SEP **TNM**

INSTITUTO TECNOLÓGICO DE TIJUANA

DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

ENFOQUE EVOLUTIVO EN LA OPTIMIZACIÓN DE PARTICIONES PARA SOLUCIÓN DE PROBLEMAS DE SATISFACCIÓN DE RESTRICCIONES

TRABAJO DE TESIS

Presentado por
M.C. LUCERO DE MONTSERRAT ORTIZ AGUILAR

Para Obtener el Grado de
DOCTOR EN CIENCIAS EN COMPUTACIÓN

Director de Tesis
DR. JUAN MARTÍN CARPIO VALADEZ

Co-Director de Tesis
DR. HÉCTOR JOSÉ PUGA SOBERANES

Tijuana, BC, Diciembre, 2020

"Año de Leona Vicario, Benemérita Madre de la Patria"

INSTITUTO TECNOLOGICO DE TIJUANA
POSGRADO EN COMPUTACION

Asunto: Se autoriza impresión de Trabajo de Tesis

Tijuana, B.C., 30 de Noviembre del 2020

C. Dra. Yazmin Maldonado Robles
Jefe de la Div. de Estudios de Posgrado e Investigación
Presente.

En lo referente al trabajo de tesis escrito, con título " **ENFOQUE EVOLUTIVO EN LA OPTIMIZACIÓN DE PARTICIONES PARA SOLUCIÓN DE PROBLEMAS DE SATISFACCIÓN DE RESTRICCIONES"**, presentado por el ***C. LUCERO DE MONTSERRAT ORTIZ AGUILAR,*** alumna del Doctorado en Ciencias en Computación con número de control **D09240932**, informamos a usted que se autoriza el escrito de tesis y se aprueba en todas sus partes, en virtud de reunir los requisitos de un trabajo de grado de Doctorado y a la vez se autoriza al interesado para que proceda de inmediato a la impresión del mismo y a presentar su examen de grado, ya que cumple con todos los requisitos.

ATENTAMENTE

DRA. ELBA PATRICIA MELIN OLMEDA
PRESIDENTE

DR. OSCAR CASTILLO LOPEZ
SECRETARIO

DR. JOSE MARIO GARCIA VALDEZ
VOCAL

DR. FEVRIER ADOLFO VALDEZ ACOSTA
VOCAL

DRA. DANIELA ADRIANA SANCHEZ VIZCARRA
VOCAL

c.c.p. Oficina de Titulación
c.c.p. División de Estudios de Posgrado e Investigación
c.c.p. Expediente
c.c.p. Interesado

EPMO/*inf

Calzada del Tecnológico S/N Esq. Av. Castillo de Chapultepec y calle Cuauhtemotzin,
Fracc. Tomás Aquino C.P. 22414, Tijuana, Baja California. Conmut. (664) 6078400 Ext. 101
e-mail: direccion@tectijuana.mx,
www.tectijuana.edu.mx

EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLOGICO NACIONAL DE MÉXICO

Instituto Tecnológico de Tijuana

"Año de Leona Vicario, Benemérita Madre de la Patria"

Tijuana, Baja California, 07/diciembre/2020

OFICIO No. 108/DEPI/2020
Asunto: Autorización de Impresión de Tesis

MARIBEL GUERRERO LUIS
JEFA DEL DEPARTAMENTO DE SERVICIOS ESCOLARES
PRESENTE

En lo referente al trabajo de tesis, **"Enfoque evolutivo en la optimización de particiones para solución de problemas de satisfacción de restricciones"**. Presentado por C. **Lucero de Montserrat Ortiz Aguilar,** alumna del Doctorado en Ciencias en Computación con número de control **D09240932**; informo a usted que a solicitud del comité de tutorial, tengo a bien **Autorizar la impresión de Tesis,** atendiendo las disposiciones de los Lineamientos para la Operación de Estudios de Posgrado del Tecnológico Nacional de México.

Sin más por el momento le envío un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
Por una Juventud Integrada al Desarrollo de México ®

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE TIJUANA
DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

YAZMIN MALDONADO ROBLES
JEFA DE DIVISIÓN DE ESTUDIOS DE POSGRADO E INVESTIGACIÓN

C.p. Interesado
C.p. Archivo

DECLARACIÓN DE ORIGINALIDAD

Tijuana, BC., 19 de Noviembre de 2020,

Yo, **Lucero de Montserrat Ortiz Aguilar** estudiante del Doctorado en Ciencias en Computación, en mi calidad de autor manifiesto que este documento de tesis es producto de mi trabajo original y que no infringe los derechos de terceros, tales como derechos de publicación, derechos de autor, patente y similaridad. Por lo tanto, la obra realizada es de mi exclusiva autoría y no infringí en copiar el texto o imágenes, de fuentes de información por lo cual soy responsable del escrito que aquí se presenta.

Así mismo, declaro que en las citas textuales que he incluido (las cuales aparecen entre comillas) y en los resúmenes que he realizado de publicaciones ajenas, indico explícitamente los datos de los autores y las publicaciones.

En caso de presentarse cualquier reclamación o acción por parte de terceros en cuanto a los derechos de autor sobre la obra en cuestión, acepto toda la responsabilidad de tal infracción y relevo de esta a mi director de tesis, así como al Tecnológico Nacional de México, al Instituto Tecnológico de Tijuana y a sus respectivas autoridades.

Lucero de Montserrat Ortiz Aguilar
Estudiante de Doctorado en Ciencias en Computación

AGRADECIMIENTOS

Agradezco al Tecnológico Nacional de México y al Consejo de Nacional de Ciencia y Tecnología por el apoyo brindado durante el desarrollo de este trabajo, con el numero de beca 446106, el cual se gestó dentro del programa de Doctorado en Ciencias en Computación, ofertado de manera interinstitucional por el Instituto Tecnológico de Tijuana y el Instituto Tecnológico de León. Agradezco a la coordinadora general del doctorado, Dra. Elba Patricia Melín Olmeda, así como al comité tutorial, Dr. Juan Martín Carpio Valadez, Dr. Héctor José Puga Soberanes, Dr. Jorge Alberto Soria Alcaraz, Dr. Manuel Ornelas Rodríguez, y Dr. Alfonso Rojas Domínguez.

A mi familia que a pesar de las vicisitudes que tuvimos estos años, siempre me brindaron su apoyo incondicional para continuar estudiando.

A mi mamá por haberme dado todo su amor, paciencia, preocupaciones mientras estaba lejos y cuidados para continuar este camino.

A mi papá por haberme dado la fortaleza cuando estuve en Italia para seguir adelante y que mi estancia fuera lo mejor posible.

A mi hermana Jenny por acompañarme en todas mis locuras y comer conmigo mientras estaba en Italia, aunque se desvelará.

A mi hermana Rosario por ser tan paciente, fuerte y ser el pilar de la casa mientras pasaba momentos difíciles.

A mi amiga Eli, por ser mi mejor amiga, cómplice y escucharme cada tarde mientras la pasaba mal. Por ser parte de eventos alegres y ser como una hermana para mí.

A mi amigo Rafa, simplemente por ser Rafa, y darme aliento de tranquilidad en momentos difíciles. Y ser compañero incondicional todo este tiempo y ser como el hermano para mí.

A mis compañeros de doctorado Adolfo, Juan y Valentín por acompañarme en este camino, en ellos vi una amistad genuina y sincera. Al Dr. Carpio, por ser mi guía en este camino de 10 años universitarios. Le agradezco el darme la oportunidad de llegar tan lejos en este ámbito educativo y ser también un gran amigo.

A todas aquellas personas que tocaron mi corazón y fueron parte de este proyecto, ¡Gracias!.

Sabes bien que te queremos y deseamos siempre que salgas adelante. Te protegimos desde antes que nacieras. Lo hicimimos mientras crecias y ahora que eres mayor de edad continuaremos haciendolo hasta el ultimo día de nuestra vida, Que dios nos de.

Te queremos y amanos
Tus hermanas y tus papas
Familia Ortiz Aguilar

El miedo al abandono es un miedo propio de criaturas que dan por hecho que, al alejarse la fuente de la vida, morirán. Asumir que no somos niñas sino mujeres adultas es un proceso interno muy complejo, de maduración emocional afectiva. Es difícil, pero indispensable. Solo con soledad podremos saber qué es realmente lo que podemos esperar de nosotras mismas y qué podemos esperar de otras personas.
Dra. Marcela Lagarde

ABSTRACT

Different task assignment problems are subject to restrictions. The problems where it is hard to satisfy a set of restrictions are graph coloring, crosswords, assignment of tasks, location of warehouses, routing problems, routing problems with time windows, scheduling problems, etc. The timetable designs can be subject to a set of restrictions, which depend on the overall problem. For this reason, developing a good timetable design can optimize resources within companies.

The University timetabling problem is one of the problems where a set of restrictions depend in some cases on: students, teachers, and the institution's classroom. For each school period (example: semester or trimesters), the demand for subjects by students can change, therefore a static design is not feasible to use. This research seeks to generate a partition design that allows the simultaneous solution of the combinatorial problems. As test instances, some proposals will be used in the Ïnternational Conference on the Practice and Theory of Automated Timetabling"(PATAT), and real instances, from historical data from the Technological Institute of León.

In this thesis, the API-Carpio and Design Methodologies proposed in the state of the art by Soria were applied, which allowed us to model a certain set of restrictions. Both methodologies gave us the facility to generate feasible solutions to the combinations of these problems. Now, the use of exact methods for the construction and evaluation of all the possible solutions is not very feasible. In this project the use of evolutionary computation techniques that involve Heuristics, Metaheuristics and Hyperheuristics and some metalearning concepts to give a solution to three different combinatorial problems.

RESUMEN

Diferentes problemas de asignación de tareas están sujetos a un conjunto de restricciones. Algunos problemas donde se busca satisfacer un conjunto de restricciones son: coloreo de grafos, crucigramas, asignación de tareas, ubicación de bodegas, ubicación de Antenas de telefonía, problemas de rutas con ventanas de tiempo, problemas de agenda y asignación de proyectos, etc. Calendarizar eventos puede estar sujeto a un conjunto de restricciones, las cuales dependen del problema en general. Por ello, elaborar un buen diseño de agenda de tareas puede permitir optimizar recursos dentro de las empresas.

El problema de University timetabling es uno de los problemas donde se contempla un conjunto de restricciones que dependen en algunos casos de: alumnos, maestros e inmueble de la institución. Dado que, en cada periodo escolar, llámese semestre, cuatrimestre, trimestres, etc., puede cambiar la demanda de materias por parte de los alumnos, un diseño estático es poco factible de emplear. En este trabajo se buscará generar un diseño de particiones que permita la solución simultánea de la combinación de los problemas: Course-Faculty-Classroom assignment Timetabling. Como instancias de prueba se utilizarán algunas propuestas en el International Conference on the Practice and Theory of Automated Timetabling"(PATAT), e instancias reales, provenientes de datos históricos del Instituto Tecnológico de León.

En este trabajo de tesis se aplicó las metodologías API-Carpio y Metodología del diseño propuesta en el estado del arte por Soria, las cuales nos permitieron modelar un cierto conjunto de restricciones. Ambas metodologías nos brindaron la facilidad de generar soluciones factibles a las combinaciones de estos problemas. Ahora bien, el uso de métodos exactos para la construcción y evaluación de todas las posibles soluciones resulta poco factible, en este proyecto se propuso el uso de técnicas de cómputo evolutivo que involucran Heurísticas, Metaheurísticas e hiperheurísticas, que nos permitirán generar soluciones aceptables en menor tiempo.

Índice general

Índice de figuras

Índice de tablas

Lista de algoritmos

Capítulo 1

Introducción

Actualmente existen diferentes metodologías y/o esquemas de diseño que nos permiten resolver problemas de calendarización de eventos. Dicha calendarización en algunos casos requiere que se satisfagan un conjunto de restricciones, por ejemplo:

- En las escuelas se busca asignar materias, considerando que los alumnos puedan tomar una determinada cantidad de clases por semestre. Además de que se debe apegar a la cantidad de aulas de la escuela y la planta docente contratada.

- En algunos centros de transporte buscan generar rutas para sus vehículos, de tal forma que puedan entregar una cierta cantidad de cargas, y optimizar el recorrido y los costos de combustible cumpliendo tiempos de entrega establecidos.

- En los hospitales, se busca la mayor cantidad de cirugías en quirófanos, considerando la prioridad de las cirugías, el horario de los quirófanos y médicos.

En este proyecto se propuso construir particiones que nos permitan tener diseños de combinatorios de forma automática con técnicas de cómputo evolutivo para diferentes problemas de optimización combinatoria.

1.1. Antecedentes

Existen diferentes problemas de Satisfacción de restricciones, los cuales pueden ser susceptibles a ser resueltos por el diseño de particiones. Peter Jeanvons en [1], define que el problema de *Timetabling* puede ser modelado como un *CSP*.

Para la elaboración de este proyecto de tesis, se tomó como referencias los trabajos realizados a lo largo de una línea de elaboración de horarios que tiene más de 15 años, de la cual surgió la metodología API-Carpio [2], propuesta por un experto humano que tiene 20 años de experiencia en el diseño de horarios en el Instituto Tecnológico de León, para la asignación de horarios.

Uno de los trabajos que preceden a esta investigación, es el elaborado por Aguayo Ríos en 2009. Su trabajo titulado "*Optimización y Automatización en la Asignación de Tareas Aplicadas en el área Educativa*", donde mediante algoritmos genéticos da una primera aproximación de solución al problema de calendarización de horarios. En dicha investigación, se generaron algunos insumos como lo son MMA, MM y HMM, los cuales permiten generar algunos diseños de "*Vectores*" , los cuales contienen una cantidad "x" de materias cada uno. En este trabajo se comenzó con utilizar un algoritmo genético con un solo tipo de heurística para la mutación, dada la información con la que contaban, se limitó solo a realizar experimentación para saber cuál es el tamaño de población que obtenía un rendimiento competitivo y bajo costo [3].

Como trabajo consecutivo Soria en 2010, elaboró la tesis titulada "*Diseño de Horarios con Respecto al Alumno mediante Técnicas de Cómputo Evolutivo*". En este trabajo presentó la creación de un módulo que permite el diseño de horarios automáticamente, ahorrando tiempo y recurso humano para una universidad, en este caso el Instituto Tecnológico de León. Su diseño de horarios contempló las demandas de los estudiantes y las restricciones de horarios de las materias. Implementó nuevamente un algoritmo bioinspirado, técnicas de agrupamiento y el enfoque Hiperheurístico, pero a diferencia del trabajo realizado en [4], Soria utilizó varias heurísticas como muta para el algoritmo genético. En su trabajo sus resultados fueron competitivos para diferentes instancias del ITL, quedando abierto a la posibilidad de ampliar su enfoque a considerar otras

variables como lo son el profesor, aula y laboratorios.

De este trabajo de maestría se tuvo la pauta para el trabajo doctoral realizado en 2015 por el mismo autor. Soria en [5], hizo una integración de un esquema de Diseño en algoritmos de dos fases con técnicas de CSP para la calendarización de eventos. En este trabajo se propuso una metodología de modelado llamada "*Metodología de Diseño*", en la cual se contemplan algunas restricciones del problema de *Timetabling*, además de que es genérica y aplicable a diversas instancias. Soria propuso un enfoque dinámico de aprendizaje (*online*) para realizar una búsqueda autónoma de soluciones y los algoritmos resultantes se aplicaron sobre un conjunto de instancias del PATAT 2002 y 2007. El enfoque dinámico (*online*) planteado se comparó con el aprendizaje estático (*offline*), siendo el dinámico el que supero estadísticamente al estático y reporto además algunos resultados previamente no reportados en el estado del arte (*best know solutions*).

Por otra parte a nivel maestría en 2016 por parte de Ortiz, se generó un diseño de horarios para alumnos y maestros mediante técnicas de *Soft Computing* en [6]. Ese trabajo da continuidad al realizado en [4], las diferencias se enfocan en generar un diseño no solo para alumnos, sino para profesores y aulas. En [6] se propuso extender la metodología del diseño propuesta por Soria en [4], la cual fue probada para instancias generadas artificialmente. En total reportó 4 tipos de experimentación, dos para instancias reales del ITL y dos para instancias generadas artificialmente. Se implementaron diferentes técnicas Metaheurísticas como: algoritmos genéticos, memético, sistema inmune, búsqueda local iterada; y el enfoque Hiperheurístico para los cuatro experimentos.

Con este preámbulo se planteó este proyecto de investigación para dar continuidad en la línea de investigación abordada previamente en [6]. La continuidad o necesidad que se identificó fue el tener una metodología genérica y general que pudiera ser aplicada a diferentes problemas de optimización combinatorio sujetos a restricciones.

1.2. Definición del Problema

Debido a que continuamente se formulan nuevos problemas de optimización combinatoria, es común ver el surgimiento de nuevas técnicas tanto heurísticas, metaheurísti-

cas e Hiperheurísticas para su solución. Lo anterior deriva una necesidad constante de proponer nuevas metodologías de solución que sean genéricas, con un nivel de generalidad y con resultados competitivos o cercanos a un óptimo. Considerando este escenario surgen el cuestionamiento de saber si todos los problemas de optimización combinatoria comparten características en común, que nos permitan identificar de forma rápida como aproximar una primera solución. Y por otra parte estos problemas representan un reto en el diseño de nuevas técnicas, para los investigadores y desarrolladores de nuevos algoritmos de Inteligencia artificial.

El conocer cómo se relaciona la herramienta de solución con el problema es una pregunta abierta de investigación vigente en el área de las ciencias de la computación [7]. En este trabajo de tesis se busca dar una respuesta al cómo es que se pueden resolver determinados problemas de optimización combinatoria a través de metodologías basadas en conceptos matemáticos como lo son estructuras algebraicas y particiones [8].

La metodología usada en este trabajo basada en particiones [2], ha demostrado previamente obtener resultados competitivos en problemas como lo son *University Course Timetabling* y *Vehicle routing problem* [9]. Otro concepto importante son los diseños combinatorios, que son conformados por una tupla de variables y bloques de diseño, los cuales son subgrupos de estas variables. Por lo tanto, la parte complementaria de nuestro problema fue identificar las características que tienen en común otros problemas de naturaleza similar, para ser solucionados por particiones.

1.3. Hipótesis

Es posible construir un k-diseño combinatorio aplicando técnicas de cómputo evolutivo, para problemas de CSP que sean susceptibles a ser tratados por particiones, donde cada bloque (parte) a su vez satisfaga un subconjunto de restricciones y el k-diseño satisfaga otro subconjunto de restricciones.

1.4. Justificación

El investigar qué características tienen diferentes problemas de optimización y cómo las soluciones a estos varían de acuerdo con las técnicas que se aplican y permtió el usar metodologías ya conocidas por su buen desempeño para la solución de estos.

Esta investigación sirve para dos propósitos, el primero es que gracias al uso de estructuras algebraicas permitió observar y desambiguar cuándo un algoritmo está trabajando sobre diferentes dominios de problemas. Mientras que el segundo es poder resolver un conjunto de problemas combinatorios definidos por [8], sujetos a diferentes restricciones, bajo el enfoque de un tipo específico de estructura algebraica que son las particiones [10]. Lo anterior permitió mejorar el diseño de herramientas Metaheurísticas e Hiperheurísticas con el análisis de las heurísticas en los problemas combinatorios analizados.

Debido a que existen una gran ambigüedad en diferentes investigaciones relacionada al nivel de generalidad que tienen el algoritmo o metodología para resolver un problema o familia de estos; en esta investigación se buscó analizar al menos un conjunto de problemas bajo el contexto de las estructuras algebraicas, que nos permita determinar si la herramienta es general o no en diferentes dominios de problema. Es decir, que estos problemas tengan funciones objetivo diferentes, números de variables distinto, restricciones distintas y por lo tanto sus espacios de búsqueda sean diferentes. La relación que existe entre la herramienta que soluciona el problema y el tipo o clase de este mismo, es un tópico que se intentó dar respuesta en este trabajo de tesis doctoral.

Un problema común dentro del área de investigación de Hiperheurísticas de selección, es el determinar cuál es el conjunto básico de heurísticas. Aunado a que se busca tener Hiperheurísticas que puedan resolver problemas de diferentes dominios.

Por lo que tener una clasificación de problemas combinatorios guiados por una base formal matemática en este caso las estructuras algebraicas, nos permitirá hacer distinción de dominios o clases de problemas. Lo anterior será una base para plantear posteriormente una taxonomía por familia o clases de problemas.

1.5. Preguntas de Investigación

Las preguntas planteadas en esta investigación doctoral fueron:

- ¿Es posible distinguir dominios de problemas combinatorios de satisfacción de restricciones?
- ¿Es posible determinar cuáles algoritmos herísticos son más adecuados para resolver un problema combinatorio?
- ¿Cuáles son los problemas de Satisfacción de Restricciones que son susceptibles a ser resueltos por una partición?
- ¿Dado un problema de satisfacción de restricciones es posible aplicar el concepto de partición asociado a un *k-diseño* para solucionar el problema?
- ¿Dados diferentes problemas de satisfacción de restricciones, es posible proponer una metodología basada en particiones para dar solución a estos?

1.6. Objetivos

El objetivo general y específicos que se plantearon en el proyecto de tesis es el siguiente:

1.6.1. Objetivo General

Construir particiones de conjuntos de variables, por medio de técnicas de cómputo evolutivo, para CSP del estado del arte que sean susceptibles a ser resueltos por una partición, donde cada parte satisfaga a su vez un conjunto de restricciones.

1.6.2. Objetivos específicos

1. Clasificar los problemas combinatorios basándose en fuentes formales y una investigación profunda del estado del arte.

2. Evaluar y modelar problemas combinatorios que puedan ser utilizados como casos de uso a ser resueltos por particiones.

3. Modelar las instancias de problemas a insumos de particiones.

4. Generar soluciones a las instancias de problemas combinatorios con la metodología de particiones.

5. Comparar estadísticamente los resultados con:

   - Los propuestos por el experto humano.
   - Los propuestos por el estado del arte.

6. Reportar la mejor solución para las instancias de problemas combinatorios.

1.7. Alcances y Limitaciones

Los alcances y limitaciones de este trabajo de tesis doctoral se mencionarán a continuación.

1.7.1. Alcances

Este proyecto tuvo como alcance el proponer soluciones para problemas de optimización combinatorios que sean susceptibles a ser resueltos específicamente por particiones. De los problemas que se resolvieron por particiones, se tomaron al menos dos de diferentes familias.

La metodología propuesta se aplicó a los problemas seleccionados y como trabajo futuro se deja la posibilidad de aplicarlo a los problemas restantes. El análisis heurístico que se propuso es aplicado solamente para apoyar en el diseño de Hiperheurísticas de selección.

1.7.2. Limitaciones

En este proyecto no se pretendió realizar un estudio profundo de todos los *GCP* del estado de arte o clasificar sus variantes. Así mismo, el análisis de heurísticas no se hará para todos los dominios de *GCP*, y siendo solamente heurísticas de perturbación las seleccionadas para su análisis.

1.8. Publicaciones científicas relacionadas con la tesis

Los artículos producto de esta investigación y los resultados mostrados se pueden ver a lo largo de esta tesis. En total se generaron dos artículos, a continuación, se da el nombre y resumen de cada uno de ellos:

- **Ortiz-Aguilar, L. D. M.**, Carpio, M., Puga, H., Soria-Alcaraz, J. A., Ornelas-Rodríguez, M., Lino, C. (2017). I*ncrease Methodology of Design of Course Timetabling Problem for Students, Classrooms, and Teachers.* In Nature-Inspired Design of Hybrid Intelligent Systems (pp. 713-728). Springer, International Publishing.

- **Lucero Ortiz-Aguilar**, Martin Carpio, Alfonso Rojas-Domínguez, Manuel Ornelas-Rodriguez, Hector Puga , Adolfo Montesino-Guerra y Jorge A. Soria-Alcaraz. *Classification of General Combinatorial Problems: a Review Guided by their Algebraic Structures.*

- **Lucero Ortiz-Aguilar**, Martin Carpio, Alfonso Rojas-Domínguez, Manuel Ornelas-Rodriguez, H. J. Puga-Soberanes, Jorge A. Soria-Alcaraz. *A Methodology for selection and determination a subset of Heuristics for hyper-heuristics through meta-learning for solving Graph Coloring and Capacitated Vehicle Routing Problems.*

- Montesino-Guerra, J. A., Puga, H., Carpio, J. M., Ornelas-Rodríguez, M., Rojas-Domínguez, A., **Ortiz-Aguilar, L.** (2020). Combinatorial Designs on Constraint

Satisfaction Problem (VRP). In Intuitionistic and Type-2 Fuzzy Logic Enhancements in Neural and Optimization Algorithms: Theory and Applications (pp. 509-526). Springer, International Publishing.

1.9. Organización de la tesis

Para probar la hipótesis y lograr los objetivos plateados, este trabajo de tesis se estructuró de la manera siguiente:

En el capítulo 2 se analiza el estado de arte hasta la fecha de los artículos encontrados tanto nacionales como internacionales relacionados a diferentes problemas de optimización combinatoria. Así como investigaciones relacionadas a heurísticas, Hiperheurística y *meta-Learning*.

El capítulo 3 abarca el marco teórico de esta tesis. En éste se muestra los conceptos teóricos necesarios y algoritmos necesarios para resolver el problema en cuestión. Además de que se dan algunas definiciones encontradas que pueden ser aplicadas a diferentes problemas.

En el capítulo 4 se da una definición extensa de cada uno de los diez problemas de optimización combinatoria (GCP) clasificados. De cada problema se documentó sus antecedentes, definición en términos de estructuras algebraicas, variantes y casos o instancias de prueba reales o artificiales con sus óptimos o mejores soluciones validadas hasta la fecha de elaboración de esta tesis.

El capítulo 5 está dedicado a la descripción de la metodología general de trabajo propuesta para la solución del problema en cuestión.

El capítulo 6 muestra los resultados obtenidos de aplicar las metodologías definidas en el capítulo 5. Además, se documentó los resultados de pruebas estadísticas no paramétricas aplicadas a los diferentes problemas. Finalmente lo complementamos con sus respectivos análisis, para cada uno de los resultados obtenidos.

En el capítulo 7 se presenta las conclusiones y trabajo futuro propuestos como continuación de esta tesis. En este resumen todos los objetivos cumplidos y una breve explicación de que fue lo que se realizó en cada uno de los tipos de experimentos y

resultados obtenidos.

Por último, se anexa una sección de apéndices de apoyo, glosario de palabras y referencias las cuales el lector podrá consultar en cualquier momento.

Capítulo 2

Estado del arte

En este capítulo se presentan algunos trabajos relacionados, encontrados en el estado del arte de Problemas de Optimización Combinatoria, *Meta-Learning* e Hiperheurísticas, además de diversas formas de solución aplicadas en los últimos años.

2.1. Estado del arte en clasificación de problemas de Optimización Combinatoria

En la literatura relacionada con clasificación de GCP, se pueden distinguir tres etapas históricas de acuerdo con las principales contribuciones de las publicaciones. Fig. 2.1 muestra una línea de tiempo de las contribuciones más importantes relacionadas con este trabajo. Se muestran tres etapas históricas de arriba a abajo: investigaciones desarrolladas en la década de 1970, trabajos a fines de la década de 1990-principios de 2000 y 2010-presente.

Primera etapa (1973-1979).- En la primera etapa, se identifican cuatro trabajos importantes [11, 12, 13, 14], que tenían como objetivo definir la base teórica y matemática de los GCP.

La primera investigación relacionada con este trabajo es una revisión de teoría combinatoria realizada por [11]. Su trabajo mencionó temas relacionados con la teoría de

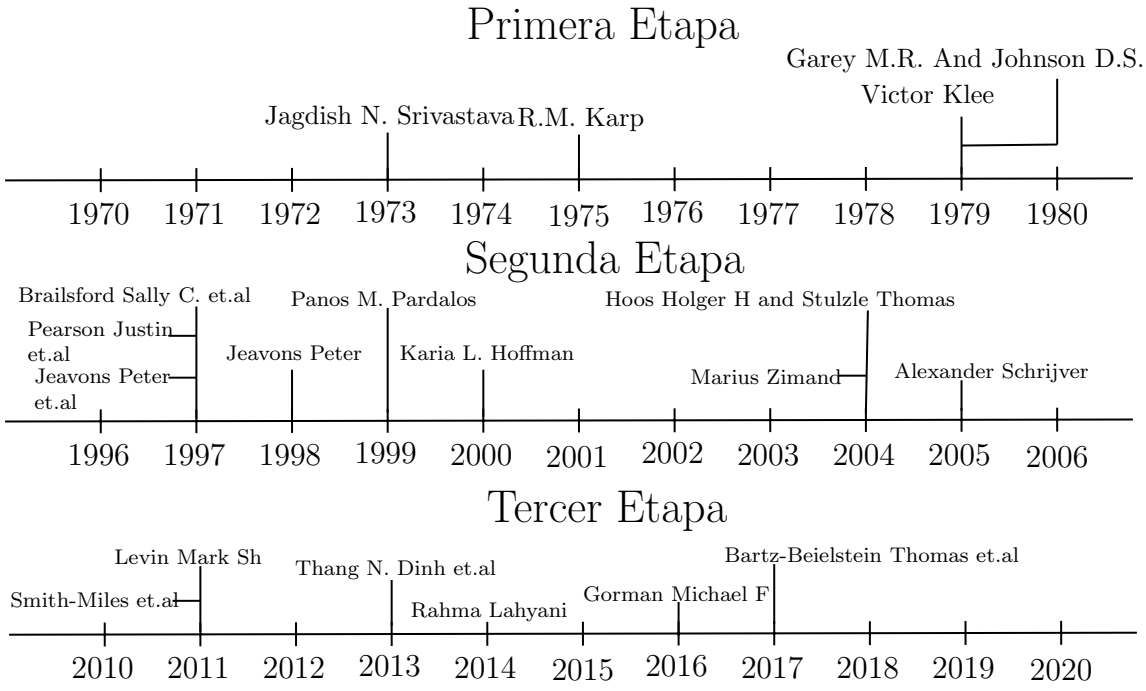


Figura 2.1: Línea del tiempo del estado del arte sobre clasificación de GCP.

grafos, cuadrados latinos, bloques de diseño y su construcción, particiones, lattice, problemas de caracterización del *combinatorial graph*, problemas de búsqueda combinatoria, problemas de geometría combinatoria y una resumen de problemas de enumeración gráfica. Esta investigación proporcionó una base para una comprensión teórica y matemática de los problemas combinatorios.

Karp definió la complejidad computacional de algunos problemas combinatorios como Coloreo de Grafos *, SAT, Clique, 3-Dimensional Matching, Cubic Hamiltonian Circuit, Graph Partition*, entre otros [12].

Garey & Johnson publicó uno de los trabajos más importantes en el área de la informática sobre la teoría de la complejidad NP [13] donde definieron varios de los GCP incluidos en nuestra revisión.

En el mismo año, [14] definió el término *optimización combinatoria*, proporcionó el hito de la teoría y revisó los avances más significativos hasta esa fecha y las posibles direcciones del área de investigación.

Segunda Etapa (1997-2005).- La segunda etapa se caracteriza por contener las primeras propuestas de clasificaciones CSP y trabajos basados en encuestas de GCP. La investigación realizada por [15, 8, 1] analizó los diferentes problemas combinatorios en términos de estructuras algebraicas, formando la base para una clasificación.

Brailsford propuso una clasificación de CSP como *Location, Scheduling, Car Sequencing, Cutting Stock, Timetabling* y *Rostering* [16]. Su investigación sirvió como el punto de partida de la agrupación de CSP para la revisión presentada en este trabajo para el Problema de Satisfacción de Restricción (CSP) particular.

A finales de los noventa, se publicó la primera edición del *Handbook* de optimización combinatoria [17], con la revisión teórica y técnica de algunos problemas combinatorios.

A partir del año 2000, el estudio de los problemas de optimización combinatoria ha generado un interés considerable. Los investigadores han centrado su atención en la revisión de la heurística, metaheurística y otras técnicas capaces de proporcionar soluciones a problemas combinatorios. En esta etapa, hubo algunos estudios orientados a la clasificación y definición de problemas combinatorios.

Hoffman revisó los desarrollos de la época y los tópicos de futuras investigaciones [18]. En [19] se dedicó un capítulo a los problemas combinatorios, su complejidad computacional, paradigmas y trabajos relacionados. En el mismo año, [20] publicó un capítulo de libro centrado en la teoría de optimización combinatoria, que enriqueció la teoría relacionada con los problemas combinatorios en ese momento.

Finalmente, [21] elaboró una revisión histórica de la optimización combinatoria hasta 1960. Mencionó que la optimización combinatoria, como disciplina matemática, disciplina es relativamente joven porque sus inicios datan de una vez que se estableció la programación lineal en la década de 1950.

Tercera Etapa (2010-presente).- En la tercera etapa, se encuentra un número creciente de propuestas para nuevos algoritmos o modelos para resolver diferentes problemas combinatorios. Además, algunas investigaciones se centraron en la generación de una taxonomía para *Rich Vehicle Routing Problems (RVRP).*

En [22] los autores revisaron las características reveladoras de dureza y discutieron las métricas correctas para cada problema. También en 2011, una investigación dirigida por [23], propuso un marco de cuatro capas para resolver problemas de optimización combinatoria. Levin propuso algunos − llamados *modelos básicos* como mochila, clasificación multicriterio, asignación, etc. Sin embargo, su investigación no mostró por qué algunos problemas se consideraban modelos básicos.

En 2013, [24] publicó la segunda edición del*handbook* de optimización combinatoria. En [25] se propuso una taxonomía integral desarrollada para el $RVRP$. En [26], se publicó un *metasurvey* que incluye 343 trabajos de los últimos 15 años en el área de Investigación de Operaciones y *Management Science*. También mencionó que solo había cinco trabajos compilatorios en el área de optimización combinatoria.

La última investigación relacionada es la realizada por T. Bartz-Beielstein y M. Zaefferer [27], que mostraron un resumen de *model-based* métodos, introdujo una taxonomía, examinó problemas de optimización discreta y propusieron un método para combinar enfoques.

2.2. Estado del Arte de Heurísticas de perturbación

La heurística $K-$flip o $K-$opt fue propuesta por Lin, S. y B. Kernighan en [28] para el TSP. Esta heurística se basó en el intercambio general, es decir, una ciudad debe cambiar su posición con otra ciudad en el mismo recorrido. Además, esta heurística es una de las más populares para TSP [29], tiene aplicación en otros problemas como *Planar Graphs, unconstrained binary quadratic programming* y el estudio de su complejidad en SAT y *MAX-SAT*. La two points Perturbation es un caso de $k-$flip, y se da una descripción detallada y algoritmos de estas heurísticas en las siguientes secciones.

La heurística de $k - swap$ es una heurística similar y frecuentemente se confunde con $k - flip$. La heurística $K - swap$ mejora su rendimiento a medida que se mueve cuando usa dos o tres movimientos [30]. La heurística $k-swap$ también se conoce como *Kempe Chain Neighborhood* para el problema de *Course Timetabling* y en *Examination Timetabling* fue llamado *S-chain neighbourhood* [31].

La heurística f también conocida *Minimizing conflicts* fue propuesta por [32]. Esta heurística demostró tener un mejor rendimiento en los problemas de $n-$ reinas que las técnicas tradicionales de retroceso. La heurística *Minimizing conflicts* se ha aplicado a diferentes áreas de la computación como el enfoque hiperheurístico, *Graph Coloring-Problem, Pickup-and-Delivery Problems* y *Scheduling problem*. La heurística *Move to les conflict* es una variante de *First Fit*, la única diferencia es que una toma una variable de forma aleatoria y cambia su valor por otro que genere un menor costo.

El *First-Fit* fue propuesta por Brenda S. Baker [33] para el problema de *Bin-Packing*. Por otro lado, en las últimas décadas, esta heurística se aplicó a los problemas más conocidos como el *virtual machine relocation problem*, y *cuttingstock*. Una notable variedad de heurística es *Worst-fit*, que fue estudiada por Brenda S. Baker [33] y J. Csirik [34], en particular, su aplicación en el problema de *Bin-Packing*.

Soria et al. [35] propuso tres heurísticas para *University Course Timetabling, Best Single Perturbation (BSP) , Statical Dynamic Perturbation (SDP)* y *Double dynamic Perturbation* como parte del conjunto de heurísticas de bajo nivel para Hiperheurísticas. Además, estas heurísticas se aplicaron al problema de VRP en investigaciones

posteriores [36].

2.3. Estado del Arte de Hiper-Heurísticas de selección

Usualmente, las Hiperheurísticas de selección con aprendizaje fuera de línea pertenecen a los métodos de *MachineLearning* [37]. Yates y Keedwell [38] demuestran que se encontraron subsecuencias de heurísticas en una base de datos de aprendizaje fuera de línea de Hiperheurísticas que es efectiva para algunos dominios problemáticos. Utilizaron la *Elman network* para calcular secuencias de heurísticas que se evaluaron en problemas no vistos en $HyFlex$, y los resultados obtenidos son generales e intra-domain learning con 99 % confianza.

Una de las cuestiones cruciales en el diseño de Hiperheurísticas es la calidad y el tamaño del conjunto heurístico de bajo nivel [39]. Soria et al. [39] propuso una metodología que utiliza pruebas estadísticas no paramétricas y mediciones de fitness landscape para el diseño de Hiperheurísticas. Esta metodología se probó en *course timetabling* y *vehicle routing problems*; su propuesta Hiperheurística tenía un conjunto heurístico compacto y competía con algunos métodos tradicionales en *course timetabling*. En el problema de *course timetabling*, obtuvieron cinco soluciones con nuevos óptimos de 24 instancias PATAT.

Finalmente, una investigación reciente de Amaya et al. [40] documentó un modelo para crear constructivamente Hiperheurísticas de selección. Amaya et al. mostró el rendimiento de la Hiperheurística generada cambiar de acuerdo con un número diferente de instancias. La efectividad del modelo propuesto por Amaya depende de los valores deltas utilizados en el órgano rector, teniendo un mejor rendimiento con deltas superiores.

2.4. Meta-learning

La importancia de *meta-learning, matching learning* y optimización han sido estudiados por Song et al. [41]. El objetivo del meta-learning es acumular y adaptar experiencias en el rendimiento de múltiples aplicaciones del aprendizaje de un sistema. El meta-learning también conocido como aprender a aprender [42], ofrece sistemas que pueden ayudar buscando patrones en diferentes tareas para controlar el proceso de exploración de la experiencia acumulada. El concepto de *meta-learning* ha estado presente en el campo de heurísticas y metaheurísticas para el *TSP* [43] , *the Quadratic Assignment Problem*, y *Hyper-heuristics*.

