diminuta. Cuando la instancia es pequeña, el tiempo de estos procedimientos no resulta mucho problema, por lo cual es conveniente usarlos, algunos ejemplos de dichos procedimientos son: programación dinámica como "divide y vencerás", ramificación y poda, un problema de estos procedimientos exactos, es para finalizar su ejecución. Es posible que lleguen a explorar de forma exhaustiva el espacio de soluciones, de forma que para problemas NP-DUROS no se pueda llegar a una solución óptima en tiempo polinomial [Papadimitrou, 1982].

2.3.1 RAMIFICACION Y PODA

Los límites de un problema de optimización combinatoria se definen como los límites entre los cuales se evalúan y se incluyen una solución óptima, por lo tanto, los límites superiores(UB) e inferiores(LB) se utilizan para estos problemas.

El método de ramificación y poda (Branch and Bound en inglés) enumera soluciones en el espacio de búsqueda de un problema de optimización combinatoria, esta enumeración está hecha por ramificación sucesiva, en subprocesos. Estos valores se realizan calculando los valores de los límites inferiores y superiores. Por ejemplo, para un problema de minimización, acelerando la solución del problema, se puede podar un subespacio si $if LB(Si) \geq UB_{best}$ donde UB_{best} es el mejor límite superior conocido durante la búsqueda, Esta poda implica que no se explorarán las soluciones del subespacio), las soluciones obtenidas mediante enfoques heurísticos representan (UB) mientras que los límites (LB)se obtienen resolviendo una versión rejalada del problema, cuando $LB = UB$ en una instancia, se dice que la instancia ya está resuelta[Clausen, 1999].

La eficiencia del tiempo de ejecución de un algoritmo ramificación y poda depende de los siguientes factores:

- El esquema delimitador: diferentes técnicas para delimitar, esto suele dar como resultado búsquedas de árboles.

- Métodos $UByLB$: cuanto más efectivos son los métodos que evalúan la UB yLB son las decisiones más eficientes que se toman durante la búsqueda.
- Estrategia de exploración: mientras se ramifica, la estrategia utilizada para generar el subespacio a explorar contribuye en gran medida a la eficiencia del algoritmo.

[Burke et al, 2011] describe un procedimiento de ramificación y corte para el problema de cursos universitarios, la formulación tiene un numero exponencial de restricciones, que solo se agregan solo en caso de violación, de igual manera se implementan otras clases de cortes que surgen de enumeración de patrones de eventos y periodos libres, la implementación correspondiente del método de ramificación y poda se evalúa con las instancias del concurso internacional de horarios 2007.

2.3.2 PROGRAMACION LINEAL ENTERA

La programación matemática es otro enfoque utilizado para resolver problemas de horarios. En los últimos años, los avances tecnológicos en las computadoras volvieron a popularizar estas técnicas. [Akkoyunlu, 1973] y [Lawrie, 1969] propusieron modelos de programación lineal y entera para el problema de horarios. Estos se pueden enumerar entre los primeros enfoques en matemática programación. Akkoyunlu tenía como objetivo prevenir cualquier conflicto durante el proceso de asignación.

[Daskalaki et al, 2004] presentó una formulación de programación entera para un caso de horarios universitarios, afirman que su modelo es completo y puede producir horarios que no tengan conflicto.

Los métodos de programación lineal entera han sido efectivos en casos de instancias medianas y pequeñas o experimentales. En el caso de instancias más grandes no ha sido fácil encontrar soluciones factibles a este tipo de problemas. Sin embargo, en algunos casos ha sido posible encontrar soluciones a instancias medianas o grandes utilizando técnicas de agrupación de eventos y preprocesamiento de variables [Tripathy, 1984].

[Saldaña et al,2007] formulo dos modelos de Programación Lineal Entera para un problema de Programación de Horarios para Universidades y presento dos estrategias de solución para cada uno de ellos. donde consiste en programar los eventos, considerando los profesores, días, horarios, salones y las necesidades de asignar los eventos en periodos consecutivos determinados. El objetivo es minimizar la asignación en periodos no deseados, balanceando la carga de trabajo diaria para cada grupo de alumnos.

2.4 HEURÍSTICAS

Los problemas NP-duros son problemas que generalmente requieren un número exponencial de operaciones para resolver con exactitud. Dado que resolverlos exactamente requiere mucho tiempo, por tal motivo generalmente se emplean heurísticas [Cruz-Chávez, 2016].

Dentro de la extensa variedad de problemas (difíciles) que existen dentro de la computación los cuales requieren ser resueltos de forma eficiente y eficaz, se proponen la metodología de enfoques eficientes para encontrar buenas soluciones, aunque estas no sean las óptimas. Estos métodos, tienden a ser rápidos en encontrar buenas soluciones en un lapso determinado de tiempo, todo esto es importante como la calidad de la solución obtenida, estos métodos mejor conocidos en el entorno científico como heurísticos. [Martí & Reinelt, 2011].

Existen muchos métodos heurísticos de naturaleza muy diferente como lo menciona [Marti,2003], por lo que es complicado dar una clasificación completa a estos, además, muchos de ellos han sido diseñados para un problema específico sin posibilidad de generalización o aplicación a otros problemas similares, algunas de las categorías amplias se muestran a continuación, donde se puede ubicar a los heurísticos más conocidos:

- Métodos de Descomposición: El problema original se descompone en subproblemas más sencillos de resolver, teniendo en cuenta, aunque sea de manera general, que ambos pertenecen al mismo problema.
- Métodos Inductivos: La idea de estos métodos es generalizar de versiones pequeñas o más sencillas al caso completo. Propiedades o técnicas identificadas en estos casos más fáciles de analizar pueden ser aplicadas al problema completo.
- Métodos de Reducción: Consiste en identificar propiedades que se cumplen mayoritariamente por las buenas soluciones e introducirlas como restricciones del problema. El objetivo es restringir el espacio de soluciones simplificando el problema. El riesgo es dejar fuera las soluciones´optimas del problema original.
- Métodos Constructivos: Consisten en construir literalmente paso a paso una solución del problema. Usualmente son métodos deterministas y suelen estar basados en la mejor elección en cada iteración. Estos métodos han sido muy utilizados en problemas clásicos como el del viajero.
- Métodos de Búsqueda Local: A diferencia de los métodos anteriores, los procedimientos de búsqueda o mejora local comienzan con una solución del problema y la mejoran progresivamente. El procedimiento realiza en cada paso un movimiento de una solución

a otra con mejor valor. El método fina liza cuando, para una solución, no existe ninguna solución accesible que la mejore.

En 2019 se propone un modelo matemático de satisfacción de restricciones que define el problema real de Programación de Cursos Universitarios en la Facultad de Ciencias Químicas en Ingeniería (FCQeI) se emplea un algoritmo de enfoque constructivo para obtener soluciones factibles del modelo propuesto que permita el tratamiento de instancias de los semestres de la FCQeI. [Cruz-Chavez et al, 2016].

2.4.1 COLORACION DE GRAFOS

[Welsh & Powell,1967] ha representado el primer problema de horarios en equivalente al problema de coloración de grafos, presentó el método que no pudo resolver los problemas, cuando había eventos asignados previamente. [DeWerra, 1985] describió el método de coloración de grafos sobre la modelación de un problema de horarios mediante el uso de un grafo no dirigido. Donde el objetivo es colorear los vértices del grafo usando un número dado de colores donde ningún vértice vecino (nodo) tiene el mismo color. El horario resultante no debe tener ningún conflicto y debe estar coloreado con el mínimo número de colores. En el problema de horarios está basado en la teoría de coloración de grafos, los eventos son como nodos, los bordes como restricciones (preferiblemente restricciones duras) y colores como intervalos de tiempo.

En la siguiente figura 2.1 se muestra un ejemplo del problema de horarios abordado con el método de coloración de grafos, donde los nodos se representan los eventos, los bordes como restricciones y los colores como los intervalos de tiempo. Donde se debe encontrar el menor número de colores con los intervalos de tiempo donde es posible asignar eventos en un periodo por día. Si dos nodos vecinos son del mismo color, entonces ocurre un conflicto de tiempo.

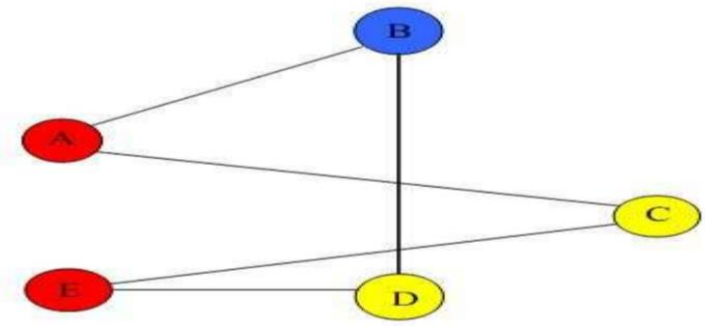

Figura 2.2 problema de programación de horarios representado por el método de coloración de grafos [Dewerra,1985].

Eventos (A, B, C, D, E)

Bordes (Restricciones) {(A, B), (A, C), (B, D), (C, E), (D, E)}

Colores (Intervalos)
A, E: Rojo
B: Azul
C, D: Amarillo

[Mouhamed & Dandashi, 2010] utilizo el método de coloración de gráficos para generar un horario de cursos, donde los nodos y los bordes representan los cursos comunes y estudiantes respectivamente y el objetivo es asignar los cursos al número dado de intervalos de tiempo (color) (cada intervalo de tiempo se puede utilizar para un número determinado de habitaciones). El objetivo de presentar un enfoque heurístico son los siguientes:

- Aumentar la distribución uniforme de cursos en colores
- Equilibrar el número de cursos para cada intervalo de tiempo en los salones actuales

2.4.2 ALGORITMO BASADO EN CLANES

El algoritmo k-Cliques realiza exactamente $n\frac{n-1}{2}$ iteraciones, por lo que el número de operaciones que realiza es $O(n^2)$. Así, el algoritmo k-Cliques revisa todas las adyacencias posibles en el grafo. En la actualidad existen pocos artículos que empleen heurísticas basadas en clique. Para resolver problemas de horarios.

[Carter & Lee,1996] señaló que una camarilla, un subgrafo donde los vértices eran adyacentes entre sí, estos representarían un conjunto de eventos en conflicto en un problema de horarios. Este conjunto de eventos tuvo que ser programado en diferentes intervalos de tiempo, por lo que el número mínimo de intervalos de tiempo utilizados no sería menor que el tamaño de cualquier camarilla en el conflicto. En su algoritmo, primero se determinó una gran camarilla y los exámenes en esta camarilla tenía mayor prioridad para ser programado.

[Liu et al, 2011] propone un algoritmo heurístico para el problema de cursos universitarios (UCTP), donde la asignación de eventos a intervalos de tiempo, el cual está basado en camarillas, donde en cada una de ellas se representa el conjunto de eventos que pueden ser asignados en el mismo intervalo de tiempo, tal algoritmo fue probado con las mismas instancias trabajadas en este trabajo logrando un total de 52 soluciones factibles de las 60 propuestas.

2.4.3 INSERCIÓN DE EVENTOS

El movimiento de kempe [Morgenstern & Shapiro, 1990] es una estructura básica para la construcción de un vecindario, en la inserción de eventos la esencia proviene la de heurística denominada inserción de kempe la cual es una técnica para mover eventos entre intervalos de tiempo en un horario. Esta heurística utiliza una exploración de vecindad computacionalmente más pesada esquema basado en el movimiento de Kempe con el fin de liberar recursos adecuados para el resto eventos y acercar gradualmente el horario a la factibilidad.

[Mühlenthaler & Wanka, 2010] presenta una novedosa estrategia de inserción de eventos para encontrar soluciones factibles para (UCTP), este método se basa en la técnica de inserción de

kempe, la cual se basa en una estructura de vecindario, cuyo objetivo es que en cada movimiento del vecindario este se acerca a una solución factible, el algoritmo consta de dos etapas, la primera una heurística secuencial simple y la segunda, inserción de kempe la cual intenta asignar recursos factibles para los eventos restantes, esta heurística fue probada con las misma instancias de este trabajo, logrando encontrar factibilidad en 53 instancias.

2.5 METAHEURÍSTICAS

El término metaheurística fue introducido por Fred Glover en 1986 desde entonces han aparecido muchas propuestas de procedimientos o guías para diseñar mejores métodos de solución de problemas de optimización combinatoria. Como lo plantea [Glover,2003] una metaheurística es un algoritmo mejor elaborado los cuales representan una gran parte de técnicas utilizadas para resolver la mayoría de los problemas de optimización combinatoria. En la actualidad existen numerosas Metaheurísticas y su concepción proviene de diversas fuentes de inspiración, algunas están hechas por analogía de otros campos científicos como la física (Recocido Simulado), biología (Colonia de Hormigas), neurología (Algoritmos Evolutivos), sociología (Búsqueda Tabú) (Búsqueda Local Iterada), etc.

Los métodos metaheurísticos comienzan con una o más soluciones iniciales y emplean estrategias de búsqueda que intentan salir de óptimos locales. Todos estos algoritmos de búsqueda pueden producir soluciones de alta calidad, pero a menudo tienen un costo computacional considerable [Burke & Petrovic, 2002].

"Los procedimientos Metaheurísticos son una clase de métodos aproximados que están diseñados para resolver problemas difíciles de optimización combinatoria en los que las heurísticas no son efectivas. Las Metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos." [Osman & Kelly, 1997]

2.5.1 BÚSQUEDA LOCAL ITERATIVA

Búsqueda local Iterada (iterated Local Search ILS en inglés) consiste en mezclar búsquedas locales con perturbación o mutación de elementos. En cada iteración la donde la solución actual S se modifica, se perturba utilizando los dos operadores. Primero la solución se perturba con el operador de mutación, en segundo lugar, se mejora mediante un operador de búsqueda local, donde si $f(s')$ es mejor que $f(s), s'$ reemplaza a s y se convierte en la solución actual [Glover, 2006].