El problema de agente viajero (TSP) ha sido tomado como problema *Benchmark* para diferentes aplicaciones de *meta-learning*. Primero, Kanda et al. [44] propuso un enfoque basado en la calidad de las meta-características que describen las instancias. Además, ellos usaron 4 diferentes conjuntos de meta-características basadas en propiedades de instancias de *TSP* como medidas de nodos y aristas, medidas de complejidad de redes, propiedades de las metaheurísticas y propiedades de *sub-sampling land markers*. La principal contribución de Kanda et al. fue la metodología para resolver detalles computaciones de dos enfoques, calcular las meta-características y combinarlas con el enfoque de *sub-sampling landmarks*.

Por otro lado, Gutiérrez-Rodríguez et al. [43] usa *VRP* con ventanas de tiempo y propuso una metodología basada en meta-aprendizaje para seleccionar la mejor metaheurística para cada instancia. Además, su propuesta compartió y explotó un esquema fuera de línea para las soluciones instantáneas de la academia y la industria. Sus principales contribuciones fueron: proponer un conjunto de características para caracterizar instancias de *VRPTW* y diseñar un proceso de clasificación que prediga la metaheurística más adecuada para cada instancia. Sin embargo, asumieron que las soluciones de las instancias establecidas podrían almacenarse, compartirse y usarse en un esquema fuera de línea para predecir buenos solucionadores para nuevas instancias no vistas.

2.5. Resumen

En este capítulo se da una visión general de estado del arte en clasificación de problema de optimización combinatoria, heurísticas de perturbación, Hiperheurísticas de selección con aprendizaje fuera de línea y trabajos relacionados al *Meta-Learning*.

Los contenidos relevantes para el desarrollo de este proyecto se encuentran divididos en tres áreas. Problemas de optimización combinatoria, heuristicas de estado del arte e Hiperheurística. Adicionalmente como herramienta en la metodología, se habló en este capítulo de meta aprendizaje. La sección de estado del arte nos permitió, identificar al menos tres etapas de desarrollo o enfoque de investigaciones. Dichas etapas fueron evolucionando de acuerdo con la llegada de nuevas técnicas y diferentes formulaciones de problemas. En cuanto a las heuristicas e Hiperheurística, los trabajos realizados se enfocan principalmente en buscar cuál de estos algoritmos da mejores soluciones a problemas específicos, lo que nos permitió identificar un área de oportunidad en cuanto al análisis y selección de la mejor herramienta para familias o grupos de problemas. Finalmente, se da una breve vista a las técnicas usadas con meta aprendizaje.

Capítulo 3

Marco Teórico

En este capítulo se da una vista de los conceptos teóricos a partir de los cuales se basa y desarrolla esta tesis. En esta sección se describe además algunos algoritmos y representaciones de las soluciones usadas en las diferentes instancias, así como los operadores, de las diferentes Metaheurísticas.

3.1. Estructuras Combinatorias

En Comellas et al. [45], nos mencionan que las estructuras combinatorios estudian las relaciones de incidencia entre un conjunto de objetos y ciertos subconjuntos de estos mismos. Otra definición es que estudian de forma sistemática, como se seleccionan objetos según un conjunto de reglas.

3.1.1. Diseños combinatorios

Un tipo de estructura combinatoria es el *diseño combinatorio*, el cual es una tupla formada por $D = (V, B)$, donde $V = v_1, v_2, \ldots, v_y$ llamado conjunto de *variables* y $B = B_i, B_i \subset V$ llamados *bloques* de diseño, que son una familia de subconjuntos de estas variables [45].

La cantidad de variables de un diseño D está dada por $|V| = v$ y el número de sus bloques por la cardinalidad de B, $b = |B|$. En el caso del problema de *University*

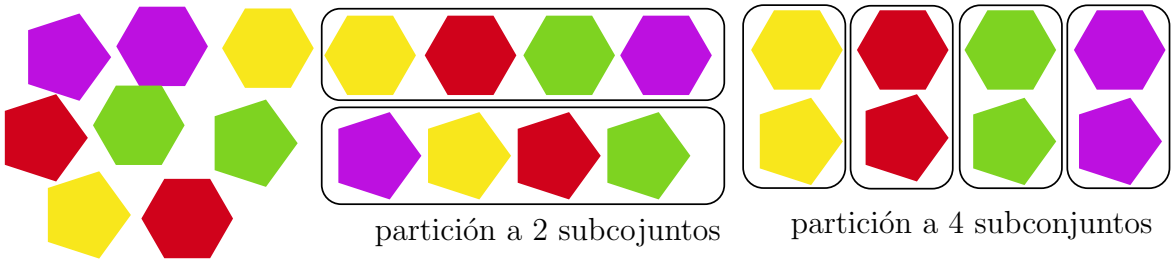


Figura 3.1: Ejemplo de Partición

Timetabling las variables y bloques son: $V = 1, 2, \ldots, n$, donde n es el número de eventos o materias. Y los bloques de diseño son subconjuntos de eventos, **los cuales forman particiones**.

En adelante se utilizará la letra V para denominar a las variables y la letra B para los subconjuntos de estas variables.

3.1.2. Partición

Una partición se define como [46]: Una ***partición*** de un**conjunto *S*** es una colección de subconjuntos **disjuntos dos a dos y no vacíos** de ***S*** tales que su unión es un todo *S*.

Es decir, la colección de subconjuntos A_i, $i \epsilon I$ (donde I es un conjunto de índices) forma un partición de S si, y sólo si, $A_i \neq \emptyset$ para $i \epsilon I$, $A_i \cap A_j = \emptyset$ si $i \neq j$ y $U_{i \epsilon I} A_i = S$ [46]. En la figura 3.1, se muestra un ejemplo de partición.

El trabajo de investigación realizado involucra a dos metodologías de solución de problemas de optimización combinatoria. La primera es la metodología API-Carpio, la cual nos permitió inicialmente hacer uso de las particiones para resolver los problemas de calendarización de horarios, así como considerar las restricciones duras de los problemas que se revolvieron posterior a este. La segunda metodología fue la propuesta por Soria [4], la cual permitió manejar y modelar las diferentes restricciones de los problemas combinatorios. En esta sección, describiremos ambas metodologías usadas para la experimentación con los dos problemas de optimización combinatoria.

3.1.3. Metodología API-Carpio

La metodología API-Carpio, considera dentro de su modelado las restricciones de los Alumnos, Profesores e Institución (tomando cada inicial tenemos el acrónimo API). Dicha metodología, ha sido aplicada desde hace más de 20 años en el diseño de horarios del Tecnológico Nacional de México/Instituto Tecnológico de León. A partir del año 2014, se logró integrar con técnicas metaheurísticas y aplicar de forma exitosa a las carreras de gestión empresarial.

La metodología API-Carpio, tiene como característica fundamental para su diseño de horarios el colocar una materia de cada semestre en diferente horario, es decir, dos materias del mismo semestre no pueden ir en el mismo horario. Lo anterior debido a que a los alumnos se les debe ofertar un horario que les permita tomar todas sus materias correspondientes al semestre que cursan.

Aunque en su aplicación inicial consideraba generar insumos matriciales para los conflictos de aulas y profesores, de forma temporal estas variables y sus restricciones son modeladas en listas para su fácil uso y procesamiento computacional.

Una característica fundamental de la metodología API-Carpio es que la resolución de las diversas variables involucradas usan el concepto de vectores o particiones de materias, estos vectores relacionados con el número de horarios ofertados por la institución a la cual se aplicará la presente metodología contienen las distintas materias que conforman la retícula de alguna carrera específica, estos vectores forman por construcción una base linealmente independiente y una partición del espacio de las materias, ya que al ser conjuntos disjuntos su intersección es vacía y su unión es el universo de materias.

Define formalmente la metodología de calendarización de horarios con la siguiente fórmula [4]:

$$f(x) = FA(x) + FP(x) + FI(x) \tag{3.1}$$

Donde:

$FA(x)$ =Número de estudiantes en conflicto dentro del horario x, *Course Timetabling*.

$FP(x)$ =Número de profesores en conflicto dentro del horario x, *Faculty Timetabling*.

$FP(x)$ =Número de aulas y laboratorios en conflicto dentro del horario x, *Classroom Timetabling*.

Con:

$$FA = \sum_{j=1}^{k} FA_{v_j} \tag{3.2}$$

$$FP = \sum_{j=1}^{k} FP_{v_j} \tag{3.3}$$

$$FI = \sum_{j=1}^{k} FI_{v_j} \tag{3.4}$$

En los cuales:

$FA_{v_j}(x)$ =Número de estudiantes en conflicto dentro del vector v_j.

$FP_{v_j}(x)$ =Número de profesores en dentro del vector v_j.

$FI_{v_j}(x)$ =Número de aulas y laboratorios en conflicto dentro del vector v_j.

Donde a su vez:

$$FA_{v_j} = \sum_{s=1}^{(M_{v_j})-1} \sum_{l=1}^{(M_{v_i})-s} (A_{j,s} \wedge A_{j,s+l}) \tag{3.5}$$

$$FP_{v_j} = \sum_{s=1}^{(M_{v_j})-1} \sum_{l=1}^{(M_{v_i})-s} (P_{j,s} \wedge P_{j,s+l}) \tag{3.6}$$

$$FI_{v_j} = \sum_{s=1}^{(M_{v_j})-1} \sum_{l=1}^{(M_{v_i})-s} (I_{j,s} \wedge I_{j,s+l}) \tag{3.7}$$

Para los que:

$(A_{j,s} \wedge A_{j,s+l})$ = Número de alumnos que demandan la inscripción simultánea de las materias $M_j, s \bigwedge M_{j,s+1}$.

$P_{j,s} \wedge P_{j,s+l}$) = Número de profesores que demandan la impartición de una misma materia M_j, s , como la materia $M_{j,s+1}$.

$I_{j,s} \wedge I_{j,s+l}$) =Número de aulas y laboratorios que pueden dar servicio tanto a la materia M_j, s , como la materia $M_{j,s+1}1$.

M_{Vi} =Número de materias diferentes contenidas en un vector determinado.

3.1.4. Metodología de diseño Soria-Alcaraz

La metodología del diseño propuesta por [5] permite que diferentes políticas del $Course - Timetabling$ y listas de restricciones sean modelados mediante la conversión de todas las restricciones de tiempo y espacio en un simple tipo de restricción: conflictos de estudiantes. En esta se proponen estructuras como lo son la matriz MMA, lista LPH, lista LPA y lista LPS. Las primeras tres representan las restricciones duras y la última representa las restricciones blandas.

La metodología en sus inicios fue enfocada a la construcción de soluciones en el trabajo realizado por Soria en [4]. Posteriormente se extendió su uso y aplicación al ámbito hiperheurístico [35]. Dicha metodología ha sido usada para modelar las diferentes restricciones y su principal potencialidad radica en que se hace la búsqueda de soluciones dentro de un espacio factible, en lugar de obtener soluciones no factibles y tener que corregirlas.

En este trabajo se utilizarán dos de las cuatro estructuras las cuales son: MMA y LPH. En [5] nos definen que sus estructuras son las siguientes:

Matriz MMA: Contiene el número de estudiantes en conflicto entre materias, por ejemplo, el número de conflictos si dos materias son asignadas en el mismo *timeslot*. Esta matriz muestra el número de estudiantes que demandan al mismo tiempo la materia fila con la materia columna. La tabla 3.1 muestra un ejemplo de matriz MMA. Y el algoritmo utilizado para la construcción de esta matriz en el caso de generar instancias artificiales está en 3.1 recuperado de [5].

Tabla 3.1: MMA Matrix

City/Node	N_1	N_2	...	$N_n - 1$	N_n
N_1	**10**	99	91	97	137
N_2	99	**101**	88	80	96
...	91	88	**58**	33	32
$N_n - 1$	97	80	33	**35**	10
N_n	137	96	32	10	**68**

Algoritmo 3.1 Construcción MMA

Entrada: int N Estudiantes, int[][] LD Demanda de Estudiantes

1: **para** $i = 0$ hasta N **hacer**
2: $Starr = LD[i]$
3: **para** $j = 0$ hasta $size(Starr)$ **hacer**
4: **para** $k = j + 1$ hasta $size(Starr)$ **hacer**
5: $MMA[Starr[j]][Starr[k]] += 1$
6: $MMA[Starr[k]][Starr[j]] += 1$
7: **fin para**
8: **fin para**
9: **fin para**
10: **devolver** MMA

Lista LPH: Esta lista nos da información acerca de cada lección (clase, evento o materia) en que posible espacio de tiempo($time - domain$) puede ser asignada. En esta estructura tiene la información correcta del espacio de búsqueda para cada variable. La lista LPH nos muestra en sus filas todos los eventos y en sus columnas son todos los días, cada celda muestra la posible asignación para un evento específico en un determinado día. Un ejemplo de esta lista lo se muestra en la tabla 3.2 obtenido de [5]. Si se desea crear una instancia artificial se puede seguir el algoritmo 3.2.

Tabla 3.2: Lista LPH. $M_1 ... M_n$ representan las materias

Eventos	Timeslots							
M_1	1	2	3	4	5	6	7	8
M_2	1	2	3	4	5		7	8
M_3	1	2	3	4	5	6	7	8
M_4	0							
$\vdots$								
M_{n-1}	1	2	3	4	5	6	7	8
M_n	1	2	3	4	5	6	7	8

Algoritmo 3.2 Construcción LPH

Entrada: int Nm CantidadMaxEventos, int[] SI SolucionInicial

1: **para** $i = 0$ hasta $size(SI)$. **hacer**
2: int $Crandom$= randomEntre(Nm)
3: int [] $lpht$ = nuevo
4: **para** $j = 0$ to $Crandom$ **hacer**
5: $lpht$[j]= radomEntre(Nm)
6: **fin para**
7: $lpht[Nm]= SI$[i]
8: $LPH[i].add(lpht)$
9: **fin para**
10: **devolver** LPH

3.2. Heurísticas de Perturbación

Las heurísticas que se aplicaron a los dos problemas y sus respectivas instancias fueron las siguientes:

k-flip / Simple Random Perturbation(SRP) (H_1) Esta heurística cambia el valor de una o más variables (en algunos casos k) por otro valor que sea válido para esta variable. En Coloreo de Grafos el objetivo de esta heurística, es cambiar el color de un determinado nodo por otro color [47]. Finalmente, para el *Capacita-*

ted Vehicle Routing Problem (CVRP) el movimiento implica que se cambie una ciudad de una ruta a otra ruta [29].

K-Swap/ Kempe Chain Neighbourhood/ S-chain neighborhood (H_2) En esta heurística se seleccionan dos o más variables y se intercambian sus valores entre ellas, respetando que no se viole alguna restricción. Para el Coloreo de Grafos se intercambian los colores entre dos nodos seleccionados. Esta heurística es usado en otras investigaciones como lo son el *TSP*, donde es conocida como *k-interchange* [48] [49].

Best Single Perturbation (BSP) (H_3) Esta heurística elige una variable de acuerdo a la lista dura de restricciones (LPH) y cambia su valor por otro de la lista. El valor que se eligió debe de dar un mejor costo o en el peor de los casos el mismo costo [35]. La lista contiene una memoria, es decir, si la primera vez se eligió la primera variable, en la siguiente llamada o aplicación de esta heurística se seleccionará la segunda variable y así sucesivamente. El siguiente nodo que tiene que cambiar de color se seleccionará de acuerdo con la última variable elegida, para el Coloreo de Grafos *(GC)*. Para *CVRP* se cambiará la ciudad de camión a otra de forma similar en cómo se hizo en el problema anterior.

Statical Dynamic Perturbation (SDP) (H_4) Basada en una distribución de probabilidad de las frecuencias de la selección de una variable en las últimas k iteraciones, esta heurística elige una variable y cambia el valor por otro seleccionado aleatoriamente de sus posibles valores [35]. Las variables con menores cambios tendrán una probabilidad más alta de ser seleccionados. Aplicando esta Heurística al GC el nodo con menor cantidad de cambio de colores es el que tendrá mayor probabilidad de ser seleccionado, mientras que para el $CVRP$ será la ciudad con menos movimientos de camión.

Two Points Perturbation (2pp) (H_5) Esta heurística también conocida como $k-opt$ es un caso particular con un $k = 2$.

Double dynamic Perturbation (DDP)(H_6) Esta heurística está basada en la SDP, en la cual recibe una solución y modifica el valor de la variable de acuerdo a la distribución de probabilidad. La diferencia entre esta y SDP, es que copia la solución inicial y al final se regresa la mejor solución entre la inicial y la modificada [35].

Move to less conflict (MLC)(H_7) En esta heurística se selecciona una variable de forma aleatoria y se le asigna el valor que genera el menor costo [36]. Para GC, el color cambia de acuerdo con otro que mejore el resultado (no tener dos nodos adyacentes con el mismo color) y para el $CVRP$ se mueve la ciudad al camión en el cual la distancia del recorrido sea minimizada.

Min-Conflicts (H_8) Selecciona una variable y le asigna el valor que tiene le menor costo [36]. Por ejemplo, para el GC debe cambiarse el color de un nodo por otro que mejore el costo. Para el $CVRP$ se selecciona una ciudad en la cual la distancia del tour mejora.

First-Fit (H_9) Cambie el valor de la variable a otra, que es la menos repetida en otras variables [36], i.e. en VRP se tomará una ciudad y la se cambia al camión que tiene menos ciudades en su ruta. Para Coloreo de Grafos, se selecciona un nodo y se le asigna el color que menos se repite.

Worse-fit (H_10) Asigna el valor más repetido si es posible, sin violar las restricciones duras a una variable seleccionada aleatoriamente [50]. Para GCP and $CVRP$, se le asigna a un nodo o ciudad al color o camión más repetido.

Burke-Abdullah (BA) (H_11) Esta heurística fue propuesta por Abdullah et al. [51], en el que elige una variable que aplicando $Fail-First$ or $Brelaz\ Heuristic$ [52] y su valor cambia de acuerdo con el que ha obtenido un mejor rendimiento mediante la aplicación de los siguientes algoritmos: *Minimum conflict, Randomly, Sequential selection* y *least Constrained*.

3.3. Hiperheurísticas con aprendizaje fuera de línea con heurísticas de perturbación

El algoritmo de búsqueda local iterada (ILS por sus siglas en inglés) fue usado como órgano rector. Esta metaheurística fue propuesta por Lourenço et al. [53], la cual va construyendo o modificando una solución mediante una heurística embebida. Las soluciones generadas son mejores, a si se generan o modifican aleatoriamente. La esencia de este algoritmo es intensificar la solución inicial, explorando vecinos de este. El algoritmo del ILS

se muestra en 3.3, tomado de Talbi El-Ghazali [54].

Algoritmo 3.3 High Level Iterated Local Search (ILS)

$s_0 = GenerateInitialSolution$
$s^* = ImprovementStage(s_0)$
mientras !$StopCriteria()$ **hacer**
  $s' = SimpleRandomPerturbation(s^*)$
  $s^{*'} = ImprovementStage(s')$
  si $f(s^{*'}) < f(s^*)$ **entonces**
    $s^* = s^{*'}$
  fin si
fin mientras
devolver s^*

Algoritmo 3.4 Improvement Stage

$ls \leftarrow IncumbentSolution$
mientras !$LocalStopCriteria()$ **hacer**
  $h_i = Perturbate()$
  $ls^* = apply(h_i, ls)$
  si $f(ls^*) < f(ls)$ **entonces**
    $ls = ls^{*'}$
  fin si
fin mientras
devolver ls

3.3.1. Operador de Selección

Para la fase de perturbación (paso 4 en el algoritmo) se elige una variable que sigue una distribución de probabilidad basada en la frecuencia de selección de variables en las últimas iteraciones k. Esta simple heurística nos permite hacer una modificación en la solución

Se uso el mismo *Framework* Hiperheurístico y de acuerdo con cada clase, se da un grupo diferente de heurísticas de bajo nivel. Por ejemplo, si para la clase 1 las mejores heurísticas fueron H_1, H_8, H_10 y H_5, estas fueron nuestro grupo para la Hiperheurística, y se realizó un diseño de procedimiento similar para cada clase. Después de este proceso, se entrena a cada Hiperheurística con sus respectivas clases, para las siguientes etapas.

3.4. Metalearning

El concepto de *Meta-learning*, es también conocida como "Aprendiendo a aprender" [55]. La diferencia principal entre *Base-learning* y *Meta-learning*, está en el alcance del nivel de adaptación; mientras que el aprendizaje a nivel básico está enfocado en acumular experiencia en especificas tareas de aprendizaje, el aprendizaje a meta nivel está enfocado a acumular experiencias en el desempeño de múltiples aplicaciones de sistemas de aprendizaje [55].

Los pasos básicos que debe tener el meta aprendizaje son:

- Revisar la información importante del problema.
- Aprender de la información.
- Aplicar lo aprendido para solucionar el problema.
- Revisar lo aprendido.
- Aprender de lo aprendido.
- Aplicar el aprendizaje para mejorar lo aprendido.

Otro concepto clave en esta investigación es el de meta-Knowledge o metaconocimiento, el cual puede ser definido como un tipo de conocimiento que es derivado en el curso de la aplicación de un sistema de aprendizaje dado [55].

3.5. Clasificador

El clasificador *Naive Bayes* está basado en probabilidad y la aplicación de los teoremas de Bayes, que asumen una fuerte independencia de los datos [56]. Este clasificador de enfoque supervisando puede ser entrenado eficientemente, siempre y cuando se tenga un modelo ideal de probabilidad de los datos. A continuación, se menciona algunas ventajas y desventajas del *Naive Bayes* [57]:

- Ventajas: la complejidad computacional es baja en comparación con los árboles de decisión. Se adapta al modelo de un contexto de aprendizaje de forma incremental y resiste a los atributos irrelevantes.
- Desventajas: Tiene dificultades con modelos simplificados como la XOR.

3.6. Resumen

En este capítulo se abordó en marco teórico con el cual se basa esta tesis.

Primero se revisó lo que son las estructuras combinatorias y su relación con las particiones. Posteriormente se habló de lo que son las heuristicas de perturbación, así como la descripción de todas las utilizadas en este trabajo de tesis. Como parte importante se describió un poco acerca del enfoque hiperheurístico con aprendizaje fuera de línea. Finalmente, de forma breve se explicó lo que es el meta aprendizaje e información del clasificador usado. Cabe mencionar que no se extendió en el tema de meta aprendizaje y clasificador, ya que fueron herramientas adicionales usadas en la metodología.

Capítulo 4

Problemas de Optimización Combinatoria (GCP)

Existen diferentes formas de definir un problema de optimización. Algunas de ellos son modelos de programación matemática, modelos no analíticos y problemas de optimización combinatoria [54]. De estos problemas combinatorios de acuerdo por la naturaleza que tienen los valores que toman sus variables el caso continuo y discreto. Este trabajo se enfocó a resolver problemas de optimización combinatoria discretos, esto debido a que se tiene un especial interés en resolver aquellos que sean susceptibles a ser resueltos por particiones y sujetos a restricciones.

Es importante mencionar que, dentro de la línea de investigación de este proyecto, se han podido obtener insumos reales de un tipo de estos problemas y aplicar las técnicas de inteligencia artificial para su solución. Dentro de los problemas discretos,

Dentro de los problemas discretos, tenemos los enfocados a optimización y dividirlos por su naturaleza en dos categorías: continuo (se puede pedir encontrar un conjunto de números reales para una función real, los cuales deben de optimizarla bajo algunas restricciones) y discretos (en los cuales el objetivo es encontrar una combinación de objetos dentro de un espacio finito de soluciones factibles, que optimicen una función en algunos casos sujeta a restricciones) [58].

El problema general combinatorio (GCP), el cual plantea la pregunta de saber si existe una solución tal que considere todas las variables y sus respectivas restricciones.

El GCP está definido en términos algebraicos por dos estructuras relacionales finitas denotadas en [1] como:

$$\langle S_1 \text{ y } S_2 \rangle$$

Donde S se define como:

$$S = \langle V, E_i(i \epsilon I) \rangle$$

Dónde:

V es un conjunto no vacío de elementos. Es el *universo* de elementos de la estructura relacional.

E_i es un sistema de relaciones finitas sobre V, indexado por los elementos de I.

I es un conjunto de elementos que permiten relacionar cuales elementos de V están incluidos en E_i.

Si denotamos una instancia de GCP como $P = \langle S_1, S_2 \rangle$, una solución a P es un homomorfismo de S_1 a S_2.

Algunos problemas de *General Combinatorial Problem* de estado del arte mencionados en [8] son:

1. Coloreo de Grafos
2. *Clique*
3. *Vertex Cover*
4. *K-Dimensional Matching*
5. *Hamiltonian Circuit* y *TSP*
6. *Bandwidth*
7. *Graph Isomorphism*
8. *Undirected Graph Reachability*

9. *Satisfiability*

10. *CSP (Constraint Satisfaction Problem)*

Cada uno de los problemas combinatorios mencionados tiene un primer conjunto S_1 que incluye los subconjuntos de variables y sus restricciones. El conjunto S_2 representa las combinaciones de valores permitidas para el subconjunto de variables.

4.1. Coloreo de Grafos

4.1.1. Definición

Un Problema Coloreo de Grafos *(GC)* está formado por un grafo G vértices y q de colores. De tal forma que se busca etiquetar cada uno de los G vértices con q colores, considerando la restricción de que dos vértices adyacentes tienen que ser de diferente color. En la figura 4.1 (fuente elaboración propia), se muestra un ejemplo con cinco colores y 15 vértices. Cada vértice está etiquetado con un color diferente y los vértices adyacentes cumplen la regla de no repetir dicho color.

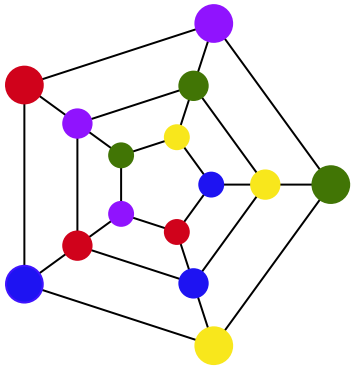


Figura 4.1: Ejemplo de Coloreo de Grafos

En términos algebraicos una instancia de GC como $\langle G, K_q \rangle$, donde k_q es un grafo completo en q vértices[1].

4.1.2. Historia

Este problema tiene orígenes desde 1852 cuando Morgan informa mediante una carta a Halmiton, que uno de sus estudiantes ha planteado una tarea de colorear los diferentes condados de Inglaterra con diferentes colores, considerando que los condados vecinos no estén con el mismo tono [58]. La demostración de que el coloreo de grafos o también conocido como "*vertex-coloring*" es un problema NP-Duro está en [59]

4.1.3. Tipos de problemas de Coloreo de Grafos

En estado del arte existen diferentes artículos que describen los diferentes problemas de coloreo de grafos [59, 60, 61, 62], así como sus aplicaciones en ingeniería electrónica, ciencias de la computación, *Operations Research* y matemática aplicada [63]. En [64] , describen la complejidad computación de diferentes problemas de coloreo de grafos. Algunos de estos problemas de coloreo de grafos son:

- *Classical coloring* [65].
- *Equitable coloring* [66, 67].
- *Chromatic sum coloring* [68].
- *T-coloring* [69].
- *Rank coloring.*
- *Harmonious coloring* [70].
- *Interval coloring* [71].
- *Circular coloring* [72].
- *Path coloring* [73].
- *List coloring* [74].

En las tablas A.1 y A.2 del anexo A, se muestra la información de diferentes instancias de prueba del problema de coloreo de grafos.

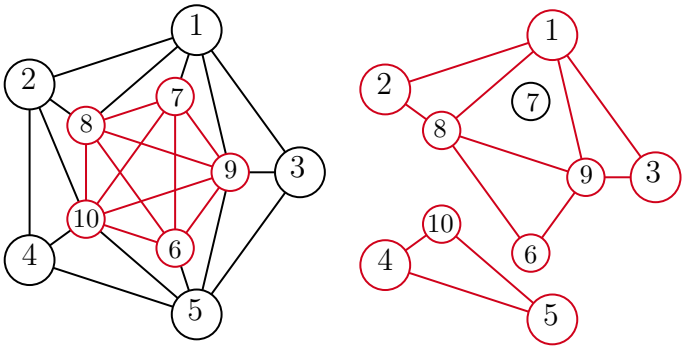


Figura 4.2: Clique ejemplo

4.2. Clique

4.2.1. Definición

Una instancia de *clique* consiste en un grafo G y un entero q en el que busca conocer cuántos subgrafos existen dentro de G con q vértices que sean cliques (este es isomorfo a un grafo completo K_q). Se puede expresar como una instancia de GCP (K_q, G), si G no contiene bucles (vértices adyacentes consigo mismos) [1].

En otras palabras, un "*clique*" es un subgrafo de G en el cual todos sus vértices están conectados entre sí. En la figura 4.2 (fuente elaboración propia) a tenemos un grafo de 10 vértices de los cuales cinco forman el subgrafo clique. Este clique está compuesto por los nodos 6, 7, 8, 9 y 10 los cuales pueden verse en color rojo; los nodos restantes no pertenecen al clique ya que por el vértice 1 solo tiene conexión con los nodos 1 y 3. En la figura 4.2 b, tenemos un grafo similar con 10 nodos, pero este tiene cinco clique compuesto por tres vértices cada uno. El primero está compuesto por los nodos 1, 2 y 8; el segundo por el 1, 3 y 9; el tercero por 8, 9 y 6; el cuarto por 4, 5 y 10 y el quinto por 1, 8 y 9.

4.2.2. Historia

El problema del clique se encuentra clasificado en la literatura como un problema *NP* [75]. Inicialmente el uso de cliques se da en el área de la sociología como método

para el análisis de estructuras de grupos en 1949 [76] y en 1957 se propone el primer algoritmo por Harary y Ross [77].

4.2.3. Problemas de clique

En las investigaciones hechas por Pardalos et al. y Bomze et al. [78, 77] , nos definen algunos problemas de clique, los cuales son:

1. Maximal clique problem [77].

2. Máximum clique problem [79].

3. Weighted máximum clique [80].

En las tablas A.2 y A.3, se muestra la información de diferentes instancias de prueba del problema de clique.

4.3. Vertex Cover Problem

4.3.1. Definición

Una instancia de Vertex Cover consiste en un grafo $G = \langle V, E\rangle$ y un número entero k . Entonces se busca si existe un subconjunto $V' \subset V$ con $|V'| \leq k$ para cualquier $\langle v, w\rangle \ \epsilon E$ ya sea que $v \epsilon V'$ o $w \epsilon V'$ [1]. Es decir, en un grafo completo se busca conocer el subconjunto de vértices que cubra todas las aristas del grafo.

En la figura 4.3(fuente elaboración propia), se muestra un ejemplo de vertex cover, donde se tienen diez nodos y 20 aristas. Un subconjunto de nodos que cubren todas las aristas es $V' = 1, 3, 4, 6, 7, 8, 10$, estos están marcados con color verde en la figura 4.3.

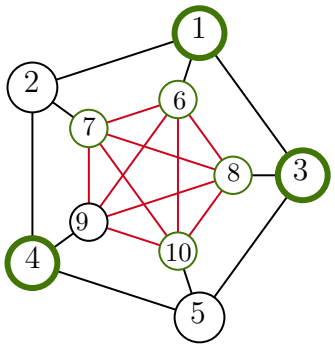


Figura 4.3: Ejemplo de *Vertex Cover*

4.3.2. Historia

Este problema es considerado un problema NP−Completo en [75], teniendo sus primeras menciones en [81], donde formulan la pregunta de cuantos son los vértices mínimos para cubrir todas las aristas de un grafo. El problema generalizado fue introducido en [82, 83] por Hassin y Levin.

4.3.3. Problemas de Vertex Cover

El Vertex Cover problema tiene diferentes variantes en el estado del arte, alunas de ellas son mencionadas por Guo et al. en [84], las cuales son:

- *Connected Vertex Cover*: Dado un grafo G, buscar un vertex cover tal que los nodos estén conectados.
- *Tree Cover*: Dentro de un grafo con aristas ponderadas, buscar si existe un subgrafo que sea vertex cover y que además sea un árbol.
- *Tour Cover*: Dentro de un grafo con aristas ponderadas, buscar si existe un subgrafo que sea vertex cover y que sea un tour.
- *Capacitated Vertex Cover*: Dentro de un grafo con aristas ponderadas, buscar si existe un vertex−weighted tal que su capacidad sea menor que la del grafo inicial

- *Soft Capacitated Vertex Cover*
- *Hard Capacitated Vertex Cover*
- *Maximum Partial Vertex Cover*: Dentro un grafo G, buscar un vertex cover tal que cubra una cantidad n de aristas.
- *Minimum Partial Vertex Cover*: Dentro un grafo G, buscar un vertex cover tal que cubra al menos una cantidad n de aristas.

En la tabla A.2 , se muestra la información de diferentes instancias de prueba del problema de *vertex Cover*, como lo es el nombre, número de instancias de cada paquete, el número de vértices por instancia y el número de vértices mínimo que cubre todas las aristas para esa instancia .

4.4. K-Dimensional Matching

4.4.1. Definición

El problema de *k-dimesional Mathing* también conocido como *3DM (3-Dimensional Matching)*, consiste en que dada un conjunto $M \subseteq W \times Y \times Z$, donde W, X y Y son conjuntos disjuntos que tienen el mismo número q de elementos, se busca que M contenga un *matching*, el cual es un subconjunto $M' \subseteq M$ tal que $|M'| = q$ y ninguna coordenada de dos elementos de M' coinciden[13].

Ahora bien puede ser definida como una instancia deGCP de tal forma que $\langle\langle V, \Diamond_V\rangle, \langle M, \Diamond_M^{\sim}\rangle\rangle$ donde $\Diamond_M^{\sim} = \langle\langle v_1, v_2, \ldots, v_k\rangle, \langle v_1', v_2', \ldots, v_k'\rangle\rangle \in M^2 | v_i \neq v_i', i = 1, 2, \ldots, k$ [1].

En la figura 4.4, se muestra un ejemplo de *3-Dimensional Matching*, donde tenemos tres subconjuntos de puntos, $W = 1, 2, 3, 4, 5$ (color verde), $X = 1, 2, 3, 4$ (color morado), y $Y = 1, 2, 3, 4$(color rojo).

El producto $M = \{(1, 1, 1), (2, 2, 2), (3, 3, 3), (4, 4, 4), (5, 4, 3)(4, 3, 2), (2, 1, 2)\}$ representados en la figura 4.4 A) (fuente elaboración propia), sombreados con color gris y lila la coordenada.

La parte b de la figura 4.4, se puede ver un ejemplo de M' que contiene a las coordenadas $M' = (1,1,1),(5,4,3)y(4,3,2)$ y estas forman un *3-Dimensional Matching*, ya que $w_1 \neq w_2$, $x_1 \neq x_2$ y $y_1 \neq y_2$.

Otros ejemplos de *3-Dimensional Matching* son $M'' = \{(1,1,1),(2,2,2),(3,3,3)(4,4,4)\}$ (ver figura 4.4 C) y $M''' = \{(2,1,2),(3,3,3)(4,4,4)\}$ (ver figura 4.4 D).

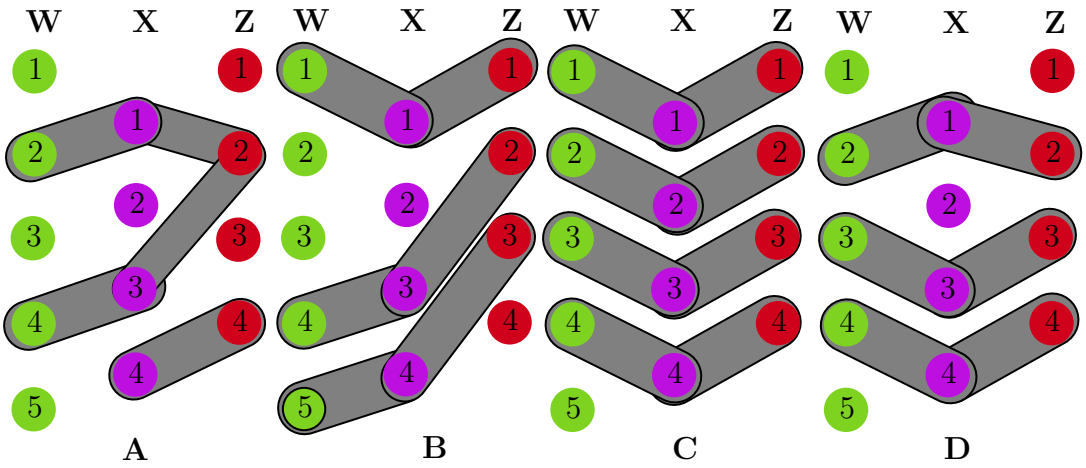


Figura 4.4: Ejemplo de *3-Dimensional Matching*

4.4.2. Historia

Inicialmente el *3-Diemensional Matching* es mencionado por Garey y Johnson, como uno de los seis problemas básico NP-Completos [13]. Posteriormente Hazan et al. en [85], mencionan el caso generalizado del *k-Dimesional Matching* y visto como un *Max SNP-Complete* Kann Viggo en [86] y en trabajos recientes continúan trabando este problema con diferentes técnicas de inteligencia artificial [87].

4.5. Hamiltonian Circuit

4.5.1. Definición

Hamiltonian Circuit

Un circuito Hamiltoniano consiste en dado un grafo G con un conjunto de vértices V y aristas E, se busca un *camino* que pasa solo una vez visitar todos los vértices del grafo G. De manera formal el problema de circuito Hamiltoniano denotado como $G = (V, E)$, se busca un ciclo ordenado de V para cada par sucesivo de nodos es adyacente en G [13].

Expresado como una instancia de GCP $\langle\langle V, C_V, \Diamond_V\rangle, \langle V, E, \Diamond_V\rangle\rangle$, donde C_V es una permutación cíclica arbitraica en V y $\Diamond_V$ es una relación desigual sobre V [1].

TSP

Dado un conjunto de ciudades y el costo de los viajes entre cada par de ellos, el problema de vendedor ambulante, o TSP en inglés para abreviar (Travel Salesman Problem), es encontrar la forma más barata de visitar todas las ciudades y volver a su punto de partida. En la versión estándar, los costos de viaje son simétricos en el sentido de que viajar de ciudad X a ciudad Y cuesta tanto como viajar de Y a X.

La simplicidad de la enunciación del problema es engañosa: el TSP es uno de los problemas más intensamente estudiados en matemáticas computacionales y, sin embargo, no se conoce un método de solución efectivo para el caso general. Se dice que la resolución del TSP solucionaría el problema de P contra NP y obtendría un premio de $\$1,000,000$ del *Clay Mathematics Institute*.

Un agente viajero debe realizar un recorrido partiendo de una ciudad de origen, pasando por cada una de las ciudades una sola vez, y regresando a la ciudad de origen. Se desea encontrar el recorrido de distancia mínima. Este problema es de compleja formulación matemática, la propuesta por Dantzig, Fulkerson y Johnson [88], que para algunos autores es una de las más aceptadas por la comunidad académica y consiste en:

Dado un conjunto de n ciudades, $V = 1, 2, 3, ..., n$, y un conjunto de arcos $(i, j) \in A$ uniendo cada una de las ciudades, donde A es el espacio de búsqueda. Si C_{ij} es la

distancia para ir de la ciudad i a la ciudad j donde $C_{ij} = C_{ji}$ en el caso simétrico y X_{ij} es la variable de decisión del problema es:

Sea X_{ij} la variable de decisión del problema

$$X_{ij} = \begin{cases} 1 & si \quad el \quad arco \quad (i,j) \quad es \quad usado \quad para \quad hacer \quad el \quad tour \\ 0 & en \quad caso \quad contrario \end{cases} \tag{4.1}$$

Así, el modelo matemático asume la siguiente forma:

$$minD(\vec{X}) = \sum_{(i,j)\in A} C_{ij}X_{ij} \tag{4.2}$$

$$\sum_{\{i:(i,j)\in A\}} X_{ij} = 1 \quad \forall j \in V \tag{4.3}$$

$$\sum_{\{j:(i,j)\in A\}} X_{ij} = 1 \quad \forall i \in V \tag{4.4}$$

$$\sum_{\{(i,j)\in A:\quad i\in U,\quad j\in(V-U)\}} X_{ij} = 1 \quad 2 \leq |U| \leq |V| - 2 \tag{4.5}$$

Donde A es el espacio de búsqueda, C_{ij} es el peso de la arista o la distancia asociada a X_{ij}, la ecuación 4.2 corresponde al cálculo de la función objetivo. Donde $\vec{X}$ es una variable vectorial que a su vez es un vector matricial, es decir cada valor de $\vec{X}$ es una ruta distinta, por lo tanto en la función $D(\vec{X})$ se busca el argumento mínimo de todas las rutas.

La restricción 4.3 indica que se puede llegar a cada ciudad desde una única ciudad anterior. La restricción 4.4 indica que desde la ciudad i se puede pasar a una única ciudad (de la ciudad i se puede salir por un único camino). La restricción 4.5 evita que se generen subrutas, donde U es el conjunto que contiene el vértice i del cual se parte. Se le resta U a V, para evitar que se elija así mismo como el siguiente nodo a visitar, evitando así bucles.

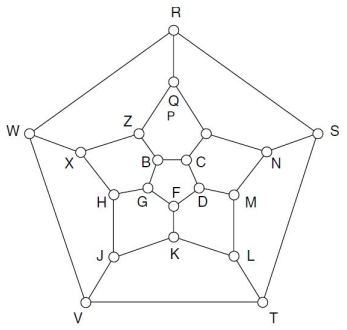


Figura 4.5: Ejemplo de Icosian

4.5.2. Historia

Hamiltonian Circuit

Este problema tiene sus orígenes desde 1850, llamado así por el irlandés Sir William Rowan Hamilton. Hamilton comenzó estudiando la forma de visitar los 20 nodos de un dodecaedron y para ello generó una abstracción gráfica llamada icosian . En la figura 4.5 se muestra una imagen del icosian (recuperado de [89]), donde las aristas del Icosian representan los bordes geométricos y los círculos representan las esquinas.

Hamilton propone primero generar un camino compuesto por cinco vértices y mediante diferentes cálculos demuestra que no importa cual camino se elijan como inicial, siempre es posible completar un tour a lo largo de los vértices restantes del Icosian.

Pero fue hasta 1976 que la NP-Completes fue demostrada por Garey y Johnson en su artículo "*The planar Hamiltonian Circuit problem is NP-Complete* "[90] . Para estado del arte y trabajos relacionados se puede encontrar en los siguientes investigaciones: [91], [92], [93], [94].

TSP

Los problemas matemáticos relacionados con el problema del vendedor ambulante fueron tratados en los años 1800 por el matemático irlandés Sir William Rowan Hamilton y por el matemático británico Thomas Penyngton Kirkman. Una buena discusión

de los primeros trabajos de Hamilton y Kirkman se puede encontrar en [95].

La forma general del TSP parece haber sido estudiada por primera vez por los matemáticos a partir de la década de 1930 por Karl Menger en Viena y Harvard. El problema fue promovido posteriormente por Hassler Whitney y Merrill Flood en Princeton. Un tratamiento detallado de la conexión entre Menger y Whitney, y el crecimiento de la TSP como un tema de estudio se puede encontrar en el documento de Alexander Schrijver "*Sobre la historia de la optimización combinatoria (hasta 1960)*".

4.5.3. Tipos de problemas

Hamiltonian Circuit

Encontramos los Circuitos hamiltonianos (HC), y dentro de estos se encuentran los *Hamiltonian-Connected From a vertex (HCV)*, se pueden ver dos subconjuntos uno de *hamitonian-Connected* y los *Uniquely Hamiltonian-Connected From a vertex*, en su intersección se dan 3 tipos específicos similares a los "*Uniquely Hamiltonian-Connected From a vertex* "definidos en [96].

TSP

Algunos de estos problemas de TSP son:

- Simétrico
- Asimétrico
- *VRP (Vehicle Routing Problem)*
    - *Capacitated VRP*
    - *Capacitated VRP with Time Windows*
    - *Capacitated VRP with Pick-up and Deliveries and Time Windows*
    - *Multiple Depot VRP*
    - *Multiple Depot VRP with Time Windows*

- *Periodic VRP*
- *Periodic VRP with Time Windows*
- *Vehicle Routing Problem with Pick-up and Deliveries*

En la tabla A.1 , se muestra la información de diferentes instancias de prueba del problema de *Hamiltonian Circuit*, como lo es el nombre, el número de vértices por instancia y el número de aristas para esa instancia. En la tabla A.4, se muestra la información de diferentes instancias de prueba de los problemas de *TSP Y VRP*.

4.6. Subgraph Isomorphism y Graph Isomorphism

4.6.1. Definición

Subgraph isomorphism

El problema de *Subgraph Isomorphism* consiste en que dado dos grafos finitos A y B, determinar si A contiene un subgrafo tal que es isomórfico a B.

Formalmente se puede definir como dado dos grafos $G_1 = (V_1, E_1)$ y $G_2 = (V_2, E_2)$, se busca encontrar un subgrafo de G_1 que sea isomórfico a G_2. Además de que los subconjuntos de $V' \subseteq V_1$ y $E' \subseteq E_1$ cumplen con $|V'| = |V_2|$, $|E'| = |E_2|$ y existe una función uno a uno $f : V_2 \longrightarrow V'$ que satisface $\{u, v\} \in E_2$ si y solo si $\{f(u), f(v)\} \in E'$ [75].

Un ejemplo de *Subgraph Isomorphism*, se puede ver en la figura 4.6, donde el grafo G está conformado por los vértices $V_1 = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$ y las aristas se pueden ver en la matriz de adyacencias de la tabla 4.1. El grafo H está conformado por los vértices $V_2 = \{A, B, C, D, E, F, G\}$ y las aristas están representadas en la matriz de adyacencias de la tabla .

	1	2	3	4	5	6	7	8	9	10
1	0	1	0	0	0	1	0	0	0	0
2	1	0	1	0	0	1	0	1	0	0
3	0	1	0	1	1	0	0	0	0	0
4	0	0	1	0	1	0	0	0	1	1
5	0	0	1	1	0	1	0	0	0	0
6	1	1	0	0	1	0	1	0	0	0
7	0	0	0	0	0	1	0	1	0	0
8	0	1	0	0	0	0	1	0	0	0
9	0	0	0	1	0	0	0	0	0	1
10	0	0	0	1	0	0	0	0	1	0

Tabla 4.1: Matriz de Adyacencias para el grafo G

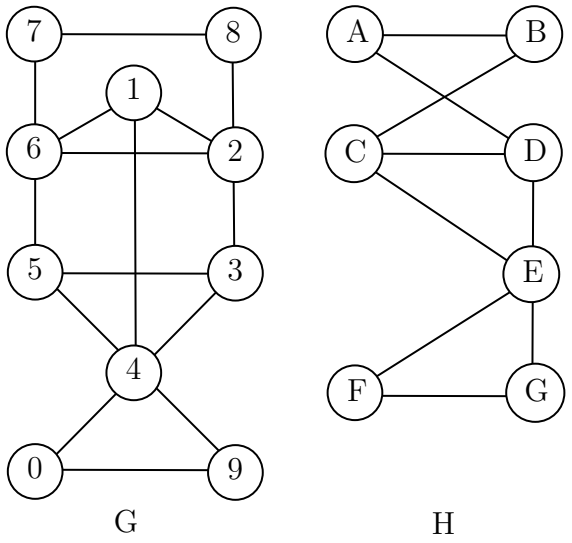


Figura 4.6: Ejemplo de *Subgraph Isomorphism*

El subgrafo G' conformado por los vértices $v = \{2, 3, 4, 5, 6, 9, 10\}$ es isomorfo a H, ya que si se tiene la siguiente función f tal que $f(2) = A$, $f(3) = D$, $f(4) = E$, $f(5) = C$, $f(6) = B$, $f(9) = G$ y $f(10) = F$ existen las conexiones ente las aristas de E' (ver tabla 4.3).

	A	B	C	D	E	F	G
A	0	1	0	1	0	0	0
B	1	0	1	0	0	0	0
C	0	1	0	1	1	0	0
D	1	0	1	0	1	0	0
E	0	0	1	1	0	1	1
F	0	0	0	0	1	0	1
G	0	0	0	0	1	1	0

Tabla 4.2: Matriz de Adyacencias para el grafo H

		A	B	C	D	E	F	G
		$f(2)$	$f(6)$	$f(5)$	$f(3)$	$f(4)$	$f(10)$	$f(9)$
A	$f(2)$	0	1	0	1	0	0	0
B	$f(6)$	1	0	1	0	0	0	0
C	$f(5)$	0	1	0	1	1	0	0
D	$f(3)$	1	0	1	0	1	0	0
E	$f(4)$	0	0	1	1	0	1	1
F	$f(10)$	0	0	0	0	1	0	1
G	$f(9)$	0	0	0	0	1	1	0

Tabla 4.3: Matriz de Adyacencias para el subgrafo G'

Graph Isomorphism

En términos general el problema de grafo isomorfo consiste en decidir si dos grafos finitos son isomorfos, buscando si es que existe un mapeo biyectivo (una permutación) de los nodos de un grafo hacia los nodos de segundo grafo tal que las aristas conectadas se respetan [97].

Una instancia del problema *Graph Isomorphism* consiste en dos grafos $G = \langle V, E\rangle$ y $\langle G = V', E'\rangle$ con $|V| = |V'|$. Entonces se busca saber si existe una biyección entre los vértices de tal forma que los vértices adyacentes en G se asignen a vértices adyacentes de G' y los vértices no adyacentes en G sean mapeados a vértices no adyacentes en G'.

En términos de GCP se expresa como $\langle\langle V, E,\bar{\mathrm{E}}'\rangle, \langle V', E',\bar{\mathrm{E}}'\rangle\rangle$, donde $\bar{\mathrm{E}}\Diamond_V - E$ y $\bar{\mathrm{E}}'\Diamond'_V - E'$

En la figura 4.7 (fuente elaboración propia), tenemos dos grafos con seis nodos. En el grafo A la matriz de adyacencias se puede observar en las tablas 4.4 para el grafo A y el grafo B en 4.5. Ambos grafos son isomorfos con la siguiente función de biyectiva f: $f(1) = C$, $f(2) = B$, $f(3) = F$, $f(4) = D$, $f(5) = E$ y $f(6) = A$.

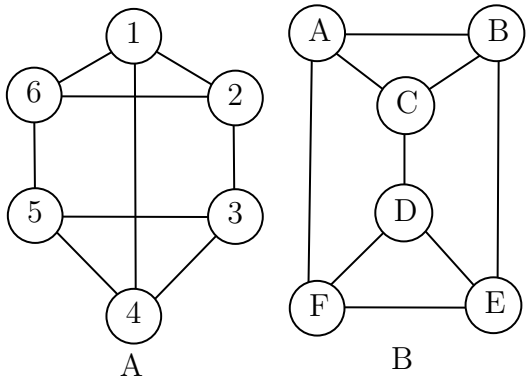


Figura 4.7: Ejemplo de Isomorfismo de grafos

4.6.2. Historia

El problema de de *Subgraph Isomorphism* es considerado un problema NP-Completo [13] y Cook demuestra que puede ser resuelto por una máquina de Turing no determi-

	1	2	3	4	5	6
1	0	1	0	0	0	1
2	1	0	1	0	0	1
3	0	1	0	1	1	0
4	0	0	1	0	1	0
5	0	0	1	1	0	1
6	1	1	0	0	1	0

Tabla 4.4: Matriz de Adyacencias para el grafo A

	A	B	C	D	E	F
A	0	1	1	0	1	0
B	1	0	1	0	0	1
C	1	1	0	1	0	0
D	0	0	1	0	1	1
E	1	0	0	1	0	1
F	0	1	0	1	1	0

Tabla 4.5: Matriz de Adyacencias para el grafo B

nista [98].

EL enfoque general de Isomorfismo de Grafos fue desarrollado por Weisfeiler y Lehman en la década de los 60 [99]. El problema de isomorfismo de grafos es visto como caso especial de *Subgraph Isomorphism* [100], sin embargo, en la actualidad no se puede asegurar que es un problema NP-Completo o P [101, 102]. Existen caso particulares del isomorfismo de grafos que pueden ser resueltos en un tiempo polinomial, pero dado que se busca saber si es que existe una permutación de los nodos de un grafo A tal que nos regrese el grafo B, esto es del orden $n!$ [99].

4.6.3. Modelos

En estado del arte existen diferentes artículos que describen los diferentes variantes de *Subgraph Isomorphism* y grafos isomorfos. Algunos de estas variantes son:

- *Automorphism problem.*
- *Tree Isomorphism.*
- *Isomorphism of Graphs with Bounded Vertex Valences* [102].
- *G-Automorphisms of a Colored Set* [102].
- *Colored Graphs with Bounded Color-Valences* [102].

De manera similar, como el problema del ancho de banda, no elaboramos un diagrama de clasificación, porque no encontramos definiciones formales de estos subproblemas.

En las tablas A.5, se muestra la información de diferentes instancias de prueba del problema de isomorfismo de grafos.

4.7. Undirected Graph Reachability

4.7.1. Definición

En la teoría de grafos, reachability"(accesibilidad) se refiere a la capacidad de llegar/alcanzar/conectar (de) un vértice a otro dentro de un gráfico. Un vértice S puede

alcanzar un vértice T (S es accesible/alcanzable desde S) si existe una secuencia de vértices adyacentes que forme un camino que comienza con S y termina con T.

En un gráfico no dirigido (Sin restricción de dirección o sentido forzoso en que se debe de recorrer dicho vértice), la accesibilidad entre todos los pares de vértices se puede determinar mediante la identificación de los componentes conectados de la gráfica, estos componentes son un subgrafo en el que dos vértices cualesquiera están conectados entre sí por caminos, y que no está conectado a ningún vértice adicional en el supergrafo.

Cualquier par de vértices en un gráfico de este tipo puede llegar a otro entre sí, sí y sólo si pertenecen a la misma componente conectada. El determinar el tamaño del subgrafo de componentes conectados y Los componentes conectados de un gráfico no dirigido se pueden identificar en tiempo lineal.

Una instancia de *Undirected Graph Reachability* consiste en que un grafo no dirigido $G = \langle V, E\rangle$ y un par de vértices $v, w, \in V$, buscar si existe un camino en el cual se conecten v a w [75].

El problema complementario seria encontrar si no existe un camino que no conecte a v y w, se puede expresar como instancia de *GCP* como: $\langle\langle V, E, \{\langle V\rangle\}\ \rangle, \{\langle w\rangle\}, \langle\{0, 1\}, \square_{\{0,1\}}, \{\langle 0\rangle\}, \{\langle 1\rangle\}\rangle\rangle$ [1].

4.7.2. Historia

En [103] se muestra la aplicación más temprana de la teoría de los grafos, desde entonces se tiene registro del uso de varios conceptos para profundizar en las investigaciones sobre estructuras matemáticas.

4.7.3. Tipos de problemas

Algunos tipos de problemas derivados del *Undirected Graph Reachability* son:

Giant component

En la teoría de redes, un componente gigante es un componente conectado de un gráfico aleatorio dado que contiene una fracción constante de los vértices del gráfico

entero [104].

Biconnected component

En la teoría de los grafos, un componente biconectado (también conocido como un bloque o un 2-componente conectado) es un subgrafo biconectado máximo. Cualquier gráfico conectado se descompone en un árbol de componentes biconectados llamado "el árbol de corte de bloques del gráfico". Los bloques se unen entre sí en vértices compartidos llamados "vértices cortados "o "puntos de articulación ". Específicamente, un vértice de corte es cualquier vértice cuya eliminación incrementa el número de componentes conectados [105].

Modular decomposition

En la teoría de grafos, al descomponer modularmente se divide un gráfico en subconjuntos de vértices llamados módulos. Estos módulos son una generalización de un çomponente conectado"de un grafo. Estos tienen cierta diferencia de los componentes conectados, ya que un módulo puede ser un subconjunto adecuado de otro. Por lo tanto, los módulos llevan a una descomposición recursiva (jerárquica) del grafo, en lugar de sólo una partición [106].

Existen variantes de descomposición modular para grafos no dirigidos y grafos dirigidos. Para cada grafo no dirigido, esta descomposición es única. Esta noción puede generalizarse a otras estructuras (por ejemplo grafos dirigidos) y es útil para diseñar algoritmos eficientes para el reconocimiento de algunas clases de grafos, para encontrar orientaciones transitivas de gráficos de comparabilidad, para problemas de optimización en gráficos y para dibujar gráficos [106].

4.8. Satisfiability

4.8.1. Definición

El problema de satisfactibilidad está formado por una fórmula en lógica proposicional $\mathscr{F}$, e cual es la conjunción de un conjunto de cláusulas C. Para cada una de las cláusulas C es una disfunción de literales, donde las literales son variables que pueden tomar el valor verdadero o falso. Entonces en este problema se busca el conjunto de valores para las variable en la función $\mathscr{F}$, tal que esta es verdadera [75].

En términos de GCP se puede expresar como $\langle\langle V, E_c(c \in C)\rangle, \langle\{0,1\}, R_c(c \in C)\rangle\rangle$, donde V es el conjunto de variables proposicionales usadas en $\mathscr{F}$, cada $E_c = \{\langle x_1, x_2, \ldots, x_{p(c)}\rangle\}$ donde $x_1, x_2, \ldots, x_{p(c)}$ son las variables que aparecen en la cláusula c y $R_c = \{\langle h(x_1), h(x_2), \ldots, h(x_{p(c)})\rangle | h$ es una asiganción de valores que satisfacen c de forma verdadera $\}$ [1].

4.8.2. Historia

Hasta donde las referencias indican, el investigador Stephen Cook en la universidad de Toronto hace mención del SAT cuando demostró que era de complejidad NP-completa en el año 1971. Presento de manera formal que todo problema de decisión en la clase de complejidad NP, puede ser reducido al problema del SAT [98].

4.8.3. Modelos

En estado del arte existen diferentes artículos que describen los diferentes problemas de $Satisfiability$ [107, 108], así como sus aplicaciones en ciencias de la computación, *Operations Research* y matemática aplicada [109, 110]. En [98] describen formalmente el problema de $Satisfiability$. Algunos de estos problemas de $Satisfiability$ son:

- *k-Satisfiability* [111]
  - *2-Satisfiability* [112].
  - *3-Satisfiability* [113].

- *Horn-Clause-Satisfiability* [114].
- *Not-All-Equal Satisfiability* [115]
- *One-in Three Satisfiability* [116].
- *XOR-satisfiability* [117].

En la tabla A.4, se muestra la información de diferentes instancias de prueba del problema de $Satisfiability$.

4.9. Constraint Satisfaction Problem

4.9.1. Definición

El problema de satisfacción de restricciones (CSP por sus siglas en ingles), se puede definir a partir de un conjunto $V = \{v_1, v_2, \ldots, v_n\}$ variables, un conjunto $W = \{w_1, w_2, \ldots, w_m\}$ de valores que pueden tomar las variables, una familia $S = \{s_k | s_k \subset V\}$ y $C(s_k)$ es el conjunto de variables en s_k. Entonces la pregunta es ¿Cuál es la asignación de valores w para las variables V, donde cada restricción se satisface? [8].

Expresado como una instancia de *GCP*, tenemos que: $\langle\langle V, S_1, S_2, \ldots, S_m\rangle, \langle D, C(S_1), C(S_2), \ldots, C(S_m)\rangle\rangle$

4.9.2. Historia

El problema fue tratado de manera formal en el área de computación en 1974 por Montanari aplicado al procesamiento de imágenes [118]. Posteriormente se realizaron diversas investigaciones acerca de los problemas de satisfacción de restricciones [119, 15]

4.9.3. Tipos

En estado del arte existen diferentes artículos que describen los diferentes problemas de CSP [16, 120, 121], así como sus aplicaciones en ciencias de la computación, *Opera-*

tions Research y matemática aplicada [122]. Algunas variantes de problemas sujetos a restricciones son:

- *Knapsack*[123]
  - *Multi-objetive* [124].
  - *Multi-Dimensional* [125].
  - *Multiple*[126].
  - *Quadratic* [127].
  - *Subset-sum* [128].
- *Location* [129].
  - *Capacitated* [130]
  - *Uncapacitated* [131]
  - *Minimax* [132].
  - *Minimum* [133].
  - *Maximum* [134].
  - *Maxmin* [135].
  - *Network Design* [136].
  - *Warehouse* [137].
  - *Fire Box Coverage* [129].
  - *Competitive* [138].
- *Scheduling* [139]
  - *Resource constrained Project Scheduling* [140].
    - *One shop* [141].
    - *Flow shop* [141].

- *Job shop* [141].
  - *Processor scheduling* [142].
  - *Bandwidth*
  - *Airport gate* [143].
  - *Repair crew* [144].

- *Parallel Machine Problems* [139]
  - *Preemptive scheduling* [141]
  - *Nonpreemptive scheduling* [141]
    - *Unit processing times* [141]
    - *General processing times* [141]

- *Single Machine Problems* [139]
  - *Batch Scheduling* [145]
  - *Single Processor* [146]
  - *Parallel Processors* [141]

- *Timetabling*

  - *Rostering* [147]
    - *Nurse Rostering* [148]
    - *Health Care* [149]
  - *Educational Timetabling*
    - *High School*
    - *University Timetabling* [150, 151].
      - *Course Timetabling* [35].
      - *Faculty Timetabling* [152].
      - *Class-room Asigment* [153].
      - *Class-teacher* [154].

- ◇ *Examination* [155].

- *Cutting stock* [156].
  - *One-dimensional* [157]
  - *Two-dimensional*
    - *Irregular (nesting problem)*
    - *Rectangular*

- Car sequencing [158, 159].

En las tablas A.4 y A.5, se muestra la información de diferentes instancias de prueba del problema de CSP.

4.10. Resumen

En este capítulo se documentó la teoría y definiciones formales de los problemas de optimización combinatoria. De forma general se abordó la historia, definición y algunos ejemplos de cada problema. Para el desarrollo de capítulos posteriores se consideró importante el reportar algunos aspectos formales de cada problema, lo cual da la pauta a comprender por qué algunos si son susceptibles a ser resueltos por particiones y otros no.

Es importante mencionar que los problemas aquí mostrados tienen una formulación en términos de estructuras algebraicas [1]. De haber algún otro definido en esos términos y que sea de la familia de los combinatorios, puede incluirse dentro de los ya reportados.

Capítulo 5

Metodología

En esta sección abordaremos la metodología que se siguió en la tesis para la elaboración de particiones, investigación del estado del arte, clasificación de GCP y selección de Hiperheurísticas.

5.1. Metodología de Revisión Sistemática de Estado del Arte (SLR)

Utilizamos el método de revisión sistemático estándar de literatura (SLR) [160], empleando una búsqueda manual de los diez GCP en seis motores de búsqueda web:

- IEEExplore (ieee.org)
- ACM Digital library (dl.acm.org)
- Google scholar (scholar.google.com)
- Citeseer library (citeseerx.ist.psu.edu)
- Keele University's electronic library (keele.ac.uk)
- ScienceDirect (sciencedirect.com)

Esta revisión incluyó trabajos que contienen una palabra clave (o sinónimo) en el título y el resumen de los GCP. Siempre que la relevancia de la investigación no fue evidente bajo este criterio, se examinaron otros elementos de los trabajos, como la introducción y las conclusiones. La búsqueda se realizó con consultas con palabras o frases como:

- *General Combinatorial Problem* or *GCP*
- *Vertex Cover* or *Vertex Cover problems.*
- *Hamiltonian path* or *Hamiltonian Circuit problem*

Las búsquedas se dirigieron a encontrar fuentes con el factor de impacto más alto posible, en diferentes formatos, como memorias de congresos, revistas, informes técnicos y repositorios de instancias de problemas. Una vez que se obtuvieron estas fuentes primarias, se revisaron en busca de referencias relevantes, a fin de identificar las tendencias a seguir y las relaciones entre las diferentes investigaciones. Esto se realizó mediante los términos clave utilizados para la búsqueda.

Para documentar el origen de cada GCP en términos de su definición matemática, buscamos investigaciones que se refieran a la primera aparición o mención formal. En particular, se examinaron publicaciones que contenían demostraciones, aplicaciones, mejoras en los enfoques de solución, pruebas experimentales, caracterizaciones o mediciones sobre los problemas de interés.

5.2. Metodología de clasificación de problemas de GCP

La investigación actual implicó la búsqueda y el análisis de diez GCP analizados por Jeavons en términos de sus estructuras algebraicas. Jeavons estableció que estos problemas son al menos NP-Duros.

En lo sucesivo, el término *clase de problema* o simplemente *problema* se usa para referirse a cada uno de los 10 problemas analizados por Jeavons (Coloreo de Grafos,

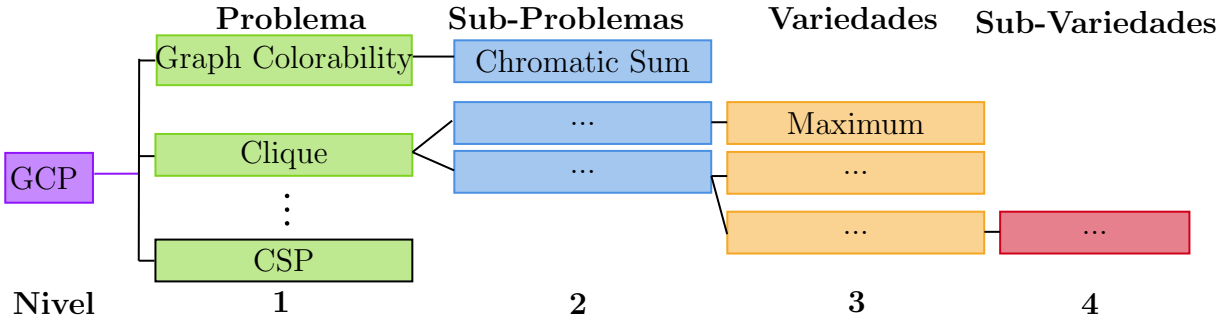


Figura 5.1: Estructura de árbol que ilustra los términos utilizados en este trabajo.

Satisfiability, Vertex Cover, etc.) y que se definen en términos de estructuras algebraicas en la literatura. El segundo nivel de agrupación incluye las formulaciones identificadas formalmente en la literatura; el término *subproblemas* se refiere a estos. El tercer nivel en esta estructura de árbol se conoce como *variedades*, el cuarto nivel representa variedades con más restricciones llamadas *sub-variedades*, y finalmente, conjuntos de datos específicos o realizaciones de los problemas, subproblemas o sub-variedades, se llaman *instancias*. La estructura de árbol utilizada para describir los problemas en las siguientes subsecciones se muestra en la Fig. 5.1. En este trabajo, dicha estructura incluye hasta cuatro niveles de profundidad. La raíz de la estructura de árbol es el GCP.

Para que un problema aparezca en el primer nivel del árbol GCP, el problema debe poseer una definición en términos de una estructura algebraica, como los problemas descritos en [8]. Los subproblemas y variedades se agregan a los niveles más profundos del árbol (nivel dos en adelante), si se puede verificar que:

- Se puede encontrar una definición matemática formal en la literatura (obligatoria para el nivel dos).
- Las restricciones y variables coinciden con las de la clase de problema a la que se va a agregar el subproblema.

Siempre que exista una definición matemática formal de un subproblema con variedades, éstas se agregan a la clase de problema respectiva. De lo contrario, el subproblema

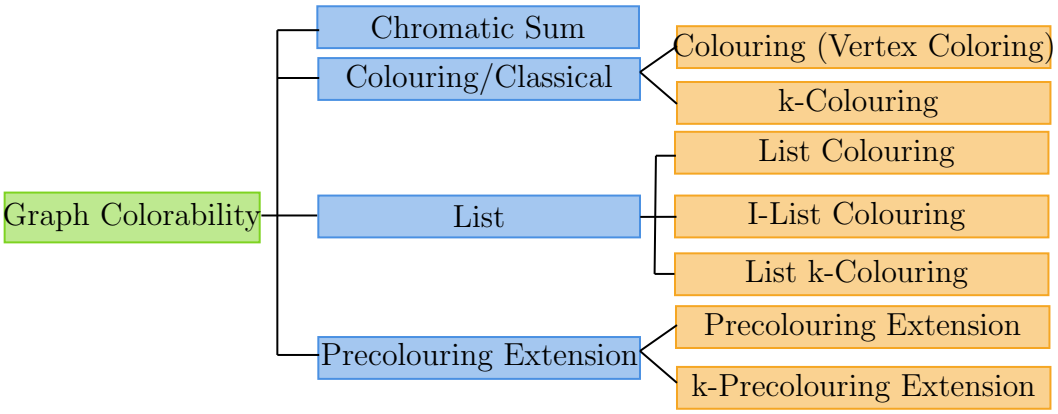


Figura 5.2: problemas de Coloreo de Grafos

no se agrega a la clase de problema correspondiente, pero se considerará agregado en el futuro, cuando se formule una definición formal (ver Fig. 5.1).

Con la información antes mencionada, se tiene que las variedades representadas por los niveles inferiores, como 3 o 4, están contenidas en los niveles anteriores, además de tener un menor grado de generalidad con respecto a la capa que la contiene anteriormente, es decir, el problema ubicado en el nivel 4 es un caso más particular (limitado o restringido) con respecto a los problemas contenidos en el nivel 1 o 2.

5.2.1. Coloreo de Grafos

Se identificaron un total de tres subproblemas de Coloreo de Grafos, representados en la Fig. 5.2. Los subproblemas incluyen al menos dos variedades definidas formalmente [161, 162, 74, 163]. Los subproblemas conocidos como: *Coloring* (también conocido como Classical- o Vertex- Coloring), *List* y *Precoloring*, así como sus variedades correspondientes. Las siguientes variedades no aparecen porque actualmente no existe un criterio formal para agruparlas: *Circular, Equitable, Harmonious, Interval, Path and Rank*.

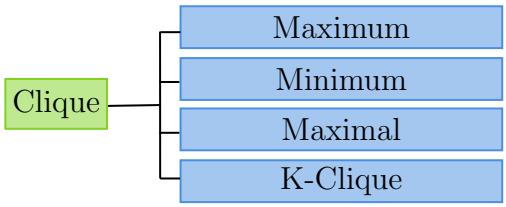


Figura 5.3: Problemas de Clique

5.2.2. Clique

En la figura 5.3(fuente elaboración propia), se muestra un diagrama que resume algunos problemas de Clique.

5.2.3. Vertex Cover

Se identificaron dos variedades, la primera con cuatro y la segunda con dos subvariedades definidas formalmente. La clasificación presentada anteriormente para el problema de *Vertex Cover* se muestra en la Fig. 5.4. Se dejo fuera del diagrama tres subproblemas porque no hay una definición formal para estos: *Connected, Tour and Tree.*

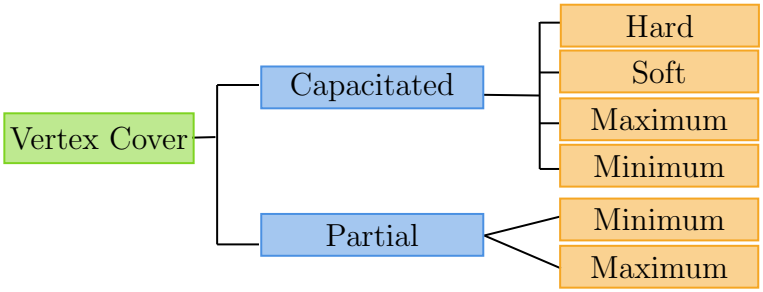


Figura 5.4: Clasificación de Problemas de *Vertex Cover*

5.2.4. Circuitos Hamiltonianos

Toda esta rama de la estructura de árbol correspondiente a los Circuitos Hamiltonianos se basa en la clasificación presentada por Garey and Johnson en [96], como se

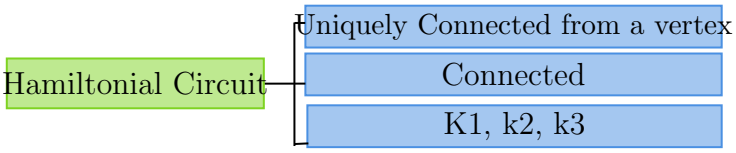


Figura 5.5: Problemas de Circuitos Hamiltonianos

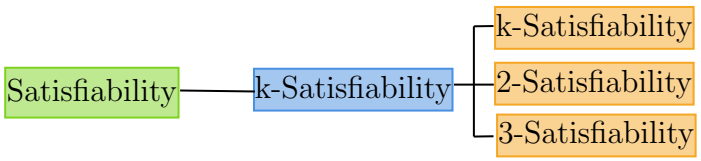


Figura 5.6: Problemas y subproblemas de *Satisfiability*

muestra en la Fig. 5.5.Posteriormente, en la figura 5.5 (fuente elaboración propia), se muestra un diagrama que resume algunos problemas de TSP.

5.2.5. Satisfiability

Se encontraron al menos cinco subproblemas del problema de Satisfacción (ver Fig. 5.6), uno de ellos ($k - Satisfabilidad$) se define formalmente junto con sus variedades en [98]. Los otros subproblemas: *Horn-Clause, Not-All, One-in-Three y XOR*, se han discutido en [107] y estos no se incluyen como subproblemas en el árbol de *Satisfiability* porque ningún estudio los ha clasificado, como variedades de cualquier caso particular de subproblema perteneciente *Satisfiability*.

5.2.6. Constraint Satisfaction Problem (CSP)

En la figura 5.7, se muestra un diagrama que resume algunos problemas de Satisfacción de Restricciones.

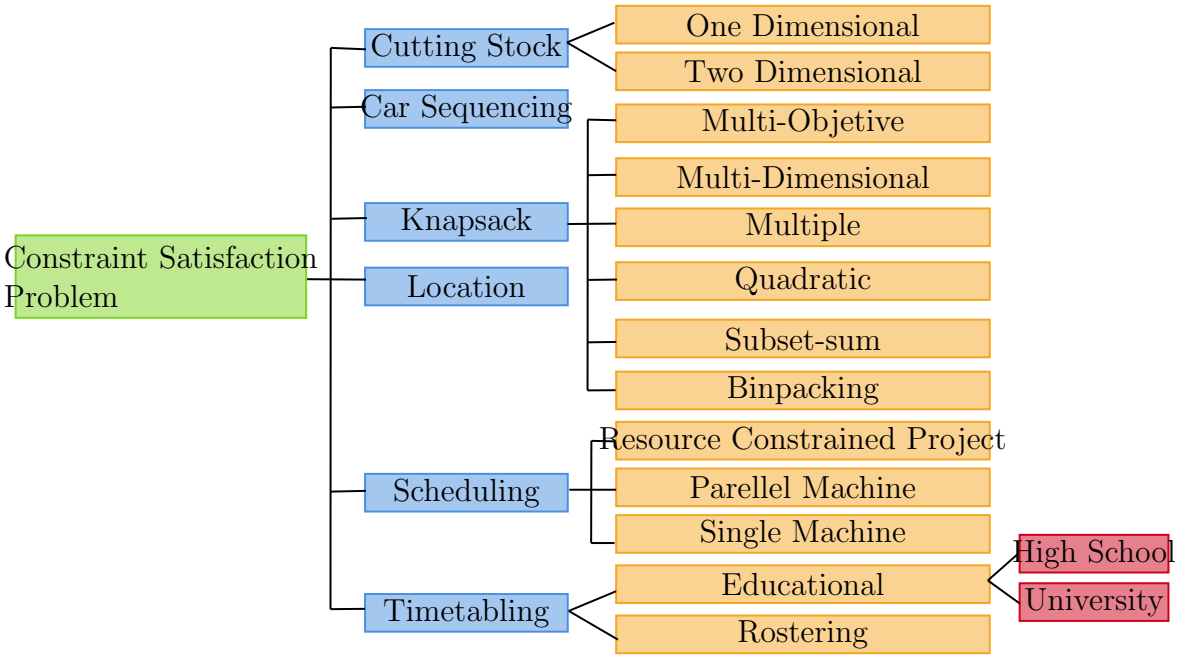


Figura 5.7: Problemas de *Constraint Satisfaction Problem*

5.3. Metodología de diseño de Particiones a GCP

Para cada uno de los 10 GCP que se estudiaron en este trabajo, en las siguientes secciones se describe si el problema pudo ser resuelto por particiones o carece de características o condiciones para ser resuelto por particiones:

- Realizar una partición de sus variables.
- Una variable no pueden estar contenida en dos partes.
- La cantidad de partes a formar en la partición de preferencia sea mayor a 2.

5.3.1. Coloreo de Gráfos

El coloreo de grafos se puede resolver mediante el uso y modelado de diseños combinatorios no regulares que denotaremos como partición, donde el conjunto de variables son los G vértices y los bloques de diseño son subconjuntos de estos vértices. La cantidad de bloques de diseño viene dada por la cardinalidad de q (número de colores).

Si el problema tiene un conjunto finito de vértices que se desean colorear con una cantidad finita de n colores, este problema es susceptible a ser resuelto por particiones. Cada parte contendrá una cantidad finita de vértices y estará etiquetada con un color.

Por ejemplo, en el siguiente grafo mostrado en la figura 5.8 (fuente elaboración propia), tenemos un total de 15 vértices, los cuales se buscan etiquetar con cinco colores diferentes. Entonces, las variedades son 15 y el número de subconjuntos es de cinco. Como información principal construiremos una matriz de adyacencias con base en el grafo de la figura 5.8, la cual se muestra en la tabla 5.1.

Si a cada color se asociamos un número romano como etiqueta, entonces tendremos lo siguiente: $I = \{morado\}$, $II = \{verde\}$, $III = \{Azul\}$, $IV = \{amarillo\}$ y $V = \{morado\}$. Pasaremos a formar cinco subconjuntos de variables donde cada una contenga al menos una variedad. En la tabla 5.2 tenemos un ejemplo de partición de estas variables.