Existen muchas variantes de ILS en la literatura, algunas al aceptar peores soluciones con determinadas condiciones, otros llegan a consideran más de un tipo de perturbación y aplicarlas en pasos específicos.

Algunas de las referencias para este trabajo las cuales han abordado las mismas instancias refieren a **[Song et al, 2018]** donde se proponen un algoritmo de búsqueda local iterada cuyo objetivo es encontrar soluciones factibles para el problema de programación de cursos universitario. Este algoritmo propone tres fases clave: inicialización, intensificación y diversificación. Una vez que se construye un horario inicial parcialmente factible, se realiza una búsqueda local basada en la metaheurística de Recocido Simulado a la par de un procedimiento de diversificación que produce una perturbación con la cual puede alcanzar una mejora en la solución actual de manera iterada hasta que se cumpla una condición de paro. este método es el que ha encontrado factibilidad en 58 de 60 instancias, hasta el momento es el que tiene más soluciones factibles.

2.5.2 BÚSQUEDA TABÚ

La búsqueda tabú [Glover, 1989], es un tipo de búsqueda por entornos, que permite moverse a una solución, aunque no sea tan buena como la actual. De este modo se puede escapar de óptimos locales y continuar la búsqueda de soluciones aún mejores. Esencialmente el método trata de usar la información de lo que ha sucedido hasta el momento para actuar en consecuencia. Es decir, la vecindad de una solución estará afectada por la eliminación de aquellas soluciones que ya hayan sido visitadas por el algoritmo en un pasado reciente (lista tabú) para evitar que el proceso cicle o que tengan atributos que mantengan una estructura poco beneficiosa según se desprendan las soluciones visitadas. El principio subyacente en este método es que es preferible tomar una decisión que localmente sea mala (moverse a una solución con peor valor de f) si está fundamentada en la información que puede extraerse de las soluciones ya visitadas. En este sentido puede decirse que hay un cierto aprendizaje durante el proceso y que la búsqueda es inteligente evitando quedar atrapados en óptimos locales.

En este artículo [**Chen et al, 2020**] se aborda el problema para el UCTP, el objetivo es encontrar una solución factible, es decir, es decir una solución que satisfaga todas las restricciones duras. Para generar la solución inicial se utiliza un algoritmo de búsqueda codiciosa, que tiene por estrategia ordenar los eventos por el número de salones candidatos. Mediante un algoritmo de búsqueda tabú con una estrategia de aleatorización controlada, que cuenta con los operadores de movimiento que son permutaciones dentro de los salones (intra-room swap) el cual consiste en intercambiar los horarios asignados de los eventos que están asignados en el salón y el

operador de intercambio de salones (inter-room swap) intercambia dos eventos asignados a salones diferentes. En términos de soluciones factibles, este método encontró un total de 55 soluciones factibles para el benchmark de Lewis.

2.5.3 ALGORITMO GENÉTICO

Algoritmo genético, o en inglés: Genetic Algorithm (GA) [Holland 1975] está inspirado en la biología molecular, GA considera una solución como una estructura cromosómica que contiene fenotipos buenos y malos. Suponiendo que los buenos fenotipos son parte de las soluciones óptimas, GA utiliza el mecanismo de reproducción, la selección natural y el principio de mutación para producir nuevas soluciones en las que el cromosoma contiene mejores fenotipos. En la práctica, en cada iteración, se selecciona un conjunto de cromosomas para su reproducción. Luego, los cromosomas seleccionados se cruzan para producir nuevos cromosomas llamados cromosomas hijos. Los hijos son mutados usando un operador de mutación, con una probabilidad baja. Por último, se calcula su aptitud (calidad de la solución) y se insertan en la población.

La población considerada después del cruce la variante del algoritmo. La primera variante es el algoritmo genético generacional. (GGA) en el que solo los nuevos cromosomas constituyen la población. La segunda variante es el algoritmo genético de estado estable (SSGA) en el que los hijos no se insertan directamente, pero compiten con los cromosomas existentes.

[Pichardo, 2018] implementa un algoritmo genético para el problema de programación Horarios el cual brinda buenas soluciones, el objetivo principal usado en el algoritmo es el operador de mutación ligero posterior a eso se implementa un algoritmo iterativo (búsqueda local) para la mejora de la solución.

[Lewis, 2007A] propuso un algoritmo de agrupación genética que se utiliza exclusivamente para encontrar soluciones factibles del UCTP. El principio de este enfoque es que el objetivo del problema puede ser visto como la tarea de agrupar eventos en número apropiado de intervalos de tiempo, de forma que se satisfagan las restricciones duras, en este caso, los intervalos de tiempo son lo que definen la construcción subyacente de los bloques del problema, este algoritmo hace uso de operadores genéticos especializados que intentan permitir que los grupos se propaguen durante la evolución, este trabajo logró encontrar 38 soluciones factibles de las 60 propuestas. En este trabajo fueron propuestas las 60 instancias difíciles para UCTP.

2.5.4 ALGORTIMO MEMÉTICO

Los algoritmos meméticos representan enfoques basados en la evolución combinados con técnicas de búsqueda. Este método de hibridación (algoritmos evolutivos junto con búsqueda

local) se le han dado varios nombres en la literatura, como genético híbrido algoritmos y algoritmos de búsqueda genética local [Ross, 2003]. La mayoría de Los algoritmos meméticos discutidos en la literatura son el resultado de la incorporación de técnicas de búsqueda con algoritmo genético.

[Qaurooni & Akbarzadeh, 2013] propuso un algoritmo memético (MA) que combina la búsqueda local con algoritmo genético, con esto mejora el proceso de transición simple que sucede en un algoritmo genético al incluir la búsqueda local. La idea es guiar la búsqueda local mediante la capacidad de los algoritmos genéticos para explorar rincones del espacio de soluciones, este algoritmo logró encontrar en factibilidad en 55 de las 60 instancias propuestas por Lewis.

2.5.5 ALGORTIMO DE COLONIA DE HORMIGAS

La optimización de colonias de hormigas o en inglés: Ant Colony Optimization (ACO) es una metaheurística propuesta por Dorigo La inspiración de ACO es el comportamiento de búsqueda de alimento de las hormigas reales. El ingrediente básico de ACO es el uso de un mecanismo de construcción de soluciones probabilísticas basado en estigmergia. ACO se ha aplicado con éxito a numerosos problemas de optimización combinatoria, incluido el problema de asignación cuadrática, problemas de satisfactibilidad, problemas de programación, etc. [Colorni , 1998].

[Nothegger et al, 2012] Presenta un nuevo enfoque para abordar el problema de programación de Horarios, el procedimiento se basa en la optimización de hormigas (ACO), donde las hormigas construyen sucesivamente soluciones basadas en feromonas (estigmergia) e información local.

2.5.6 GRASP

La Metaheuristica Gredy Randomize Adaptive Search Procedure (GRASP), x es una de las más jóvenes técnicas surgidas en los años 80, que se destaca en el campo de optimización combinatoria. Se ha aplicado para establecer problemas de cobertura, árbol de expansión, entre otros. [Resende & Ribeiro, 2014].

En el trabajo de [De Souza et al, 2012] donde implementa un algoritmo GRASP para la generación de horarios utilizando la formulación del ITC-2007. El algoritmo tiene una fase inicial donde se produce una solución aleatoria codiciosa. Las clases son clasificadas en orden de dificultad (más difícil a más fácil) y se seleccionan uno por uno para ser asignadas al horario. La selección aleatoria de eventos en algoritmo de explosión evita conflicto de evento. Para la fase de mejora GRASP aplica una búsqueda local a la solución de programación de horarios

inicial, La iteración GRASP (fase inicial y fase de mejora) se repite varias veces generando distintos horarios, donde la solución final es la mejor de todos los horarios generados.

2.5.7 RECOCIDO SIMULADO

El Recocido Simulado o en inglés Simulated Annealing (SA) es una búsqueda local inspirada en el proceso de recocido en física [Kirkpatrick et al, 1983]. Es ampliamente utilizado para resolver problemas de optimización combinatoria, especialmente para evitar quedar atrapado en óptimos locales, cuando se utiliza la búsqueda local, el más simple método de búsqueda.

Esto se hace de la siguiente manera: siempre se acepta un movimiento de mejora mientras que uno que empeora se acepta de acuerdo con una probabilidad que depende de la cantidad de deterioro en el valor de la función de evaluación, de modo que mientas más "malo" sea el movimiento es menos probable que lo acepte.

[AlHadid et al, 2020] realiza una investigación sobre el funcionamiento de recocido simulado empleado en el problema de programación de horarios de cursos universitarios, dado que recocido simulado se ha utilizado ampliamente para resolver problemas complejos de optimización, debido que es más accesible en su implementación y su capacidad de salir de óptimos locales, la investigación se enfoca en 3 componentes del SA, Temperatura inicial, estructura de enfriamiento y vecindarios.

[Koustuck, 2003] ganador del primer lugar en la competición internacional de horarios en 2003, presentó un enfoque heurístico para la solución de estos problemas, el cual se divide en dos etapas, al inicio construye un horario factible sin tener en cuenta la función objetivo en la cual emplea coloreo de grafos y coincidencia máxima en la segunda etapa se emplea recocido simulado para mejorar la solución utilizando un vecindario de pares.

Algunas de diversas opciones surgen al aplicar metaheurísticas para resolver el problema de programación de horarios de cursos universitarios, surgiendo la necesidad de una metodología que puede orientar para la toma de decisiones. Como parte de la investigación de la red de metaheurística, se abordaron cinco metaheurísticas y compararon los resultados, se incluyeron versiones básicas como búsqueda tabú, recocido simulado, búsqueda local iterada, algoritmos evolutivos y colonia de hormigas. Todas estas heurísticas utilizaron el mismo procedimiento de búsqueda local descrita por [Rossi-Doria et al, 2002], las instancias para la comparación de estas metaheurísticas fueron creadas por el mismo generador aleatorio, pero diferentes a la competencia, las cuales estaban agrupadas en tres clases (pequeñas, medianas y grandes. El patrón principal encontrado fue de búsqueda tabú y búsqueda local iterada, donde búsqueda local iterada se comportó mejor que otras metaheurísticas con respecto a la capacidad de

alcanzar la viabilidad, pero solo cuando se encontró un horario factible, recocido simulado pareció ser la mejor opción para minimizar las infracciones de las restricciones suaves, todos los detalles están descritos en los trabajos de [Rossi-Doria et al,2003] y [Chiarandini et al, 2006].

[Tuga et al, 2007] aborda el problema de cursos universitarios con una reformulación relajando una de las restricciones duras y luego crear una restricción blanda para abordar la restricción relajada, tal problema lo resuelve en dos pasos, primero una heurística basada en grafos se utiliza para construir un horario factible del problema relajado, segundo se compone de un enfoque basado en recocido simulado el cual se emplea para minimizar la violación de la restricción blanda, esta heurística fue probada con las mismas instancias de este trabajo, logrando encontrar 50 de ellas.

[Frausto-Solis et al, 2008] propuso un algoritmo de recocido simulado para el problema de UCTP, donde la solución inicial se creó agregando intervalos de tiempo adicionales a los horarios para crear una solución factible, las instancias trabajadas tenían 400 eventos, hasta 20 salones, máximo 20 características y 1000 estudiantes. En este trabajo se encontraron 54 del benchmark de 60 instancias.

[Lewis et al, 2007B] aplicó dos algoritmos el primero de agrupamientos evolutivos y el segundo de búsqueda local. Ambos algoritmos fueron mejorados por búsqueda local, posteriormente se mejoraron en el contexto de un algoritmo de recocido simulado, el algoritmo de recocido simulado obtuvo un total de 43 soluciones factibles de las 60 propuestas por Lewis.

[Ceschia et al, 2012] aplicó un enfoque metaheurístico basado en el recocido simulado para resolver soluciones Factibles para el problema de UCTP, se describe la técnica de búsqueda local en seis etapas las cuales son preprocesamiento, espacio de búsqueda, solución inicial, relaciones de vecindad, función de costos y metaheurística de recocido simulado este enfoque resolvió un total de 53 instancias de las 60.

[Song et al, 2021] propone un algoritmo novedoso de búsqueda local en múltiples vecindarios, En comparación con los métodos metaheurísticos clásicos de la literatura, el algoritmo propuesto incluye tres objetivos principales, En primer lugar, se presenta una nueva forma de combinar múltiples vecindarios, En segundo lugar, se proponen dos reglas heurísticas para determinar las probabilidades de seleccionar la vecindad, en tercer lugar, se propone una estrategia de reinicio.

2.6 MODELO MATEMÁTICO

La modelación matemática para obtener horarios factibles se formula de la siguiente forma con los elementos descritos a continuación.

Tabla 2.2 Simbología de conjuntos de Programación de Horarios

Símbolo	Descripción
E	Conjunto de Eventos
P	Conjunto de Periodos
R	Conjunto de Salones
U	Conjunto de Estudiantes
F	Conjunto de Características
c_i	Capacidad del Salón i
n	Número de eventos
m	Número de salones
s	Número de estudiantes
t	Número de características

Se tiene un conjunto de eventos $E = \{e_1, e_2, \dots, e_n\}$ que se programarán en 45 intervalos de tiempo $P = \{p_1, p_2, \dots, p_{45}\}$ (5 días de 9 horas cada uno). Se tiene un conjunto de salones $R = \{r_1, r_2, \dots, r_m\}$ en las que se pueden llevar a cabo los eventos. Se tiene un conjunto de estudiantes $U = \{u_1, u_2, \dots, u_s\}$ que asisten a los eventos. Existe un conjunto de características $F = \{f_1, f_2, \dots, f_t\}$ que poseen los salones y es requerido por los eventos. Cada estudiante asiste a una serie de eventos y cada salón tiene cierta capacidad representada en el conjunto $\mathrm{C} = \backslash\backslash\{c_1, c_2, \dots, c_m\backslash\backslash\}$.