Contaremos el número de adyacencias por bloque o parte, para el ejemplo anterior, se tiene que en el bloque $(1, 6) = 0; (1, 14) = 0; (6, 14) = 0$ por lo que $I = 0 + 0 + 0 = 0$,

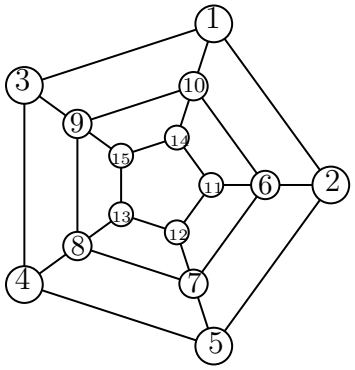


Figura 5.8: Ejemplo 1 grafo con 15 vértices

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	1	1	0	0	0	0	0	0	1	0	0	0	0	0
2	1	0	0	0	1	1	0	0	0	0	0	0	0	0	0
3	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0
4	0	0	1	0	1	0	0	1	0	0	0	0	0	0	0
5	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0
6	0	1	0	0	0	0	1	0	0	1	1	0	0	0	0
7	0	0	0	0	1	1	0	1	0	0	0	1	0	0	0
8	0	0	0	1	0	0	1	0	1	0	0	0	1	0	0
9	0	0	1	0	0	0	0	1	0	1	0	0	0	0	1
10	1	0	0	0	0	1	0	0	1	0	0	0	0	1	0
11	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0
12	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0
13	0	0	0	0	0	0	0	1	0	0	0	1	0	0	1
14	0	0	0	0	0	0	0	0	0	1	1	0	0	0	1
15	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0

Tabla 5.1: Matriz de adyacencias de la figura 5.8

Bloque	Variables		
I	1	6	14
II	2	7	15
III	3	8	13
IV	4	9	12
V	5	10	11

Tabla 5.2: Bloques de diseño (particiones) para el ejemplo de la figura 5.8

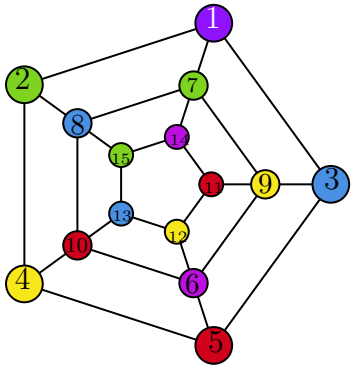


Figura 5.9: Grafo del ejemplo 1 con colores segun la tabla 5.2

para los bloques II, III, IV y V es cero, por lo tanto este diseño de particiones no tiene nodos adyacentes con el mismo color. La representación gráfica se muestra en la figura 5.9(fuente elaboración propia).

5.3.2. Clique

En este problema se pueden identificar un conjunto de variables las cuales son los nodos y una familia de subconjuntos de estas variables que son los cliques. Pero dado que los cliques pueden compartir nodos con otros cliques, esto no empataría con la definición de partición. En una partición la intersección de dos subconjuntos debe ser el conjunto vacío y dado que se puede dar el caso de compartir nodos entre cliques.

Por ejemplo en la figura 4.2 b, dado que los cliques formados por los nodos 1,2 y 8 comparten al nodo 1 con el clique 1,3 y 9; la intersección de estos dos subconjuntos

	1	2	3	4	5	6	7	8	9	10
1	0	1	1	0	0	1	0	0	0	0
2	1	0	0	1	0	0	1	0	0	0
3	1	0	0	0	1	0	0	1	0	0
4	0	1	0	0	1	0	0	0	1	0
5	0	0	1	1	0	1	0	0	0	1
6	1	0	0	0	1	0	1	1	1	1
7	0	1	0	0	0	1	0	1	1	1
8	0	0	1	0	0	1	1	0	1	1
9	0	0	0	1	0	1	1	1	0	1
10	0	0	0	0	1	1	1	1	1	0

Tabla 5.3: Matriz de adyacencias del grafo de la figura 4.3

es el nodo uno, dadas estas restricciones no se puede aplicar el diseño de particiones a este problema.

5.3.3. Vertex Cover Problem

Este problema puede ser susceptible a ser tratado como particiones en el caso de que busque saber la cantidad de nodos mínima que cubren todas las aristas (puede ser con o sin ponderación de nodos). Las variables serían los nodos y el número de bloques de diseño estarían limitados a dos, donde uno contendrá el *Vertex Cover* y segundo los nodos restantes.

Para el ejemplo de la figura 4.3, se puede decir que las partes $p_1 = 1, 3, 4, 6, 7, 8$ y $p_2 = 2, 5, 9$ al unirlas nos regresa el conjunto universo de nodos y su intersección es el conjunto vacío. Además de que para la evaluación se puede apoyar con la matriz de adyacencias que se muestran en la tabla 5.3. Para cada parte se contarán el número de aristas que cubre y aquella que las contenga todas y tenga la menor cantidad de vértices, será considerado como solución factible. Para el ejemplo de la figura 4.3, p_1 cubre las 20 aristas y tiene siete vértices, mientras que p_2 parte tiene tres vértices y cubre solamente diez aristas.

5.3.4. K-Dimensional Matching

El problema de *k-Dimensional Matching* se puede resolver mediante en enfoque de particiones. El conjunto de variables serían las coordenadas denominadas como M y el conjunto de bloques en este caso sería de dos, uno que tendrá las variables *Matching* y en otro las que quedan fuera del matching por las restricciones.

En el caso mostrado en la figura 4.4, tenemos un total de cinco variables y dos bloques de diseño. Entonces una solución de partición para M puede ser $m_1 = \{(1,1,1),(5,4,3)y(4,3,2)\}$ y $m_2 = \{(4,4,4),(2,1,2)\}$, la unión nos regresa el conjunto M y su intersección nos regresa el conjunto vacío. Otro ejemplo es: $m_1 = \{(1,1,1),(2,2,2),(3,3,3),(4,4,4)\}$ y $m_2 = \{(5,4,3)(4,3,2),(2,1,2)\}$.

5.3.5. Hamiltonian Circuit

Partiendo del supuesto: *el conjunto de variables son los nodos y los bloques de diseño subconjuntos de estos nodos*, dicho problema se puede aplicar el diseño de particiones pero no es recomendable. Esto debido a que no se busca hacer subconjuntos de esos nodos, sino se busca generar la ruta que pase todo todos los nodos, sin pasar dos veces por la misma arista.

5.3.6. TSP

En el *TSP* se puede decir que el conjunto de variables son las ciudades y en la formulación inicial no se busca hacer subrutas de estas. Para el *TSP* donde se busca generar la ruta de menor costo y que pase por todos los nodos, no es susceptible a ser resuelto por particiones.

En el caso de del problema de *VRP*, este si puede ser resuelto mediante el enfoque de particiones, dado que es un problema que tiene menos restricciones y nos permite formar particiones. Nuestras variables serán el conjunto de ciudades y dando que se busca hacer subrutas de estas ciudades, los bloques de diseño serán estas estos subrutas.

Por ejemplo tenemos la instancia $A-n32-k5$, la cual tiene 32 ciudades y se busca hacer cinco surutas para cubrir una demanda. En total se generan cinco partes de

Ciudad	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
Sub-ruta (Parte)	1	4	5	5	5	4	5	5	1	2	4	3	3	4	2	3	5	4	4	3	3	3	3	4	4	4	3	2	1	5	2	3	5	1	2	2	1

Tabla 5.4: Ciduades de Instancia $A - n32 - k5$ con un número de tour

	Ciudad								
Parte 1	1	9	29	34	37				
Parte 2	10	15	28	31	35	36			
Parte 3	12	13	16	20	21	22	23	27	32
Parte 4	2	6	11	14	18	19	24	25	26
Parte 5	3	4	5	7	8	17	30	33	

Tabla 5.5: Partición de la Instancia $A - n32 - k5$

esta ruta, si a cada ruta le asignamos un número de parte (ver tabla 5.4), podríamos generar cinco tours y la intersección de estos tours es vacía, ya que no se permite que dos camiones pasen por la misma ciudad (salgo el punto de partida inicial, ver tabla 5.5).

5.3.7. Subgraph Isomorphism y Graph Isomorphism

En los problemas *Subgraph Isomorphism* e isomorfismo de grafos, se puede plantear que los nodos o vértices son las variables.

Para *Subgraph Isomorphism* se puede identificar la formación dos subconjuntos de estas variables, uno que contenga los nodos que forman un subgrafo que es isomorfo a otro grafo y el segundo que contiene los nodos restantes. Sin embargo, aquí importa el orden que puedan tener los nodos y aunque la intersección de los dos subconjuntos es vacía, no son condiciones recomendables para la aplicación del enfoque de modelación a particiones, ya que solo se necesitan hacer dos subgrupos.

Para el isomorfismo de grafos no se necesita elaborar un subconjunto de las variables. Además de que al igual que el problema anterior nos importa el orden de los nodos, entonces no es recomendable la aplicación de modelación a particiones, ya que solo se necesitan hacer dos subconjuntos y al menos se recomienda que sean 3.

5.3.8. Undirected Graph Reachability

Para el problema de *Undirected Graph Reachability*, nuestras variables será el conjunto de nodos denotados como v y en este caso se busca generar dos bloques de diseño. Uno de ellos contendrá el subgrafo o camino que conecta a v a w y otro los nodos restantes (por lo tanto, no pueden compartir nodos y la intersección de estos dos conjuntos es vacía).

En este caso dado que se tienen solamente dos particiones, es posible utilizar el enfoque de particiones, pero no se puede garantizar obtener resultados competitivos en los problemas a aplicar.

5.3.9. Satisfiability

Este problema puede ser modelado a particiones de la siguiente forma: el conjunto de variables serán las variables $x_1, x_2, \ldots, x_n$ y en este problema se busca que estas tomen un valor de verdadero o falso, por lo que se agruparan en su grupos y entonces nuestros bloques de diseño serán dos (uno que contengan a las variables con valores verdaderos y otro con los falsos). La evaluación de las variables en cada una de clausuras se puede realizar sin verse afectado.

Sin embargo, queda a comprobación es recomendable aplicar particiones a este problema. Aunque el orden en el que aparecen las variables dentro del bloque de diseño es irrelevante, queda por demostrar que se pueden obtener resultados competitivos con el estado del arte al aplicar el enfoque por particiones.

5.3.10. Constraint Satisfaction Problem

Jeavons en [8], define al conjunto de variables con la letra V, al igual que en la definición de diseños combinatorios. Ahora los bloques de diseño son los subconjuntos s_k de la definición del *CSP*. Cabe mencionar que estos boques de diseño están sujetos a un conjunto de restricciones. Por lo tanto, este problema puede ser susceptible a ser tratado por particiones, siempre y cuando se satisfagan restricciones.

Algunos de los problemas mencionados anteriormente que pueden ser resueltos como peticiones son:

- *High School.*
- *University Timetabling.*
- *Car sequencing.*
- *Scheduling.*
- *Nurse Rostering.*

5.4. Metodología para seleccionar y determinar un conjunto mínimo de heurísticas

En esta sección, se describe un nuevo enfoque para seleccionar y determinar un subconjunto de heurísticas para resolver instancias de *GCP* y *CVRP*. Describimos nuestra metodología en los siguientes pasos y la representación gráfica de nuestra metodología se muestra en la figura 5.10.

De acuerdo con el análisis previo de los diez problemas de optimización combinatoria, se seleccionaron dos, ya que con base en sus características esto son compatibles con la metodología de particiones. Posteriormente se estudiaron del estado del arte diferentes heuristicas que pudieran ser aplicadas a ambos problemas, para conformar lo que sería nuestro conjunto total de heurísticas.

Paso 1 Identificación de las variables y restricciones del problema. Primero, se identifican las variables y restricciones del problema de acuerdo con las metas u objetivos del problema. Para modelar el GCP, los valores en la matriz MMA representan los pesos de las aristas que conectan los nodos. Si hay un cero en una determinada posición (x, y) en la matriz, esto representa que no hay conexión entre dichos nodos. Para el coloreo de grafos, cada nodo se colorea considerando que los nodos adyacentes no tengan el mismo color. *CVRP* está alineado con nuestra

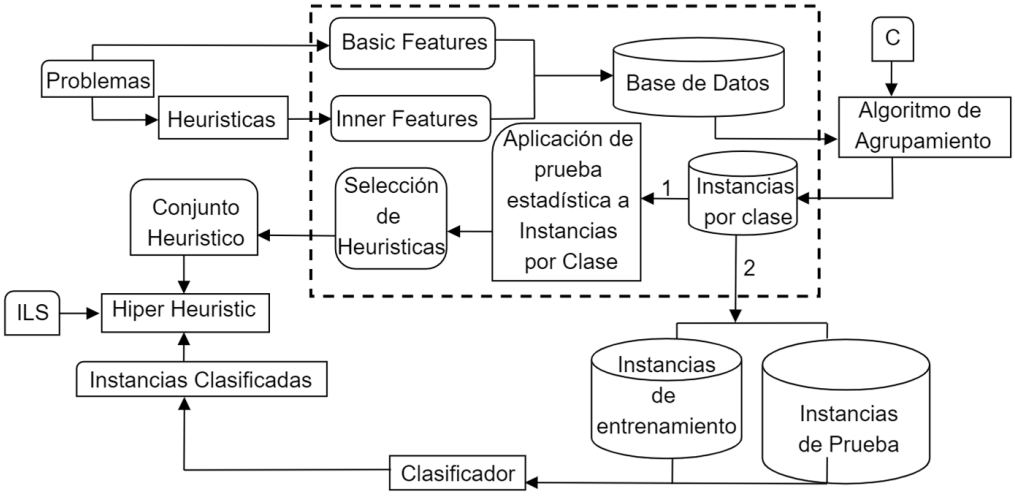


Figura 5.10: Metodología para seleccionar y determinar un conjunto mínimo de heurísticas

metodología debido a que su objetivo busca obtener subrutas en las que el costo del tour (subgrupo) debe ser del mínimo costo o el más económico. Además de que tenemos el insumo de matriz de distancias que puede ser traducido fácilmente a MMA, en el cual podemos identificar las variables y sus restricciones.

Paso 2 Modelación de las restricciones del problema a estructuras En ambos problemas, debemos diseñar una partición de nodos o ciudades. Primero, es necesario representar todas las restricciones para cada variable en una LPH, por ejemplo, un nodo no puede ser etiquetado por cierto color o una ciudad estar restringida para un tour. Luego se debe diseñar la MMA para cada problema; en GC la matriz de adyacencia es la MMA y en CVRP la matriz MMA será la matriz que tiene las distancias de ciudad a ciudad. Para GC, la LPH se construye en función del número de colores en los que se pueden etiquetar los nodos. En caso de que el problema restrinja los colores a cinco, la lista será similar a la que se muestra en la tabla 3.2. De manera similar, esta lista se creará para el CVRP, donde el número de camiones es el número de intervalos de tiempo que deben representarse en la lista (consulte la tabla 3.2). Para los problemas utilizados en este trabajo, no fue necesario elaborar estructuras adicionales para restricciones suaves. Además, para una descripción extensa y cómo modelar restricciones adicionales, la investigación propuesta por Ortiz en [164] detalla todos los casos posibles y diferentes características.

Paso 3 Dado un conjunto de instancias denotado como I. Para cada una de las instancias I_i se aplica un número k de veces la heurística H_j. Después de modelar los insumos para cada problema, el siguiente paso es aplicar las heurísticas. En esta etapa, cada instancia se procesa previamente para obtener más información sobre el problema. Esto permite mejorar el diseño y las pruebas de la Hiperheurística. La descripción de los algoritmos utilizados se verá en la sección 3.2. El objetivo de esta etapa es determinar cuáles heurísticas pueden tener un mejor rendimiento individualmente para todas las instancias de una clase de problema. Esto mejora el proceso de selección del conjunto de heuristicas

para las Hiperheurísticas que es una parte fundamental de la misma [165]. Para todos los casos, se aplicará k veces cada heurística. Es posible que después de éste paso se obtengan soluciones cercanas al óptimo de cada instancia. Esto significa que es posible evitar ejecutar un proceso hiperheurístico completo y costoso a la hora de resolver problemas con la aplicación de una simple heurística.

Paso 4 Aplicar las pruebas estadísticas no paramétricas a las instancias a todas las clases de problemas. Ejemplo: Friedman, Friedman alineado y Quade para todos los casos de Coloreo de grafos y por separado las mismas 3 pruebas para el CVRP. En esta etapa de meta aprendizaje, primero tenemos que aprender de las instancias, para hay que identificar y generar los patrones con características relevantes del problema. Dichos patrones estarán conformados por las $inner - features$ y $basicfeatures$. Finalmente, todas las características en las instancias del conjunto de patrones creado se normalizaron antes de aplicar el clasificador. Las $innerfeatures$ representan los datos básicos e importantes de cada instancia. Para $inner - feature$, es necesario obtener los rangos (basados en el rendimiento) de cada heurística por problema.

Paso 5 Determinar el número c . Para determinar la cantidad de clases con Sturges Rule [166] con:

$$c = 1 + \frac{log(T)}{log(2)} \tag{5.1}$$

Donde T es el número total de instancias.

Paso 6 Generación de patrones basados en características *inner* y *basic features* , por instancia. La *basic features* será la información del problema y las *inner features* será la aptitud obtenida por la heurística aplicada k veces al problema. La descripción y generación de características que permiten diferenciar en al menos dos grupos de instancias dentro de la misma clase de problema. Usamos el término $basic - feature$ y $inner - feature$ con base a la propuesta realizada por Gutierrez-Rodríguez et al. [43]. Como características básicas, estas vienen dadas por el problema, por ejemplo, el número de nodos,

Tabla 5.6: Basic-Features para CVRP y Coloreo de Grafos

Problema	Min Partición	Max Partitión	Variables	Aristas
VRP	Trucks	Trucks	Ciudades	conexiones entre ciudades
Coloreo de Grafos	Lower Bound	Upper Bound	nodes	Aristas

Tabla 5.7: Caracteristicas para las instancias

	Lower Bound	Upper Bound	Nodos	Aristas	H_0	H_1	H_2	H_3
Instance 1	3	3	8	50	3	2	1	4
Instance 2	5	10	6	36	4	3	1.5	1.5
Instance 3	3	8	25	80	3	1	2	4
Instance 4	10	10	50	80	2	4	3	1

colores, camiones, etc., dependiendo de la información de cada problema. Para ambas clases de problemas, el número de $basic - features$ diferentes se resume en la tabla 5.6.

Los valores de rendimiento de todas las heurísticas son la clave entre funciones. Finalmente, el patrón por instancia es *Basic-feature+inner-feature*. El patrón final se muestra en la tabla 5.7. Por ejemplo, la instancia 1 tiene un patrón (3, 3, 8, 50, 3, 2, 1, 4), el número de $inner - feature$ depende de la heurística del grupo (ocho características para el ejemplo dado).

Paso 7 Aplicar el algoritmo $k - means$ para agrupar todas las instancias en c grupos. Agrupar las instancias con el algoritmo k-means. El número de clúster correspondiente de cada instancia es el número de clase.

Paso 8 Etiquetar cada patrón de acuerdo con el número de grupo en el que se clasificó cada patrón (instancia).

Paso 9 Aplicar nuevamente las tres pruebas estadísticas a los resultados de las heurísticas (paso 3), pero ahora por clase de problema (las obtenidas de la agrupación de $k - means$, pasos 7 y 8). La heurística con el menor valor de jerarquía corresponde a la de mejor desempeño [39].

Paso 10 Según el r_j de cada una de las pruebas no paramétricas como referencia, se establece un límite para cada prueba: FC_{rj} para Friedman, FAC_{rj} para Friedman Alineado y QC_{rj} en Quade.

Paso 11 Reducción del número de heurísticas adecuadas para cada clase eligiendo solo las heurísticas que estén por debajo del nivel establecido en las tres pruebas por clase. Al final, estas heurísticas serán el conjunto mínimo por clase..

Paso 12 Separar los patrones (paso 6) en un conjunto de entrenamiento y prueba para pasar a la fase de clasificación. Es importante considerar al menos un patrón de cada clase para el conjunto de prueba..

Step 13 Utilizar el clasificador con el conjunto de entrenamiento para realizar los ajustes necesarios. Después de describir y obtener todas las características del patrón por instancia, el siguiente paso es entrenar y probar todas las instancias mediante un clasificador. Para nuestro enfoque, preferimos utilizar un clasificador simple bayesiano, porque nuestro objetivo no era comparar el rendimiento entre algoritmos de clasificación o diseñar clasificadores ad-hoc para nuestra investigación.

El NBC simplifica el aprendizaje asumiendo por clase que todas las características son independientes [167]. En nuestra metodología, asumimos que cada desempeño heurístico es independiente porque aplicamos cada heurística en experimentos independientes. En la etapa anterior, cada experimento debe ejecutarse con una sola heurística, por lo que no aplicamos dos o más heurísticas en ese momento.

Paso 14 Finalmente, el conjunto de instancias de prueba se utilizará con el clasificador para asignar un *grupo* y resolverla con su correspondiente conjunto de heurísticas.

5.5. Resumen

En esta sección abordamos 4 metodologías que apoyaron el desarrollo de esta tesis. La primera relacionada a la forma metodológica de revisión del estado del arte (SLR). Posteriormente, la metodología que se propuso para clasificar los problemas de GCP. La tercera sección, detalla las metodologías de diseño que fueron base para los insumos y solución de GCP. Finalmente, la metodología propuesta para seleccionar y determinar un conjunto de heurísticas para el enfoque hiperheurístico. Las metodologías fueron mencionadas conforme al orden de su aplicación y uso a lo largo de este trabajo doctoral.

Capítulo 6

Experimentos y análisis de resultados

En este capítulo se detallan los experimentos y resultados obtenidos de durante esta investigación. Una vez visto el marco teórico y la metodología pasamos a presentar la aplicación de las técnicas Metaheurísticas e Hiperheurísticas al diseño de horarios.

6.1. Resultados de la revisión del Estado del Arte con SLR

Bajo los criterios de búsqueda empleados, se recopilaron aproximadamente 750 referencias con posible relevancia. Los resultados obtenidos por los diferentes motores de búsqueda fueron numerosos y parecían pertinentes, pero bajo un análisis detallado, algunos de ellos fueron descartados de esta investigación. Después de filtrar este resultado inicial, la selección se redujo a aproximadamente 130 referencias. Estas búsquedas se realizaron durante el período comprendido entre diciembre de 2016 y febrero de 2019.

Se descubrió que varios motores de búsqueda en la web producen tales resultados que su relación con el concepto objetivo es demasiado débil. Es decir, en algunos casos se encuentra una sola palabra de interés en un trabajo no relacionado sobre el tema específico.

No hubo un solo motor de búsqueda que superara a los demás en la calidad de sus

resultados, con respecto a una consulta específica. Además, los artículos devueltos por algunos de los motores de búsqueda tenían poca relevancia para la consulta, por lo que sería difícil concluir que su rendimiento es uniforme.

En muchos casos, el contenido de los resúmenes devueltos carecía de información sustancial que delineara los artículos correspondientes de una manera útil, por lo que se recurrió a la inspección de todos los documentos para justificar su inclusión.

La tabla 6.1 muestra el resumen de los resultados de búsqueda que se obtuvieron. La tabla contiene el número de artículos obtenidos para las doce búsquedas (una para cada problema, incluidos GCP y TSP) en los seis motores de búsqueda.

La tabla A.1 muestra el resumen de los resultados en términos de porcentaje, todos ellos normalizados por el motor de búsqueda. Se consideró la suma de todos los trabajos devueltos por los motores de búsqueda y luego se extrajo el porcentaje. Los valores subrayados indican el porcentaje más alto de resultados por motor de búsqueda, mientras que los resultados en negrita indican los resultados más altos por problema. Por ejemplo, el problema con la mayoría de los éxitos para los motores de ACM y Keele University es el problema de satisfacción (SAT). Se puede observar que ACM, Citeseer y Keele University obtuvieron los porcentajes más altos de resultados devueltos para la mayoría de los problemas (números en negrita). Sin embargo, no debe olvidarse que una gran cantidad de resultados pueden ser engañosos, porque muchos de los documentos relacionados"no discuten exhaustivamente la definición de un determinado problema.

Para cada uno de los motores de búsqueda web, los problemas más populares fueron:

- ACM Digital library: *SAT*.
- Google Scholar: *Graph Colorability*.
- Citeseer library: *General Combinatorial Problem*.
- IEEE Xplore and Keele University's: *CSP*.
- ScienceDirect: *Clique*.

Los problemas menos populares fueron:

Problema	**IEEE**	**ACM Digital library**	**Google scholar**	**Citeseer library**	**Keele U.**	**Science Direct**
GCP	879	1	255	303	45	48
GC	857	301	46,900	196	5,841	3,503
Clique	820	535	13,000	10	2,166	19,632
VC	294	170	7,900	54	1,475	2,073
k-DM	33	8	218	321	909	25
HC	341	19	6,140	12	1,075	1,374
Bandwidth	2,030	8,238	7,850	24	1,028	599
GI	1,789	105	23,000	115	3,725	2,083
UGR	7	1	100	42	11	3
SAT	2,783	10,335	29,000	169	4,029	9,378
CSP	4,641	429	34,000	179	3,804	3,052

Tabla 6.1: Resultados de la búsqueda de literatura para diferentes GCP diferentes

Problema	**IEEE**	**ACM Digital library**	**Google scholar**	**Citeseer library**	**Keele U.**	**Science Direct**
GCP	6.07	0.00	0.15	**21.2**	0.19	0.11
GC	5.92	1.49	**27.8**	13.75	24.23	8.39
Clique	5.67	2.66	7.72	0.70	8.98	**47.0**
VC	2.03	0.84	4.69	3.79	**6.12**	4.96
k-DM	0.23	0.04	0.13	**22.5**	3.77	0.06
HC	2.36	0.09	3.65	0.84	**4.46**	3.29
Bandwidth	14.03	**40.9**	4.66	1.68	4.26	1.43
GI	12.36	0.52	13.66	8.07	**15.4**	4.99
UGR	0.05	0.00	0.06	**2.95**	0.05	0.01
SAT	19.23	**51.3**	17.22	11.86	16.71	22.4
CSP	**32.0**	2.13	20.19	12.56	15.78	7.31

Tabla 6.2: Resultados de búsqueda (%) para diferentes GCP en seis motores de búsqueda web

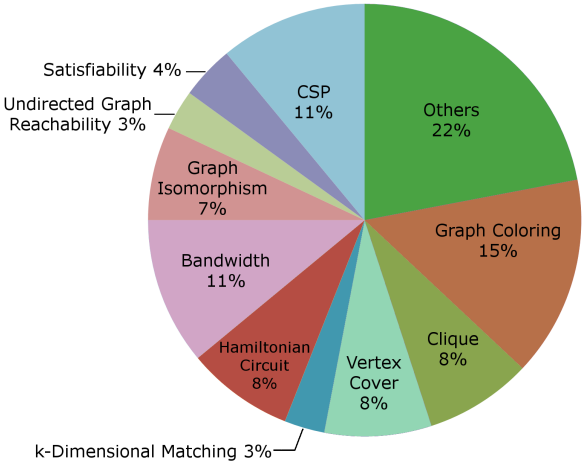


Figura 6.1: Porcentaje de investigaciones por GCP

- IEEE Xplore, Google Scholar, Keele University's, ACM and ScienceDirect: *Undirected Graph Reachability.*
- Citeseer library: *Clique.*

6.2. Resultados de la revisión de Estado de Arte de GCP

En el capítulo 4 enumeramos 10 *GCP*. Cabe mencionar que el orden en que se presentaron estos no implica un orden de importancia o nivel de generalidad entre ellos. Los porcentajes por *GCP* en las referencias analizadas en este trabajo se muestran en la Fig. 6.1.

El Coloreo de Grafos es quizás uno de los más antiguos en su formulación, ya que data de mediados del siglo XIX y este es el problema con más referencias en nuestro trabajo. Dentro del estado del arte consultado, se encontraron 3 subproblemas diferentes del Coloreo de Grafos; estos están contenidos en el nivel 2. Del mismo modo, los problemas de *Clique, Undirected Graph Reachability* y *Subgraph Isomorphism* tienen uno o más subproblemas en el Nivel 3 de sus diagramas de árbol correspondientes. Finalmente, fue

complejo generar subproblemas o clasificaciones de CSP debido al número de familias que abarca. Además, es el segundo problema más popular, con un 11 por ciento de referencias en nuestro trabajo.

A continuación, se resumen tres ejemplos de cómo la clasificación propuesta puede ayudar a diseñar mejor los estudios sobre problemas combinatorios generales:

- El estudio en [168] analiza tres problemas de optimización combinatoria: *Bin Packing, Permutation Flow Shop y Personnel Scheduling*. Los autores se refieren a estos problemas como tres dominios de problemas diferentes, indicando que pertenecen a diferentes clases de problemas. Sin embargo, con base en la clasificación propuesta en este trabajo, se puede identificar que se puede identificar que Burke et al. discutió tres subproblemas del problema de satisfacción de restricciones.

- En [169], los autores describen un diseño hiperheurístico basado en *Q-Learning* para seis dominios de problemas diferentes: *Boolean Satisfiability, One-dimensional Bin-Packing, Permutation Flow Shop, Personnel Scheduling, Traveling Salesman Problem y Vehicle Routing Problem*. Bajo la clasificación propuesta en este trabajo, se puede identificar que [169] cubrió solo tres problemas diferentes: Satisfabilidad, CSP (que incluye las variedades: *Bin-Packing, Flow Shop y Personal Scheduling*) y H*amiltonian Circuit* (que incluye las variedades: TSP y VRP).

- Como tercer ejemplo, en [170] los autores seleccionaron lo que supusieron que eran dos *clases de problema* diferentes para evaluar un diseño de algoritmo. Utilizando la clasificación descrita en este trabajo, se puede identificar que dicho estudio no es tan general como se ha descrito, ya que las clases de problemas asumidas, a saber, *VRP y Multi-Knapsack*, son de hecho subproblemas del *Hamiltonian Circuit* y familia de *Constraint Satisfaction*, respectivamente.

6.3. Selección de Conjunto de Heurísticas

Esta sección describe nuestros experimentos en detalle para las instancias del coloreo de grafos y $CVRP$ utilizados en nuestro enfoque. Describimos la configuración en la aplicación del *Iterated Local search Hyper-heuristic*. Finalmente, describimos las prue-

bas estadísticas que utilizamos para comparar nuestros resultados de la metodología experimental.

Para cada heurística, se proporcionó un límite de 100,000 llamadas de función en cada ejecución, para todas las instancias. Aplicamos la prueba de *Shapiro Wilks* para verificar si los resultados eran normales o no y elegir un mejor representante (promedio o medio). Cuando los datos eran normales, el promedio se tomaba como valor representativo y en otro caso la mediana.

6.3.1. Resultados de heurísticas para Coloreo de Grafos y CVRP

Con la finalidad de conocer el desempeño de la heuristicas, se hicieron 41 ejecuciones de cada una de ellas en todas las instancias. Se usó el conjunto de instancias DIMACS el cual fue propuesto para la competencia *The second DIMACS* en coloreo de grafos leighton1979graph. En las tablas 6.3 y 6.4 se muestra los resultados, donde se denotan en negritas los mejores resultados. La única instancia que fue resuelta con optimalidad fue *myciel3*.

Aplicamos pruebas estadísticas no paramétricas con un nivel de significancia de $\alpha = 0{,}05$ para comprobar que existen diferencias entre el rendimiento de cada heurística. La tabla 6.5 muestra los rangos obtenidos en las tres pruebas estadísticas para instancias de Coloreo de grafos. Para las tres pruebas ómnibus indicadas, existen diferencias significativas entre las heurísticas.

Las heurísticas que obtuvieron los rangos más altos en las tres pruebas fueron H_6 y H_{11}. El H_3 y H_8 fueron los mejores resultados para las instancias de problemas, porque en las tres pruebas estadísticas Friedman, Friedman Alineado y Quade esas heurísticas obtuvieron los rangos más bajos (ver tabla 6.5 marcadas en negrita).

6.3.2. Vehicle Routing Problem Capacitated (CVRP)

Se utilizaron y probaron tres datasets del estado del arte con 41 corridas:

1. Augerat et al. (SET A) 9 instancias, propuesto por [171]

#	Nombre	Nodos	Colores	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
1	1-Insertions_4	67	3	12	18	11	15	**10**	66	61	13	25	28	75
2	2-Insertions_3	37	3	**4**	7	**4**	5	**4**	24	15	**4**	11	10	29
3	2-Insertions_4	149	3	33	41	**16**	32	32	148	158	29	45	47	155
4	3-Insertions_3	56	3	5	9	**4**	**4**	5	33	24	**4**	9	9	40
5	4-Insertions_3	79	2	14	19	16	13	**11**	61	66	13	25	22	71
6	anna	138	10	17	17	14	17	16	59	45	**13**	35	30	68
7	ash331GPIA	662	3	99	**76**	85	96	82	1067	724	89	164	156	1082
8	ash608GPIA	1216	3	1497	1448	**729**	1541	1370	2003	1120	1083	1513	1485	2066
9	ash958GPIA	1916	3	2618	2553	**1900**	2621	2462	3161	1999	2039	2616	2624	3206
10	david	87	10	16	18	**14**	16	16	49	48	15	35	37	58
11	DSJC1000.1	1000	19	1847	1801	**610**	1837	1781	2565	1019	942	1789	1821	2634
12	DSJC1000.5	1000	82	2432	2513	**838**	2425	2379	3130	1126	1230	2327	2327	3234
13	DSJC1000.9	1000	214	1856	2073	**828**	1850	1824	2354	1204	1062	2375	1768	2414
14	DSJC125.1	125	4	48	59	**36**	48	54	161	97	37	56	66	173
15	DSJC125.5	125	15	113	156	72	114	124	275	133	**68**	250	112	298
16	DSJC125.9	125	42	125	173	77	124	131	210	132	**72**	208	112	221
17	DSJC250.5	250	25	350	415	**146**	349	368	658	278	157	339	326	681
18	DSJC250.9	250	70	311	388	**152**	315	319	477	234	153	487	294	485
19	DSJC500.1	500	11	629	646	**197**	634	598	1085	532	279	623	614	1114
20	DSJC500.5	500	47	927	1007	**290**	911	905	1381	471	368	859	853	1412
21	DSJC500.9	500	122	771	927	**326**	759	763	1044	487	382	1051	731	1129
22	DSJR500.1	500	11	124	130	**33**	125	119	334	128	46	129	127	343
23	DSJR500.1c	500	84	1257	1439	**203**	1262	1259	1566	1162	475	1539	1210	1589
24	DSJR500.5	500	121	376	425	**201**	372	369	619	299	211	627	352	662
25	fpsol2.i.1	496	64	141	183	**82**	139	122	263	169	91	255	262	267
26	fpsol2.i.2	451	29	52	81	59	**51**	52	337	281	52	204	215	348
27	fpsol2.i.3	425	29	51	82	55	**51**	54	364	272	54	211	232	368
28	games120	120	8	16	24	11	17	17	82	34	**10**	23	23	88
29	homer	561	12	23	**22**	**22**	24	23	165	139	24	53	44	176
30	huck	74	10	13	13	**12**	13	13	46	38	**12**	27	24	47
31	inithx.i.1	864	53	**78**	95	81	80	80	423	320	115	264	263	444
32	inithx.i.2	645	30	**55**	100	62	56	59	525	459	85	350	324	531
33	inithx.i.3	621	30	59	108	65	**56**	63	510	462	78	338	331	539
34	jean	80	9	13	**12**	13	14	**12**	47	31	13	27	26	46
35	le450_15a	450	14	292	293	**103**	291	291	580	410	137	292	289	597
36	le450_15b	450	14	289	297	**106**	290	285	589	427	125	300	288	624
37	le450_15c	450	14	716	762	**358**	713	713	1187	887	424	734	707	1179
38	le450_15d	450	14	723	751	**371**	718	733	1163	799	418	728	725	1197
39	le450_25a	450	24	160	162	**50**	158	153	372	229	63	164	162	395
40	le450_25b	450	24	156	161	**51**	164	160	377	188	58	156	156	404
41	le450_25c	450	24	407	417	**150**	400	393	736	510	180	396	394	780
42	le450_25d	450	24	400	422	**150**	400	403	736	461	179	397	394	791
43	le450_5a	450	4	780	772	**484**	755	738	1195	966	551	753	771	1209
44	le450_5b	450	4	769	768	**498**	759	738	1192	886	565	764	769	1280

Tabla 6.3: Resultados de las heurísticas para instancias de coloreo de grafos parte 1

#	Nombre	Nodos	Colores	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
45	le450_5c	450	4	1459	1476	**909**	1422	1394	2024	1692	1083	1428	1464	2073
46	le450_5d	450	4	1464	1442	**843**	1449	1385	1981	1712	1060	1431	1436	2004
47	miles1000	128	41	52	58	**49**	51	49	133	75	49	129	70	137
48	miles1500	128	73	81	107	82	82	81	149	94	**79**	123	100	169
49	miles250	128	7	14	14	15	15	**13**	73	37	16	24	28	75
50	miles500	128	19	26	**24**	25	25	24	87	48	24	81	43	90
51	miles750	128	30	38	41	38	38	**37**	102	64	37	101	53	119
52	mugg100_1	100	3	5	12	**4**	6	5	55	19	5	12	12	57
53	mugg100_25	100	3	5	12	**4**	6	6	49	25	**4**	12	14	55
54	mugg88_1	88	3	5	10	**4**	5	5	43	19	5	9	10	50
55	mugg88_25	88	3	5	10	**4**	5	6	46	20	5	11	11	61
56	mulsol.i.1	197	48	60	68	60	60	**57**	142	76	59	131	80	153
57	mulsol.i.2	188	30	49	56	48	50	**47**	172	118	48	143	95	186
58	mulsol.i.3	184	30	50	56	48	49	**47**	177	117	48	160	99	176
59	mulsol.i.4	185	30	49	53	50	49	**48**	165	119	48	142	94	192
60	mulsol.i.5	186	30	50	62	**48**	50	50	171	114	48	107	100	191
61	myciel3	11	3	4	6	4	**3**	**3**	10	6	4	10	4	12
62	myciel4	23	4	5	9	6	**5**	**5**	24	12	**5**	15	9	27
63	myciel5	47	5	**8**	13	8	9	**8**	53	32	**8**	20	17	56
64	myciel6	95	6	18	24	**12**	19	19	135	105	13	37	51	143
65	myciel7	191	7	84	96	**18**	85	78	347	303	23	107	136	328
66	qg.order30	900	29	554	544	**152**	554	525	912	223	227	533	531	953
67	qg.order40	1600	39	1174	1105	627	1160	1078	1626	**378**	558	1131	1128	1671
68	qg.order60	3600	59	3076	2907	2425	3079	2897	3692	**831**	2234	3070	3008	3738
69	queen10_10	100	10	41	58	37	38	**35**	155	65	37	60	71	163
70	queen11_11	121	10	57	76	54	57	**50**	209	101	53	87	82	216
71	queen12_12	144	12	51	74	52	54	49	224	93	**48**	81	74	233
72	queen13_13	169	12	70	107	**67**	74	68	281	112	69	108	117	299
73	queen14_14	196	15	58	83	54	56	**53**	288	117	53	93	101	317
74	queen15_15	225	15	75	111	**71**	81	74	370	138	74	124	112	368
75	queen16_16	256	16	85	108	83	90	80	407	159	**79**	125	117	438
76	queen5_5	25	4	17	38	16	17	**15**	45	25	17	25	23	50
77	queen6_6	36	6	19	27	17	17	18	49	28	**16**	47	27	55
78	queen7_7	49	6	29	51	**27**	30	33	90	42	30	36	39	87
79	queen8_12	96	11	29	52	30	32	**25**	138	60	28	48	48	138
80	queen8_8	64	8	32	56	**25**	31	33	100	45	27	80	42	104
81	queen9_9	81	9	31	48	33	30	**27**	124	55	34	49	58	129
82	school1	385	13	997	1008	**626**	1009	987	1423	1330	691	1076	1037	1440
83	school1_nsh	352	13	723	742	**445**	735	720	1100	1048	490	734	781	1135
84	will199GPIA	701	3	438	**418**	438	453	430	1762	1542	442	547	555	1788
85	zeroin.i.1	211	48	62	72	62	61	**60**	147	80	**60**	126	89	159
86	zeroin.i.2	211	29	42	55	**41**	**41**	**41**	165	113	45	163	89	169
87	zeroin.i.3	206	29	40	55	43	40	**39**	179	111	40	162	91	164

Tabla 6.4: Resultados de las heurísticas para instancias de coloreo de grafos parte 2

	estadístico	p-valor	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
F	255.72	0	4.64	6.40	**2.18**	4.68	3.33	10.01	7.52	2.48	7.40	6.47	10.89
FA	77.86	0	374.88	461.35	**209.30**	370.94	340.22	820.66	575.50	212.84	568.01	505.86	829.45
Q	99.04	0	5.27	6.49	**1.91**	5.24	3.68	9.99	6.84	2.44	7.11	6.13	10.90

Tabla 6.5: Resultados de pruebas estadísticas para Coloreo de Grafos de Friedman (F), Friedman Alineado (FA) y Quade (Q)

2. Christofides, Mingozzi y Toth (CMT) 14 instancias, propuesto por [172].

3. Golden, Wasil, Kelly y Chao (GWKC) 20 instancias, propuesto por [173]

4. Uchoa et al. 9 instancias, propuesto por [174]

En la tabla 6.6 mostramos los resultados de las instancias y el costo del recorrido más bajo se indica en negrita; donde n es el número de nodos, Q es la capacidad de cada camión y k es el número de camiones (colores en el caso de Graph Coloring).

De forma similar al problema de coloreo de grafos, se aplicaron las pruebas estadísticas de Friedman, Friedman alineado y Quade para tener evidencia de la diferencia del comportamiento de las heurísticas. Establecimos 0.05 como un nivel de significancia y nuestro h_0 que no hay diferencias entre el desempeño de las heurísticas y como h_a que existen diferencias entre el desempeño de las heurísticas. La tabla 6.7 muestra los rangos obtenidos en las tres pruebas estadísticas.

En este caso, la heurística H_5 tiene los rangos más bajos para todas las pruebas, y H_6 tiene el segundo rango más bajo para Quade y Friedman.

6.3.3. Selección de características y clases por pruebas estadísticas

Siguiendo los pasos mencionados en la sección 5.4, primero debemos determinar el número de grupos o clases que dividen todas nuestras instancias de prueba. En este caso teniendo 139 instancias de prueba el criterio de Sturges nos arrojó un número de 8 clases.

Se consideraron 8 clases y se usó $k - means$ y se esperaba que las instancias fueran distribuidas uniformemente en los clústeres. El algoritmo de $k - means$ se aplicó con

	Instancia	Ciudades	camiones	OP	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
88	CMT1	50	5	524.61	1583	1161	1569	1568	1117	1141	1664	1576	1664	1664	1664
89	CMT2	75	10	835.26	2353	1591	2351	2349	1728	1898	2577	2547	2577	2577	2577
90	CMT3	100	8	826.14	1904	1821	1939	2017	1699	1743	3293	2249	3293	3293	3293
91	CMT4	150	12	1028.42	2578	2674	2520	2519	2347	2462	5096	3115	5096	5096	5096
92	CMT5	199	17	1291.29	3159	4469	6490	3144	3008	3155	6742	6707	6742	6692	6742
93	CMT6	50	6	555.43	1027	1160	1569	1028	941	933	1664	1573	1664	1571	1664
94	CMT7	75	11	909.68	1492	1616	2351	1465	1357	1407	2577	2517	2577	2531	2577
95	CMT8	100	9	865.95	1743	1890	1799	1906	1598	1636	3293	2203	3293	3195	3293
96	CMT9	150	14	1162.55	2308	2768	2489	2325	2185	2256	5096	2971	5096	4918	5096
97	CMT10	199	18	1395.85	4607	4607	2852	2852	2791	2870	6742	6707	6742	6513	6742
98	CMT11	120	7	1042.12	2077	2077	2372	2372	2046	2110	2608	2449	2608	2608	2608
99	CMT12	100	10	819.56	1692	1692	1733	1733	1472	1528	2306	2144	2287	2300	2306
100	CMT13	120	11	1541.14	2040	2040	2041	2041	1827	1906	2608	2476	2608	2577	2608
101	CMT14	100	11	866.37	1717	1717	1609	1609	1444	1517	2306	2171	2306	2270	2306
102	GWKC1	240	9	5623.47	9497	9155	8607	9261	7350	8565	13312	10546	13312	13312	13312
103	GWKC2	320	10	8404.61	14571	16631	15161	13399	12320	10903	20961	16801	20787	20735	21080
104	GWKC3	400	9	11036.2	19197	21734	24161	20070	17167	18793	26117	22304	26117	26117	26117
105	GWKC4	480	10	13590	27046	30204	32382	26643	27133	26504	38732	31793	38892	38892	38892
106	GWKC5	200	5	6460.98	14224	15677	13009	14207	12800	13435	19558	15763	21378	21378	21378
107	GWKC6	280	7	8412.9	15407	21043	14925	15701	14022	14539	26699	17688	26699	26699	26699
108	GWKC7	360	8	10102.7	25909	23841	25909	25909	24611	25718	25909	25909	25909	25909	25909
109	GWKC8	440	10	11635.3	21500	23969	20817	20483	18560	18677	31506	26204	31506	31506	31506
110	GWKC9	255	14	579.71	1102	989	2323	1264	954	916	2581	2319	2581	2581	2581
111	GWKC10	323	16	735.66	1344	1312	3476	1277	1222	1229	3589	3451	3589	3589	3589
112	GWKC11	399	17	912.03	1751	1919	4758	1640	1572	1485	5040	4478	5039	5039	5040
113	GWKC12	483	19	1101.5	2295	1979	6738	2462	2192	2363	5416	6520	6750	6750	6750
114	GWKC13	252	26	857.19	2010	1301	2062	2031	1475	1615	2117	2078	2158	2151	2158
115	GWKC14	320	29	1080.55	2351	1637	2637	2322	2221	2389	2748	2665	2745	2744	2748
116	GWKC15	396	33	1337.87	3262	2062	3322	3207	2793	3025	3366	3308	3366	3364	3366
117	GWKC16	480	36	1611.56	3978	2596	4107	3820	3492	3783	4198	4092	4217	4212	4217
118	GWKC17	240	22	707.76	1754	1518	2163	1729	1611	1669	2376	2165	2376	2376	2376
119	GWKC18	300	27	995.13	2130	2165	3443	2150	2073	2155	3443	3443	3443	3425	3443
120	GWKC19	360	33	1365.6	3238	2808	4540	3024	3226	3575	4713	4466	4713	4713	4713
121	GWKC20	420	38	1817.59	3519	3746	6190	3511	3189	3327	6186	5992	6191	6084	6191
122	A-n32-k5	32	5	784	1538	1476	1431	1474	1265	1255	2230	1662	2229	2229	2230
123	A-n45-k6	45	6	1733.36	1733	1733	1733	1733	1273	1156	1733	1733	1733	1733	1733
124	A-n55-k9	55	9	1073	1660	1413	1453	1642	1439	1328	2376	1901	2376	2376	2376
125	A-n61-k9	61	9	1034	2500	1814	2338	2464	1731	1838	2592	2517	2592	2592	2592
126	A-n62-k8	62	8	1288	1772	1724	2039	1792	1551	1615	3194	2219	3194	3194	3194
127	A-n63-k9	63	9	1314	2480	1879	2452	2396	1813	1835	2677	2564	2677	2677	2677
128	A-n64-k9	65	9	1401	1935	1893	2169	1948	1744	1715	2613	2203	2613	2613	2613
129	A-n65-k9	65	9	1174	2609	1940	2600	2546	1820	1892	2949	2808	2949	2949	2949
130	A-n69-k9	69	9	1159	2149	1803	2080	1979	1626	1687	3220	2477	3220	3220	3220
131	X-n148-k46	148	46	43448	71826	54940	75062	71665	68924	71179	77623	78715	79116	79116	79116
132	X-n153-k22	153	22	21220	36700	39843	52741	36155	34116	34337	71801	56820	71799	71793	71801
133	X-n157-k13	157	13	16876	36364	24507	36364	36364	27917	29210	36364	36364	36364	36364	36364
134	X-n162-k11	162	11	14138	39232	37225	36381	38162	32918	34059	79973	60657	79969	79963	79973
135	X-n367-k17	367	17	22814	71209	85326	97502	72482	68648	70027	154244	112919	154244	154244	154244
136	X-n393-k38	393	39	38260	141432	74629	152513	143398	130939	148728	158058	155203	158058	158058	158058
137	X-n401-k29	401	29	66187	140100	106590	163158	141582	141056	145043	197323	163230	197323	197323	197323
138	X-n411-k19	411	19	19718	67963	79642	81034	67090	63038	66173	173269	88152	173269	173269	173269
139	X-n420-k130	420	130	10798	182376	132745	207213	183053	203757	200726	217483	216871	217483	217483	217483

Tabla 6.6: Resultados para las instancias de VRP-Capacitated

	Estadistico	p-valor	H_1	H_2	H_3	H_4	H_5	H_6	H_7	H_8	H_9	H_{10}	H_{11}
F	420.84	0	4.49	8.79	9.69	3.53	5.59	4.16	1.60	2.46	9.32	6.92	9.45
FA	47.3	0	170.82	441.25	443.96	143.53	280.22	169.32	127.19	147.94	441.70	342.00	443.57
Q	79.97	0	4.29	8.87	9.72	3.68	5.59	3.96	1.67	2.57	9.26	6.91	9.47

Tabla 6.7: Resultados de pruebas estadísticas para Capacitated Vehicle Routing Problem para Friedman (F), Friedman Alineado (FA) y Quade (Q)

Grupo	Instancias	%	GC	CVRP	Min Nodos	Max Nodos	Min colors/camiones	max Colors/camiones
1	18	12.9	18	0	11	149	3	6
2	10	7.2	8	2	420	3600	3	214
3	11	7.9	0	11	148	480	5	46
4	5	3.6	0	5	367	420	17	130
5	19	13.7	19	0	64	256	7	30
6	31	22.3	31	0	125	900	11	122
7	11	7.9	11	0	352	701	3	14
8	34	24.5	0	34	32	483	5	33

Tabla 6.8: Resumen de Grupos/clases con el algoritmo de $k-$means

un número máximo de iteraciones = 500, con los centroides iniciales generados de forma aleatoria. Para considerar una distribución uniforme de las clases en grupos, utilizamos la distancia de Manhattan obtenida después de una prueba experimental, con los mejores resultados.

La tabla 6.8 contiene los detalles sobre las clases, el número de instancias por grupo / clase, el número de instancia para el coloreo de grafos (3ra columna) o CVRP (4ta columna) por clase, los nodos mínimo y máximo, el número mínimo y máximo de color o nodos. En esta experimentación para los clústeres 1,5,6 y 7 solo hubo casos de coloreo de grafos; los grupos 3, 4 y 8 para CVRP, y solo el grupo 2 tiene ambos tipos de instancias.

6.3.4. Diseño y prueba de la Hyper-heurística con aprendizaje fuera de línea con K-folds

El siguiente paso, es tomando los resultados de las pruebas estadísticas de Friedman, Friedman Alineado y Quade mencionadas anteriormente, formarán las características de nuestras instancias Coloreo de grafos y CVRP por clase. En esta fase, elegimos según las

jerarquías de las heurísticas que tienen una jerarquía menor a $(minrank+maxrank)/2$. Si, por ejemplo, el Friedman Alineado tiene un rango mínimo = 776,5 y un rango máximo = 3604,5, el valor a considerar para una heurística debe ser 2190,5. Debido al hecho de que la heurística 11 tiene el peor rendimiento para ambos conjuntos de datos, no se consideró esta heurística para esta fase de experimentación.

La tabla 6.9 muestra las jerarquías y la heurística correspondiente para cada clase. Por ejemplo, la clase 1 para Quade y Friedman alineado tiene las mismas heurísticas H_5, H_6, H_4, H_1y H_2, y H_3, mientras que Friedman no consideró H_3. Entonces, solo se considera H_5, H_6, H_4, H_1y H_2 y H_3 como un conjunto mínimo.

Las heurísticas seleccionadas para cada clase se pueden resumir en cinco grupos:

clase 1, 5 y 6: H_3, H_8, H_5, H_4 y H_1.

clase 2: H_3, H_8, H_7, H_5 y H_2.

clase 3: H_5, H_6, H_2, H_4, H_1 y H_3.

clase 4 y 8: H_2, H_1, H_5, H_4 y H_6.

clase 7: H_3, H_8, H_5 y H_4.

Por lo tanto, solo diseñamos 5 algoritmos para estas 8 clases. Esto significa que la Hiperheurística es la misma, pero el conjunto de heurísticas es diferente de acuerdo con grupos de clases (ejemplo clases 1, 5 y 6) . Entrenamos y probamos la Hiperheurística con los nuevos subconjuntos. Dejamos el 10 % de instancias como no visto para la Hiperheurística y los resultados se muestran en la tabla 6.10.

La configuración para la Hiperheurística fue de 10 iteraciones para búsqueda local y 100,000 llamadas a función. Para algunas instancias de coloreo de grafos, obtuvimos el número óptimo de colores (indicado en negrita en la tabla 6.10). Además, para los casos A-n45-k6, A-n55-k9, A-n62-k8, A-n64-k9, CMT13, GWKC1, GWKC2, GWKC3 y X-n148-k46 obtenemos valores cercanos al óptimo con un máximo de 20 % de distancia (ver tabla 6.11).

	Clase 1						Clase 2					
Fr	H_3	H_8	H_5	H_1	H_4	H_2	H_3	H_8	H_7	H_5	H_2	H_{10}
	1.50	1.69	3.78	4.03	4.08	6.69	2.8	3	3.4	4	5.7	6.25
A-Fr	H_3	H_8	H_1	H_5	H_4	H_2	H_3	H_8	H_7	H_5	H_2	H_4
	627	638	1089	1091	1102	1760.5	293	320	335	428	508	540
Qu	H_3	H_8	H_5	H_4	H_1	H_2	H_3	H_8	H_7	H_5	H_2	H_{10}
	1.55	1.59	3.77	4.04	4.15	6.81	2.24	2.73	2.98	4.18	5.51	6.39
	Class 3						Class 4					
Fr	H_5	H_6	H_2	H_4	H_1	H_3	H_5	H_2	H_1	H_4	H_6	H_3
	1.64	2.18	4.27	4.27	4.82	5.18	2.4	2.6	2.8	3.6	3.6	6
A-Fr	H_5	H_6	H_4	H_2	H_1	H_3	H_2	H_1	H_5	H_4	H_6	H_3
	242	283	403	418	419	568	33	74	78	79	91	128
Qu	H_5	H_6	H_2	H_4	H_1	H_3	H_2	H_1	H_5	H_4	H_6	H_3
	1.71	2.26	4.20	4.52	4.88	5.38	2.07	2.67	2.67	3.67	3.93	6
	Class 5						Class 6					
Fr	H_8	H_3	H_5	H_1	H_4	H_2	H_3	H_8	H_5	H_4	H_1	H_{10}
	1.45	1.55	3.53	4.37	4.68	6.03	1.32	1.69	4.35	4.92	5.05	5.92
A-Fr	H_8	H_3	H_5	H_1	H_4	H_2	H_3	H_8	H_5	H_4	H_1	H_2
	646	651	1144.5	1197	1225.5	2010.5	1306	1345	4112	4211	4227.50	5058
Qu	H_8	H_3	H_5	H_1	H_4	H_2	H_3	H_8	H_5	H_4	H_1	H_2
	1.47	1.53	3.67	4.44	4.69	5.78	1.48	1.53	3.90	4.55	4.60	6.47
	Class 7						Class 8					
Fr	H_3	H_8	H_5	H_4	H_1	H_9	H_5	H_6	H_2	H_4	H_1	H_3
	1.32	2.45	3.45	5.32	5.86	5.91	1.47	2.38	3.44	4.21	4.59	5.46
A-Fr	H_3	H_8	H_5	H_4	H_1	H_9	H_5	H_6	H_2	H_4	H_1	H_3
	108.50	171	477.50	578.50	637.50	642	2328	2691	2989	3847	4070	6169.50
Qu	H_3	H_8	H_5	H_4	H_9	H_1	H_5	H_6	H_2	H_4	H_1	H_3
	1.12	2.17	3.65	5.09	5.74	5.82	1.50	2.28	3.27	4.46	4.70	5.43

Tabla 6.9: jerarquías de las heurísticas por Classes, Optimo conocido (OP), Friedman (Fr), Friedman alineado (A-Fr) y Quade (Qu)

Instancia	C	Mediana	DE	Promedio	instancia	C	Mediana	DE	Promedio	Instancia	C	Mediana	DE	Promedio
1-Insertions_4	1	**4**	12.42	8.9	huck	5	**10**	4.44	11.9	ash331GPIA	7	22	283.97	161.6
2-Insertions_3	1	**3**	2.1	3.7	jean	5	**10**	5.02	12.2	DSJC500.1	7	160	212.25	490.2
2-Insertions_4	1	**4**	38.94	29.9	miles250	5	8	10.26	14.6	le450_15c	7	326	184.79	622.8
3-Insertions_3	1	**3**	4.8	4.6	miles500	5	20	11.35	25.5	le450_15d	7	321	183.32	619.5
4-Insertions_3	1	**3**	9.69	7.1	miles750	5	32	13.56	39.9	le450_5a	7	407	159.81	674.3
DSJC125.1	1	18	28.4	40.1	myciel7	5	**7**	72.64	72.2	le450_5b	7	402	167.55	671.6
mugg100_25	1	**3**	8.07	5.8	queen10_10	5	22	25.25	36.4	le450_5c	7	294	398.06	1146.6
mugg88_1	1	**3**	5.4	4.8	queen11_11	5	37	32.93	55.5	le450_5d	7	293	389.99	1155.5
mugg88_25	1	**3**	6.3	5.1	queen12_12	5	32	42.16	54.5	school1_nsH_6	7	340	171.09	658.4
myciel3	1	**3**	0.3	3.1	queen13_13	5	52	55.33	78.6	will199GPIA	7	320	412.68	547.9
myciel4	1	**4**	2.4	4.8	queen14_14	5	35	57.53	63.9	A-n45-k6	8	1820.8	260.88	1523.61
myciel5	1	**5**	5.93	7.3	queen16_16	5	60	80.19	101.2	A-n55-k9	8	1146.77	517.26	1598.48
myciel6	1	**6**	24.45	18.5	queen8_8	5	15	14.65	25.1	A-n61-k9	8	1365.76	467.86	2082.18
queen5_5	1	**4**	6.96	7.6	queen9_9	5	18	21.28	28.8	A-n62-k8	8	1409.65	767.2	2052.18
queen6_6	1	8	7.88	11.9	DSJC125.9	6	60	32.83	99	A-n63-k9	8	1582.36	437.33	2145.09
queen7_7	1	16	10.64	22.7	DSJC250.9	6	127	82	255.3	A-n64-k9	8	1509.79	454.54	1975.24
ash608GPIA	2	573	364.58	1175.8	DSJC500.5	6	247	285.35	715.3	A-n65-k9	8	1433.14	582.78	2270.86
ash958GPIA	2	1011	509.58	2190.3	DSJC500.9	6	285	205.63	627.7	A-n69-k9	8	1419.67	747.39	2120.39
DSJC1000.1	2	541	534.62	1434.6	DSJR500.1	6	21	58.93	96.3	CMT1	8	908.56	291.31	1339.03
DSJC1000.9	2	771	478.09	1541.8	DSJR500.5	6	181	97.33	318.2	CMT10	8	2468.67	1897.36	4621.8
DSJR500.1c	2	153	440.93	961.7	fpsol2.i.1	6	75	42.52	132.8	CMT11	8	1707.36	341.91	2174.43
GWKC16	2	2178.58	775.82	3343.23	fpsol2.i.2	6	42	77.33	98.3	CMT12	8	1115.92	426.34	1604.75
GWKC20	2	2858.1	1520.11	4527.67	fpsol2.i.3	6	40	79.55	99.5	CMT13	8	1548.33	418.17	2050.83
qg.order40	2	318	330.18	896.6	homer	6	14	35.18	34.8	CMT14	8	1116.7	420.25	1563.92
qg.order60	2	753	689.31	2522.4	inithx.i.1	6	68	86.46	130.8	CMT3	8	1441.42	787.3	2105.22
GWKC2	3	9428.95	4587.4	13966.73	inithx.i.2	6	45	130.31	141.6	CMT4	8	2049.16	1312.99	3099.24
GWKC3	3	13210.58	5617.19	19240.3	inithx.i.3	6	46	135.61	144.8	CMT5	8	2619.42	1841.8	4701.71
GWKC5	3	11375.74	4024.59	14761.77	le450_15a	6	85	103.65	238.6	CMT6	8	814.78	342.48	1165.11
GWKC6	3	12566.36	6118.29	17696.2	le450_15b	6	85	108.2	238.3	CMT7	8	1188.25	572.48	1800.2
GWKC7	3	13221.35	4905.24	23107.37	le450_25a	6	37	68.3	125.7	CMT8	8	1436.09	727.28	2024.26
GWKC8	3	14390.1	7054.95	21411.11	le450_25b	6	35	69.49	123.7	GWKC1	8	6023.22	3021.03	8873.57
X-n148-k46	3	50633.67	9126.38	68655.72	le450_25d	6	127	136.84	331.9	GWKC10	8	1055.68	1172.48	2200.88
X-n153-k22	3	30323.45	17172.99	47025.38	miles1000	6	43	15.39	53.3	GWKC11	8	1340.47	1671.43	2996.3
X-n157-k13	3	22174.99	5419.57	32883.27	miles1500	6	74	14.64	85.7	GWKC12	8	1744.01	2286.76	4038.1
X-n162-k11	3	28769.02	21922.43	45678.37	mulsol.i.1	6	50	17.28	61.9	GWKC13	8	1138.39	421.72	1657.68
X-n367-k17	4	65097.7	59474.96	160616.68	mulsol.i.2	6	40	28.22	58.7	GWKC14	8	1446.5	522.49	2153.6
X-n393-k38	4	67183.83	29543.56	127827.15	mulsol.i.3	6	37	26.41	57.1	GWKC15	8	1810.5	574.4	2736.95
X-n401-k29	4	97074.28	34914.6	149749.99	mulsol.i.5	6	40	27.31	58.3	GWKC17	8	1299.72	462.26	1810.28
X-n420-k130	4	125874.52	27755.61	192093.21	qg.order30	6	131	192.72	417	GWKC18	8	1623.17	837.3	2586.99
david	5	**10**	5.95	15	zeroin.i.1	6	53	17.47	65.4	GWKC19	8	2370	1031.94	3518.91
DSJC125.5	5	45	47.54	94.9	zeroin.i.2	6	34	27.88	54.8	GWKC9	8	823.14	802.58	1602.79
games120	5	8	14.41	14.2	zeroin.i.3	6	33	25.61	53.4					

Tabla 6.10: resultados de la Hiper-heurística para coloreo de grafos y CVRP. Los mejores resultados están resaltados en negrita

N	Instancia	Ciudades	Camiones	Optimo	Mejor Solución	Distancia
123	A-n45-k6	45	6	1733.36	1733	1
124	A-n55-k9	55	9	1073	1328	1.23765144
126	A-n62-k8	62	8	1288	1551	1.20419255
128	A-n64-k9	65	9	1401	1715	1.22412562
90	CMT3	100	8	826.14	1699	2.05655216
102	GWKC1	240	9	5623.47	7350	1.30702218
103	GWKC2	320	10	8404.61	10903	1.29726424
104	GWKC3	400	9	11036.2	17167	1.5555173
131	X-n148-k46	148	46	43448	54940	1.26450009

Tabla 6.11: Relación de óptimos conocidos con resultados obtenidos.

Correctly classified	114	91.20 %
Incorrectly classified	11	8.80 %
Weighted Avg		
TP Rate	0.912	
FP Rate	0.022	
Precision	0.918	
Recall	0.912	

Tabla 6.12: Resultados de la clasificación de etapa de entrenamiento

6.3.5. Entrenamiento y pruebas con el clasificador

Después de diseñar los grupos para la Hiperheurística, dividimos nuestro conjunto de datos en dos subconjuntos, uno para entrenamiento y otro para prueba. El subconjunto de entrenamientos se formó con 125 instancias con 15 características (*basic + inner*) y el subconjunto de prueba constó de las instancias no vistas fueron por 14 instancias. Los resultados de la clasificación con Naive Bayes se muestran en la tabla 6.12.

De manera detallada, la tabla 6.13 contiene la matriz de confusión de la clasificación. Observamos que para algunas clases como 3,4, 7 y 8, todos los patrones se clasificaron correctamente. Por otro lado, el resto de las clases tienen algunos patrones clasificados incorrectamente. Por ejemplo, para 3 patrones de la clase 1 fueron asignados en 5 y 6,

Class	1	2	3	4	5	6	7	8	TPR	FPR	P
1	13	0	0	0	2	1	0	0	0.813	0	1
2	0	7	0	0	0	0	0	2	0.778	0	1
3	0	0	10	0	0	0	0	0	1	0	1
4	0	0	0	4	0	0	0	0	1	0	1
5	0	0	0	0	12	5	0	0	0.706	0.019	0.857
6	0	0	0	0	0	27	1	0	0.964	0.062	0.818
7	0	0	0	0	0	0	10	0	1	0.009	0.909
8	0	0	0	0	0	0	0	31	1	0.021	0.939

Tabla 6.13: Confusion Matrix, TP Rate (TPR), FP Rate (FPR) y Precision (P) para cada clase

utilizando el mismo grupo de heurísticas de bajo nivel y esto no representa un problema para el siguiente paso de la metodología, debido a que esas instancias comparten heuristicas con la clase que se les asigno.

6.3.6. Clasificación de las instancias de prueba y aplicación de la HiperHeurística a las instancias de prueba

Finalmente, para las 14 instancias no vistas, usamos el clasificador Naive Bayes y determinamos la clase a la que pertenecen. Posteriormente aplicamos la Hiperheurística con las heurísticas correspondientes según el diseño a cada clase y se obtuvieron los resultados mostrados en las tablas 6.14 y 6.15.

La tabla 6.14 muestra la matriz de confusión TP rate, FP Rate y Precision de la prueba de clasificación para las instancias. En estos resultados, dos patrones que pertenecen a la clase cinco se clasificaron incorrectamente, pero esto no afecta la solución Hiperheurística porque esta clase comparte las mismas heurísticas con la clase 6.

Comparación estadística de la Metodología

Finalmente, para comprobar si existen diferencias entre los resultados aplicando la metodología y sin aplicar la metodología. Se hizo un experimento donde se ejecutaron

class	1	2	3	4	5	6	7	8	TPR	FPR	P
1	2	0	0	0	0	0	0	0	1	0	1
2	0	1	0	0	0	0	0	0	1	0	1
3	0	0	1	0	0	0	0	0	1	0	1
4	0	0	0	1	0	0	0	0	1	0	1
5	0	0	0	0	0	2	0	0	0	0	-
6	0	0	0	0	0	3	0	0	1	0.182	0.6
7	0	0	0	0	0	0	1	0	1	0	1
8	0	0	0	0	0	0	0	3	1	0	1

Tabla 6.14: Confusion Matrix, TP Rate (TPR), FP Rate (FPR) y Precision (P) para cada instancia de prueba

Instance	Class	Fitness	Instance	Class	Fitness
anna	1	**3**	DSJC250.5	6	27
mugg100_1	1	**3**	le450_25c	6	26
DSJC1000.5	2	83	mulsol.i.4	6	30
GWKC4	3	14255.68	school1	7	**13**
X-n411-k19	4	20005.37	A-n32-k5	8	876.19
queen15_15	5	29	CMT2	8	911.37
queen8_12	5	21	CMT9	8	1258.57

Tabla 6.15: Resultados de las instancias de prueba con Hiperheurística

33 veces la Hiperheurística con todo el conjunto de heurísticas y 100,000 llamadas a función. Los resultados se muestran en la tabla 6.16.

Primero se analizaron las distribuciones que sigue cada conjunto de resultados por clase, es decir, se aplico la prueba de saphiro wilks [175] para saber si los resultados de la metodología, Hiperheurística sin metodología (HHPC) y óptimos del estado del arte seguían una distribución normal. Lo anterior nos permitió elegir mejor la prueba estadística para comparar los resultados. La prueba se aplicó con un $\alpha = 0{,}05$ de significancia, estableciendo como hipótesis las siguientes:

h_0 : *Los datos son iguales a una distribución normal.*

h_a : *Los datos no son iguales a una distribución normal.*

Los resultados de la prueba se muestran en la tabla 6.17, se tomarón los datos mostrados en las tablas 6.10 y6.16. Cabe resaltar de que los resultados de la metodología solo para el clúster 4 y 5 fueron resultados normales; para los óptimos del estado del arte solo el clúster 4 y para HHPC los clústeres normales fueron 1, 4, 5 y 7.

Cluster	Normalidad	Cluster p_{value}	Edo arte	Edo. Arte p_{value}	HHPC	HHPC p_{value}
1	no	2.26E-05	no	2.69E-04	si	1.13E-01
2	no	0.0309	no	1.57E-04	no	7.52E-03
3	no	0.01438	no	1.80E-03	no	4.28E-03
4	si	0.3551	si	7.91E-01	si	3.29E-01
5	si	0.05466	no	1.05E-03	si	9.61E-02
6	no	1.20E-05	no	6.53E-05	no	2.25E-05
7	no	0.0255	no	3.32E-03	si	7.31E-01
8	no	4.95E-08	no	2.40E-09	no	4.59E-07

Tabla 6.17: Pruebas de normalidad para Resultados de la metodología (Clúster), Estado del Arte e Hiperheurística con pool completo

Se utilizó una prueba estadística no paramétrica, ya que en los tres grupos de datos algunos clústeres no siguen una distribución normal. Para saber que tiempo de prueba elegir, se aplica una prueba de homocedasticidad con la prueba de Bartlett [176]. Se Estableció un nivel de confianza de $\alpha = 0{,}01$, comparando por parejas (Metodología y Edo. Arte; Metodología y HHPC) y como hipótesis las siguientes:

Instancia	c	HH PC	Instancia	c	HH PC	Instancia	c	HH PC	Instancia	c	HH PC
1-Insertions_4	1	34	X-n367-k17	4	155287	le450_15b	6	242	CMT12	8	2345
2-Insertions_3	1	11	X-n393-k38	4	158696	le450_25a	6	137	CMT13	8	2769
2-Insertions_4	1	60	X-n401-k29	4	198461	le450_25b	6	140	CMT14	8	2361
3-Insertions_3	1	14	X-n420-k130	4	218854	le450_25d	6	341	CMT3	8	3335
4-Insertions_3	1	28	david	5	27	miles1000	6	72	CMT4	8	5185
DSJC125.1	1	70	DSJC125.5	5	138	miles1500	6	97	CMT5	8	6785
mugg100_25	1	15	games120	5	31	mulsol.i.1	6	74	CMT6	8	1671
mugg88_1	1	20	huck	5	22	mulsol.i.2	6	87	CMT7	8	2582
mugg88_25	1	14	jean	5	20	mulsol.i.3	6	74	CMT8	8	3355
myciel3	1	6	miles250	5	24	mulsol.i.5	6	70	GWKC1	8	13602
myciel4	1	9	miles500	5	41	qg.order30	6	351	GWKC10	8	3630
myciel5	1	26	miles750	5	51	zeroin.i.1	6	78	GWKC11	8	5057
myciel6	1	48	myciel7	5	117	zeroin.i.2	6	70	GWKC12	8	6775
queen5_5	1	26	queen10_10	5	82	zeroin.i.3	6	78	GWKC13	8	2191
queen6_6	1	29	queen11_11	5	108	ash331GPIA	7	433	GWKC14	8	2793
queen7_7	1	42	queen12_12	5	105	DSJC500.1	7	477	GWKC15	8	3413
ash608GPIA	2	746	queen13_13	5	141	le450_15c	7	618	GWKC17	8	2398
ash958GPIA	2	1157	queen14_14	5	143	le450_15d	7	643	GWKC18	8	3467
DSJC1000.1	2	1090	queen16_16	5	184	le450_5a	7	632	GWKC19	8	4756
DSJC1000.9	2	1625	queen8_8	5	50	le450_5b	7	699	GWKC9	8	2604
DSJR500.1c	2	930	queen9_9	5	69	le450_5c	7	861	A-n32-k5	T	2235
GWKC16	2	4264	DSJC125.9	6	119	le450_5d	7	792	anna	T	27
GWKC20	2	6248	DSJC250.9	6	255	school1_nsh	7	671	CMT2	T	2611
qg.order40	2	623	DSJC500.5	6	656	will199GPIA	7	898	CMT9	T	5145
qg.order60	2	2098	DSJC500.9	6	611	A-n45-k6	8	1856	DSJC1000.5	T	1765
GWKC2	3	21504	DSJR500.1	6	100	A-n55-k9	8	2474	DSJC250.5	T	336
GWKC3	3	26716	DSJR500.5	6	314	A-n61-k9	8	2681	GWKC4	T	39592
GWKC5	3	21807	fpsol2.i.1	6	137	A-n62-k8	8	3200	le450_25c	T	321
GWKC6	3	27213	fpsol2.i.2	6	136	A-n63-k9	8	2774	mugg100_1	T	17
GWKC7	3	26506	fpsol2.i.3	6	131	A-n64-k9	8	2709	mulsol.i.4	T	104
GWKC8	3	32073	homer	6	50	A-n65-k9	8	2977	queen15_15	T	164
X-n148-k46	3	80822	inithx.i.1	6	190	A-n69-k9	8	3258	queen8_12	T	52
X-n153-k22	3	72255	inithx.i.2	6	188	CMT1	8	1709	school1	T	850
X-n157-k13	3	37747	inithx.i.3	6	178	CMT10	8	6763	X-n411-k19	T	174414
X-n162-k11	3	80314	le450_15a	6	251	CMT11	8	2624			

Tabla 6.16: Resultados de la Hiperheurística con el conjunto completo y sin metodología (HHPC), Cluster (C) y T son las instancias de entrenamiento

Cluster	Metodología y Edo. Arte	p_{valor}	Metodología y HHPC	p_{valor}
1	no	1.81E-01	si	2.99E-06
2	no	3.28E-01	si	0.0479
3	no	6.45E-01	no	0.06401
4	no	5.80E-01	no	0.9019
5	si	1.11E-03	si	4.10E-05
6	no	5.59E-02	si	3.92E-05
7	si	8.63E-03	no	0.4522
8	no	4.38E-01	si	2.57E-06

Tabla 6.18: Resultados de la prueba Bartlett de homocedasticidad

h_0 : *Las varianzas de los datos son iguales.*

h_a : *Las varianzas de los datos no son iguales.*

Los resultados de las pruebas se muestran en la tabla 6.18. Podemos observar que en la mayoría de los casos para la metodología y Edo. del arte los datos tienen varianzas diferentes (solo en el clúster 5 y 7 son similares). Mientras que para la metodología y HHPC los datos tienen varianzas diferentes solo en los clústeres 3, 4 y 7. Aunando con los resultados de las pruebas de shapiro wilks, se tiene suficiente evidencia para hacer pruebas $t - student$ para las comparaciones a pares.

Se aplicaron pruebas *T-student* para Metodología Edo. Arte y Metodología e HHPC Establecimos $\alpha = 0{,}05$ como un nivel de significancia. Las hipótesis nulas y alternativas son: Para Metodología e HHPC

- h_0 : *No hay diferencias entre el desempeño de la Hiperheurística con la metodología y sin la metodología.*
- h_a : *Hay diferencias entre el desempeño de la Hiperheurística con la metodología y sin la metodología.*

Para Metodología Edo. Arte

- h_0 : *No hay diferencias entre el desempeño de la hiperheurística con el óptimo del estado del arte.*

- h_a : *Hay diferencias entre el desempeño de la hiperheurística.*

Los resultados estadísticos de las pruebas se muestran en la tabla 6.19 . Con los valores de podemos observar lo siguiente:

Metodología y HHPC se puede inferir que los resultados de la metodología son significativamente diferentes a los que tiene la Hiperheurística con todo el conjunto de heuristicas. Lo cual indica que la metodología mejoró el rendimiento y permitió limitar el conjunto de heuristicas para cada uno de los clústeres.

Metodología y Edo. Arte se puede inferir que no se encontró evidencia estadística de que los resultados de la metodología difieran con los óptimos del estado del arte exceptuando en los clústeres 5 y 7. Lo anterior se debe a que es donde se tienen más datos atípicos o que fueron mal clasificados. Lo cual abre un área de oportunidad para el refinamiento de la metodología.

	Metodología y Edo. Arte		Metodología y HHPC	
Cluster	p_{valor}	Acepta h_0	p_{valor}	Acepta h_0
1	0.1641	si	5.12E-05	No
2	0.1489	si	0.01496	No
3	0.3211	si	0.02586	No
4	0.02814	si	0.004389	No
5	0.00601	No	0.0002388	No
6	0.01144	si	0.0009709	No
7	4.43E-07	No	6.65E-06	No
8	0.06235	si	2.52E-05	No

Tabla 6.19: Resultados de las pruebas $T - student$ para Metodología y Edo. Arte y Metodología e HHPC

Adicionalmente se muestra en las figuras 6.2 y 6.3, los resultados de cada para cada uno de los clústeres por la metodología, HHPC y estado del arte. Es importante recordar que nuestra metodología es general, ya que utilizamos dos dominios de problemas

diferentes y que nuestro objetivo no es obtener en todos los casos la solución óptima. Nuestro objetivo es mejorar la selección del conjunto heurístico para la Hiperheurística con aprendizaje fuera de línea.