De igual manera se cuenta con 3 matrices binarias, las cuales se muestran a continuación.

- Matriz de evento-estudiante, a los que asiste cada estudiante: D_{nxm}

$$D_{nxm} = \begin{matrix} d_{11} & d_{12} \dots & d_{1n} \\ d_{21} & d_{22} \dots & d_{2n} \\ d_{m1} & d_{m2} \dots & d_{mn} \end{matrix}$$

Donde $d_{il} = 1$ si el estudiante l asiste al evento$i$, y 0 en caso contrario.

- Matriz *s*, describiendo las características que poseen los salones:s_{txr}.

$$s_{txr} = \begin{matrix} s_{11} & s_{12} \dots & s_{1t} \\ s_{21} & s_{22} \dots & s_{2t} \\ s_{r1} & s_{r2} \dots & s_{rt} \end{matrix}$$

Donde $s_{jf} = 1$ si el salón j posee la característica $f, y0$ en caso contrario.

- Matriz Q, describiendo las características que requieren los eventos:Q_{txn}

$$Q_{txn} = \begin{matrix} q_{11} & q_{12} \cdots & q_{1t} \\ q_{21} & q_{22} \cdots & q_{2t} \\ q_{n1} & q_{n2} \cdots & q_{nt} \end{matrix}$$

Donde $q_{if} = 1$si el evento i requiere la característica $f, y0$en caso contrario.

Para este modelo se define la variable X_{ijk} cuyo valor será 1 cuando se asigne el evento i al salón j en el periodo k, en caso contrario será 0. Por ejemplo, cuando la variable $x_{418} = 1$, el evento 4 se asignará al salón 1, en el período 8 [Alonso, 2008].

2.6.1 RESTRICCIONES DURAS

Para lograr un horario factible para el problema de cursos universitarios es necesario satisfacer todas las restricciones duras que se establezcan, a continuación, se presenta
las restricciones duras:

- **Ningún estudiante asiste a más de un evento al mismo tiempo**.

Esta restricción indica que dentro de cualquier periodo k, puede programarse a lo más un evento $i \in Q_i$, donde $Q_i$es el conjunto de eventos que se encuentra en conflicto con el evento i, eventos en conflicto se entiende por qué tienen a lo más un estudiante en común.

- **El salón es lo suficientemente grande para albergar a todos los estudiantes que asisten y satisface todas las características requeridas por el evento.**

Esta restricción puede dividirse en dos partes:

- Primera: el salón es lo suficientemente grande para atender a todos los estudiantes que asisten al evento.
- Segunda: el salón debe satisfacer todas las características requeridas por el evento.

La primera parte de la restricción indica que en cualquier salón j , donde se puede programar un evento i, la capacidad c_j del salón j, debe ser mayor o igual al número total de estudiantes que asisten al evento i, para modelar esta restricción la variable definimos B_i como el número total de alumnos inscritos para el evento i.

La segunda parte de esta restricción indica que para cada uno de los periodos k, el salón j debe cumplir todos los requisitos para cualquier evento i, programado en el salón,

- **Solo hay un evento en cada salón en cada intervalo de tiempo**.

Lo que indica la restricción es que en cada periodo$k$, dentro de un salón j, a lo más se puede tener programado un evento i.

Existe una restricción implícita la cual es indispensable cumplir para el problema la cual nos dice:

- ***Todos los eventos deben estar programados en algún período.***

lo que significa que tomando en cuenta los 45 periodos, todos los eventos i deben estar programados exactamente solo una vez.

Una nota en cuanto al espacio de soluciones: Un horario donde se asignan todos los eventos y satisface todas las restricciones duras considera factible. El número total de posibles asignaciones (horarios) se establece con la siguiente formula: $(t * r)^e$, donde t = el número de intervalos de tiempo, r = el número de salones, y e = el número de eventos). En todo, menos en casos triviales, la mayoría de estas asignaciones probablemente contendrán algún nivel de inviabilidad [Lewis, 2007B].

A continuación, se presenta el modelo matemático, para obtener un horario factible, basado del modelo de [Alonso, 2008]:

$$MinimizarZ = |B| * 1000 + \sum_{i \in B} s_\alpha \qquad \textbf{(2.1)}$$

$$Sujetoa$$

$$Hcv = 0$$

$$\sum_{i \in Q_i} x_{ijk} \leq 1 j = 1, \ldots . rk = 1, \ldots . 45 i = 1, \ldots . n \qquad \textbf{(2.2)}$$

$$b_i = \sum_{l=1}^{m} d_{li} i = 1, \ldots \ldots . n \qquad \textbf{(2.3)}$$

$$\forall X_{ijk} = 1, b_i \leq c_j, i = 1 \ldots\ldots, n; j = 1 \ldots\ldots, r; k = 1..,45 \quad \textbf{(2.4)}$$

$$X_{ijk} = 1 \rightarrow q_{if} \leq s_{if}, \forall f \in Fi = 1 \ldots\ldots, n; j = 1 \ldots\ldots, r; k = 1..,45 \quad \textbf{(2.5)}$$

$$\sum_{i=1}^{n} x_{ijk} \leq 1 j = 1, \ldots\ldots\ldots r; k = 1, \ldots\ldots 45 \quad \textbf{(2.6)}$$

$$\sum_{k=1}^{45} \sum_{j=1}^{r} x_{ijk} = 1; i = 1, \ldots\ldots, n \quad \textbf{(2.7)}$$

En la ecuación 2.1 se presenta la función objetivo del modelo, donde |B| es la cardinalidad del conjunto de eventos que se programarán en periodos virtuales, se considera un periodo virtual cuando no se encuentra en el rango de (1 a 45), multiplicado por la constante de 1000 para ir decrementado el valor en la función objetivo, más la sumatoria del total de estudiantes donde i pertenece a B donde se encuentran los eventos que están asignado en periodos virtuales donde s corresponde a los estudiantes, puede ser que existan dos eventos los cuales tengan en total 25 estudiantes y en otra iteración 1 sole evento con un total de 45 estudiantes, para ello es importante la constante la cual puede ser 100,1000, 10000, todo a ello para que la función objetivo pueda ir decrementando en base a los eventos virtuales.

La restricción de que ningún estudiante asiste a más de un evento al mismo tiempo se expresa en la ecuación 2.2, donde el índice de i pertenece a los eventos, j a los salones y k a los periodos. La restricción el salón es lo suficientemente grande para atender a todos los estudiantes que asisten al evento se expresa en las ecuaciones 2.3 y 2.4 donde la variable i pertenece al evento, y la variable l a los estudiantes, d_{li} = 1 si el estudiante l asiste al evento i y 0 en caso contrario. La restricción el salón debe satisfacer todas las características requeridas por el evento se expresa en la ecuación 2.5, donde q_{if} representa la característica f asociada al evento i, y s_{jf} representa la característica f satisfecha por el salón j. La restricción solo hay un evento en cada salón en cada intervalo de tiempo se expresa en la ecuación 2.6. La restricción implícita: todos los eventos deben estar programados en algún período se puede ver en la ecuación 2.7

2.6.2 BENCHMARK

En el 2005, en el trabajo de Lewis [Lewis, 2005A] fueron creadas 60 instancias en las cuales es difícil encontrar una solución factible, con el mismo formato del PATAT usando un generador de instancias, el Benchmark se divide en tres clases, small, medium y big. Se garantiza que todas estas instancias contienen al menos un horario factible, las características de dichas instancias se muestran en las siguientes tablas.

Tabla 2.3 Instancias small del Benchmark [Lewis, 2005A]

Instancia	Eventos	Salones	Características	Estudiantes
small_1	200	5	5	200
small_2	210	6	5	400
small_3	200	6	5	400
small_4	200	5	8	500
small_5	200	5	8	500
small_6	200	5	3	1000
small_7	200	5	3	800
small_8	225	5	10	1000
small_9	225	5	10	900
small_10	220	5	10	1000
small_11	200	5	4	1000
small_12	225	5	10	1000
small_13	225	5	10	1000
small_14	225	5	3	1000
small_15	200	5	3	900
small_16	200	5	3	900
small_17	200	5	3	900
small_18	225	5	3	1000
small_19	225	5	3	1000
small_20	225	5	3	1000

Tabla 2.4 Instancias médium del Benchmark [Lewis, 2005A]

Instancia	Eventos	Salones	Características	Estudiantes
med_1	400	10	10	400
med_2	390	10	10	400
med_3	390	10	10	400
med_4	410	10	9	400
med_5	410	10	9	450
med_6	410	11	10	450
med_7	410	11	10	450
med_8	400	10	10	400
med_9	400	10	10	400
med_10	400	10	8	500
med_11	400	10	8	800
med_12	400	10	8	800
med_13	400	10	8	800
med_14	400	10	8	1000
med_15	425	10	8	500
med_16	400	10	8	1000
med_17	400	10	8	800
med_18	400	10	8	1000
med_19	410	10	8	1000
med_20	410	10	8	1000

Tabla 2.5 Instancias Big del Benchmark [Lewis, 2005A]

Instancia	Eventos	Salones	Características	Estudiantes
big_1	1000	28	20	1000
big_2	1000	25	20	1000
big_3	1000	25	20	900
big_4	1050	25	20	800
big_5	1075	25	20	1000
big_6	1075	25	20	1000
big_7	1050	25	20	1100
big_8	1025	25	20	1000
big_9	1050	25	20	800
big_10	1075	25	20	1000
big_11	1075	25	20	1000
big_12	1000	26	25	1000
big_13	1000	25	25	1000
big_14	1000	25	25	1000
big_15	1000	25	25	1000
big_16	1000	25	10	1000
big_17	1000	25	10	1200
big_18	1000	25	10	1000
big_19	1000	25	10	1000
big_20	1000	25	10	1000

CAPÍTULO III METODOLOGÍA

En este capítulo se muestra la estrategia de cómo se abordó para el problema de programación de horarios factibles universitarios. Se utilizaron tres algoritmos, el primero es una heurística para encontrar el "evento más restringido", la segunda aceptación por umbral, y tercero se aplicó un Recocido Simulado. En la figura 3.1 describe la metodología, donde en primer lugar la solución inicial genera un horario completo, pero no necesariamente factible, dado que puede llegar a tener periodos virtuales incrementando el número de periodos. La solución inicial pasa al segundo algoritmo: Aceptación por umbral, el cual mejora la solución, el algoritmo termina si es posible encontrar un horario factible o en caso contrario regresa la mejor solución encontrada. Por último, Recocido Simulado trabaja con la mejor solución encontrada por aceptación por umbral, esta metaheurística se itera 10 veces, donde de igual manera retorna un horario factible o caso contrario la mejor solución encontrada.

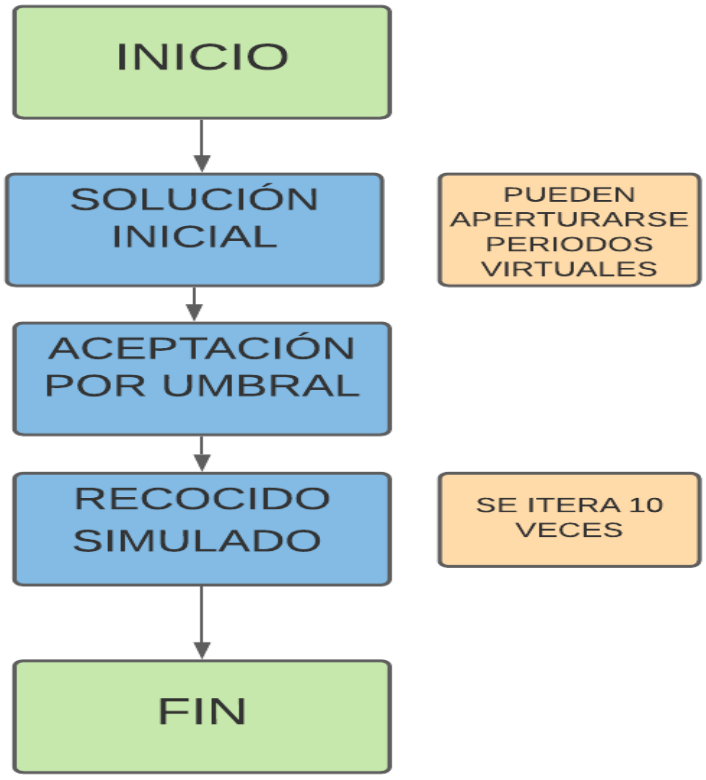

Figura 3.1 Proceso de desarrollo

3.1 SOLUCIÓN INICIAL.

En la figura 3.2 describe el procedimiento que se emplea para general una solución inicial.