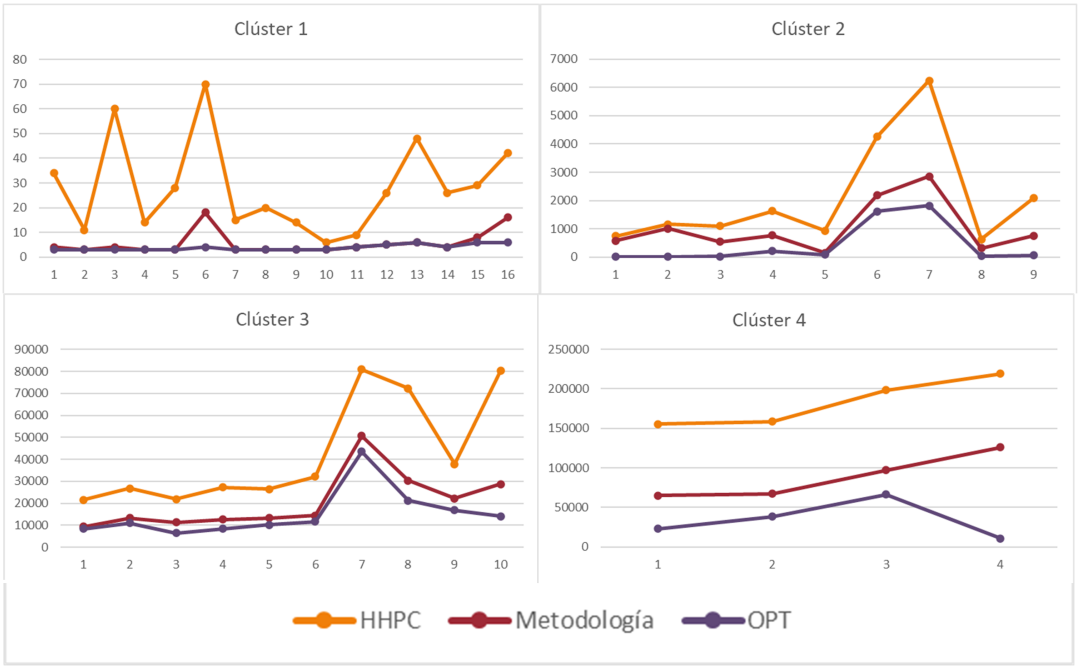


Figura 6.2: Resultados para los clústeres 1 al 4

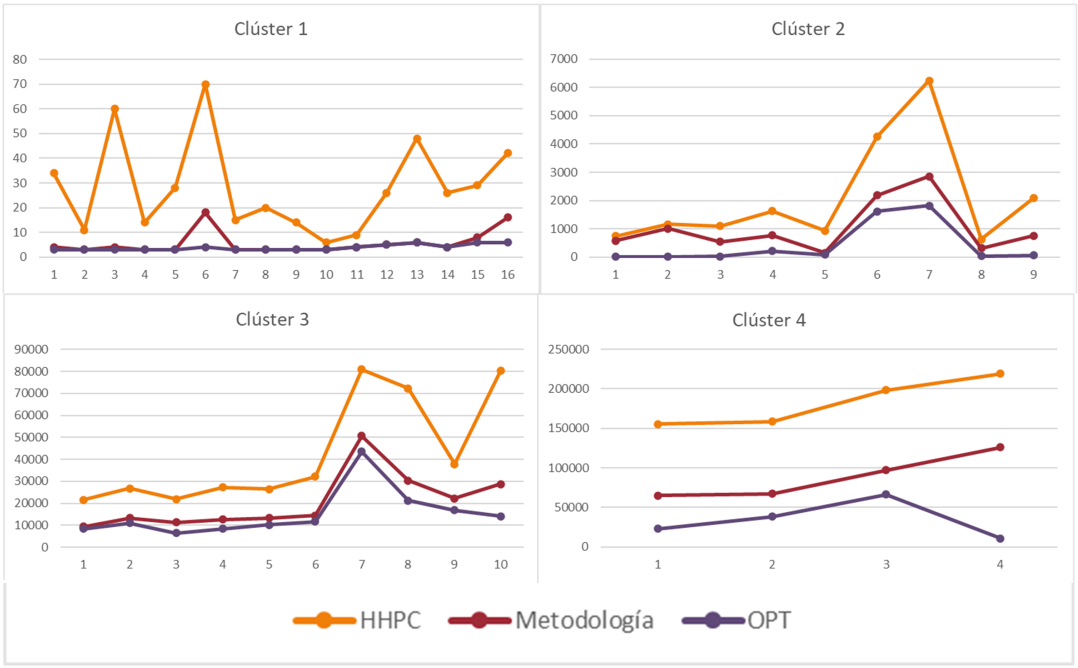


Figura 6.3: Resultados para los clústeres 4 al 8

6.4. Resumen

En este capítulo se mostraron los resultados de la investigación realizada en el estado del arte aplicando el Systematic Literature Review. Posteriormente mostramos los resultados en cuanto a la aplicación de heuristicas a instancias del estado del arte de GCP. Finalmente, se muestran los resultados de aplicar la metodología de selección de heurísticas a los problemas de Coloreo de grafos y CVRP.

Capítulo 7

Conclusiones

7.1. Resumen esquemático de la investigación

En este proyecto se desarrolló una metodología que involucra el diseño de particiones para diferentes problemas de *GCP* y selección de conjunto mínimo de heurísticas para Hiperheurística con aprendizaje fuera de línea. La trayectoria de la investigación la podemos ver en cuatro etapas:

1. Planteamiento del problema

2. Desarrollo de Metodología e implementación con técnicas Heurísticas,

3. Metaheurísticas e Hiperheurística.

4. Cierre del proyecto y reporte de resultados.

En lo que respecta al planteamiento del problema, se inició con la propuesta de investigación, así como la selección de problema a resolver. De aquí se desprendió lo que fue el protocolo de investigación, seguido de una investigación documental tanto de las técnicas como de los problemas más recientes del estado de arte relacionados a problemas combinatorios. Como resultado, se obtuvo información para realizar un Survey de los problemas de *GCP* y una propuesta de clasificación de estos.

Posteriormente, la metodología general del trabajo se plasmó en la segunda etapa. La metodología fue descrita en capítulos anteriores, una parte es la de modelar el *GCP* a particiones y una vez logrado este paso podemos pasar a resolverlo con una técnica heurística o metaheurística. En este caso se decidió enfocar a solucionar el problema de conjunto mínimo de heurísticas para una Hiperheurística, apoyado con el enfoque de meta aprendizaje.

Finalmente, en el cierre se reportan todos los productos entregables, incluyendo el trabajo realizado en la estancia de investigación en universidad de Udine, Italia. La representación gráfica de la trayectoria se muestra en la figura B.4.

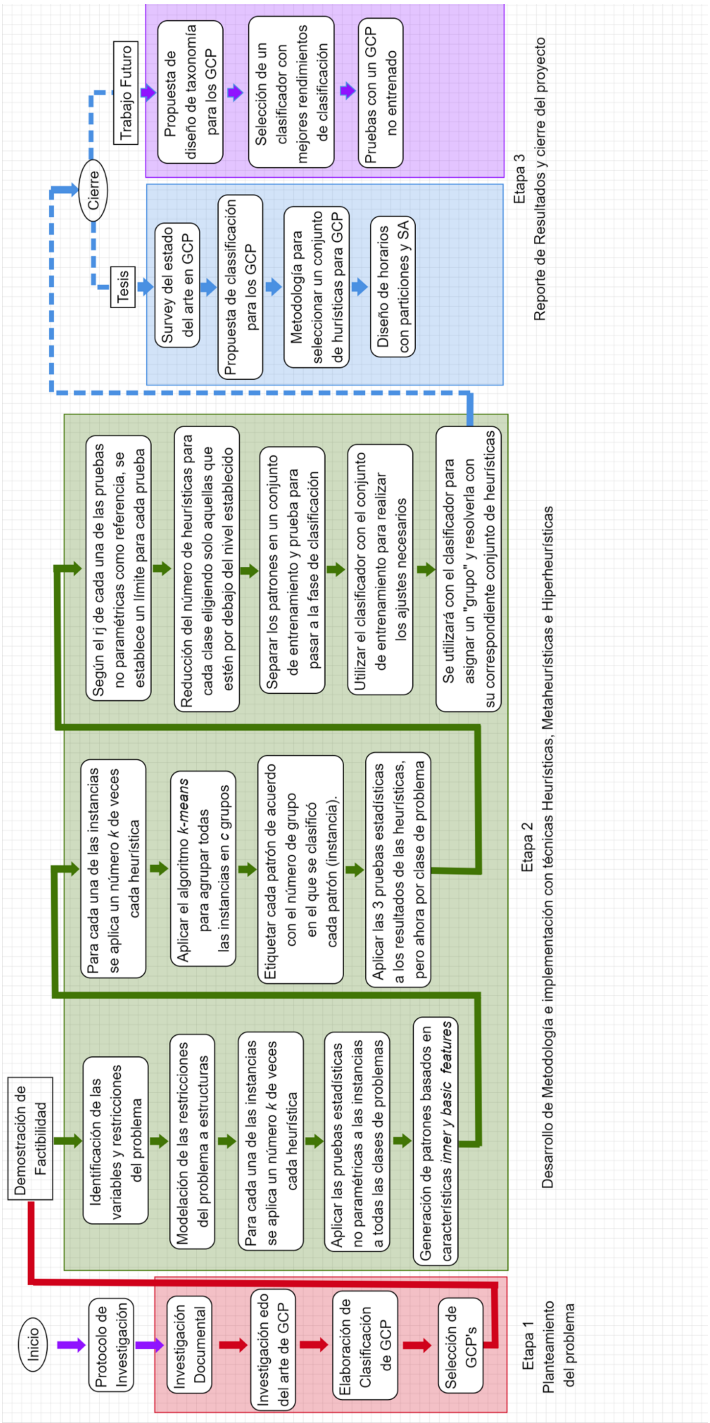


Figura 7.1: Trayectora del trabajo doctoral de investigación

7.2. Conclusiones generales

De acuerdo con los objetivos planteados en el capítulo 1, cada objetivo se puede ver cumplido en las siguiente parte de la tesis:

1. Clasificar problemas de GCP basándose en la característica de su planteamiento como estructura algebraica. La metodología para hacer esta clasificación se observa en el capítulo 6 y en el articulo *Classification of General Combinatorial Problems: a Review Guided by their Algebraic Structures*.

2. Evaluar y modelar los GCP que puedan ser utilizados como casos de uso a ser resueltos por particiones. Este objetivo fue parte del procesamiento de información y cuyos resultados fueron reportados en el capítulo 6.

3. Modelar las instancias de problemas a insumos de particiones. Este objetivo fue parte del procesamiento de información y cuyos resultados fueron reportados en el capítulo 6.

4. Generar soluciones a las instancias de GCP con la metodología de particiones. Este objetivo fue reportado en el capítulo 6.

5. Comparar estadísticamente los resultados con:

   - Los propuestos por el experto humano. Este objetivo se reportó en el artículo *Increase Methodology of Design of Course Timetabling Problem for Students, Classrooms, and Teachers*.
   - Los propuestos por el estado del arte.

   . Este objetivo se cumplió con el entregable de *A Methodology for selection and determination a subset of Heuristics for hyper-heuristics through meta-learning for solving Graph Coloring and Capacitated Vehicle Routing Problems*. 6.

6. Reportar la mejor solución para las instancias de GCP. Este objetivo fue reportado en el capítulo 6.

Las aportaciones de este trabajo fueron:

- Proponer el uso de estructuras algebraicas como criterio de agrupamiento para problemas que se conocen como diferente dominio.
- Identificar diferentes clases de problemas, a partir del uso estructuras algebraicas, definidas por Jeavons [8].
- Con el uso de particiones [45], resolver problemas de diferente clase cumpliendo por diseño las restricciones de cada uno.
- Establecer una metodología genérica con base en el uso de particiones, la cual permitió mejorar la selección de heurísticas para un tipo de enfoque hiperheurístico.

7.2.1. Clasificación de GCP

El estado del arte de los problemas combinatorios (en particular, los problemas NP-duros) incluye un desarrollo sustancial de investigación relacionada con su definición y métodos de solución. Sin embargo, en general no se encontraron criterios claramente definidos para la clasificación de estos problemas en esos estudios. Esto motivó el presente trabajo, en el que se presentó una revisión de los *General Combinatorial Problems*, partiendo del análisis de Jeavons de dichos problemas en términos de estructuras algebraicas [8].

El uso de dicho análisis nos permitió identificar los problemas, subproblemas, variedades y las subvariedades que pertenecen a cada clase de problema de GCP que no se habían categorizado previamente como tales. Porque los trabajos revisados no utilizaron un criterio formal para respaldar su categorización.

7.2.2. Metodología de Selección de Heurísticas

En relación con la selección de heurísticas, se propuso una metodología con enfoque de particiones en combinación con meta-Learning y pruebas estáticas para seleccionar el mejor grupo de heurísticas para Hiperheurística.

El primer paso en nuestra metodología fue generar los insumos donde se modelan las restricciones de cada variable. En este paso podemos identificar cuales problemas pueden ser susceptibles a ser resueltos por particiones.

Como segundo paso hizo un agrupamiento las instancias y seleccionar las mejores heurísticas para la Hiperheurística de acuerdo con estas clases o subconjuntos. Para predecir las mejores heurísticas e Hiperheurística de grupo o para una instancia determinada, utilizamos el clasificador Naïve Bayes utilizando nuestro conjunto de datos de prueba. Este proceso implica la fase previa de generación de características (**basic + inner**) y, finalmente, el Naive Bayes predijo la clase y las heurísticas del grupo correspondiente para cada instancia. Luego, el conjunto de instancias de entrenamiento se probó en la Hiperheurística. Con el último paso, podemos reducir nuestro conjunto de heurísticas de acuerdo con cinco grupos diferentes y diseñar un mejor algoritmo de acuerdo con las características y resultados anteriores de cada heurística.

Una tarea importante para los investigadores y los tomadores de decisiones es conocer la mejor técnica o algoritmo para una instancia de problema dado, ya que es una tarea difícil y compleja. La metodología con uso de particiones propuesta abona a este punto.

Nuestros resultados experimentales muestran que, para cada uno de los 8 clústeres, son significativamente mejores los resultados del uso de la metodología de selección de heuristicas a que si no se aplicará. Este punto es un tema de investigación que esta abierto, vigente y en lo que este proyecto de investigación abona a ese tema.

Sin embargo, para complementar la investigación y como paso para probar que se tiene una metodología que es competitiva con las mejores soluciones que se encuentran en el estado del arte, se hizo una comparación estadística con los óptimos. De los resultados obtenidos en las pruebas estadísticas, podemos inferir que no hay suficiente evidencia estadística para decir que los resultados obtenidos de la Hiperheurística que usa la metodología están por debajo de dichos óptimos. Es preciso señalar que en dos clústeres no se obtuvieron buenos resultados y esto de acuerdo con el teorema de no-free luch. Lo cual nos abre la pauta a tener un área de oportunidad de investigación para mejorar estos resultados.

Capítulo 8

Trabajo Futuro

Se identificaron las siguientes líneas de investigación relacionadas al trabajo mostrado: desarrollo de una investigación para la clasificación de problemas de GCP, Refinamiento de la metodología de selección de heurísticas con particiones y aplicación de la metodología de selección de heuristicas a modelos con aprendizaje en línea.

En cuanto a la revisión y clasificación de los diferentes GCP, sus subproblemas y variedades, se requiere evidenciar a través de la identificación de características comunes bajo un marco matemático. Lo anterior con fin de avanzar en dirección a formalizar una base para elaborar una clasificación o taxonomía. Además de sugerir los criterios objetivos que deberían respaldar la definición formal de cada subproblema, variedad e instancias lo cual posibilite la identificación de las metodologías de solución óptimas para cada uno.

Se propone como trabajo futuro, utilizar este enfoque para seleccionar el conjunto de heurísticas a otros enfoques de heurísticas como construcción e hiperheurístico de aprendizaje en línea. Como segundo paso, se debe realizar una investigación más profunda para describir o mejorar la caracterización de cada instancia. Es decir, mejorar la caracterización de las características (*inner y basic*). Para aquellos problemas con las siguientes características: problema de optimización combinatoria cuyo objetivo sea dividir sus variables en diferentes grupos cuya característica sea Partición y problemas en el que al menos se necesite formar 3 partes de variables, aplicar la metodología y

comparar los resultados con los de otros algoritmos. Algunos de estos problemas son: cobertura de vértices, *car sequencing, nurse rostering*, CVRP con ventanas de tiempo entre otros.

8.1. Áreas de oportunidad

Después de realizar este trabajo de investigación, estudio y análisis exhaustivo del problema, se observan áreas de oportunidad:

- Para cada uno de los clústeres, realizar una reclasificación de aquellos patrones atípicos o que causan ruido en la clase.
- Establecer un criterio formal que permita uniformizar el mínimo de instancias para todos los clústeres.
- Considerar el aprendizaje para patrones no vistos y/o modelados en todo el proceso de entrenamiento.
- Aplicar el modelado de particiones a problemas reales y revisar si se extiende la generalidad de la metodología.
- Usar un método de componentes principales para determinar cuáles y cuentas son la heuristicas base para una Hiperheurística.
- Finalmente, consideramos relevante mejorar nuestro enfoque utilizando otros clasificadores

Apéndice A

Tablas de Instancias de GCP

Hamiltonian Circuit

Name	I	N_{max}	E_{max}	Reference
hc 1-8	8	200	1,250	} 53.5 mm
np 10-90	9	90	8,010	
nv50a-70a 240 -580	16	70	540	di.univaq.it
4xp20	4	80	396	
2xp30	5	60	318	

Clique

Name	I	N_{max}	E_{max}	Reference
brock 200-800	12	800	208,166	} 73.5 mm
C 125-2000	5	200	1,799,532	
DSJC1000_5	1	1,000	499,652	
DSJC500_5	1	500	125,248	cse.unl.edu
gen 200-400	5	400	71,820	
hamming	6	-	-	
johnson	4	32	-	
c-fat 200-500	7	500	-	[177]
keller	3	3,361	4,619,898	[178]
p_hat 300-1500	15	1,500	847,244	[179]
san 400-100	12	100		[180]
MANN	4	3,321	5,506,380	turing.cs.hbg.psu.edu

Tabla A.1: Graph colorability, Vertex Cover, Hamiltonian Circuit and Clique instances. Including the number of instances I, maximum number of nodes N_{max} and maximum number of edges E_{max}.

Graph Colorability

Name	I	N_{max}	E_{max}	**Reference**
DSJC125-1000	15	100	898,898	[181]
Le450 05-25	12	450	17,425	[182]
anna	1	138	493	} 163 mm
david	1	87	406	
fat	6	1,000	246,708	
fpsol2	3	496	11,654	
games120	1	120	638	
homer	1	561	1,629	
huck	1	74	301	
inithx	3	864	18,707	mat.gsia.cmu.edu
jean	1	80	254	
latinsquare10	1	900	307,350	
miles750-1000	5	128	3,216	
mulsol1-5	5	197	3,973	
Myciel3-7	5	191	2,360	
queen5-16	13	256	12,640	
school1-1nsh	1	385	19,095	
zeroin	3	211	4,100	

Vertex Cover

Name	I	N_{max}	E_{max}	**Reference**
frb 30-59 - 15-26 mis	25	1,534	1,475	nlsde.buaa.edu.cn

Tabla A.2: Graph colorability, Vertex Cover, Hamiltonian Circuit and Clique instances. Including the number of instances I, maximum number of nodes N_{max} and maximum number of edges E_{max}.

Bandwidth

Name	I	N_{max}	E_{max}
Path 20-1000	15	1000	999
cycle 20-100	15	1000	1000
mesh2D-3D	15	2197	1930
tree	12	1023	1022
caterpillar	15	1034	1033
hypercube	3	8192	53248
can_24.mtx	1	24	68
jgl009.mtx	1	9	32
jgl011.mtx	1	11	49
rgg010.mtx	1	10	45
A-pores_1.mtx	1	30	103
B-ibm32.mtx	1	32	90
C-bcspwr01.mtx	1	39	46
D-bcsstk01.mtx	1	48	176
E-bcspwr02.mtx	1	49	59
F-curtis54.mtx	1	54	124
G-will57.mtx	1	57	127
H-impcol_b.mtx	1	59	281
I-ash85.mtx	1	85	219
J-nos4.mtx	1	100	247
K-dwt__234.mtx	1	117	162
L-bcspwr03.mtx	1	118	179
M-bcsstk06.mtx	1	420	3720
N-bcsstk07.mtx	1	420	3720
O-impcol_d.mtx	1	425	1267
P-can__445.mtx	1	445	1682
Q-494_bus.mtx	1	494	586
R-dwt__503.mtx	1	503	2762
S-sherman4.mtx	1	546	1341
T-dwt__592.mtx	1	592	2256
U-662_bus.mtx	1	662	906
V-nos6.mtx	1	675	1290
W-685_bus.mtx	1	685	1282
X-can__715.mtx	1	715	2975

Tabla A.3: Bandwidth instances, including the number of instances I, maximum number of nodes N_{max}, and maximum number of edges E_{max}. These instances can be found online at: `optsicom.es` and `tamps.cinvestav.mx`

TSP

Name	Reference
Benchmark TSP LIB	{ 23 mm iwr.uni-heidelberg.de math.uwaterloo.ca

VRP

Name	Reference
Benchmark Vehicle routing	neo.lcc.uma.es

Satisfiability

Name	Reference
Model RB Forced Satisfiable CSP and SAT	nlsde.buaa.edu.cn
Benchmarks SAT	aloul.net
Software Verification	cs.ubc.ca

Knapsack

Name	Reference
Data for the 01 Multiple Knapsack Problem	{ 33 mm [183] [184] people.sc.fsu.edu
Data for the 01 Knapsack Problem	people.sc.fsu.edu

Location

Name	Reference
Capacitated Facility Location Problems	[185]
Beasley	[186]
Real data instance , Italian cookie factory	di.unipi.it

Scheduling

Name	Reference
Resource-Constrained Project Scheduling (RCPS)	} 23 mm om-db.wi.tum.de
RCPS with time-dependent resource capacities	
RCPS with Minimal and Maximal Time Lags	} 33.5 mm
Multi-Mode RCPS with Min./Max. Time Lags	wiwi.tu-clausthal.de
Resource Investment with Min./Max. Time Lags	
Multi-Mode RCPS (MRCPS)	[187]

Timetabling

Name	Reference
ITC2007	cs.qub.ac.uk
ITC2002 PATAT	sferics.idsia.ch
Benchmark Data Sets in Exam Timetabling	{Qu:2009survey cs.nott.ac.uk

Tabla A.4: TSP, VRP, Satisfiability, Knapsack, Location, Scheduling, and Timetabling instances.

Graph Isomorphism

Name	**Reference**
Graphs for Practical Graph Isomorphism	[188]
	www.lii.rwth
Nauty and Traces	[189]
	pallini.di.uniroma1.it
bliss: A Tool for Computing Automorphism Groups	www.tcs.hut.fi
Algorithm conauto for Graph Isomorphism Testing	sites.google.com
Saucy3: Fast Symmetry Discovery in Graphs Sakallah	vlsicad.eecs.umich.edu

Cutting Stock (CS)

Name	**Reference**
Standard one-dimensional CS (single stock length)	[190]†
One-dimensional CD (multiple stock lengths)	[191]†
The hard28 set for 1D-BPP	[192]†
One-dimensional setup minimization	[193]†
Two-dimensional two-stage constrained cutting problem	[192]†
Two-dimensional strip-packing problem	[194]†
3D OPP	[195]
Two-dimensional orthogonal feasibility problems	[196]
Optimal Clustering Problem	[197]
Hard selection for 1D-CSP and 1D-BPP	www.math.tu-dresden.de
Set of 53 non-IRUP instances	www.math.tu-dresden.de

Tabla A.5: Graph Isomorphism and Cutting Stock instances. †Dataset available from: www.math.tu-dresden.de

Apéndice B

Gráficos de caja y bigote

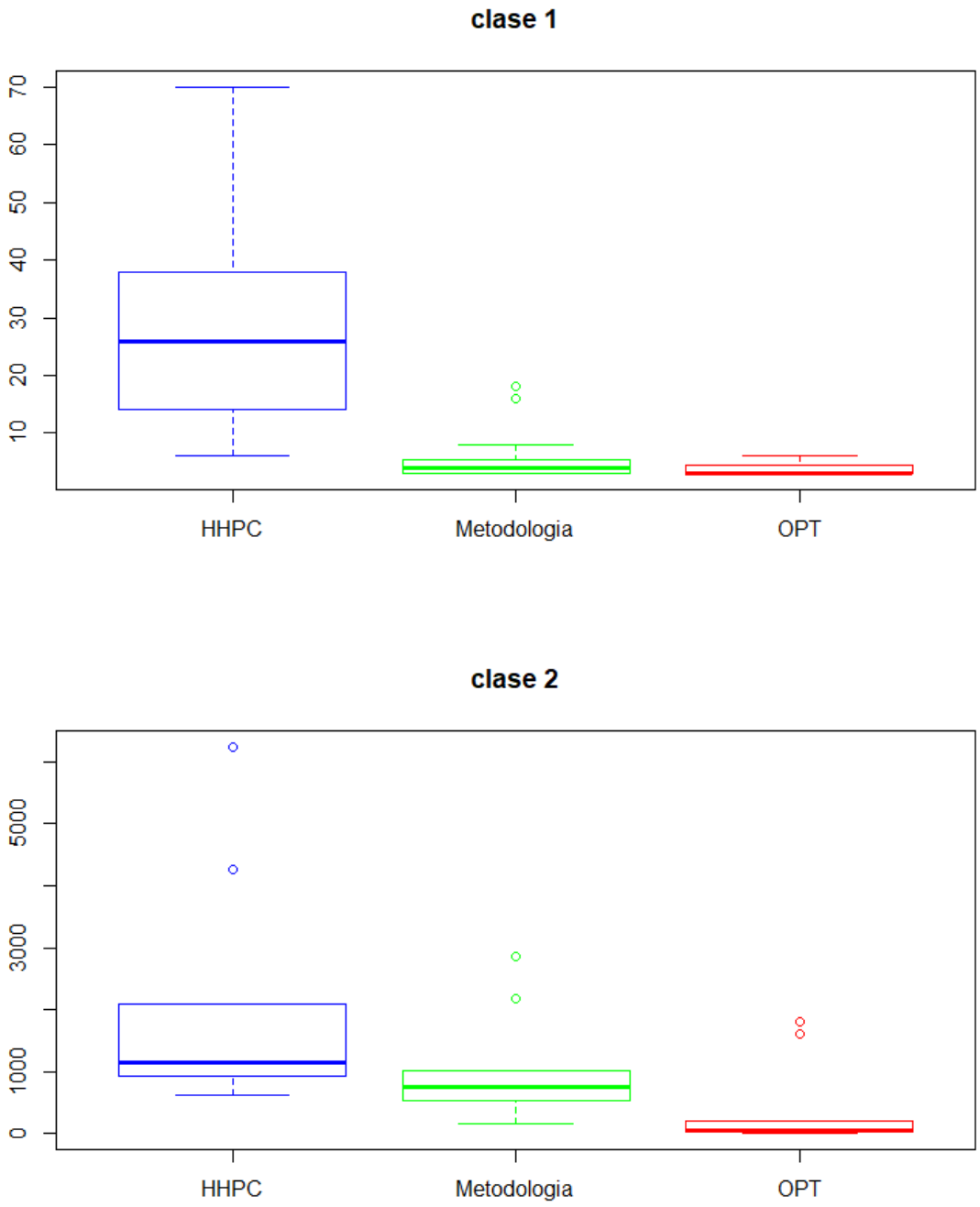


Figura B.1: Resultados de la clase 1 y 2

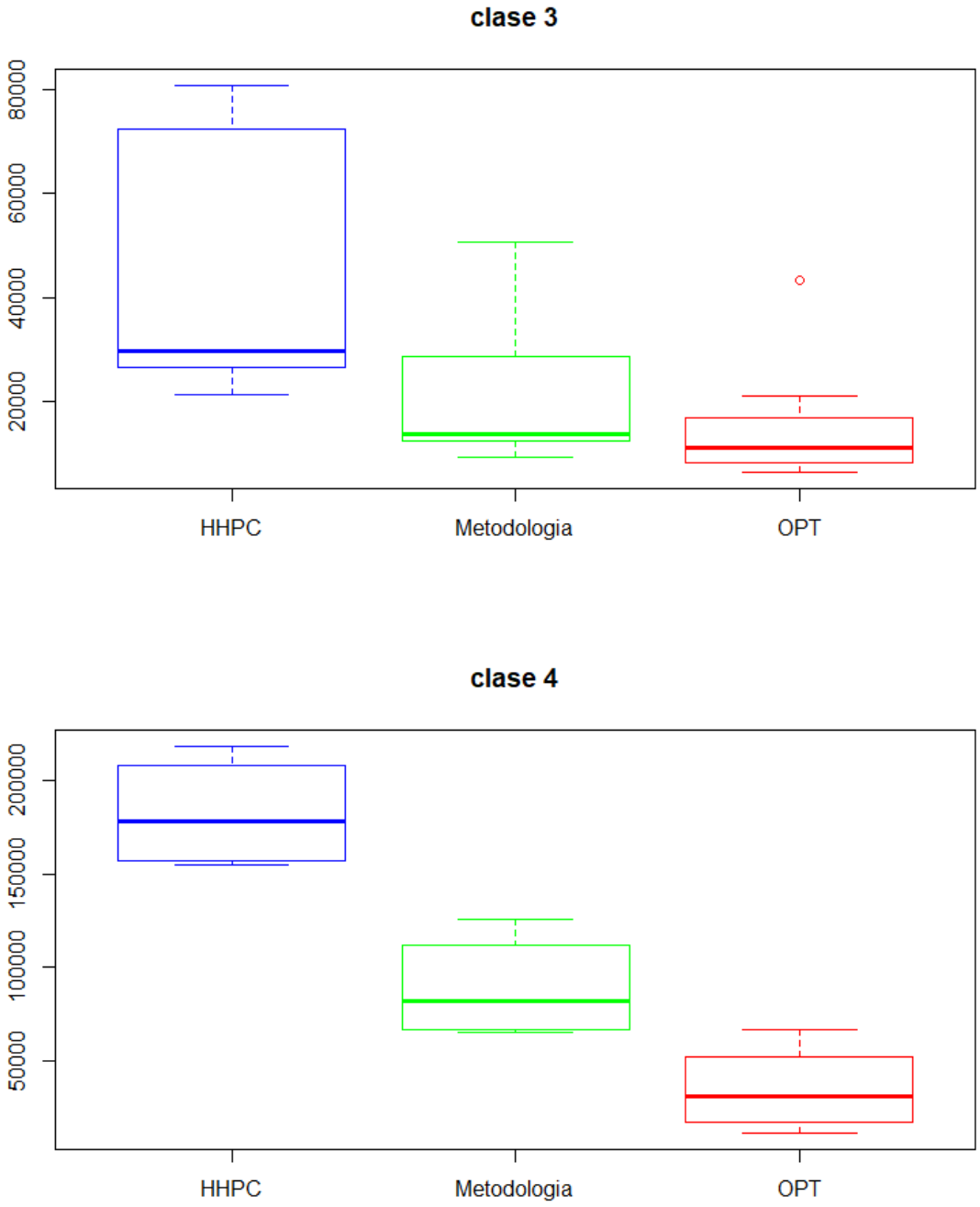


Figura B.2: Resultados de la clase 3 y 4

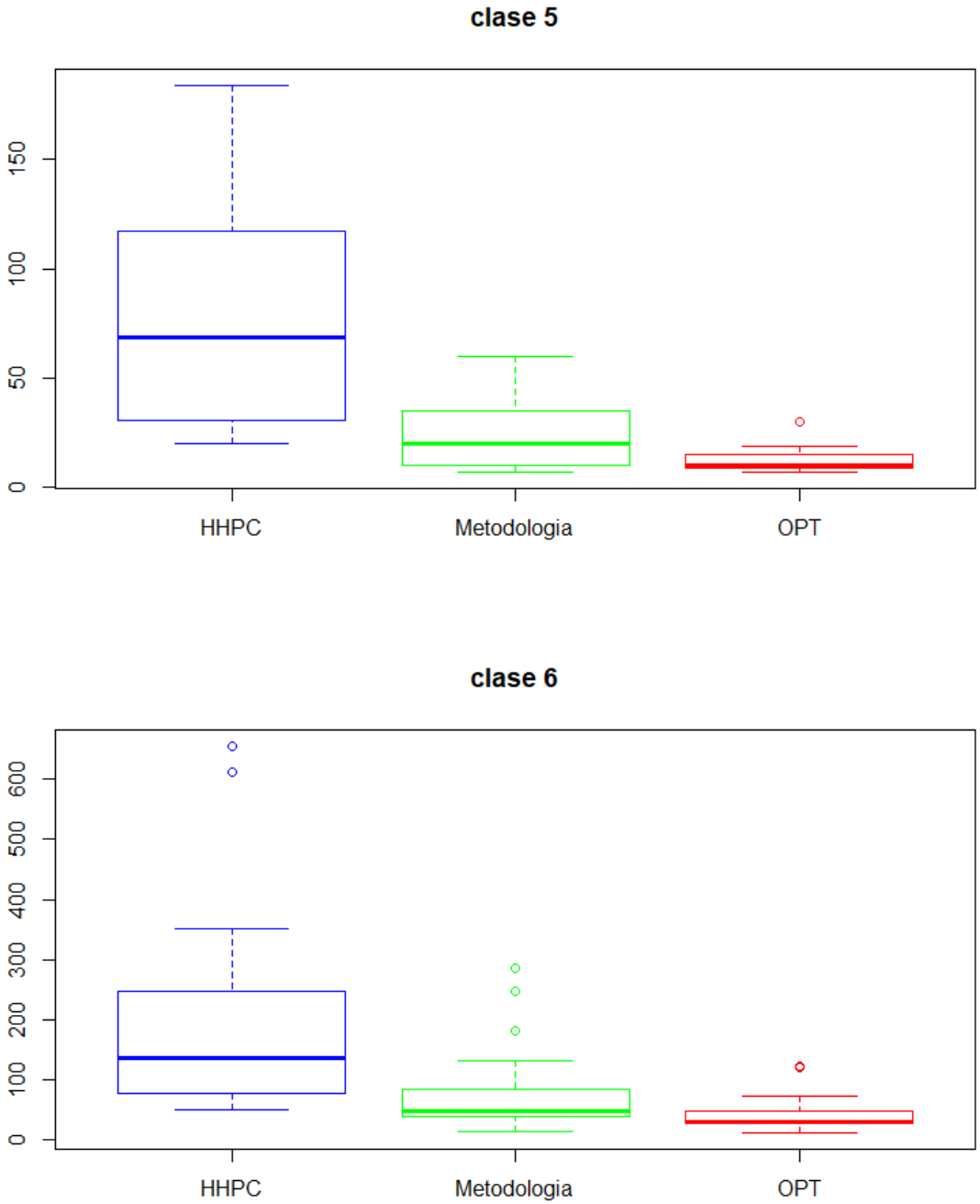


Figura B.3: Resultados de la clase 5 y 6

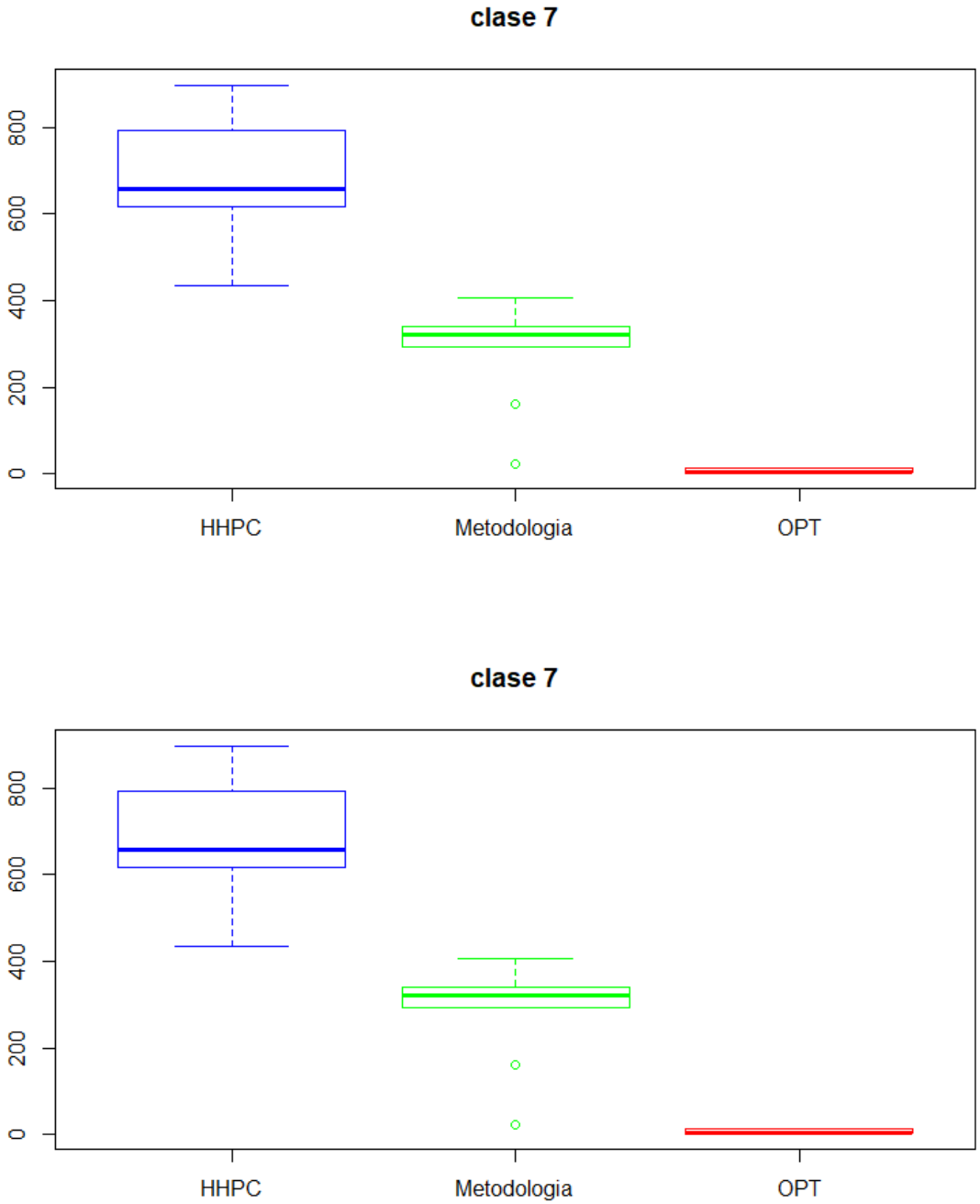


Figura B.4: Resultados de la clase 7 y 8

Classification of General Combinatorial Problems: a Review Guided by their Algebraic Structures

Lucero Ortiz-Aguilar[a], Martin Carpio[a], Alfonso Rojas-Domínguez[a], Manuel Ornelas-Rodriguez[a,*], Hector Puga [a], Adolfo Montesino-Guerra[a] and Jorge A. Soria-Alcaraz[b]

[a]*División de Estudios de Posgrado e Investigación, Tecnológico Nacional de México/I. T. León, León Guanajuato, México*
[b]*Universidad de Guanajuato, División de Estudios de Ciencias Económico Administrativas*

ARTICLE INFO

Keywords:
Combinatorial Problems
Classification of Combinatorial Problems
Constraint Satisfaction Problems
Algebraic Structures
Optimization Problem

ABSTRACT

In Computer Science there is a wide range of optimization problems; one specific type is the General Combinatorial problem (GCP), where the aim is to generate an assignment of a finite set of objects while satisfying a set of restrictions. A great amount of effort has been devoted to investigate the best algorithms for solving instances from different combinatorial problem domains; however, in the literature there is a lack of an adequate classification of the GCPs. The aim of this study is to present a review of the GCPs described in the literature, guided by the analysis of their algebraic structures due to Peter Jeavons (1998). This review includes the mathematical definition of the GCPs in terms of algebraic structures, historical precedents, models and benchmark instances. Through the identification of the different generality levels, a hierarchical structure of the GCPs (including problems, sub-problems and their varieties) is presented. This hierarchization will enable researchers to visualize the relationship between the different formulations and may help to design, select and implement algorithms for solving instances of different combinatorial problems in a more efficient manner.