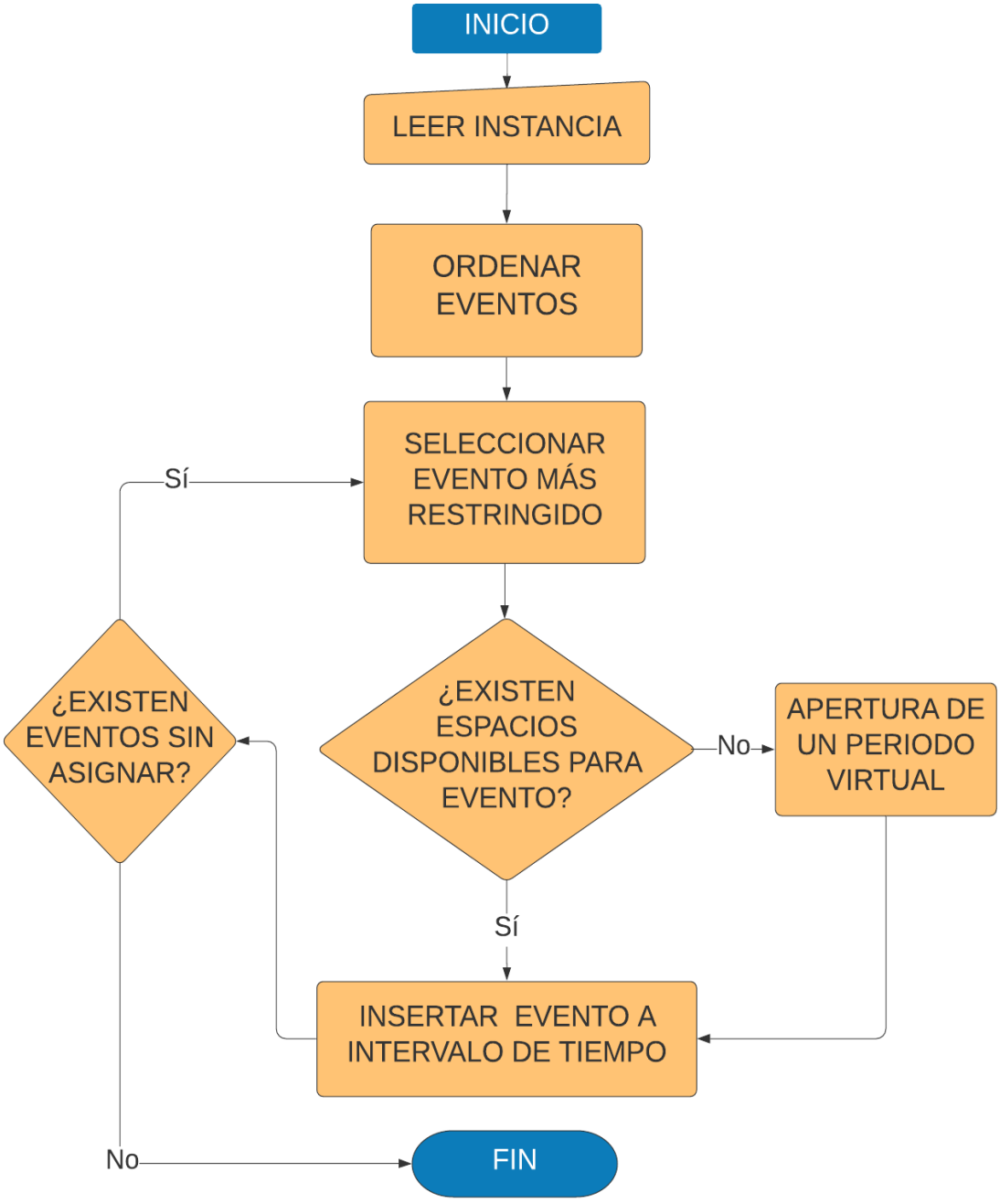

Figura 3.2. Diagrama de solución inicial

Para obtener la solución inicial, se implementa una heurística para encontrar el "evento más restringido", donde el evento más restringido consiste en el evento que tenga menos combinaciones periodo-salón para ser asignado. Por ejemplo, si el evento 1 tiene 3 combinaciones factibles para ser asignado y el evento 2 una combinación factible, se le considera al evento 2 como el evento más restringido. Para que una combinación del evento e_1 periodo-salón se considere factible, no solamente debe de estar disponible el lugar, sino que no debe existir algún evento e_2 programado en el mismo periodo con el cual e_1 tenga algún conflicto.

El primer paso para obtener una solución inicial es realizar un ordenamiento de eventos. Los eventos se ordenan de forma creciente de acuerdo con el número de las posibles combinaciones periodo-salón factibles para ser asignado. Por ejemplo, si un evento tiene 2 salones, el número total de combinaciones seria 90, pero no quiere decir que los 90 sean factibles, se considera una combinación factible, cuando no hay ningún evento asignado a ese periodo-salón.

En la tabla 3.1 se muestra un ejemplo de las posibles combinaciones de los eventos ordenados.

Tabla 3. 1 Ordenamiento de eventos

Evento	Combinaciones
46	45
76	45
79	45
103	45
12	45
4	90
43	135

Después de haber ordenado los eventos, se escoge el evento con menos combinaciones disponibles. Si existe más de un evento con el mínimo número de combinaciones, se escoge un evento aleatorio y se asigna a una combinación periodo-salón, después se actualiza la lista eventos y nuevamente se vuelve a ordenar, si un evento ya no cuenta con posibles combinaciones, se apertura un periodo virtual y se asigna el evento, en la tabla 3.2, se muestra el evento 43 el cual se ha quedado sin espacio disponible para poder ser asignado.

Tabla 3. 2 Actualización de eventos

Evento	Combinaciones
43	0
91	10
203	10
11	17
7	18
23	20
11	20

Cada evento que se quede sin espacio disponible será asignado en un periodo virtual, en la tabla 3.3 se muestra un ejemplo de las asignaciones en un periodo virtual, donde el evento 43 se encuentra en el salón 3 o cualquier salón que tenga factibilidad y periodo 46.

Tabla 3. 3 Horario con periodos virtuales

Evento	Salón	Periodo
43	3	46
91	2	3
203	0	22
11	0	12
7	2	44
23	1	45
11	4	32

Al finalizar la solución inicial se tendrá un horario donde todos los eventos son asignados, pero es posible que no sea factible dado que pueden existir periodos virtuales. Cuando se finaliza la solución inicial se calcula la función objetivo, la cual se describió anteriormente en el modelo matemático. La tabla 3.4 se muestra un ejemplo de un horario generado por la solución inicial y a continuación se describe como se calcula la función objetivo.

El evento 43 está asignado al periodo virtual 46 el cual tiene 14 estudiantes asignados. El evento 89 está asignado al periodo virtual 46 el cual tiene 27 estudiantes asignados. El evento 3 está asignado al periodo virtual 46 el cual tiene 15 estudiantes asignados.

El evento 23 este asignado al periodo virtual 47 el cual tiene 24 estudiantes asignados. Por lo tanto, se toma en cuenta en la función objetivo, en la tabla se muestran dos periodos virtuales 46 y 47 en los que se encuentran asignados un total de 4 eventos, al sustituir los valores en nuestra función objetivo quedaría lo siguiente:

4 eventos * 1000 + 80 (suma total de estudiantes de eventos en periodos virtuales)

El valor de la función será de 4080, el cual se tendrá que minimizar.

Tabla 3. 4 Horario de solución inicial

Salones/periodos	1	…..	45	46	47
1	103	…	34	43 (14)	23 (24)
2	198	…	78	1	-1
3	172	…	2	89 (27)	-1
4	176	…	-1	-1	-1
5	-1	…	56	3 (15)	-1

La solución inicial, cuando no es factible se pasará al siguiente algoritmo para tratar de minimizar los eventos en los periodos virtuales para obtener un horario factible.

3.2 VECINDARIOS

Para los algoritmos de Aceptación por Umbral y Recocido Simulado se implementaron dos vecindarios. Ambos algoritmos tienen como criterio de paro el haber encontrado un horario factible o en caso contrario haber terminado de ejecutarse sin haber logrado encontrar un horario factible, en cuyo caso retornarán el horario "menos infactible" que hayan logrado obtener.

3.2.1 VECINDARIO MOVER EVENTO

El primer vecindario consiste en escoger aleatoriamente un evento, si el evento se encuentra dentro del rango [1, 45] de los periodos, se mueve solo dentro de ese rango, si el evento se encuentra en alguno de los periodos virtuales, el evento puede moverse en cualquiera de los periodos, al moverse a un periodo del rango [1-45] se valida si puede eliminarse un periodo para poder ir disminuyendo la función objetivo, antes de realizar cualquier movimiento se debe verificar que no viole ninguna restricción dura.

Como se muestra en la tabla 3.5 un horario de solución inicial y en la tabla 3.6 un horario implementado el vecindario mover evento, donde el evento 89 que se encuentra en periodo virtual, este evento logra ser asignado en un periodo de rango (1-45) en cierta interacción, con este movimiento cambia de valor la función objetivo, la cual había iniciado con un valor de 4080 que anteriormente se describió en la generación de la solución inicial y cuyo valor finalizo en 3053 al ejecutar el vecindario mover evento.

Tabla 3. 5 Horario Antes de Vecindario Mover Evento

Salones/periodos	1	2	…..	45	46	47
1	103	72	…	34	43	23
2	198	-1	…	78	-1	-1
3	172	122	…	2	89	-1
4	176	62	…	-1	-1	-1
5	-1	5	…	56	3	-1

Tabla 3. 6 Horario Después de Vecindario Mover Evento

Salones/periodos	1	2	…..	45	46	47
1	103	-1	…	34	43	23
2	198	89	…	78	-1	-1
3	72	122	…	-1	-1	-1
4	56	172	…	2	-1	-1
5	176	5	…	42	3	-1

3.2.2 VECINDARIO INTERCAMBIO DE EVENTO.

Este vecindario consiste en seleccionar dos eventos aleatoriamente e intercambiarlos siempre y cuando no viole las restricciones duras. Siguiendo el ejemplo anterior en la tabla 3.7 se muestra el evento 23 y 176 intercambiándolos y así generando una nueva solución.

Tabla 3. 7 Vecindario intercambio de Evento

Salones/periodos	1	2	…..	45	46	47
1	103	-1	…	34	43	176
2	198	89	…	78	-1	-1
3	72	122	…	-1	-1	-1
4	56	172	…	2	-1	-1
5	23	5	…	42	3	-1

3.3 ACEPTACIÓN POR UMBRAL.

Después de haber terminado la heurística para el evento más restringido, se implementa el algoritmo de Aceptación por Umbral, el cual trabajará con la solución generada por esta heurística siempre que esta no sea factible. A continuación, se muestra el pseudocódigo de aceptación por umbral para el problema de programación de horarios universitarios.

Pseudocódigo Aceptación por Umbral

```
1.  Inicio
2.  x = solución inicial
3.  num_p = obtener_periodos_virtuales(x);
4.  mejor_costo = z= f(x)
5.  U = Uini;
6.  Uf =Ufin;
7.  L = LMAX;
8.  α = ALPHA;
9.  Iter = 0; x* =x;
10. Mientras (Uini > Ufin && num_p>0)
11.         Mientras (Iter < LMAX)
12.                 T_per=seleccionar_perturbacion ();
13.                 x_new= perturba (x, T_per);
14.                 z_new = f(x_nueva);
15.                 t =obtener_periodos_virtuales(x_nueva);
16.                 Si (z_new <U) or (t < num_p) Entonces
17.                         num_p = t;
18.                          z = z-new;
19.                          x = x_nueva;
20.                     Fin_Si
21.                     Si (z* > z) Entonces
22.                             x* = x;
23.                             z* = z;
24.                     Fin_Si
25.                     Iter = Iter + 1
26.                 Fin_Mientras
27.                 Uini = Uini * ALFHA;
28.                 Iter=0;
29.          Fin_Mientras
30.          Imprimir (x*)
31. Final
```

Línea 2 se obtiene una solución generada por la heurística del evento más restringido.
Línea 3 se calcula el total de periodos virtuales generada por la heurística del evento más restringido.
Línea 4 se computa el valor de la función objetivo.
Línea 5 a la 9 se inicializan los parámetros de Aceptación Por Umbral.
Línea 10 se muestran los dos criterios de paro para este algoritmo, el primero que el umbral inicial sea mayor al umbral final o que el total de periodos virtuales sea cero.
Línea 11 se encuentra el ciclo interno, el cual itera N veces, tratando de encontrar un horario factible.
Línea 12 se selecciona probabilísticamente uno de los vecindarios anteriormente mencionados.
Línea 13 se genera una nueva solución.
Línea 14 se calcula el nuevo costo de la solución.
Línea 15 se actualiza el valor de los periodos virtuales.
Línea 16 a 20 se evalúa si el nuevo costo es menor que el umbral o si algún periodo virtual ha disminuido, si alguno de estos casos se cumple, se actualiza el número de periodos virtuales, se actualizan el costo y la solución.
Línea 21 a 24, se evalúa si la solución global es peor que la actual, si la solución actual es mejor entonces se actualiza la solución global.
Línea 26 termina el ciclo interno.
Línea 27 se actualiza el valor de U al multiplicarse por el parámetro de APLHA.
Línea 29 termina el primer ciclo.
Línea 30 Imprime la mejor solución encontrada por el algoritmo.

3.4 RECOCIDO SIMULADO

Al término de la metaheurística de Aceptación por Umbral se evalúa si la solución generada es factible, en caso de que no haber encontrado un horario factible, la mejor solución generada pasa al siguiente algoritmo de Recocido Simulado, el cual realiza 10 iteraciones, a continuación, el pseudocódigo de Recocido Simulado.

Pseudocódigo Recocido Simulado

```
1.  Inicio
2.  x = solución inicial
3.  num_p = obtener_periodos_virtuales(x);
4.  mejor_costo = z = f(x)
5.  T = T_ini;
6.  T_F = T_fin;
7.  L = LMAX;
8.  α = ALPHA;
9.  Iter = 0;
10.     Mientras (T_ini > T_fin; && num_p>0)
11.                 Mientras (Iter < LMAX)
12.                     T_per=seleccionar_perturbacion ();
13.                     x_nueva= perturba (x, T_per);
14.                     costo_nuevo = f(x_nueva);
15.                     t =obtener_periodos_virtuales(x_nueva);
16.                     U=uniforme (0,1);
17.                     D=(costo_inicial-mejor_costo);
18.                     Si ((costo_nuevo < costo_actual) or (U < exp (D/T)) Entonces
19.                             num_p = t;
20.                             costo_inicial = costo_nuevo;
21.                             x = x_nueva;
22.                     Fin_Si
23.                     Si (costo_actual< mejor_costo) Entonces
24.                                 x* = x;
25.                                 mejor_costo = costo_inicial;
26.                     Fin_ Si
27.                     Iter = Iter + 1
28.                 Fin_Mientras
29.                 T_ini; = T_ini; * ALFHA;
30.                 Iter=0;
31.     Fin_Mientras
32.     Imprimir (x*)
33. Final
```

Línea 2 se obtiene una solución generada por la Metaheurística Aceptación por Umbral.
Línea 3 se calcula el total de periodos virtuales generada por Metaheurística Aceptación por Umbral.
Línea 4 se computa el valor de la función objetivo.
Línea 5 a la 9 se inicializan los parámetros de Recocido Simulado.
Línea 10 se muestran los dos criterios de paro para este algoritmo, el primero que la temperatura inicial sea mayor a la temperatura final o que el total de periodos virtuales sea cero.
Línea 11 se encuentra el ciclo de metrópolis, el cual itera LMAX veces, tratando de encontrar un horario factible.
Línea 12 se selecciona con cierta probabilidad uno de los vecindarios anteriormente mencionados.
Línea 13 se genera una nueva solución.
Línea 14 se calcula el valor del costo nuevo de la solución.
Línea 15 se actualiza el valor de los periodos virtuales.
Línea 16 se encuentra la variable uniforme de 0 y1 que se utiliza para la distribución de Boltzman.
Línea 17 se evalúa la diferencia entre la mejor solución y la solución inicial.
Línea 18 a 22 se evalúa si el costo nuevo es menor que el costo actual o en dado caso la distribución de Boltzman que permite al principio escapar de óptimos locales hacia una solución peor, pero entre la tempera decremento esta se vuelve más estricta al aceptar soluciones peores, si alguno de estos casos se cumple, se actualiza el número de periodos virtuales, se genera un mejor costo y se actualiza la solución.
Línea 23 a 26, se evalúa si la solución global es peor que la actual, si la solución actual es mejor entonces se actualiza la solución global.
Línea 28 termina el ciclo de metrópolis.
Línea 29 actualiza el valor T_{ini} al multiplicarse por el parámetro de APLHA.
Línea 31 termina el primer ciclo.
Línea 32 imprime la mejor solución encontrada por el algoritmo.

CAPÍTULO IV SINTONIZACIÓN Y RESULTADOS

En este capítulo de describen la sintonización que se usó para los vecindarios y los algoritmos de Aceptación por Umbral y Recocido Simulado, también se muestran los resultados obtenidos en este proyecto.

4.1 SINTONIZACIÓN DE VECINDARIOS.

La sintonización de los vecindarios (y toda la sintonización en general) se realizó con las instancias small_8, small_14, med_18, med_20, big_17 y big_20, debido a que son de las instancias más difíciles para obtener una solución factible de cada categoría. Para sintonizar la utilización de los vecindarios se ejecutaron 30 veces cada una de las pruebas con una Búsqueda Local (Hill Climbing), se realizó seis configuraciones de sintonización con diferente probabilidad de escoger un tipo de vecindario. La primera combinación se probó utilizar el vecindario mover evento solamente. La segunda combinación se da una probabilidad de 90% a mover evento de ser elegido y un 10% al vecindario de intercambio de evento. La tercera combinación con 80% al vecindario mover evento y 20% al vecindario de intercambio evento. La cuarta combinación tiene una probabilidad de 70% y 30% respectivamente. La quinta configuración utiliza una probabilidad de 60% y 40%. Por último, se establece un 50% para cada vecindario. Los resultados obtenidos se muestran en la tabla 4.1 donde MV es el vecindario mover evento y Swap el intercambio de eventos, se aprecia que los mejores resultados se obtuvieron con la probabilidad de 90% mover evento y 10% intercambio de evento.

Tabla 4.1 Sintonización de vecindarios con Hill Climbing. Se sombrearon los mejores resultados

	SMALL_8	SMALL_14	MED_18	MED_20	BIG_17	BIG_20
MV 100%	5326.321	8522.5	48441.61	24025.43	217933.8	207234.3
MV 90% – SWAP 10%	4237.964	8471.03	41071.71	21156.61	218233.7	201833.3
MV 80% – SWAP 20%	4317.7	8787	46725.03	17125.81	22233.63	219873.1
MV 70% – SWAP 30%	4498.679	9251.92	42154.5	23783.19	227164.2	208460.6
MV 60% – SWAP 40%	4567.83	8823.93	43839.93	24232.93	2354164	209923.3
MV 50% – SWAP 50%	4692.821	8539.96	41332.79	26921.36	225848	208719.2

Tabla 4. 2 Tiempo (S) promedio en segundos de la sintonización de vecindarios

	SMALL_8	SMALL_14	MED_18	MED_20	BIG_17	BIG_20
MV 100%	141.9539	56.28714	61.41464	92.98964	400.2	1293.3
MV 90% – SWAP 10%	293.0393	125.83	294.2354	107.7296	406.4682	1294.2
MV 80% – SWAP 20%	292.568	140.4	246.765	163.5059	567.93	1302.32
MV 70% – SWAP 30%	403.6182	375.22	320.4307	360.2858	732.6366	1375.349
MV 60% – SWAP 40%	560.13	400.52	420.73	440.83	982.02	1923.43
MV 50% – SWAP 50%	846.2057	454.54	494.1114	561.9139	1396.906	2141.592

4.2 SINTONIZACIÓN ACEPTACIÓN POR UMBRAL.

Se presenta la sintonización de los parámetros del Algoritmo de Aceptación por Umbral, U_{ini} , U_{fin} , LMAX, ALPHA y, para Recocido Simulado, T_{ini} , T_{fin} , LMAX, ALPHA para la sintonización se utilizaron las mismas instancias small_8, small_14, med_18, med_20: big_17 y big_20, se ejecutaron 30 veces obteniendo el porcentaje de aceptación y promedio de la calidad- tiempo de la función objetivo.

La primera sintonización se empleó con los parámetros U_{ini} y U_{fin} , en la tabla 4.3 y 4.4, se muestra el porcentaje de aceptación de las instancias small_8 y big_20, en la parte superior se encuentra en número de iteraciones las cuales se emplean para la sintonización de Umbral Inicial, la primera columna se encuentran los valores de Umbral Final, se realizó las configuraciones de estos dos valores y obtuvieron los porcentajes de aceptación inicial y final. A partir de una temperatura inicial de 1500, todas las instancias dan 100% de aceptación. En cuanto a la temperatura final, a partir de 0.01 el porcentaje de aceptación es similar. Los valores que se escogieron fueron de 2000 en U_{ini} y de 0.01 para U_{fin}, debido a que se considera que con 0.01 es suficiente para parar el algoritmo y que en los primeros ciclos se requiere un umbral alto para aceptar prácticamente cualquier solución.

Tabla 4.3 Porcentaje de aceptación inicial (A_ini) y aceptación final (A_fin) en la instancia small_8

U_{ini} \ U_{fin}	1000		1500		2000		2500		3000	
	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin
0.1	39%	30%	100%	25,62%	100%	30,68%	100%	26,03%	100%	24,52%
0.01	35,35%	19,15%	100%	25,19%	100%	20,97%	100%	25,69%	100%	26,30%
0.001	25,36%	19,56%	100%	30,81%	100%	21,94%	100%	28,26%	100%	23,90%
0.0001	36,46%	21,38%	100%	27,5%	100%	23%	100%	28%	100%	16%

Tabla 4.4 Porcentaje de aceptación inicial (A_ini) y aceptación final (A_fin) en la instancia big_20

U_{ini} \ U_{fin}	1000		1500		2000		2500		3000	
	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin
0.1	57%	27,2%	100%	23,98%	100%	24,8%	100%	25%	100%	27,0%
0.01	56,35%	26,62%	100%	23,88%	100%	23,62%	100%	23,71%	100%	24,12%
0.001	55,94%	30%	100%	24,57%	100%	23,70%	100%	24%	100%	24,50%
0.0001	50%	30%	100%	26,60%	100%	23,80%	100%	23,7%	100%	25,50%

El parámetro LMAX indica el número de iteraciones internas del algoritmo de aceptación Umbral. La sintonización de LMAX se muestran en la tabla 4.5, donde se muestra que 3000 es el mejor resultado en calidad de la función objetivo.

Tabla 4. 5 Sintonización de LMAX Aceptación por Umbral

INSTANCIA	LMAX							
	1000	Tiempo(s)	2000	Tiempo(s)	3000	Tiempo(s)	4000	Tiempo(s)
SMALL_8	5588.40	144.31	5522.00	280.41	4790.60	427.76	5641.55	636.93
SMALL_14	11396.93	23.52	9365.27	44.85	7838.53	67.72	8051.47	92.72
MED_18	57029.64	51.46	96574.36	100.11	43826.60	150.64	46548.67	195.30
MED_20	31803.47	31.05	25169.00	63.04	17624.47	88.24	18138.27	128.58
BIG_17	236733.57	138.10	225830.93	312.22	219496.80	407.15	219546.53	873.76
BIG_20	187565.60	519.52	112655.93	263.40	96403.93	364.16	102566.73	498.46

En la tabla 4.6 se muestra la sintonización del parámetro ALPHA que utiliza el algoritmo, donde se puede observar que se obtuvieron los mejores resultados en cuanto a calidad de la función objetivo se obtiene con el valor de 0.999. Sin embargo, debido a que, al tomar este valor, el tiempo crece demasiado, el valor para Alpha se escogió el 0.99, siendo que con este valor se obtiene buenos resultados en calidad y tiempo.

Tabla 4. 6 Calidad promedio con 30 corridas al sintonizar ALPHA en Aceptación por Umbral

Instancia	0.85	Tiempo(s)	0.9	Tiempo(s)	0.99	Tiempo(s)	0.995	Tiempo(s)	0.999	Tiempo(s)
SMALL_8	6241.93	8.40	7112.67	12.52	4064.40	135.22	2612.13	452.18	2466.47	1938.05
SMALL_14	13649.13	7.97	13575.67	8.36	9798.13	133.77	7979.13	278.63	6005.67	1607.90
MED_18	65108.67	10.88	59764.80	15.46	40937.60	147.12	40263.00	294.43	36202.47	1356.66
MED_20	40954.20	7.85	38178.00	11.23	19234.80	109.70	13867.67	211.93	10237.20	1104.84
BIG_17	271331.73	30.11	253759.20	43.60	229394.87	397.16	208621.71	765.07	172699.80	3663.22
BIG_20	172286.87	17.41	158627.00	26.15	101811.20	380.47	93934.47	600.23	82817.57	2740.54

4.3 SINTONIZACIÓN DE RECOCIDO SIMULADO.

La sintonización de los parámetros del algoritmo de Recocido Simulado, se establecen parecidos al de Aceptación por Umbral, los parámetros se encuentran establecidos las siguientes tablas siguientes 4.7 a 4.10.

En las primeras sintonizaciones de los parámetros t_{ini} y t_{fin} , los valores a escoger son U_{ini} 3000 y U_{fin} 0.1 los cuales se muestran en las tablas 4.7 y 4.8

Tabla 4. 7 Sintonización de t_{ini} y t_{fin} de Recocido Simulado (small_8)

t_{ini} \ t_{fin}	1000		1500		2000		2500		3000	
	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin
0.1	60,32%	19%	86,82	21,2%	76,3%	17,29%	82,2%	20,2%	90,3%	17,3%
0.01	68,3%	21%	80,2%	23,6%	77,03%	27,4%	88,3%	24,3%	88,3%	18,3%
0.001	66,89%	22%	79,34%	25,3%	67,3%	24,3%	80,66%	23,23%	89,90%	17,67%
0.0001	65,3%	23,2%	82%	22,76%	72,3%	17,4%	87,64%	19,4%	86,39%	21%

Tabla 4. 8 Sintonización de t_{ini} y t_{fin} Recocido Simulado (Instancia big_20)

t_{ini} \ t_{fin}	1000		1500		2000		2500		3000	
	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin	A_ini	A_fin
0.1	73%	24,3%	77,9%	26,9%	82,1%	26,74%	90,2%	25,96%	95,4%	20,1%
0.01	75,23%	25,43%	80,3%	27,54%	84,8%	25,87%	92,93%	22,56%	92,1%	24%
0.001	80,1%	27,2%	86,93%	29,90%	83,63%	24,76%	94,3%	23,59%	95,67%	27,3%
0.0001	71,2%	24,94%	84,56%	29,39%	83,26%	25,21%	92,2	26,22%	94,32%	29,3%

LMAX es la longitud del ciclo interno de Recocido Simulado, se muestran los resultados en la tabla 4.9, el valor que mejor resultados brinda en calidad es de 3000.