1. Introduction

Since the beginning of research in the area of computer science Garey1979, the enthusiasm for combinatorial research has been growing considerably as well as the theory and application of algorithms that provide solutions to various problems. To develop approaches for generating and searching problem solutions have been the objective of many investigations in the literature HHSurvey2013,Malan2013,Hussain2018, leading to the formulation of new optimization problems solvers over time. Although, nowadays a few dozens of such proposals can be distinguished, their relationship to one another has been only rarely studied. As a consequence, a consensus cannot be observed in the state of the art regarding a definite classification or grouping of the different optimization combinatorial problems.

In fact, just the opposite can be argued from an inspection of the literature, since it can be found that different studies have been carried out from very different perspectives and have contributed their findings and techniques to a cluster of information that continues to grow without a clearly defined direction or guidance.

For instance, when burke5 proposed an algorithm for "Timetabling" problems, their experimentation was focused on Educational Timetabling only, leaving out other varieties, like Rostering. Later on, raey3 proposed a new algorithm for Educational Timetabling, and observed that they were aware that Educational Timetabling is just a variety of Timetabling problems, which seems to be in disagreement with their earlier criterion in burke5. In other words, often the term *Problem Instance* is used indistinctly in reference to a specific case or to a whole set of problems, making it confusing for new researchers to assess the level of generality that certain algorithms possess. In turn, this ambiguity in how general these algorithms are, makes it hard to determine the significance of the results generated for a certain set of problem instances or family of problems in the search of *Optimal* solutions. Most of the research in the area of Optimization Algorithms focuses on designing an *ad-hoc* tool for certain problem instances.

Due to the complexity of optimization problems, several groupings of said problems exist in the literature. Often, optimization problems are grouped together based on characteristics such as: their restrictions (constrained or unconstrained Shubin19942), their objective function to be minimized or maximized, their solution process (Linear/Quadratic/Nonlinear Programming), and whether their domain is discrete or continuous Bhatti2000. Some combi-

*Corresponding author
ORCID(s):

natorial problems that have been a particular focus of research are: Graph Colorability golovach2017survey,pardalos1998graph,Ji Timetabling babaei2015survey; Scheduling zarandi2018state, Clique mascia2012analysis,bomze1999maximum,szwarcfiter200 Vertex Cover Boria2014; k-Dimensional Matching bowyer2004survey; Hamiltonian Circuit gould1991updating,witte1984survey Bandwidth lai1999survey,chinn1982bandwidth; Graph isomorphism V-1985,Derksen-2013,9781461203339; Undirected Graph Reachability 1454786,parlangeli2012reachability; Satisfiability Schaefer1978,du1997satisfiability and Constraint Satisfaction (CSP) pearson1997survey,krokhin2017dfu,Cooper2017. In these works, the problems have been studied mainly with a perspective of grouping them into so-called families (for instance, Scheduling, VRP and Timetabling all belong to the same familiy brailsford1999constraint), but without providing a formalism used to relate the different problem families with each other.

It is also possible to group optimization problems based on the method used to search for their respective solutions. Most studies in the Computers and Operational Research (COR), from 1980 to the present, were focused in Constraint Satisfaction Problems, and these were classified according to characteristics and performance of the techniques used to solve it. In fact, these problems have been studied mainly from a perspective based on solution-searching methods brailsford1999constraint, HHSurvey2013,Pillay2014, zarandi2018state, choong2018automatic, Hussain2018, rather than on a formal mathematical relationship. Grouping the problems by the level of generality of the solution searching approach results too ambiguous, because the solution approach required may be different depending on the characteristics that each combinatorial problem may present. For example, some instances of TSP can be efficiently solved by means of the Concorde Algorithm Cook2012, but as the number of cities to be visited and/or the number of restrictions increase, different solution techniques are required, such as constructive or perturbative Heuristics and Metaheuristic (ACO, Evolutionary Algorithms, intensifiers, etc. Talbi). Although we known that VRP belongs to the same family of problems as the TSP, using the same solution algorithm on both problems would require significant modifications to the algorithm and the problems inputs. The non-deterministic nature of many solution techniques may also introduce an extra degree of variability into a classification solely based on the solution-searching methods.

Several works focus on creating varieties of algorithms that can solve instances of problems in different domains. A study by Hussain2018 presents a review (from 1983 to 2016) of approximately 140 metaheuristics for the solution of different problems. Some of these metaheuristics were tested on supposedly different problems, until now there is no formal basis to show that these are different problem domains rather than simply different instances of the same problem (or of the same family of problems).

Based on what has been discussed above, authors consider it is necessary a better classification of problems designed around formal mathematical foundations that can be employed to better differentiate between levels of generality of both the problems and their related solution techniques. A classification of problems should not be limited to simply describe them; it would allow practitioners to distinguish the scope and the generality of the approaches used on those problems in the search for solutions. In other words, said classification could enable practitioners to know whether they are dealing with generally applicable methods, or rather, specialized algorithms for a certain family of problems.

A classification can lead to the development of a broader and deeper understanding of these problems, so that researchers may identify the generality of their solution proposals in a confident manner and avoid the use of over-complex algorithms that can in turn complicate their implementation, experimentation and analysis of the results. Furthermore, based on a problems classification one can perform comparisons between algorithms in a fairer manner by better identification of the nature of the problem instance to be solved.

In this work, we present a Review of General Combinatorial Problems (GCPs) and exemplify their classification based on algebraic structures, using a previously proposed guideline from Jeavons1998. We focus on the optimization problems of discrete cases, and specifically those of a combinatorial nature, with the aim to extend each of the problem domains defined according to the classification made by Jeavons. Our approach is not meant to be final or exhaustive and indeed it has limitations. However, we consider that this work constitutes a starting point with clear contributions towards a formal taxonomy, including:

1. A review of ten GCPs and their subproblems, including data repositories.
2. An extended classification of GCPs based on Jeavons' analysis, including the use of a formal criterion to support said grouping.
3. An analysis of the current relevance of the problems based on their frequency of appearance within search results of six different scientific search engines.

The rest of this work is structured as follows: previously existing related work is discussed in Section 2. Our methodology for literature review and hierarchization of the GCPs and its sub-problems is described in Section 3.

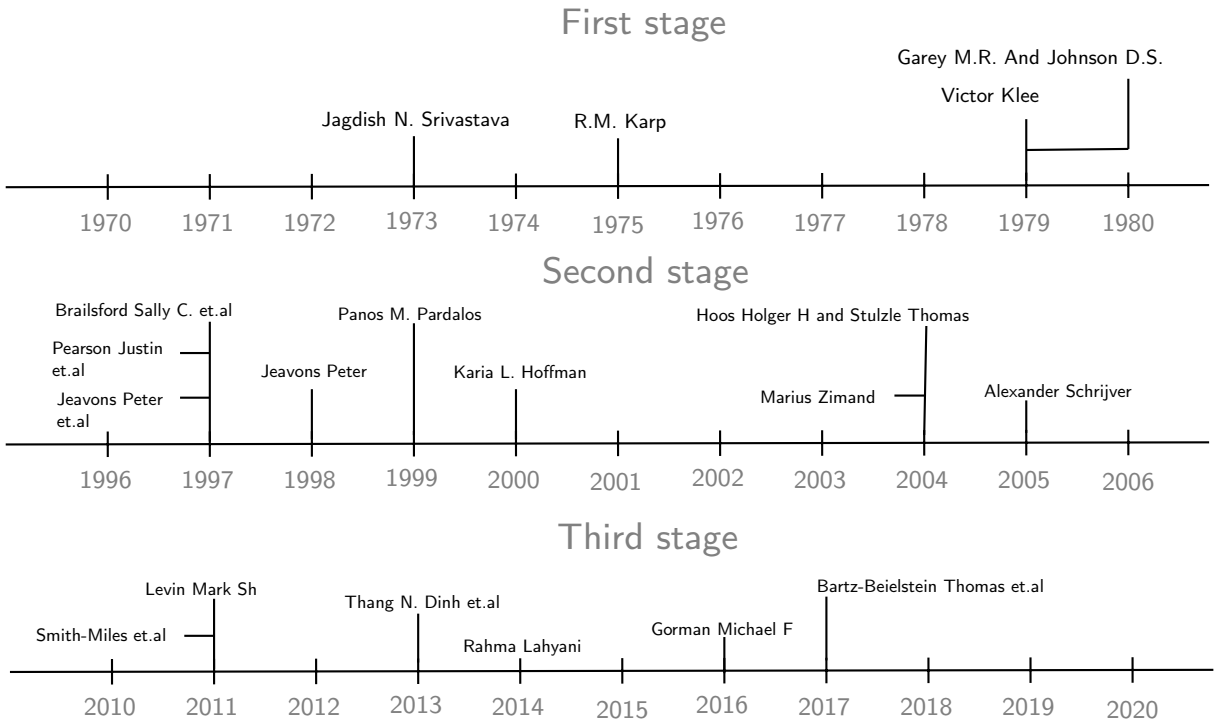


Figure 1: Time line of related work on classification of GCPs. Three historical stages are shown from top to bottom: works developed in the 1970's, works in the late 1990's-early 2000's and 2010-present.

Our review and extended classification of ten GCPs is contained in Section 4. Results and discussion are presented in Section 5. Finally, conclusions and directions for future work are offered in Section 6.

2. Related Work

In the literature related to this research, three historical stages can be distinguished according to the main contributions of the publications. Fig. 1 shows a time-line of the most important contributions related to this work.

First Stage (1973-1979).- In the first stage, four important works are identified GOETHALS1973, karp1972b, Garey1979, Klee1979, which were aimed at defining the theoretical and mathematical basis of the GCPs.

The first research related to this work is a survey of combinatorial theory conducted by GOETHALS1973. His work mentioned topics related to graph theory, Latin squares, block designs, and their construction, partitions, lattice, characterization problems of the combinatorial graph, combinatorial search problems, problems in combinatorial geometry and a survey of graphical enumeration problems. This research provided a basis for a theoretical and mathematical understanding of combinatorial problems.

Karp defined the computational complexity of some combinatorial problems such as Graph Colorability, SAT, Clique, 3-Dimensional Matching, Cubic Hamiltonian Circuit, Graph Partition, among otherskarp1972b.

Garey & Johnson published one of the most important works in the area of computer science about the theory of NP-completeness Garey1979 where they defined several of the GCPs included in our review.

In the same year, Klee1979 defined the term *combinatorial optimization*, provided the landmark of the theory and reviewed the most significant advances to that date and possible directions of the research area.

Second Stage (1997-2005).- The second stage is characterized by containing the first proposals of CSP classifications and works based on surveys of GCPs. The research by pearson1997survey,Jeavons1998,Jeavons1998a analyzed the different combinatorial problems in terms of algebraic structures, forming the basis for a classification.

Brailsford proposed a classification of CSPs such as Location, Scheduling, Cutting Stock, Car Sequencing, Rostering and Timetabling brailsford1999constraint. Their investigation served as the starting point of CSP grouping for the review presented in this work for the particular Constraint Satisfaction Problem (CSP).

At the end of the nineties, the first edition of the Combinatorial Optimization Handbook Pardalos1999 was published, with the theoretical and technical revision of some combinatorial problems.

Starting from the year 2000, the study of combinatorial optimization problems has generated considerable interest. Researchers have focused their attention on the revision of the heuristics, metaheuristics and other techniques able to provide solutions to combinatorial problems. In this stage, there were a few studies oriented to the classification and definition of combinatorial problems.

Hoffman reviewed the developments of the time and the possible topic of future research Hoffman2000. In hoos2004stochastic a chapter was dedicated to the combinatorial problems, their computational complexity, paradigms and related works. In the same year, 2004225 published a book chapter focused on combinatorial optimization theory, which enriched the theory related to the combinatorial problems at that time.

Finally, Schrijver2005 elaborated a historical review of the combinatorial optimization until 1960. He mentioned that combinatorial optimization, as a mathematical discipline, discipline is relatively young because its beginnings date from once linear programming is established in 1950's.

Third Stage (2010-present).- In the third stage, an increasing number of proposals for new algorithms or models to solve different combinatorial problems are found. Furthermore, some investigations focused on the generation of a taxonomy for Rich Vehicle Routing Problems (RVRP).

In smith2012measuring the authors reviewed hardness-revealing features and discussed the right metrics for each problem. Also in 2011, an investigation directed by levin2011four, proposed a four-layer framework for solving combinatorial optimization problems. Levin proposed some so−called *basic models* such as knapsack, multicriteria ranking, assignment, etc. However, their research did not show why some problems were considered as basic models.

In 2013, the Second Edition of the Handbook of Combinatorial Optimization was published by book:1016601. In Lahyani2015 a comprehensive taxonomy developed for the RVRP was proposed. In gorman2016metasurvey, published a *metasurvey* including 343 reviews of the last 15 years in the area of Operations Research and Management Science. He also mentioned that there were only five surveys in the area of combinatorial optimization.

The latest related research is bartz2017model, which showed a survey of model-based methods, introduced a taxonomy, examined discrete optimization problems and proposed a method to combine approaches by stacking them.

3. Methodology

3.1. Literature Review

We used the standard systematic literature review method kitchenham2009systematic, employing a manual search for ten GCPs in six web search engines:

- IEEExplore (ieee.org)
- ACM Digital library (dl.acm.org)
- Google scholar (scholar.google.com)
- Citeseer library (citeseerx.ist.psu.edu)
- Keele University's electronic library (keele.ac.uk)
- ScienceDirect (sciencedirect.com)

Our review included works that contain a keyword (or synonym) in the title and the abstract for the different GCPs. Whenever the relevance of the investigation was not evident under this criterion, other elements of the works were examined, such as the introduction and conclusions. The search was made with proposed queries of words or phrases such as:

- "*General Combinatorial Problem*" or "*GCP*"
- "*Vertex Cover*" or "*Vertex Cover problems*"
- "*Hamiltonian path*" or "*Hamiltonian Circuit problem*"

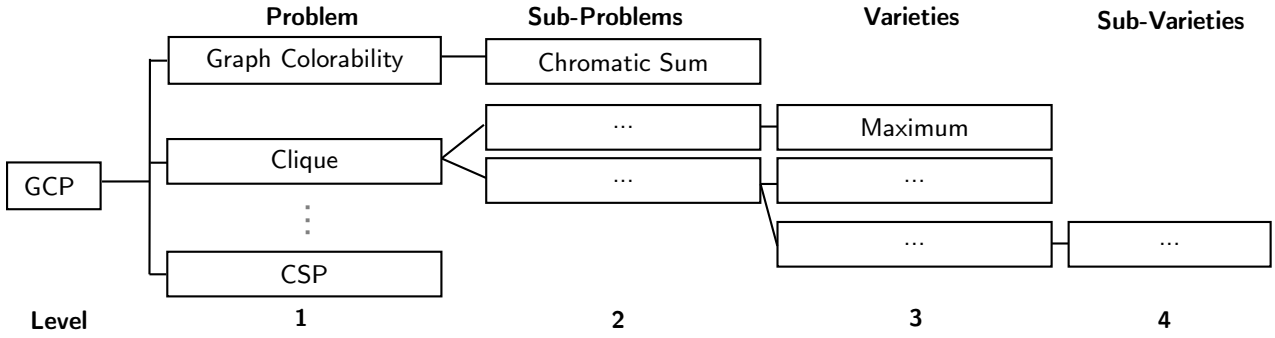


Figure 2: Tree structure illustrating the terms used in this work.

The search efforts were directed to find sources with as high an impact factor as possible, in different formats such as conference proceedings, journals, technical reports, and problem repositories. Once these primary sources were obtained, these were reviewed in search of relevant references, in order to identify trends to follow up and the relationships between the different investigations. This was performed by means of the key terms used for the search.

To document the origin of each GCP in terms of its mathematical definition, we searched for investigations that refer to the first appearance or formal mention. In particular, publications containing demonstrations, applications, improvements in solution approaches, experimental tests, characterizations or measurements on the problems of interest, were examined.

3.2. General Combinatorial Problem Tree

The current investigation involved searching and analyzing ten GCPs analyzed by Jeavons in terms of their algebraic structures. Jeavons established that these problems are at least NP-Hard.

Hereinafter, the term *problem class* or simply *problem* is used to refer to each of the 10 problems analyzed by Jeavons (Graph Colorability, Satisfiability, Vertex Cover, etc.) and that are defined in terms of algebraic structures in the literature. The second level of grouping includes the formulations formally identified in the literature; the term *sub-problems* refers to these. The third level in this tree structure is referred to as *varieties*, the fourth level represents varieties with more restrictions named *sub-varieties*, and finally, specific data sets or realizations of the problems, sub-problems or varieties, are called *instances*. The tree structure used to describe the problems in the following subsections is shown in Fig. 2. In this work, said structure includes up to four levels in depth. The root of the tree structure is the GCP.

In our work, a problem in the first level in GCP tree must possess a definition in terms of an algebraic structure, such as for the problems described in Jeavons1998. Sub-problems and varieties are added to the deeper levels of the tree (level two and onward), if one can verify that:

- A formal mathematical definition can be found in the literature (obligatory for level two).
- The restrictions and variables match those of the problem class to which the sub-problem is to be added.

Whenever there exists a formal mathematical definition of a sub-problem with varieties, these are added to the respective problem class. Otherwise, the sub-problem is not added to the corresponding problem class but will be considered to be added in the future, when a formal definition is formulated (see Fig. 2).

With the aforementioned information, we will have that the varieties represented by the lower levels such as 3 or 4 are contained in the previous levels, besides having a lower degree of generality with respect to the layer that previously contains it, i.e. the problem found in level 4 is a more particular case, (limited or restricted) with respect to the problems contained in level 1 or 2.

4. Classification of General Combinatorial Problems

In 1998, Peter Jeavons conducted a formal analysis and defined several combinatorial optimization problems in terms of algebraic structures Jeavons1998, thus identifying an important characteristic that these problems share. The

present work takes the research conducted by Jeavons as a starting point to propose a classification of these problems Jeavons1998.

The GCPs aim to find the combination of objects in a set of finite solutions that improves some function or objectives, subject to a set of restrictions karp1972b. In the context of algebraic structures Jeavons1998 defines GCP as two relational structures S_1 and S_2; where $S = \langle V, E_{i\in I}\rangle$ and V is the *universe* of elements from the relational structure, and is strictly a nonempty set. E_i is a system of finite relations over V, sorted according to the elements of I, which is a set of relationships between V and E_i. Finally a GCP instance has $P = \langle S_1, S_2\rangle$ which is a homomorphism from S_1 to S_2.

The following problems are described by Jeavons in terms of algebraic structures, within the context of universal algebra:

1. Graph Colorability (GC, Section 4.1)
2. Clique (Section 4.2)
3. Vertex Cover (VC, Section 4.3)
4. K-Dimensional Matching (Section 4.4)
5. Hamiltonian Circuit (HC, Section 4.5)
6. Bandwidth (Section 4.6)
7. Graph Isomorphism (GI, Section 4.7)
8. Undirected Graph Reachability (Section 4.8)
9. Satisfiability (Section 4.9)
10. CSP (Constraint Satisfaction Problem) (Section 4.10)

Each of the aforementioned combinatorial problems has a first set S_1 that includes the subsets of variables and their restrictions. The set S_2 represents the combinations of values allowed for the subset of variables.

Starting from the fact that the Generalized Satisfiability (SAT) problem was demonstrated by Schaefer in 1978 as P while a least one of six conditions are met, in other case is NP Schaefer1978. Jeavons demonstrated that these conditions can be derived in Boolean algebraic structures, these conditions are listed below, let Γ be a set of Boolean relations over a set of domain values D:

1. Every relation in Γ holds when all variables are false.
2. Every relation in Γ holds when all variables are true.
3. Every relation in Γ is definable by a formula in conjunctive normal form in which each conjunct has at most one negated variable.
4. Every relation in Γ is definable by a formula in conjunctive normal form in which each conjunct has at most one unnegated variable.
5. Every relation in Γ is definable by a formula in conjunctive normal form in which each conjunct contains at most two literals.
6. Every relation in Γ is the set of solutions of a system of linear equations over the finite field 0,1.

Given this, it is based that some of the problems that are at least NP class can be related through algebraic structures with the SAT and its conditions, so that they belong at least to the complexity of the problem of generalized Satisfiability. With this, Jeavons establishes a strong relationship by modeling the instances of the problems listed in his work, with the study of the Algebraic Closure Operations and with the complexity of these problems.

In the following sections, we will review the formal definition of each of these problems in terms of algebraic structures, as well as their historical background and some types of problems stemming from them. A summary of all the instances discussed can be found in Appendix-A, which contains 4 tables. Table 3 includes Graph Colorability, Clique, Vertex Cover and Hamiltonian Circuit; Table 4 contains Bandwidth; Table 5 summarizes Timetabling, Scheduling, TSP, VRP; and Table 6 GI, and Satisfiability problems.

4.1. Graph Colorability

This problem has its origins since 1852 when Morgan Cook2012, reports through a letter to Hamilton, that one of his students has proposed a task of coloring the different counties of England with different colors, considering that the neighboring counties are not to be labeled with the same color marek. The demonstration that the coloring of graphs or also known as vertex−coloring is an NP-Hard problem was proposed in Karp1972.

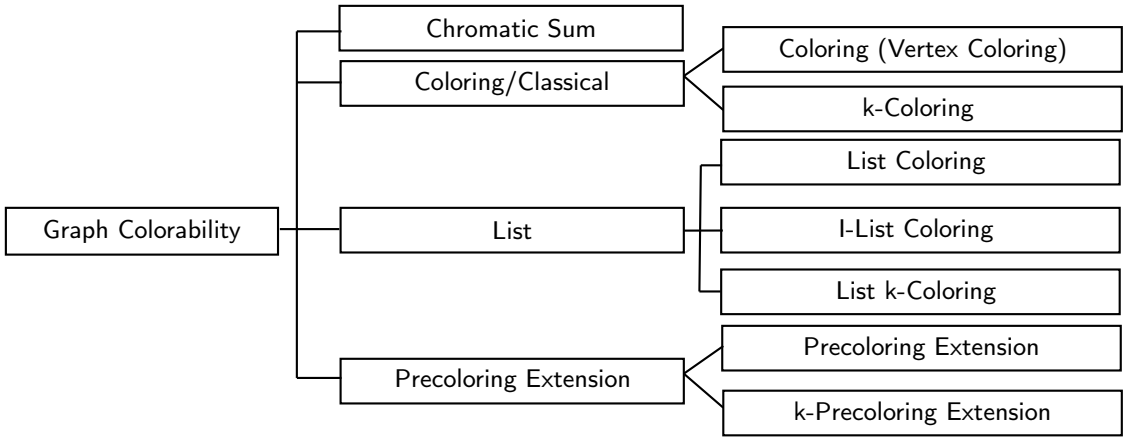


Figure 3: Graph Colorability problems

4.1.1. Definition

The Graph Colorability Problem (*GC*) has a graph G and q colors. The problem consists in labeling each vertex with one of q colors, considering the restriction that two adjacent vertex must be labeled with a different colors. Jeavons expressed *GC* as $\langle G, K_q \rangle$, where K_q is a complete graph with q colors Jeavons1998.

4.1.2. Variants

Some researchers describe the different problems of Graph Colorability Karp1972,read2014graph,BORODIN2013517,Maffr also their applications in electronic engineering, computer science, operations research and applied mathematics Deo2016. Golovach et al. describe the computational complexity of different Graph Colorability problems golovach2017survey. Graph Colorability problems are Classical Colorability kosowski2004classical, Equitable Coloring koster2017flow,Meyer1973, Chromatic Sum Coloring Jin2017, T-coloring Kawarabayashi2017, Rank Coloring rankCP, Harmonious Coloring Gao2017, Interval Coloring pyatkin2004interval, Circular Coloring deuber1996circular, Path Coloring Corteel2003 and List Coloringwang2005list.

A total of three sub-problems of Graph Colorability were identified, as shown in Fig. 3. Three of said sub-problems include at least two formally defined varieties biro1992precoloring, kosowski2004classical, wang2005list, marx2005np. The sub-problems known as: *Coloring* (also known as Classical- or Vertex- Coloring), *List*, and *Precoloring*, as well as their corresponding varieties, are in the figure. The following varieties do not appear because currently there is no formal criterion to group them otherwise: Circular, Equitable, Harmonious, Interval, Path and Rank.

4.2. Clique

The clique problem is classified in the literature as NP in Garey1979. The first applications and studies about cliques initially were on sociology as a method for the analysis of group structures in luce1949method. In 1957, the first algorithm was proposed by Harary & Ross bomze1999maximum.

4.2.1. Definition

Given a graph denoted as G, a clique is a subgraph of G in which all nodes are connected to each other. A clique instance consists of a graph G and an integer q, and the objective is to know how many subgraphs exist within G with q vertexes that are cliques (i.e. isomorphic to a complete graph denoted as K_q). It can be expressed as an instance of GCP $\langle K_q, G \rangle$, provided that G does not contain loops (vertex adjacent to itself) Jeavons1998.

4.2.2. Variants

The clique problem has different variants, in pardalos1994maximum and bomze1999maximum classified them into three: Maximal clique problem, Maximum clique problem carraghan1990exact and Weighted maximum clique wu2015solving. Based on this prior classification in Fig. 4 these variants are shown.

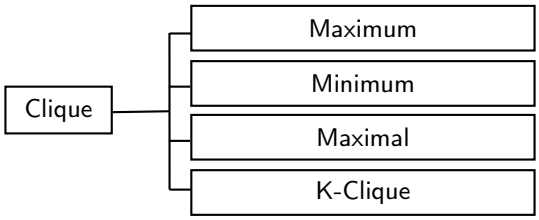


Figure 4: Clique problems and sub−problems

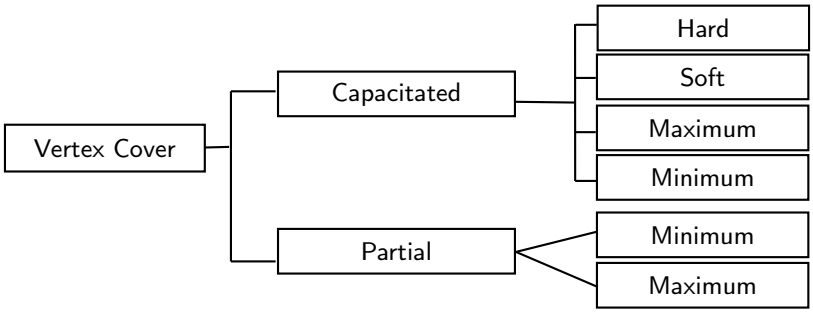


Figure 5: Vertex Cover problem, sub-problems and their varieties.

4.3. Vertex Cover

This problem is considered an NP-Complete problem in Garey1979, also having its first mentions by erdos1961minimal, where they questioned what was the minimum number of nodes needed to cover all the edges of a graph. The generalized problem was introduced in hassin2003minimum hassin2006minimum.

4.3.1. Definition

An instance of Vertex Cover consists of a graph $G = \langle V, E\rangle$ and an integer k. The aim is to prove if there exists a subset $V' \subset V$ with $|V'| \leq k$ for any $\langle v, w\rangle \in E$, either $v \in V'$ or $w \in V'$ Jeavons1998. In a complete graph we search the subset of nodes that covers all the edges of the graph.

4.3.2. Variants

There are different varieties of Vertex Cover problem, some of them are discussed in Jiong2005: Connected Vertex Cover escoffier2010complexity, Tree Cover, Tour Cover, Capacitated Vertex Cover guha2003capacitated, Soft Capacitated Vertex Cover Bar-Yehuda2007, Hard Capacitated Vertex Cover chuzhoy2006covering, Maximum Partial Vertex Cover and Maximum Partial Vertex Cover.

Two varieties were identified, with four and two sub-varieties with formal definitions respectively. The classification presented above for the Vertex problem is shown in Fig. 5. We left out three subproblems because there is no formal definition for these: Connected, Tour and Tree.

4.4. K-Dimensional Matching

In 1979, Garey and Johnson Garey1979 proposed a set of six basic NP-Complete combinatorial problems, one of them is 3-Dimensional Matching. Hazan et al. defined this as a general problem named $k-$Dimensional Matching hazan2003complexity and Kann Viggo proposed it as Max SNP-Complete kann1991maximum.

A recent study developed by Cigan focuses on applying an approach to local search for k-Set Packing where considered a special type of swaps and he improved the best-known approximation ratio for k-Set Packing cygan2013improved.

4.4.1. Definition

The k-Dimensional Matching also named as 3DM (3-Dimensional Matching) involves: given a set of $M \subseteq W \times Y \times Z$, where W, X y Y are disjoint sets with the same q elements. In this problem, the objective is to find M' contains a *matching* which is the subset $M' \subseteq M$ such that no two elements share the same coordinate Garey1979.

In terms of GCP instance, Jeavons1998 defined as tuple $\langle\langle V, \diamondsuit_V\rangle, \langle M, \diamondsuit_M^{\sim}\rangle\rangle$ where $\diamondsuit_M^{\sim}$ denotes the tuple

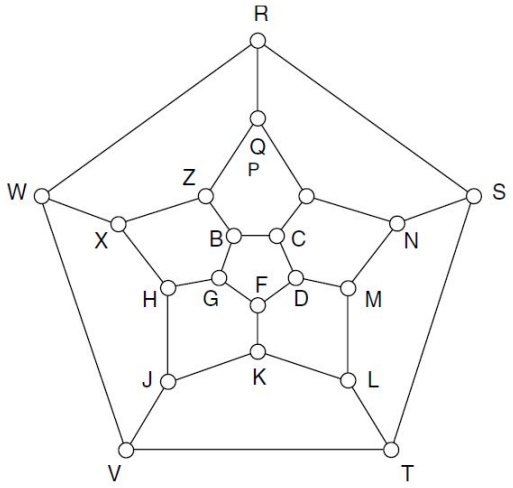


Figure 6: Icosian example (taken from [**?**])

$\langle\langle v_1, v_2, \ldots, v_k\rangle, \langle v'_1, v'_2, \ldots, v'_k\rangle\rangle \in M^2 | v_i \neq v'_i, i = 1, 2, \ldots, k$, $\bigtriangleup_V$ is the disequality relation over V and V is set called the *universe* of the relational structure.

For the case of the k-Dimensional Matching, no formal definitions of sub-problems or varieties were found.

4.5. Hamiltonian Circuit

The Hamiltonian Circuit has its origins in 1850, named by Sir William Rowan Hamilton. Hamilton began to study how to visit 20 nodes of a dodecaedron and created a graphical abstraction called icosian. The edges of icosian are geometric borders and the corners are circles as in Fig. 6 .

Hamilton proposed generating a cycle with five vertex and he proved that it does not matter which cycle is initially followed, it is always possible to complete the tour across the remaining vertex of icosian.

It was until 1976 that this problem was proposed as NP−Complete problem by Garey & Johnson in their investigation "*The planar Hamiltonian Circuit problem is NP-Complete* " garey1976planar. Some Halmitonian Circuit studies, surveys and advances are witte1984survey,gould1991updating,dean1991survey,Kuehn2012,gould2003advances, these researches described topics about history, a formal definition of variants, restrictions and constraints.

4.5.1. Definition

Given a graph G, with a vertex set V and edges E, the aim is to find one tour which covers all edges of G. In formal terms, the Hamiltonian Circuit problem denoted as $G = \langle V, E\rangle$, needs an ordered cycle from V for each successive pair of nodes and that those are adjacent to G Garey1979. In terms of a GCP instance, the Hamiltonian Circuit problem can be defined as: $\langle\langle V, C_V, \bigtriangleup_V\rangle, \langle V, E, \bigtriangleup_V\rangle\rangle$, where C_V is an arbitrary cyclic permutation on V and $\bigtriangleup_V$ denotes the inequality relation over V with $\bigtriangleup_V = \{\langle v, v'\rangle \in V^2 \, | v \neq v'\}$ Jeavons1998, .

4.5.2. Variants

Inside of the HC problem is Hamiltonian-Connected From a vertex (HCV), which can be divided into three subproblems: *Hamiltonian-Connected*, *Uniquely Hamiltonian-Connected From a vertex* and $k1$, $k2$, $k3$ GR1984. This whole branch of the corresponding tree structure is based on the classification presented in GR1984, as shown in Fig. 7.

One of the most studied problems in the state-of-the-art of the Hamiltonian Circuit is the Traveling Salesman Problem (TSP).

Traveling Salesman Problem (TSP)

The mathematics problems related to TSP were proposed in 1800 by the mathematicians Irish's Sir William Rowan Hamilton and Thomas Penyngton Kirkman. A review of the discussion about their first investigation can be found in Fran-1977.

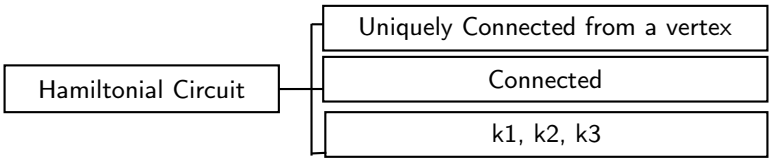


Figure 7: Hamiltonian circuit problems and subproblems

As a combinatorial problem, the TSP was studied for the first time in the 1930's by Karl Menger in Viena and Harvard. Later this problem was popularized by Hassler Whitney and Merrill Flood in Princeton. A detailed study about the connection between Menger and Whitney and the faster popularization of TSP as a topic of study can be found in the document created by Alexander Schrijver in"*History about combinatorial optimization (until 1960)* "Fran-1977.

Given a set of cities and cost of a trip between each pair of them, the aim in TSP is to find the lowest cost tour that passes through all the cities and returns to the initial city. In the standard version of this problem, all the costs are the same. Sometimes this problem is described as "*The most popular studied problem in computational mathematics*". This definition can be deceptive because in most of the state-of-the-art investigations there is not an effective method for the general case.

The TSP has with three sub-varieties, which has as characteristic a weighted graph. These subproblems can be divided into Symmetric, Asymmetric and VRP. It is worth mentioning that only the most general versions of these problems have been considered in this work, since the objective is not to perform a comprehensive review of all the varieties in each sub-problem. A subvarieties of TSP is Vehicle Routing Problem (VRP), a non-extensive list of sub-varieties includes:

- Symmetric
- Asymmetric
- VRP (Vehicle Routing Problem)
  - Capacitated VRP
  - Capacitated VRP with Time Windows
  - Capacitated VRP with Pick-up and Deliveries and Time Windows
  - Multiple Depot VRP
  - Multiple Depot VRP with Time Windows
  - Periodic VRP
  - Periodic VRP with Time Windows
  - Vehicle Routing Problem with Pick-up and Deliveries

4.6. Bandwidth

Historically, the matrix Bandwidth problem was born in 1950 when a group of engineers had the aim to compute matrix operations (as matrix inversions) and determine if they could be computed in less time and with less computational cost chinn1982bandwidth. They discovered that this could be possible if all non-zero elements of a matrix are in a certain diameter or bandwidth of the diagonal.

The bandwidth problem for graphs was formulated in 1962 at the Jet Propulsion Laboratory at Pasadena, for the edge differences in a hypercube whose vertexes were words from code. In this laboratory L.H. Harper and A. W. Hales searched for the code which minimized the average and the maximum absolute error chinn1982bandwidth.

4.6.1. Definition

An instance of Bandwidth consists of a graph $G = \langle V, E \rangle$, and a positive integer k, where $V = \{v_1, v_2, \ldots, v_n\}$. The aim is to find if there exists a linear ordering of V such that all the adjacent nodes in the graphs are separated at least k positions in the ordering Garey1979.

Expressed in terms of a GCP the Bandwidth problem is: $\langle\langle V, E, \bigtriangleup_V\rangle, \langle V, B_k, \bigtriangleup_V\rangle\rangle$ where $B_k = \{\langle v_i, v_j\rangle \in V^2 | i - j| \leq k\}$ and $\bigtriangleup_V = \{\langle v, v'\rangle \in V^2 | v \neq v'\}$ is an inequality relation over V Jeavons1998.

Chinn et al. defined the Bandwidth problem in two different approaches, first one consists in labeled n vertexes denoted as v_i from a graph G with different integers $f(v_i)$ from $\{1, 2, 3, \ldots, n\}$ such that the maximum value of $|f(v_i) - f(v_j)|$, taken from all tuples of adjacent vertexes, must be the minimum chinn1982bandwidth. In the second, given one symmetric matrix M, the aim is to find the permutation M' of M such that the maximum value from $|i - j|$ taken from all non-zero inputs m'_{ij}, must to the sum minimum.

4.6.2. Variants

Some researches describe different problems of Bandwidth 1454786,chinn1982bandwidth,Josep2002,Leung-1984, furthermore we found some applications in Computer Science, Operations Research and applied Mathematics proposed in Monien-1986,Estefani-2004,book2066451,Jayaswal-2017.

Harper defines formally the Bandwidth problem in HARPER1966385. In this work, we decide elaborate a diagram of classification because we did not find sufficiently formal definitions of these problems, where it can be assured that all the subproblems belong to the second level or deeper levels. Some subproblems of Bandwidth are: Topological Bandwidth Makedon-1985, Graceful alfalayleh2004towards, Cross J-1981 and Cycle Leung-1984 Satsangi-2012, Directed GareyM.1978, Sum lai1999survey, Directed-Sum chinn1982bandwidth, and Sum-Minimization chinn1982bandwidth.

4.7. Graph Isomorphism

The general approach to Graph Isomorphism was developed by Weisfeiler and Lehman in the 1960's Derksen-2013. The problem of the isomorphism of graphs is seen as a special case of Subgraph Isomorphism Ullmann1976, however, at present, it cannot be assured that it is an NP−Complete or P problem Kobler1993, V-1985. There are particular cases of isomorphism of graphs that can be solved in a polynomial time, but since we are looking to know if there is a permutation of the nodes of a graph A that returns the graph B, this is of the order $n!$ Derksen-2013.