Tabla 4. 9 Sintonización de LMAX Recocido Simulado

INSTANCIA	LMAX							
	1000	Tiempo(s)	2000	Tiempo(s)	3000	Tiempo(s)	4000	Tiempo(s)
SMALL_8	8549.2	147.378	8201.867	290.3247	8347.133	427.4133	8927.467	592.1087
SMALL_14	17350.6	350.4133	13757.73	75.194	13721.07	132.2253	14955.07	199.582
MED_18	72829.4	87.588	72328.07	176.8993	64387.07	287.1227	65055.53	394.8987
MED_20	33142.2	36.46133	31103.53	71.75067	24335.57	107.8747	24641.87	143.328
BIG_17	155318.4	117.6193	129713.8	231.4647	148372.2	354.17	139557.2	470.266
BIG_20	277201.9	105.4	275910.7	224.896	271456.8	317.6793	287910.5	387.0193

En la tabla 4.10 se muestra la sintonización del parámetro ALPHA. Aunque el parámetro 0.999 da mejores resultados se establece el valor de ALPHA a 0.99, dado que el tiempo es diez veces menor y el deterioro de la calidad es relativamente pequeño.

Tabla 4. 10 Sintonización de ALPHA Recocido Simulado

Instancia	ALPHA									
	0.85	Tiempo(s)	0.90	Tiempo(s)	0.99	Tiempo(s)	0.995	Tiempo(s)	0.999	Tiempo(s)
small_8	9435.33	35.13	8839.73	90.32	6169.73	361.66	7548.87	778.82	4586.93	3690.20
small_14	18076.87	7.62	16334.22	11.50	13063.53	147.94	13575.27	268.94	7835.67	827.04
medium_18	80722.60	18.98	73303.00	27.95	58980.47	258.38	59907.07	519.70	45109.27	2640.20
medium_20	50089.13	6.22	46415.67	9.09	30322.87	90.36	22604.53	178.80	13779.40	932.99
big_17	291820.27	28.01	280470.47	39.77	251264.93	353.49	254103.60	692.22	238582.29	3379.22
big_19	188917.27	14.95	179261.07	23.41	134404.73	222.90	125148.40	445.41	91720.13	2284.79

4.4 SINTONIZACIÓN DE ITERACIONES DE RECOCIDO SIMULADO.

La sintonización para detectar el número ideal de iteraciones completas de recocido simulado, se realizan con 8, 9, 10, 11 y 12 iteraciones dado que en corridas con 7 o menores tiene el mismo resultado, a partir de 8 se encuentra mejoría con respecto a la función objetivo incluso factibilidad. Una iteración completa se refiere a la ejecución de una corrida completa de Recocido simulado, desde su temperatura inicial hasta su temperatura final, o hasta que encuentre una solución factible. Los resultados muestran que 12 iteraciones de recocido generan mejor calidad en la función objetivo, siendo esta utilizada, dado que exponer más de 12 iteraciones causaría mucho desgaste en tiempo, el cual no mejora la calidad y empeora el tiempo, en la tabla 4,11 se muestra la calidad y en la tabla 4.12 el tiempo de ejecución.

Tabla 4. 11 Calidad de función objetivo iteraciones recocido simulado

ITERACIONES	SMALL_8	SMALL_14	MED_18	MED_20	BIG_17	BIG_20
Recocido _Itera_8	948.25	1813.133	4798.6	5645.867	13954.52	59753.63
Recocido _Itera_9	814.5	1306.067	4697.333	3667.133	13685.65	54832.07
Recocido _Itera_10	615.3	798.3333	2022.6	1907.6	130618.2	44522.47
Recocido _Itera_11	870.6	870.8667	1807.267	4765.8	135328.1	49729.47
Recocido _Itera_12	725.6667	725.3333	4839	2646	128265.1	41369.6

Tabla 4. 12 Tiempo de función objetivo iteraciones recocido simulado

ITERACIONES	SMALL_8	SMALL_14	MED_18	MED_20	BIG_17	BIG_20
Recocido _Itera_8	1725.3	1472.565	1985.6	858.1627	4087.6	2459.834
Recocido _Itera_9	1545.6	1697.377	1976.517	920.9347	3848.69	2465.3
Recocido _Itera_10	1152.6	1581.066	2252.499	921.37	3536.539	2262.191
Recocido _Itera_11	1502.403	1921.737	2303.846	1046.671	3790.37	2702.762
Recocido _Itera_12	1610.317	1864.546	2765.552	1050.842	4086.168	2937.989

4.5 RESULTADOS

La metodología propuesta está compuesta por tres algoritmos clave, el primero una Heurística para encontrar el evento más restringido, en segundo lugar, el algoritmo de Aceptación por Umbral y por último el algoritmo de Recocido Simulado, a continuación, se muestran los resultados obtenidos en calidad-tiempo individualmente por cada algoritmo, y el resultado general de los tres algoritmos trabajando secuencialmente, cada instancia se ejecutó 30 veces, donde se obtuvo el mejor resultado, promedio y tiempo.

Los resultados de las instancias factibles resueltas por la Heurística del evento más Restringido se muestran en la tabla 4.11. Se obtuvieron 5 instancias factibles en el conjunto de instancias tipo “small”

Tabla 4. 13 resultados de Factibilidad Heurística del evento más restringido

Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)	Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)
Small_1	0	0	0.54	Small_11	0	0	0.30
Small_2	0	0	0.23	Small_12	0	0	0.40
Small_3	28342	30628.83	0.09	Small_13	53345	55742.8	0.15
Small_4	5255	11402.6667	0.08	Small_14	51196	53657.07	0.14
Small_5	29383	34702.4667	0.06	Small_15	9968	11973.6	0.09
Small_6	0	0	0.43	Small_16	25039	31300.2	0.15
Small_7	18044	23268.4	0.07	Small_17	52094	53174.53	0.14

Small_8	39200	40964.7333	0.11	Small_18	28497	32065	0.11
Small_9	63788	66890.7333	0.13	Small_19	66334	111517.9	0.19
Small_10	27600	49050	0.14	Small_20	4372	7993.333	0.08

Los resultados de las instancias factibles resueltas por el algoritmo de Aceptación por Umbral, después de utilizar el algoritmo de la Heurística más restringida se muestran en la tabla 4.12. Se obtuvo Factibilidad en 42 instancias, lo que significa un incremento de otras 37 instancias factibles con respecto a utilizar solamente la Heurística para obtener soluciones iniciales. Se obtuvieron 17 instancias factibles en el conjunto "small", 14 instancias factibles en el conjunto "medium" y 13 instancias factibles en el conjunto "big".

Tabla 4.14 Resultados de Factibilidad Aceptación por Umbral Instancias Small.

Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)	Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)
Small_1	0	0	0.54	Small_11	0	0	0.30
Small_2	0	0	0.23	Small_12	0	0	0.40
Small_3	0	0	5.47	Small_13	0	6457.6	327.77
Small_4	0	0	5.24	Small_14	4354	9433.66	121.91
Small_5	0	0	23.86	Small_15	0	0	2.78
Small_6	0	0	0.43	Small_16	0	0	4.17
Small_7	0	74.6	9.46	Small_17	0	0	12.01
Small_8	1087	5298.6	129.39	Small_18	0	221.6	41.05
Small_9	2201	8650.4	152.68	Small_19	0	4053.2	69.37
Small_10	0	834.53	37.03	Small_20	0	0	7.38

Tabla 4.15 Resultados de Factibilidad Aceptación por Umbral Instancias Medium

Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)	Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)
Med_1	0	0	4.24	Med_11	0	0	12.77
Med_2	0	0	9.57	Med_12	0	0	11.55
Med_3	0	0	11.74	Med_13	0	2814.33	63.80
Med_4	0	0	12.60	Med_14	0	0	34.73
Med_5	0	205.667	30.14	Med_15	0	615.2	52.17
Med_6	0	0	9.64	Med_16	10739	24065.2	89.58
Med_7	16451	28073.3	623.32	Med_17	0	0	13.19
Med_8	0	0	28.37067	Med_18	24893	51099.4	133.184
Med_9	13451	29383.4	67.17733	Med_19	27075	43314.4	110.358
Med_10	0	0	7.9	Med_20	10979	16737.4	88.6

Tabla 4.16 Resultados de Factibilidad Aceptación por Umbral Instancias Big

Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)	Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)
Big_1	0	0	15.98	big_11	0	0	76.588
Big_2	0	0	16.308	big_12	0	656.92	19.306
Big_3	0	0	13.1	big_13	0	0	52.828
Big_4	0	0	121.256	big_14	0	417.8	24.056
Big_5	2042	3471.6	128.174	big_15	3081	9895.8	203.954
Big_6	16340	26760.4	916.518	big_16	5133	9038.6	205.552
Big_7	100635	115485.3	517.676	big_17	187936	199773	144828.6
Big_8	0	0	479.81	big_18	93369	101482.8	217.55
Big_9	0	407.2	996.304	big_19	202924	212405.4	275.038
Big_10	0	1357.4	1130.11	big_20	73566	96152.6	256.342

A continuación, se muestran los resultados de Factibilidad de las instancias en conjunto de los tres algoritmos, a esta parte se agrega el algoritmo de recocido simulado con el cual se obtiene la Factibilidad en 55 instancias. Los resultados se observan en las tablas 4.17, 4.18 y 4.19. Se hicieron 30 corridas por cada instancia. Se obtuvieron resultados factibles para todas las instancias tipo “Small” y “Medium” y 15 instancias factibles para las instancias tipo “Big”

Tabla 4. 17 Resultados de Factibilidad Final instancia Small

Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)	Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)
Small_1	0	0	0.54	Small_11	0	0	0.304
Small_2	0	0	0.23	Small_12	0	0	0.401
Small_3	0	0	5.47266667	Small_13	0	6457.6	327.7707
Small_4	0	0	5.24	Small_14	0	9433.667	1864.546
Small_5	0	0	23.866	Small_15	0	0	2.783333
Small_6	0	0	0.43	Small_16	0	0	4.176667
Small_7	0	74.6	9.468	Small_17	0	0	12.01133
Small_8	0	725.66	1610.31	Small_18	0	221.6	41.05133
Small_9	0	1171.73	1123.65	Small_19	0	4053.2	69.37533
Small_10	0	834.53	37.03867	Small_20	0	0	7.388667

Tabla 4. 18 Resultados de Factibilidad Final instancias Medium

Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)	Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)
Med_1	0	0	4.24	Med_11	0	0	12.778
Med_2	0	0	9.57	Med_12	0	0	11.55
Med_3	0	0	11.74	Med_13	0	2814.33	63.80
Med_4	0	0	12.60	Med_14	0	0	34.73
Med_5	0	205.667	30.14	Med_15	0	615.2	52.17
Med_6	0	0	9.64	Med_16	0	179.1	518.34
Med_7	0	38408.8	7421.13	Med_17	0	0	13.196
Med_8	0	0	28.37	Med_18	0	4839	2765.55
Med_9	0	9503.86	723.99	Med_19	0	7620.26	2279.06
Med_10	0	0	3.9	Med_20	0	725.66	1610.31

Tabla 4. 19 Resultados de Factibilidad Final instancias Big

Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)	Instancia	Mejor	Promedio F(x)	Promedio Tiempo (S)
Big_1	0	0	15.98	big_11	0	0	76.588
Big_2	0	0	16.308	big_12	0	656.92	19.306
Big_3	0	0	13.1	big_13	0	0	52.828
Big_4	0	0	121.256	big_14	0	417.8	24.056
Big_5	0	238.233	2261.24	big_15	0	0	3806.65
Big_6	0	6992.67	39462.2	big_16	0	0	2375.23
Big_7	75986	89206.3	4653.45	big_17	96564	128265.1	4086.168
Big_8	0	0	479.81	big_18	39450	57823.7	5011.4
Big_9	0	407.2	996.304	big_19	177163	198246	2695.7
Big_10	0	1357.4	1130.11	big_20	29014	41369.6	2937.989

En la siguiente tabla 4.20 se muestra la comparación con los algoritmos propuestos en la literatura que tratan las mismas instancias, se muestra que el algoritmo de este trabajo genera buenos resultados al obtener 55 instancias factibles, esta solo por debajo del algoritmo de "Búsqueda local iterativa" y a la par de los algoritmos de clanes y búsqueda tabú.

Tabla 4. 20 Comparación de resultados de los diferentes algoritmos propuestos en la literatura.

Grupo	ILS	TAA	CBA	TSA	MA	SA-M	EIS	HSA	H	GGA
Pequeña (20)	20	20	20	20	20	20	20	19	20	18
Mediana (20)	20	20	20	20	20	20	20	17	16	15
Grande (20)	18	15	15	15	14	13	12	7	7	5
Total	58	55	55	55	54	53	52	43	43	38

Siglas	Algoritmo	Siglas	Algoritmo
CBA	Algoritmo basado en clanes	TSA	Algoritmo búsqueda tabú
MA	Algoritmo memético	EIS	Estrategia de inserción de eventos
GGA	Algoritmo de agrupamiento genético	ILS	Algoritmo de búsqueda local iterativa
H	Algoritmo de búsqueda local	SA-M	Recocido simulado
HSA	Algoritmo de hibridación recocido simulado		

CAPÍTULO V CONCLUSIONES Y TRABAJOS FUTUROS

En este capítulo se muestra las conclusiones y trabajos futuros del presente trabajo.

5.1 CONCLUSIONES

En el presente trabajo se propuso una heurística para encontrar el evento más restringido con la cual puede lograr asignar todos los eventos en un intervalo de tiempo, esta asignación en algunas instancias apertura periodos virtuales, los cuales se tratan de eliminar con ayuda de los algoritmos de Aceptación por Umbral y Recocido Simulado, todo esto para poder llegar a encontrar factibilidad. Para probar el presente algoritmo se utilizaron 60 instancias propuestas para probar factibilidad

Se utilizó el algoritmo de Aceptación por Umbral al finalizar la heurística del evento más restringido, el cual empieza a trabajar en base a la solución generada por esta heurística, el objetivo es tratar de eliminar los periodos virtuales o mejorar la calidad de la función objetivo al eliminar periodos virtuales y/o eventos asignados en los periodos virtuales. Cabe señalar que la metaheurística de aceptación por umbral brinda muy bueno resultados, ya que puede lograr encontrar Factibilidad en 70% del Benchmark.

Por último, se implementa la Metaheurística de Recocido Simulado, con la cual se pretende encontrar Factibilidad en la mayoría de las instancias o mejorar la calidad de la función objetivo, cabe destacar que cada metaheurística emplea los dos vecindarios para su funcionamiento.

Al utilizar en forma secuencial la Heurística del evento más restringido, Aceptación Por Umbral y Recocido Simulado con dos vecindarios, con la cual se obtuvo un total de 55 soluciones factibles, que representa 91.6% del benchmark.

El presente Trabajo es bastante competitivo con los de la literatura, quedando por debajo de “Búsqueda local iterativa” que hasta la fecha de hoy cuenta con un total de 58 instancias factibles.

5.2 TRABAJOS FUTUROS

- Implementar un nuevo vecindario que ayude a mejorar la calidad de la función objetivo y tiempo, así poder asignar los eventos que tenga poca disponibilidad y crear espacios que ayuden a la generación de Factibilidad en las instancias. Por ejemplo, permitiendo asignar los eventos que quedan sin espacio disponible en un intervalo de tiempo, quitando todos los eventos que se encuentran asignados en ese intervalo y asignando el evento que se quedó sin espacio, posteriormente a la asignación se tratan de asignar los eventos que previamente se quitaron para volver a buscar espacios disponibles para su asignación dentro del mismo intervalo o uno distinto del rango (1-45).

- Analizar e implementar diferentes enfoques metaheurísticos para aplicarlos, como lo pueden ser algoritmos de población, o hibridaciones como Meméticos que se ha probado que tienen buenos resultado en los problemas de Horarios.

- Sintonizar dinámicamente los parámetros de los Algoritmos de Aceptación por umbral y Recocido simulado, en un grupo de instancias “difíciles”, para mejorar la calidad de la función objetivo.

REFERENCIAS.

[AlHadid, 2020] AlHadid, I., Kaabneh, K., Tarawneh, H., & Alhroob, A. (2020). Investigation of simulated annealing components to solve the university course timetabling problem. Italian journal of pure and applied mathematics, 291 ISSN 2239-0227.

[Akkoyunlu, 1973] Akkoyunlu, E. A. (1973). A linear algorithm for computing the optimum university timetable. The Computer Journal, 16(4), 347-350 https://doi.org/10.1093/comjnl/16.4.347.3

[Alonso, 2008] Alonso Pecina, F. A. (2008). Programación de horarios escolares con recocido simulado y búsqueda tabú (Doctoral dissertation).

[Andre, 2018] Andre A. y Dinata H, (2018) Interaction Design to Enhance UX of University Timetable Plotting System on Mobile Version. Interaction, IOP Conf. Series: Materials Science and Engineering 407(1), (2018) 012174 doi:10.1088/1757-899X/407/1/012174, IOP Publishing, 2018.

[Avella, 2007] Avella, P., D'Auria, B., Salerno, S., & Vasil'ev, I. (2007). A computational study of local search algorithms for Italian high-school timetabling. Journal of Heuristics, 13(6), 543-556. DOI 10.1007/s10732-007-9025-3

[Frausto-Solis] Frausto-Solis, J., Alonso-Pecina, F., & Mora-Vargas, J. (2008, October). An efficient simulated annealing algorithm for feasible solutions of course timetabling. In Mexican International Conference on Artificial Intelligence (pp. 675-685). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-88636-5_64.

[Banczyk, 2006] Banczyk, K., Boinski, T., & Krawczyk, H. (2006, September). Parallelisation of genetic algorithms for solving university timetabling problems. In International Symposium on Parallel Computing in Electrical Engineering (PARELEC'06) (pp. 325-330). IEEE.Doi 10.1109/PARELEC.2006.64

[Bardadym, 1995] Bardadym, V. A. (1995, August). Computer-aided school and university timetabling: The new wave. In international conference on the practice and theory of automated timetabling (pp. 22-45). Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-61794-9_50

[Burke, 2002] Burke, E. K., & Petrovic, S. (2002). Recent research directions in automated timetabling. European Journal of Operational Research, 140(2), 266-280. https://doi.org/10.1016/S0377-2217(02)00069-3

[Burke, 2011] Burke, E.K., Mareček, J., Parkes, A.J. et al. A branch-and-cut procedure for the Udine Course Timetabling problem. Ann Oper Res 194, 71–87 (2012). https://doi.org/10.1007/s10479-010-0828-5

[Carter, 1996] Carter, M., Laporte, G. & Lee, S. Examination Timetabling: Algorithmic Strategies and Applications. J Oper Res Soc 47, 373–383 (1996). https://doi.org/10.1057/jors.1996.37

[Ceschia, 2012] Ceschia, S., Di Gaspero, L., & Schaerf, A. (2012). Design, engineering, and experimental analysis of a simulated annealing approach to the post-enrolment course timetabling problem. Computers & Operations Research, 39(7), 1615-1624 https://doi.org/10.1016/j.cor.2011.09.014

[Chen, 2020] Chen, M., Tang, X., Song, T., Wu, C., Liu, S., & Peng, X. (2020). A Tabu search algorithm with controlled randomization for constructing feasible university course timetables. Computers & Operations Research, 123, 105007.https://doi.org/10.1016/j.cor.2020.105007

[Chen, 2021] Chen M. N, S. N. Sze, S. L. Goh, N. R. Sabar and G. Kendall, "A Survey of University Course Timetabling Problem: Perspectives, Trends and Opportunities," in IEEE Access, vol. 9, pp. 106515-106529, 2021, doi: 10.1109/ACCESS.2021.3100613.

[Chiarandini,2006] Chiarandini, M., Birattari, M., Socha, K., & Rossi-Doria, O. (2006). An effective hybrid algorithm for university course timetabling. Journal of Scheduling, 9(5), 403-432. DOI:10.1007/s10951-006-8495-8

[Clausen,1999] Clausen, J. (1999). Branch and bound algorithms-principles and examples. Department of Computer Science, University of Copenhagen, 1-30.

[Colorni, Dorigo, Maniezzo, 1998] Colorni, A., Dorigo, M. & Maniezzo, V. Metaheuristics for High School Timetabling. Computational Optimization and Applications 9, 275–298 (1998). https://doi.org/10.1023/A:1018354324992

[Cook, 2006] Cook, S. (2006). The P versus NP problem. The millennium prize problems, 87-104.

[Cooper, 1996] Cooper T.B., Kingston J.H. (1996) The complexity of timetable construction problems. In: Burke E., Ross P. (eds) Practice and Theory of Automated Timetabling. PATAT 1995. Lecture Notes in Computer Science, vol 1153. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-61794-9_66

[Cruz-Chavez 2016] Cruz-Chávez, M. A., Flores-Pichardo, M., Martínez-Oropeza, A., Moreno-Bernal, P., & Cruz-Rosales, M. H. (2016). Solving a real constraint satisfaction model for the university course timetabling problem: a case study. *Mathematical Problems in Engineering*, *2016*.

[Daskalaki, 2004] Daskalaki, S., Birbas, T., & Housos, E. (2004). An integer programming formulation for a case study in university timetabling. European journal of operational research, 153(1), 117-135, https://doi.org/10.1016/S0377-2217(03)00103-6

[De Souza, 2012] De Souza Rocha, W., Claudia, M., Boeres, S., & Rangel, M. C. (2012). A GRASP algorithm for the university timetabling problem. In Proceeding of 9th international conference of the practice and theory of automated timetabling (PATAT) (pp. 404-406).

[DeWerra, 1985] De Werra, D. (1985). An introduction to timetabling. European journal of operational research, 19(2), 151-162. https://doi.org/10.1016/0377-2217(85)90167-5

[Even, 1975] S. Even, A. Itai and A. Shamir, "On the complexity of time table and multi-commodity flow problems," *16th Annual Symposium on Foundations of Computer Science (sfcs 1975)*, USA, 1975, pp. 184-193, doi: 10.1109/SFCS.1975.21.

[Garey, 1974] M. R. Garey, D. S. Johnson, and L. Stockmeyer. 1974. Some simplified NP-complete problems. In Proceedings of the sixth annual ACM symposium on Theory of computing (STOC '74). Association for Computing Machinery, New York, NY, USA, 47–63. DOI: https://doi.org/10.1145/800119.803884

[Glover, 1989] Glover, F. (1989). Tabu Search—Part I. ORSA Journal on Computing, 1(3), 190–206. doi:10.1287/ijoc.1.3.190

[Glover, 2006] Glover, F. W., & Kochenberger, G. A. (Eds.). (2006). Handbook of metaheuristics (Vol. 57). Springer Science & Business Media, ISBN 0-306-48056-5

[Gotlieb, 1963] Gotlieb, C. C. (1963). The construction of class-teacher timetables. In IFIP congress (Vol. 62, pp. 73-77). ISSN 02865580.

[Harrabi, 2020] Harrabi O. and J. C. Siala, "A New Sum Coloring-based Integer Linear Programming for Solving the University Course Timetabling Problem," 2020 International Multi-Conference on: “Organization of Knowledge and Advanced Technologies” (OCTA), 2020, pp. 1-2, doi: 10.1109/OCTA49274.2020.9151838.

[Henry, 2010] Henry Obit , J. (2010). Developing novel meta-heuristic, hyper-heuristic and cooperative search for course timetabling problems (Doctoral dissertation, University of Nottingham).isni: 0000 0004 2702 941X

[Holland, 1975] Holland, J. H. (1975). Adaptation in natural and artificial systems, university of Michigan press. Ann arbor, MI, 1(97), 5.

[ITC, 2002] International Timetabling Competition, http://sferics.idsia.ch/Files/ttcomp2002/ fecha de consulta 05 abril 2021.

[Kirkpatrick, 1983] Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. science, 220(4598), 671-680.

[Kostuch, 2003] Kostuch, P. (2003). Timetabling competition-sa-based heuristic. International Timetabling Competition.

[Lawrie, 1969] Lawrie, N. L. (1969). An integer linear programming model of a school timetabling problem. The Computer Journal, 12(4), 307-316. https://doi.org/10.1093/comjnl/12.4.307

[Lewis, 2005] Instancias http://www.rhydlewis.eu/hardTT/ Fecha de consulta 01 abril 2021

[Lewis, 2005B] Lewis R., Paechter B. (2005) Application of the Grouping Genetic Algorithm to University Course Timetabling. In: Raidl G.R., Gottlieb J. (eds) Evolutionary Computation in Combinatorial Optimization. EvoCOP 2005. Lecture Notes in Computer Science, vol 3448. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-31996-2_14

[Lewis, 2007A] Lewis R., Paechter B., Rossi-Doria O. (2007) Metaheuristics for University Course Timetabling. In: Dahal K.P., Tan K.C., Cowling P.I. (eds) Evolutionary Scheduling. Studies in Computational Intelligence, vol 49. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-48584-1_9

[Lewis, 2007B] Lewis, R., & Paechter, B. (2007). Finding feasible timetables using group-based operators. IEEE Transactions on Evolutionary Computation, 11(3), 397-413.doi 10.1109/TEVC.2006.885162

[Lewis, 2008] Lewis, R. (2008). A survey of metaheuristic-based techniques for university timetabling problems. OR spectrum, 30(1), 167-190. DOI 10.1007/s00291-007-0097-0.

[Limota, 2021] Limota, U., Mujuni, E., & Mushi, A. (2021). Solving the University course timetabling problem using bat inspired algorithm. Tanzania Journal of Science, 47(2), 674-685.

[Liu, 2011] Liu, Y., Zhang, D., & Chin, F. Y. (2011). A clique-based algorithm for constructing feasible timetables. Optimization Methods & Software, 26(2), 281-294. https://doi.org/10.1080/10556781003664739

[Marti, 2003] Martı, R. (2003). Procedimientos metaheurısticos en optimización combinatoria. Matemátiques, Universidad de Valencia, 1(1), 3-62.

[Martí, 2011] Martí, R., & Reinelt, G. (2011). The linear ordering problem: exact and heuristic methods in combinatorial optimization (Vol. 175). Springer Science & Business

[Morgenstern, 1990] Morgenstern, C., & Shapiro, H. (1990, January). Coloration neighborhood structures for general graph coloring. In Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms (pp. 226-235).

[Mouhamed, 2010] Al-Mouhamed, M., & Dandashi, A. (2010). Graph Coloring for class scheduling. In IEEE/ACS Internation Conference on Computer Systems and Applications (AICCSA) (pp. 1-4). doi: 10.1109/AICCSA.2010.5586963.

[Mühlenthaler, 2010] Mühlenthaler, M., & Wanka, R. (2010). A novel event insertion heuristic for creating feasible course timetables. In Proc, International Conference on the Practice and Theory of Automated Timetabling (pp. 294-304).

[Naupari, 2010] Naupari Quiroz, R. E., & Rosales Gerónimo, G. K. (2010). Aplicación de algoritmos genéticos para el diseño de un sistema de apoyo a la generación de horarios de clases para la Facultad de Ingeniería de Sistemas e Informática de la UNMSM.

[Nothegger, 2012] Nothegger, C., Mayer, A., Chwatal, A., & Raidl, G. R. (2012). Solving the post enrolment course timetabling problem by ant colony optimization. Annals of Operations Research, 194(1), 325-339.

[Osman, 1997] Osman, I. H., & Kelly, J. P. (1997). Meta-heuristics theory and applications. Journal of the Operational Research Society, 48(6), 657-657. https://doi.org/10.1057/palgrave.jors.2600781.

[PATAT, 1995] PATAT 1995. Lecture Notes in Computer Science, vol 1153. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-61794-9_51

[Pichardo, 2019] PICHARDO, M. F. (2019). Heurística constructiva para el modelo de restricciones de cursos universitarios de la facultad de ciencias químicas e ingeniería.

[Pichardo, 2018] Pichardo, M. F. (2018). Revisión de algoritmos genéticos aplicados al problema de la programación de cursos universitarios. Programación Matemática y Software.

[Qaurooni, 2013] Qaurooni, D., & Akbarzadeh-T, M. R. (2013). Course timetabling using evolutionary operators. Applied Soft Computing, 13(5), 2504-2514. https://doi.org/10.1016/j.asoc.2012.11.044

[Resende, 2014] Resende, M. G., & Ribeiro, C. C. (2014). GRASP: Greedy randomized adaptive search procedures. In Search methodologies (pp. 287-312). Springer, Boston, MA.

[Rezaeipanah, 2021] Rezaeipanah, A., Matoori, S.S. & Ahmadi, G. A hybrid algorithm for the university course timetabling problem using the improved parallel genetic algorithm and local search. Appl Intell 51, 467–492 (2021). https://doi.org/10.1007/s10489-020-01833-x

[Roos, 2003] Ross P., Hart E., Corne D. (2003) Genetic Algorithms and Timetabling. In: Ghosh A., Tsutsui S. (eds) Advances in Evolutionary Computing. Natural Computing Series. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-18965-4_30

[Rossi-Doria, 2002] Rossi-Doria, O., B. Paechter, C. Blum, K. Socha, and M. Samples, "A local search for the timetabling problem." in E. Burke and P. Causmaecker (eds.), Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling, PATAT 2002, pp. 124–127 Gent, Belgium (August 2002).

[Rossi-Doria, 2003] Rossi-Doria O. et al. (2003) A Comparison of the Performance of Different Metaheuristics on the Timetabling Problem. In: Burke E., De Causmaecker P. (eds) Practice and Theory of Automated Timetabling IV. PATAT 2002. Lecture Notes in Computer Science, vol 2740. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-45157-0_22

[Saldaña, 2007] Saldaña Crovo, A., Oliva San Martín, C., & Pradenas Rojas, L. (2007). Modelos de programación entera para un problema de programación de horarios para universidades. Ingeniare. Revista chilena de ingeniería, 15(3), 245-259 http://dx.doi.org/10.4067/S0718-33052007000300005

[Salvador, 2021] Salvador, N. (2019). Algoritmo genético en la asignación de horarios de un instituto de educación superior (Artículo científico). Repositorio de la Universidad Privada del Norte. Recuperado de https://hdl.handle.net/11537/26582

[Schaerf, 1999] Schaerf, A. A Survey of Automated Timetabling. Artificial Intelligence Review 13, 87–127 (1999). https://doi.org/10.1023/A:1006576209967

[Schaerf, 2001] Schaerf, A., & Di Gaspero, L. (2001, September). Local search techniques for educational timetabling problems. In Proceedings of the 6th International Symposium on Operational Research (SOR-01), Preddvor, Slovenia (pp. 13.23).https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.24.343&rep=rep1&type=pdf.

[Song, 2018] Song, T., Liu, S., Tang, X., Peng, X., & Chen, M. (2018). An iterated local search algorithm for the University Course Timetabling Problem. Applied Soft Computing, 68, 597-608.https://doi.org/10.1016/j.asoc.2018.04.034

[Song, 2021] Song, T., Chen, M., Xu, Y., Wang, D., Song, X., & Tang, X. (2021). Competition-guided multi-neighborhood local search algorithm for the university course timetabling problem. Applied Soft Computing, 107624 https://doi.org/10.1016/j.asoc.2021.107624.

[Tripathy, 1984] Tripathy, A. (1984). School Timetabling—A Case in Large Binary Integer Linear Programming. Management Science, 30(12), 1473–1489. doi:10.1287/mnsc.30.12.1473

[Tuga, 2007] Tuga, M., Berretta, R., & Mendes, A. (2007, July). A hybrid simulated annealing with kempe chain neighborhood for the university timetabling problem. In 6th IEEE/ACIS international conference on computer and information science (ICIS 2007) (pp. 400-405). IEEE.doi 10.1109/ICIS.2007.25

[Welsh, 1967] Welsh, D. J., & Powell, M. B. (1967). An upper bound for the chromatic number of a graph and its application to timetabling problems. The Computer Journal, 10(1), 85-86 https://doi.org/10.1093/comjnl/10.1.85.

[Wren, 1996] Wren A. (1996) Scheduling, timetabling and rostering — A special relationship?. In: Burke E., Ross P. (eds) Practice and Theory of Automated Timetabling.

[Yang, 2003] Yang XS. (2013) Metaheuristic Optimization: Nature-Inspired Algorithms and Applications. In: Yang XS. (eds) Artificial Intelligence, Evolutionary Computing and Metaheuristics. Studies in Computational Intelligence, vol 427. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29694-9_16

Cuernavaca, Morelos a 8 de noviembre del 2021.

DR. AUGUSTO RENATO PÉREZ MAYO
SECRETARIO DE INVESTIGACIÓN Y POSGRADO DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la **Maestría en Optimización y Computo Aplicado**, del estudiante **Lorenzo Antonio Cardoso Contreras**, con matrícula **10034118**, con el título **Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios** por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Federico Alonso Pecina
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

FEDERICO ALONSO PECINA | Fecha:2021-11-08 16:54:34 | Firmante

j62lzXT0+LgkoNAvT8RECPmx0CXIPoOz5wI/vBk8bA5Lnjcr5hwAhpt8USoEBlrBqeSSMoiCsjYyhxR1c2jcI9CPL4maWhxNjMux7cFH+FYzp9Yq7NiuQe1kYSRyXc2ep69LujkwgjLHoC7VUKWr68LTP/yEArbcR9XQ0hvZxCFphZo1ce8eJnwuz6yTsYKCcftUuESZOjYHvFQzsK0ix3aTJnDCv9YWB2KTut/ZzIrDZywPsBjxS4D/UUM+pPGigzfrvE0/gHgjvPa7hzc4ncwEyc1gw3WuXcMYbSYkps9UqMgdiDWzWz9z+nr3jPUmKPyiFJysKKA+wwEL/AtaZQ==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

E9ljvibcC

https://efirma.uaem.mx/noRepudio/3ZYIyQtFTPXV8CAkrSIOtVG4I4BEyYAA

Cuernavaca, Morelos a 04 de Noviembre del 2021.

DR. AUGUSTO RENATO PÉREZ MAYO
SECRETARIO DE INVESTIGACIÓN Y POSGRADO DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la **Maestría en Optimización y Computo Aplicado**, del estudiante **Lorenzo Antonio Cardoso Contreras**, con matricula **10034118**, con el título **Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios** por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Marco Antonio Cruz Chávez
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

MARCO ANTONIO CRUZ CHAVEZ | Fecha:2021-11-04 15:16:26 | Firmante

e3gGWTH/M491cUeOPGNpumF7XIX7vySZ4+HsUlUHM6McRugy+TlwMLoTmAxj3lgkwKoeI7IzA3Pcm4W3nP0mpXYWliW+ijKStjPLRPWN9gts5Z4BgnZi+Q/s3ZY7U8vit1pJvbwLv0+H0jzYLoeOvy2wsdclKcyEhgFG3FyJcWKw7wDJVwX6Idf7QdlU/A1cV/Uy0dFa7wmPMKdfJ4H76oH+oBSgnjysYOPgamGNuxaw5/anhGOdwDcEeuUakHDtn4G1MN0oX3l3t5/bhnvT51KJmDagvcUvGOyEdrcM/0W2/wDDU3fYH/QSEGgXvAgxwwSKQL1D1pfGSZDpTba/1Q==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

wN0YxKIUH

https://efirma.uaem.mx/noRepudio/S12eY0YKaAezIKPM0eeaGBq96kK2FrSV

Cuernavaca, Morelos a 22 de marzo del 2022.

DR. AUGUSTO RENATO PÉREZ MAYO
SECRETARIO DE INVESTIGACIÓN Y POSGRADO DE LA FCAeI
PRESENTE

En mi carácter de revisora de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la **Maestría en Optimización y Computo Aplicado**, del estudiante **Lorenzo Antonio Cardoso Contreras**, con matrícula **10034118**, con el título **Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios** por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dra. Irma Yazmín Hernández Báez
Profesora-investigadora
Universidad Politécnica del Estado de Morelos

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

IRMA YAZMIN HERNANDEZ BAEZ | Fecha:2022-03-24 07:15:25 | Firmante

Ler2kzkqYEaXCec01Pe7aT5utpy2RA/pcb/bS3XrSxHGryjdVkNmcOql7Nce21SPnJeuclYMpM8PQxEECNKQNE3ewebK+5fO6nqjMsOROVXacMGKPHN0SZIasVsjAsjE/NW6UtlkB0qwYDvVeGPWhm+cuGfsGmVzaJ9eGEghByh/WXjZ6blXpBLFLzbybgKnWGM2iUGHK6YbVtY4HRAvpEPxApqoZ2OT9zDoLrbGnD153LETe7jEWA0MKEx0o4c61WuWI4m2pxPppioJWJPBMQMYaCYgf5GsRMLGSshy3zD3adv4wQiPr9YCKPYDtO7lK+hfP+1sVADyVFjZKC6L+A==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

nc6th4G5F

https://efirma.uaem.mx/noRepudio/9BgYYT3WFBELj985kn4rhHuEZNK1YSg6

Cuernavaca, Morelos a 22 de marzo del 2022.

DR. AUGUSTO RENATO PÉREZ MAYO
SECRETARIO DE INVESTIGACIÓN Y POSGRADO DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la **Maestría en Optimización y Computo Aplicado**, del estudiante **Lorenzo Antonio Cardoso Contreras**, con matricula **10034118**, con el título **Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios** por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. José Crispín Zavala Díaz
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

JOSE CRISPIN ZAVALA DIAZ | Fecha:2022-03-27 14:04:45 | Firmante

DJ3TAKQ5xTqgM0KUeWXAKS5GINxEbGdRoFPRHPBcft2wip7kL/cynTAE3rBIa5fdNN+NXejbbgWCni2ODmkQmsobv0p2RuC8Vm9Ag9TwBVPzc6kl+eqsh2pQ9kLzaEVK8+hxJpzLRchyr7T/4V/KovB7SlM8gF1bZEzYrAXZwj6MTAChgGCflEyzaomvsUyBtACjFKixyFk53rs0AdvBlqUXQTilat8qw7M6ULnVDmv6ewujX5XsoqX5EPCKvpjFkit/nGDgIhUWHcsBZjTe4b54gvKWwwh2syCcA1i+sqMYj44Z61qa8w425dUD3HZra78Uk0HhBdSHxq0HJHvZmw==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

l4pNMSB5z

https://efirma.uaem.mx/noRepudio/XAf0uT9hdwFZ6r9ZmnoCV2N8pqhiFVQP

Cuernavaca, Morelos a 04 de noviembre del 2021.

DR. AUGUSTO RENATO PÉREZ MAYO
SECRETARIO DE INVESTIGACIÓN Y POSGRADO DE LA FCAeI
PRESENTE

En mi carácter de revisor de Tesis, hago de su conocimiento que he leído con interés la tesis para obtener el grado de la **Maestría en Optimización y Computo Aplicado**, del estudiante **Lorenzo Antonio Cardoso Contreras**, con matrícula **10034118**, con el título **Búsqueda de Soluciones Factibles para el Problema de Horarios de Cursos Universitarios** por lo cual, me permito informarle que después de una revisión cuidadosa de dicha tesis, concluyo que el trabajo se caracteriza por el establecimiento de objetivos académicos pertinentes y una metodología adecuada para su logro. Además, construye una estructura coherente y bien documentada, por lo cual considero que los resultados obtenidos contribuyen al conocimiento del tema tratado.

Con base en los argumentos precedentes me permito expresar mi **VOTO APROBATORIO** por lo que de mi parte no existe inconveniente para que el estudiante continúe con los trámites que esta Secretaría de Investigación y Posgrado tenga establecidos para obtener el grado mencionado.

Atentamente
Por una humanidad culta
Una universidad de excelencia

Dr. Martín Heriberto Cruz Rosales
Profesor- investigador
Facultad de Contaduría, Administración e Informática

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

MARTIN HERIBERTO CRUZ ROSALES | Fecha:2021-11-04 21:33:11 | Firmante

YAMI2KGM+PuV6n3vgVBJt9Ft0gDYUd0YR2Lh52U4+kGGBMI67s3Oa6EpI0NXXoTruIz7hjMCrXOS1FlQX3aie+I+B9oqqOpNlh5xsGx3+tKnYhAi/zkajX+aIGeot6QslCDB8d+5QFtld7jVR6iNYLA91AV+sqOP8FqrpkYjvHp5d8PW75LG3ur57tVcj1VQD4808wQ9oBbVYeN5SJNL5exOBfXUCNPyPjGdhLrl0vxUusJx/2mF9r0kwv7OP1mO6OtpmYHFd24tXGGqrq3gdUynveLnfe5V0H1mGZWgVLpv435TM5oLVYl6NMGypP3yoQtwhcaEvOucy3iom3w5fg==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

YcDIJWtEz

https://efirma.uaem.mx/noRepudio/RzFbhUIFCaDq1YxYx0Cb0YCctDmOHU7s