The problem of Subgraph Isomorphism is considered an NP−Complete problem Garey1979 and Cook shows that it can be solved by a non−deterministic Turing machine Cook1971.

4.7.1. Definition

Subgraph Isomorphism

Formally, Garey and Johnson define Subgraph Isomorphism, as the problem where, given two graphs $G_1 = \langle V_1, E_1\rangle$ and $G_2 = \langle V_2, E_2\rangle$, the aim is to find a subgraph of G_1 that is isomorphic to G_2 Garey1979. In addition, the subsets of $V' \subseteq V_1$ and $E' \subseteq E_1$ comply with $|V'| = |V_2|$, $|E'| = |E_2|$ and there is a one−to−one function $f : V_2 \longrightarrow V'$ that satisfies$\{u, v\} \in E_2$ if and only if $\{f(u), f(v)\} \in E'$ Garey1979.

Graph Isomorphism

Generally, the isomorphic graph problem consists of deciding if two finite graphs are isomorphic, looking for a bijective mapping (a permutation) of the nodes of a graph towards the nodes of the second graph, such that the connected edges are respected 9781461203339.

An instance of the Graph Isomorphism problem consists of two graphs $G = \langle V, E\rangle$ and $G' = \langle V', E'\rangle$ with $|V| = |V'|$. Then it is sought to know if there is a bijection mapping between the vertexes in such a way that the adjacent vertex in G are assigned to adjacent vertex of G' and the vertex that are not adjacent in G are mapped to non−adjacent vertexes in G'. In terms of GCP this problem it is expressed as $\langle\langle V, E, \bar{E}\ \rangle, \langle V', E',\ \bar{E}'\ \rangle\rangle$, where $\bar{E}\bigtriangleup_V - E$ and $\bar{E}'\bigtriangleup_{V'} - E'$, and $\bigtriangleup_V = \{\ \langle v, v'\ \rangle \in V^2 \mid v \neq v'\}$ is the inequality relation over V Jeavons1998.

4.7.2. Variants

In the state-of-the-art, there are different articles that describe the different subproblems of Graph Isomorphism, some of these subproblems are Subgraph and Tree Isomorphism valiente2002tree. In similar way such as the Bandwidth problem, we did not elaborate a diagram of classification, because we didn't find formal definitions of these subproblems.

4.8. Undirected Graph Reachability

In 1741 Euler showed the earliest application of graph theory, since that time there is a record of the use of several concepts to deepen the investigations on mathematical structures euler1741solutio.

4.8.1. Definition

In the theory of graphs, the action to find a vertex which connects to another within a graph is called *reachability*. Given two vertexes denoted S and S', if there exist a sequence of adjacent vertexes which forms a path between these two vertexes, it is said that S is reachable to S'.

In an undirected graph, the reachability between at least one pair of vertexes can be identified with the connected nodes of the graph. These connected nodes are a subgraph in which two of them are connected to each other by paths and avoid the connections to any additional node in the supergraph.

Any pair of vertexes in a graph of this type can reach another one among themselves, if and only if they belong to the same connected component. The determination of the size of the subgraph of connected components and the connected components of an undirected graph can be identified in linear time.

An instance of Undirected Graph Reachability consists of an undirected graph $G = \langle V, E\rangle$ and a pair of vertexes $\{v, w\}, \in V$, to determine if there is a path in which v and w are connected Garey1979.

The complementary problem would be to find if there is no path that does not connect to v and w. This can be expressed as an instance of the GCP as Jeavons1998: $\langle\langle V, E, \{\langle V\rangle\}\ \rangle, \{\langle w\rangle\}, \langle\{0,1\}, \square_{\{0,1\}}, \{\langle 0\rangle\}, \{\langle 1\rangle\}\}\rangle$, where $\square_{\{0,1\}} = \{\langle(0,1),(0,1)' \in (0,1)^2 | (0,1) = (0,1)'\rangle\}$ is an equality relation over $(0,1)$.

4.8.2. Variants

Some varieties derived from Undirected Graph Reachability are:

Giant component.- In network theory, a component is a subset of connected vertex of a given random graph, and this is considered large if it has more than $n^{2/3}$ of the total of the vertex of the graph. If proven that said large component is unique among all the other components, then it is considered "giant"bollobas2001evolution.

Biconnected component.- In graph theory, it is important to determine if a graph contains: connected components, biconnected components, and simple paths. There are algorithms that look for biconnected components. These algorithms break a graph in its biconnected components by performing a depth-first search along the edges of the graph. Generating a stack of vertexes connected from the first to the last, where the penultimate vertex in said stack, it is called "*articulation point*"and it is considered an edge of the biconnected graph. The aforementioned is done in all the vertex of the original graph, until it is determining how many components exist and their kind of connection. We can also get evidence of the existence of isolated vertexes (a vertex without adjacent edges) Hopcroft-1973.

Modular decomposition.- This problem consists of dividing a given graph into subsets called modules, with certain characteristics. The resulting modules are a generalized case of a "connected component". The decomposition varies if an initial graph is directed or not. For undirected graphs this decomposition takes place if it meets certain properties that can be determined in a unique way Gallai1967.

4.9. Satisfiability

The references indicate that the researcher Stephen Cook at the University of Toronto first mentioned the Boolean Satisfiability Problem (SAT) when he showed that it was of NP−complete complexity in the year 1971. He formulated a formal way which all decision problems in the class of NP complexity can be reduced to the problem of SAT Cook1971.

4.9.1. Definition

The satisfiability problem is formed by a propositional logic formula $\mathscr{F}$, which is the conjunction of a set of clauses c. Each of the clauses c is a disjunction of literals, where the literals are variables that can take the true or false value. So in this problem, we look for the set of values for the variables in the function $\mathscr{F}$, where the result is true Garey1979.

In terms of the GCP it can be expressed as $\langle\langle V, E_c(c \in C)\rangle, \langle\{0,1\}, R_c(c \in C)\rangle\rangle$ where V is the set of propositional variables used in $\mathscr{F}$, each $E_c = \{\langle x_1, x_2, \ldots, x_{\rho(c)}\rangle\}$ where $x_1, x_2, \ldots, x_{\rho(c)}$ are the variables that appear in the clause c and $R_c = \{\langle h(x_1), h(x_2), \ldots, h(x_{\rho(c)})\rangle | h\}$, $\rho(c)$ is the arity of E_c , and h is an assignment of values that satisfy c in true form Jeavons1998.

4.9.2. Variants

Different articles describe the different varieties of Satisfiability du1997satisfiability,Roman2006, as well as their applications in Computer Science, Operations Research and Applied Mathematics Du-1999,mezard2005clustering.

Figure 8: Satisfiability problem and its subproblems

At least five sub-problems of the Satisfiatility problem were found (see Fig. 8), one of them (k-Satisfiability) is formally defined together with its varieties in Cook1971. The other sub-problems: Horn-Clause, Not-All, One-in-Three and XOR, have been discussed in du1997satisfiability and these are not included as sub-problems in the Satisfiability tree because no study has classified them as varieties of any particular case of Satisfiability sub-problem.

4.10. Constraint Satisfaction Problem (CSP)

The problem was firstly treated formally in the computer science area in montanari1974networks, applied to image processing. Subsequently, various investigations were carried out on Constraint Satisfaction Problem mackworth1992constraint,freuder2006constraint,pearson1997survey.

4.10.1. Definition

The CSP, can be defined from a set of variables $V = \{v_1, v_2, \dots, v_n\}$, a set of values $W = \{w_1, w_2, \dots, w_m\}$ that those variables can take, a family $S = \{s_k | s_k \subset V\}$ and the set of variables in s_k denoted $C(s_k)$. Then the problem is described by the question: "What is the assignment of values W for variables V, such that all constraints are satisfied?" Jeavons1998a. Expressed as an instance of the GCP Jeavons1998, we have: $\langle\langle V, S_1, S_2, \dots, S_m\rangle, \langle D, C(S_1), C(S_2), \dots\ , C(S_m)\rangle\rangle$

4.10.2. Variants

Different varieties of CSP can be found in brailsford1999constraint,Cooper2017,BODIRSKY201687, while applications can be found in krokhin2017dfu, VANDENBERGH2013367, chaudhry2016research. The diagram with some Constraint Satisfaction Problems is given in Fig. 9. We describe some of these CSP and their respective varieties and sub-varieties:

- **Sequencing Mixed-Model Assembly Lines (SMMAL):** this problem lets to enable such a highly diversified product portfolio without jeopardizing the benefits of an efficient flow-production. A set of 5 varieties were found in the state of the art by boysen2009sequencing:
  - **Work overload:** the objective of this problem is to avoid the work overloads due to the installation of varying components typically leads to variations in processing times at workstations
  - **Just-in-time objectives**. In this problem the aim is to distribute the material requirements evenly over the planning horizon. Every model is composed of different product options and thus require different materials and part. According to this, the model sequence influences the progression of material demands over time.
  - **Mixed-model sequencing**. The aim of this problem is based on detailed scheduling which explicitly takes characteristics of the line into account and the sequence-dependent work overload must be avoiding/minimizing.
  - **Car sequencing :** The car sequencing problem definition was proposed by Parello et al. in 1986 parrello1986job. The aim of this problem is to optimize the resources along an assembly car line. In this car line different options must be installed (eg. sun-roof, radio, or air-conditioning) on cars. Each option is installed in different stations, which were designed to handle a certain percentage of the cars passing along the assembly line. Every car which passes on each station must be spaced such that the capacity of every station is never exceeded.
  - **Level scheduling** . The aim of this problem is to find sequences that are in line with the JIT (Just-In-time) philosophy. The problem's characteristics are defined rates and sequenced models in such a manner that deviations between actual and ideal rates are minimized

- **Knapsack**: An instance consists: a set of items is given, each with a size and value, which must be placed into a knapsack that has a certain number of dimensions having each a limited capacity. Some varieties according to laabadi20180 are:
  - **Multi-Dimensional**: deterministic standard, deterministic multiple, deterministic multiple-Choice, deterministic multi-Objective, Non-Deterministic and Fuzzy.
  - Non-standard: Edward Yu-Hsien Lin studied some non-standard problems as Multiple-choice, Maximin and Collapsing lin1998bibliographical.
  - Christophe Wilbaut and Said Hanafi studied the application of Equality, Collapsing, Fractional, Bi-level, Two-Dimensional, Precedence Constraint, Disjunctively Constrained, multiple-choice multidimensional and Multi-period varieties wilbaut2008survey.
  - Finally, some varieties of knapsack well know are Multi-objective, Multiple, Quadratic, Subset-sum and Bin Packing. The Bin Packing problem has been studied extensive by Henrik I. Christensen et al. christensen2017approximation, and they document two variants and their respective sub-variants : One dimensional (offline 1-D and Online 1-D), Multidimensional (Geometric and Vector). The Geometric Bin packing has at least 7 cases : Geometric, Online, Square, Resource, Strip, Shelf and Guillotine; the Vector Bin Packing has Offline Vector, Online Vector, Vector Knapsack, Vector Scheduling and Vector Bin Covering.
- **Location**:
  - Fixed-Charge Facility
  - **Covering Location** with some models Set Covering, Weighted Set Covering, Redundant Covering, Hierarchical Covering, Maximal Covering, Backup Set Covering, Maximum Expected Covering, Probabilistic Location Set Covering, Maximum Availability, Covering, Minimum Cost Maximal Covering and p-Median studied by Sergio García and Alfredo Marín in garcia2015covering.
  - Classified according two characteristics Capacitated and Uncapacitated and some sub-varieties are Minimax, Minimum, Maximum, Maxmin, Network Design, Warehouse, Fire Box Coverage and Competitive.
  - **Anti-Covering**: These problems can be divided in two types according to the interactions take place, Individual-Facility and Facility-Facility carrizosa2019anti.
  - **Location of Dimensional Facilities in a Continuous Space**. These kinds of problems were proposed by Wesolowsky in wesolowsky1972rectangular, where facilities must be modeled as dimensional structures because those cannot be represented by isolated points. An extended review of different models and variations of this problem was reported by Anita Schöbel in schobel2019locating.
  - **Discrete Facility Location Problems**: In the research conducted by Correia and Saldanha da Gama correia2019facility, they put emphasis on modeling some problems as Robust Facility Location, Stochastic Facility Location, and Chance-Constrained Facility Location.
  - **Location Problems with Multiple Criteria**. In the review guided by Stefan Nickel et al. nickel2019location they analyzed the multicriteria continuous, network, and discrete locations problems.
  - **Ordered Median Location**: These problems can be studied in three different frameworks: Continuous, discrete, and networks, according to Puerto and Rodríguez-Chía puerto2015ordered.
  - **Multi-Period Facility Location** with variations on continuous, discrete, and network nickel2019multi.
  - Hub-Location, Location-Routing, Location-Arc, and Competitive Location Models.
- **Cutting stock**: One-dimensional (akin to Bin Packing) and Two−dimensional (Irregular, also known as nesting problem, and Rectangular).
- **Scheduling**: is a decision-making process that deals with the allocation of resources to task over given time-periods and its aim is to optimize one or more objectives pinedo2012scheduling. Some variants were found and documented as follows:

  - **Resource-constrained Project Scheduling**: One shop, Flow shop (Limited and unlimited Intermediate Storage), Flexible shops, Proportionate Flow Shops with Unlimited and Limited Intermediate Storage, and Job shop (Processor scheduling, Personnel scheduling, Bandwidth, Airport gate, Flow Shop, Open Shops, and Repair crew)
  - **Parallel Machine Problems**: Preemptive scheduling and Nonpreemptive scheduling (Unit and General processing times); Makespan and total Completion time both with variants on Preemptions and without Preemptions.
  - **Single Machine Problems**: Batch Scheduling, Single Processor, Parallel Processors and Online-over-Time (clairvoyant and Non-clairvoyant).
  - Advanced Single Machine Models : total tardiness, total earliness, Makespan with sequence dependent and Batch Jobs.
  - Stochastic Models: Single machine and Single Machine with Release Dates, Parallel Machine, Flow Shops, Job Shops and Open Shops.

- **Timetabling**: Rostering (Nurse Rostering and Health Care) and Educational Timetabling :High School and University Timetabling (Course, Faculty, Classroom Assignment, Class-teacher, and Examination).

5. Results and Discussion

5.1. Results of our Literature Review

Under the search criteria employed, approximately 750 references were collected with possible relevance. The results obtained by the different search engines were numerous and seemed pertinent, but under a detailed analysis, some of them were discarded from this research. After filtering this initial result, the selection was reduced to approximately 130 references. These searches were conducted during the period from December 2016 to February 2019.

We found that several web search engines produce such results that their relationship to the target concept is too weak. That is, in some cases a single word of interest is found in a non related work about the specific topic.

There was not a single search engine that surpassed the others in the quality of its results, with respect to a specific query. In addition, the documents returned by some of the search engines had little relevance to the query, so it would be difficult to conclude that their performance is uniform.

In many cases, the content of the returned abstracts lacked substantial information delineating the corresponding articles in a useful way, so that we had to resort to inspection of the whole documents to justify their inclusion.

Table 1 shows the summary of the search results that were obtained. The table contains the number of articles obtained for the twelve searches (one for each problem, including GCP and TSP) in the six search engines.

Table 2 shows the summary of the results in terms of percentage, all of them normalized by search engine. The sum of all the works returned by the search engines was considered and then the percentage was extracted. Underlined values indicate the highest percentage of results per search engine, while the results in bold face indicate the highest results per problem. For instance, the problem with most hits for the ACM and Keele University engines is the Satisfiability problem (SAT). It can be observed that the ACM, Citeseer and Keele University obtained the highest percentages of results returned for most of the problems (bold face numbers). However, it should not be forgotten that large number of results can be misleading, because many of the *related documents* do not discuss comprehensively the definition of a certain problem.

For each of the web search engines, the most popular problems were:

- ACM Digital library: SAT.
- Google Scholar: Graph Colorability.
- Citeseer library: General Combinatorial Problem.
- IEEE Xplore and Keele University's: CSP.
- ScienceDirect: Clique.

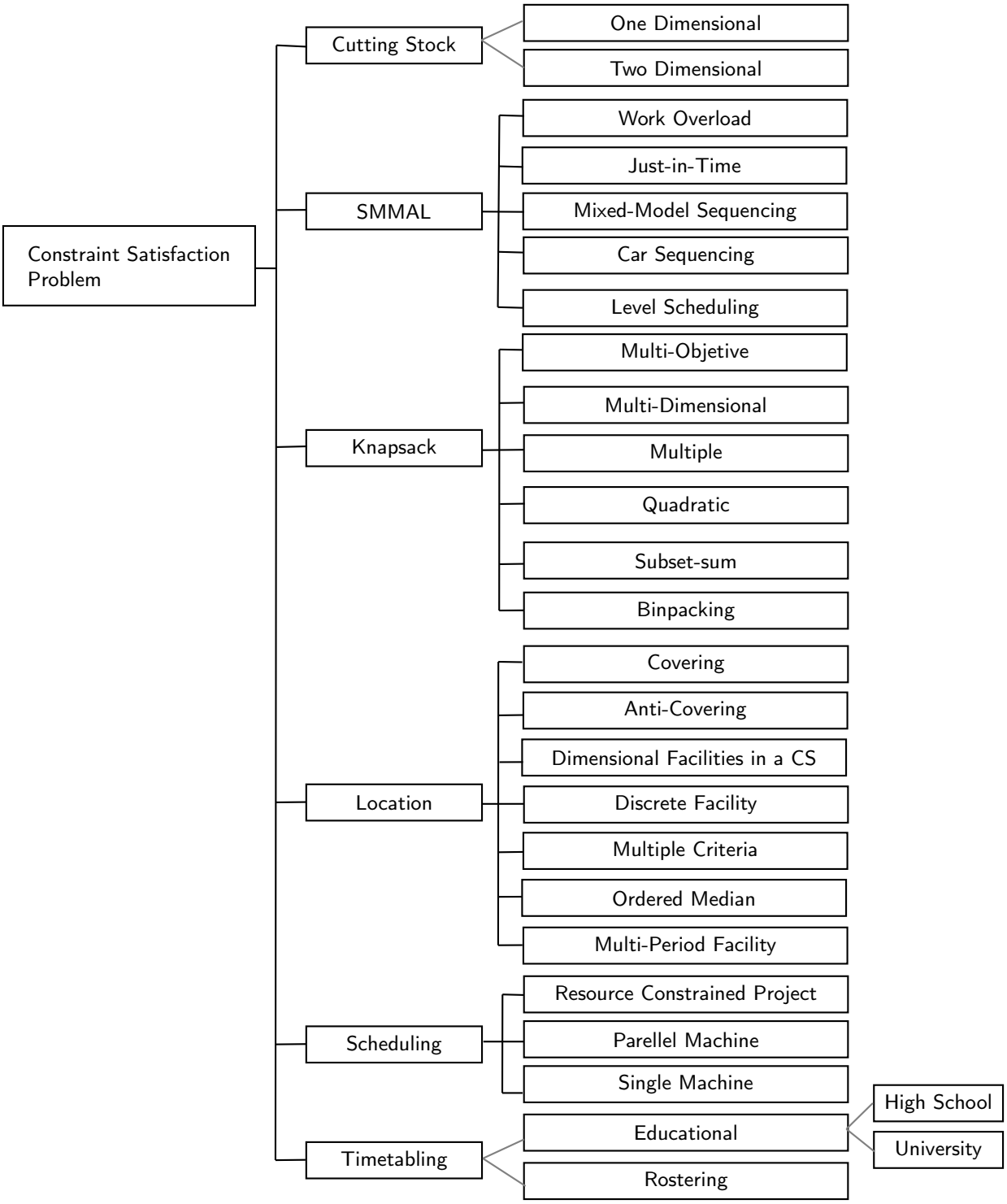


Figure 9: Constraint Satisfaction Problem, its subproblems and varieties.

The least popular problems were:

- IEEE Xplore, Google Scholar, Keele University's, ACM and ScienceDirect: Undirected Graph Reachability.
- Citeseer library: Clique.

Problem	IEEE	ACM Digital library	Google scholar	Citeseer library	Keele U.	Science Direct
GCP	879	1	255	303	45	48
GC	857	301	46,900	196	5,841	3,503
Clique	820	535	13,000	10	2,166	19,632
VC	294	170	7,900	54	1,475	2,073
k-DM	33	8	218	321	909	25
HC	341	19	6,140	12	1,075	1,374
Bandwidth	2,030	8,238	7,850	24	1,028	599
GI	1,789	105	23,000	115	3,725	2,083
UGR	7	1	100	42	11	3
SAT	2,783	10,335	29,000	169	4,029	9,378
CSP	4,641	429	34,000	179	3,804	3,052

Table 1
Results of literature search for different different GCP

Problem	IEEE	ACM Digital library	Google scholar	Citeseer library	Keele U.	Science Direct
GCP	6.07	0.00	0.15	**21.2**	0.19	0.11
GC	5.92	1.49	**27.8**	13.75	24.23	8.39
Clique	5.67	2.66	7.72	0.70	8.98	**47.0**
VC	2.03	0.84	4.69	3.79	**6.12**	4.96
k-DM	0.23	0.04	0.13	**22.5**	3.77	0.06
HC	2.36	0.09	3.65	0.84	**4.46**	3.29
Bandwidth	14.03	**40.9**	4.66	1.68	4.26	1.43
GI	12.36	0.52	13.66	8.07	**15.4**	4.99
UGR	0.05	0.00	0.06	**2.95**	0.05	0.01
SAT	19.23	**51.3**	17.22	11.86	16.71	22.4
CSP	**32.0**	2.13	20.19	12.56	15.78	7.31

Table 2
Search results (%) for different GCPs in six web-search engines

5.2. Results of the GCP review

In Section 4 we listed 10 GCPs. It should be mentioned that the order in which these were presented does not imply an order of importance or level of generality between them. The percentages per GCP in the references analyzed in this work are shown in the Fig. 10.

The Graph Colorability problem is maybe one of the oldest in its formulation, since it dates from the middle of the 19th century and this is the problem with most references in our work. Within the state of the art consulted, 3 different sub-problems from the Graph Colorability were found; these are contained in level 2. Likewise, the problems of Clique, Undirected Graph Reachability and Subgraph Isomorphism have one or more sub-problems in Level 3 of their corresponding tree diagrams. Finally, it was complex to generate sub-problems or classifications of CSP due to the number of the families it encompasses. Besides, it is the second most popular problem, with 11 percent of references in our work.

Below, three examples of how the proposed classification can help to better design studies around General Combinatorial Problems, are summarized:

– The study in burke2010iterated discusses three combinatorial optimization problems: Bin Packing, Permutation Flow Shop, and Personnel Scheduling . The authors refer to these problems as three different problem domains, stating that they belong to different problem classes. However, based on the classification proposed in this work, it can be identified that it can be identified that Burke et al. discussed three subproblems of the Constraint Satisfaction Problem.

– In choong2018automatic the authors describe a Q-Learning based hyperheuristic design for six different problem domains: Boolean Satisfiability, One-dimensional Bin-Packing, Permutation Flow Shop, Personnel Scheduling,

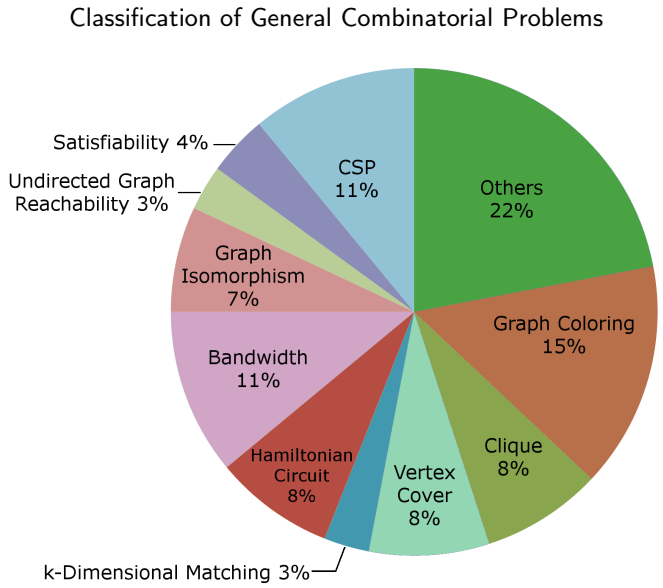


Figure 10: Percentage of Researches per GCP

Traveling Salesman Problem and Vehicle Routing Problem. Under the proposed classification in this work, it can be identified that choong2018automatic covered only three different problems: Satisfiability, CSP (which includes the varieties: Bin-Packing, Flow Shop and Personnel Scheduling), and Hamiltonian Circuit (which includes the varieties: TSP and VRP).

– As a third example, in soria2018statistical the authors selected what they assumed to be two different *problem classes* to evaluate an algorithm design. Using the classification described in this work, it can be identified that said study is not as general as it has been described, since the assumed problem classes, namely VRP and Multi-Knapsack, are in fact sub-problems of the Hamiltonian Circuit and of Constraint Satisfaction family, respectively.

6. Conclusion and Future Work

The state of the art of combinatorial problems (in particular, NP-hard problems) include a substantial body of research related to their definition and solution methods. However, in general no clearly defined criteria for the classification of these problems were found in those studies. This motivated the present work, in which a review of the General Combinatorial Problems was presented, guided by Jeavons' analysis of said problems in terms of their algebraic structure Jeavons1998.

The use of said analysis allowed us to recognize the existing sub-problems and varieties belonging to each problem class of GCPs that had not been previously identified as such because the works reviewed did not use a formal criterion to support their categorization of problems.

As has been exemplified above, this work can help new researchers to identify those algorithms that have been applied successfully and with good results to certain problems; it could also help researchers to design studies related to GCPs with increased confidence, allowing time to be saved in the identification of the complexity level of the problem(s) that researchers intend to study.

In our review, the relationship between the different problems of GCPs, their sub-problems and varieties, is evidenced through the identification of common characteristics under a mathematical framework and the organization of the different agreements given by the developments and findings of the experts in their different areas. This work needs to be enriched with the guidelines of the foundations of a formal taxonomy. To move forward in that direction, an important aspect in a future work is to include the objective criteria into the formal definition of each sub-problem and variety, possibly considering the identification of the optimal solution strategies.

Acknowledgements

Authors thank to Tecnológico Nacional de México/I. T. León and Universidad de Guanajuato. This work was supported by the National Council of Science and Technology of Mexico (CONACYT) via the Scholarships for Pos-

graduate Studies: 446106 (L. Ortiz) and 446105 (A. Montesino), and Research Grant: CÁTEDRAS-2598 (A. Rojas).

Appendix A.- Problem Instances

Graph Colorability

Name	I	N_{max}	E_{max}	Reference
DSJC125-1000	15	100	898,898	johnson1989optimization
Le450 05-25	12	450	17,425	leighton1979graph
anna	1	138	493	mat.gsia.cmu.edu
david	1	87	406	
fat	6	1,000	246,708	
fpsol2	3	496	11,654	
games120	1	120	638	
homer	1	561	1,629	
huck	1	74	301	
inithx	3	864	18,707	
jean	1	80	254	
latinsquare10	1	900	307,350	
miles750-1000	5	128	3,216	
mulsol1-5	5	197	3,973	
Myciel3-7	5	191	2,360	
queen5-16	13	256	12,640	
school1-1nsh	1	385	19,095	
zeroin	3	211	4,100	

Vertex Cover

Name	I	N_{max}	E_{max}	Reference
frb 30-59 - 15-26 mis	25	1,534	1,475	nlsde.buaa.edu.cn

Hamiltonian Circuit

Name	I	N_{max}	E_{max}	Reference
hc 1-8	8	200	1,250	di.univaq.it
np 10-90	9	90	8,010	
nv50a-70a 240 -580	16	70	540	
4xp20	4	80	396	
2xp30	5	60	318	

Clique

Name	I	N_{max}	E_{max}	Reference
brock 200-800	12	800	208,166	cse.unl.edu
C 125-2000	5	200	1,799,532	
DSJC1000_5	1	1,000	499,652	
DSJC500_5	1	500	125,248	
gen 200-400	5	400	71,820	
hamming	6	-	-	
johnson	4	32	-	
c-fat 200-500	7	500	-	Berman1990
keller	3	3,361	4,619,898	lagarias1992keller
p_hat 300-1500	15	1,500	847,244	gendreau1993solving
san 400-100	12	100		sanchis1994test
MANN	4	3,321	5,506,380	turing.cs.hbg.psu.edu

Table 3
Graph colorability, Vertex Cover, Hamiltonian Circuit and Clique instances. Including the number of instances I, maximum number of nodes N_{max} and maximum number of edges E_{max}.

Bandwidth			
Name	I	N_{max}	E_{max}
Path 20-1000	15	1000	999
cycle 20-100	15	1000	1000
mesh2D-3D	15	2197	1930
tree	12	1023	1022
caterpillar	15	1034	1033
hypercube	3	8192	53248
can_24.mtx	1	24	68
jgl009.mtx	1	9	32
jgl011.mtx	1	11	49
rgg010.mtx	1	10	45
A-pores_1.mtx	1	30	103
B-ibm32.mtx	1	32	90
C-bcspwr01.mtx	1	39	46
D-bcsstk01.mtx	1	48	176
E-bcspwr02.mtx	1	49	59
F-curtis54.mtx	1	54	124
G-will57.mtx	1	57	127
H-impcol_b.mtx	1	59	281
I-ash85.mtx	1	85	219
J-nos4.mtx	1	100	247
K-dwt__234.mtx	1	117	162
L-bcspwr03.mtx	1	118	179
M-bcsstk06.mtx	1	420	3720
N-bcsstk07.mtx	1	420	3720
O-impcol_d.mtx	1	425	1267
P-can__445.mtx	1	445	1682
Q-494_bus.mtx	1	494	586
R-dwt__503.mtx	1	503	2762
S-sherman4.mtx	1	546	1341
T-dwt__592.mtx	1	592	2256
U-662_bus.mtx	1	662	906
V-nos6.mtx	1	675	1290
W-685_bus.mtx	1	685	1282
X-can__715.mtx	1	715	2975

Table 4
Bandwidth instances, including the number of instances I, maximum number of nodes N_{max}, and maximum number of edges E_{max}. These instances can be found online at: `optsicom.es` and `tamps.cinvestav.mx`

TSP

Name	Reference
Benchmark TSP LIB	iwr.uni-heidelberg.de math.uwaterloo.ca

VRP

Name	Reference
Benchmark Vehicle routing	neo.lcc.uma.es

Satisfiability

Name	Reference
Model RB Forced Satisfiable CSP and SAT	nlsde.buaa.edu.cn
Benchmarks SAT	aloul.net
Software Verification	cs.ubc.ca

Knapsack

Name	Reference
Data for the 01 Multiple Knapsack Problem	kreher1998combinatorial Martello1990 people.sc.fsu.edu
Data for the 01 Knapsack Problem	people.sc.fsu.edu

Location

Name	Reference
Capacitated Facility Location Problems	Ka-1999
Beasley	Beasley-1990
Real data instance , Italian cookie factory	di.unipi.it

Scheduling

Name	Reference
Resource-Constrained Project Scheduling (RCPS)	om-db.wi.tum.de
RCPS with time-dependent resource capacities	om-db.wi.tum.de
RCPS with Minimal and Maximal Time Lags	wiwi.tu-clausthal.de
Multi-Mode RCPS with Min./Max. Time Lags	wiwi.tu-clausthal.de
Resource Investment with Min./Max. Time Lags	wiwi.tu-clausthal.de
Multi-Mode RCPS (MRCPS)	Kolisch1999

Timetabling

Name	Reference
ITC2007	cs.qub.ac.uk
ITC2002 PATAT	sferics.idsia.ch
Benchmark Data Sets in Exam Timetabling	Qu:2009survey cs.nott.ac.uk

Table 5
TSP, VRP, Satisfiability, Knapsack, Location, Scheduling, and Timetabling instances.

Graph Isomorphism

Name	Reference
Graphs for Practical Graph Isomorphism	Neuen2017
	www.lii.rwth
Nauty and Traces	McKay201494
	pallini.di.uniroma1.it
bliss: A Tool for Computing Automorphism Groups	www.tcs.hut.fi
Algorithm conauto for Graph Isomorphism Testing	sites.google.com
Saucy3: Fast Symmetry Discovery in Graphs Sakallah	vlsicad.eecs.umich.edu

Cutting Stock (CS)

Name	Reference
Standard one-dimensional CS (single stock length)	Scheithauer2001†
One-dimensional CD (multiple stock lengths)	G-2002†
The hard28 set for 1D-BPP	BELOV200685†
One-dimensional setup minimization	Belov2007†
Two-dimensional two-stage constrained cutting problem	BELOV200685†
Two-dimensional strip-packing problem	below2008lower†
3D OPP	belov2009lp
Two-dimensional orthogonal feasibility problems	MESYAGUTOV20122425
Optimal Clustering Problem	Belov2013
Hard selection for 1D-CSP and 1D-BPP	www.math.tu-dresden.de
Set of 53 non-IRUP instances	www.math.tu-dresden.de

Table 6
Graph Isomorphism and Cutting Stock instances. †Dataset available from: www.math.tu-dresden.de

Complexity

Research Article

A Methodology for selection and determination a subset of Heuristics for hyper-heuristics through meta-learning for solving Graph Colouring and Capacitated Vehicle Routing Problems

Lucero Ortiz-Aguilar[1], Martín Carpio[1], Alfonso Rojas-Domínguez[1], Manuel Ornelas-Rodriguez[1], H. J. Puga-Soberanes[1], and Jorge A. Soria-Alcaraz[2]

[1] Tecnológico Nacional De México, Instituto Tecnológico de León, Guanajuato, Mexico.
[2] Department of Organizational Studies, University of Guanajuato, Guanajuato, Mexico.

Correspondence should be addressed to H. J. Puga-Soberanes; pugahector@yahoo.com

Abstract

In this work, we focus on the problem of selecting low-level heuristics in a hyperheuristic approach with offline learning, for the solution of instances of different problem domains. The objective is to improve the performance of the offline hyperheuristic approach, identifying equivalence classes in a set of instances of different problems and selecting the best performing heuristics in each of them. A methodology is proposed, as the first step of a set of instances of all the problems, generic characteristics of each instance and the performance of the heuristics in each one of them are taken into account to define vectors of characteristics and make a grouping in classes. Statistical tests and Meta-learning are used to select the heuristics for each class. Finally, we used the Naive Bayes to test the set instances with k-folds cross validation and we compared all the results statistically with the best-known values. In this research, the methodology was tested by applying it to the problems of vehicle routing (VRP) and graph coloring (GCP). The experimental results show that the proposed methodology can improve the performance of the offline hyperheuristic approach, correctly identifying the classes of instances and applying the appropriate heuristics in each case. This is based on the statistical comparison of the results obtained with those of the state of the art of each instance.

1 Introduction

The Vehicle Routing Problem (VRP) has different restrictions such as minimizing distance, time, capacity, and deliveries. This problem aims to find the sub-tour of n cities, without repeating two cities on the same tour or different tours. In the Capacitated Vehicle Routing Problem's (CVRP) state of the art research exist different variants that consider this basic definition and extra restrictions. On the other hand, the Graph colouring problem is a well know problem, which has been solved by exact methods, heuristics, meta-heuristics, and hyper-heuristics.

In the literature, the exact methods have been applied to both problems in which the restrictions are flexible limited, therefore it is possible to propose a solution in a short period

of time before the problem expires. By contrast, the heuristic and meta-heuristic performances depend strongly on their design and parameter tuning. This design in some cases must be ad-hoc to each problem or problem domains.

In recent years, the born of new approaches as Hyper-heuristics, whose aim is focused on their generality level to solve different problem domains. For many years, the hyper-heuristics has been growing in popularity in both of its main categories: heuristic selection and Heuristic generation. In the context of combinatorial optimization, Hyper-heuristics was defined as "heuristics to choose heuristics" [9]. In recent research conducted by Burke et al. [9], it was defined as "an automated methodology for selecting or generation heuristics to solve computational search problems".

However, designing or selecting an appropriate algorithm for a given problem sometimes requires expert knowledge. Likewise, each problem can have different characteristics or restrictions, which in some cases makes it relatively difficult to propose a good solution. For this reason, several questions arise, within the three heuristic schemes mentioned above:

1. Is it possible to design methods that significantly improve the results obtained instances sets? (This question has been previously proposed by Amaya et al. [2]).
2. Is it possible to design methods that improve all results for whole a family of problems? For example, a method to solve all the Vehicle Routing problems: Capacitated VRP, Capacitated VRP with Time Windows, Capacitated VRP with Pick-up and Deliveries and Time Windows, Multiple Depot VRP, Multiple Depot VRP with Time Windows, Periodic VRP, and Vehicle Routing Problem with Pick-up and Deliveries.
3. Is it possible to predict the best solution method for a specific instance? (this question has been previously proposed and given a solution proposal by Amaya et al. [2]).
4. Is it possible to design a method or a low-level heuristic that improves results for different problem domains?
5. Is it possible to have an instance design methodology that allows solving different problems with the same tool, without making ad-hoc modifications for each instance?

It is important to mention the steps that are needed to solve a problem:

- Modelling the variables and problem restrictions: For example, we can give a mathematical formal definition, a set of constraints for each variable, a range of values for each variable, etc.
- Selection of solution method: According to the nature of the variables, whether they are continuous or discrete, a more suitable method will be chosen for them (i.e. a Genetic Algorithm for the discrete case and EDA for continuous case).
- Design of solution method: For some Metaheuristics and Hyper-heuristic it is necessary a parameter adjustment.
- Application of the solution to the problem.

Based on the questions and the steps aforementioned, We denoted there are important questions to be solved: knowing the best method to solve a problem and having a methodology that allows us to apply it with several problems without ad-hoc adjustments. Our proposal presented on this work responds to these two points.

For many years, some researchers aim were to know which the best algorithm is to be used for a specific instance or set of problems was reported by Rice J. [31]. Rice J. proposed

how to choose a method from a preselected set of algorithms, achieving effective results for a set of problems. Some Hyper-heuristics variations depend on the type of learning used or the nature of heuristics, e.g. with online learning [29] and offline learning [42]. The meta-learning attracted much attention in the optimization field [35] [21] and its use in hyper-heuristics can be found in Amaya et al. [2]. Song et al. [35] studied the different meanings used in both fields and presented a taxonomy for each interaction between meta-learning and optimization.

In our work, we make three contributions.

- First is a methodology to refine the stage of inputs modelling information for a problem. There are some cases in which there are difficulties to model all the restrictions to each of the Meta-heuristics or heuristics. With the use of structures extended by the API-Carpio methodology, it is not necessary to make ad-hoc modifications to the algorithm for the restrictions, these consider hard requirements or restrictions of the problem. Therefore, this allows us to reduce the preparation time to solve the problem and make the human expert interaction with the optimization technique more cooperative.
- The second contribution is characterizing and classifying the instances from different domains using meta-learning knowledge. The correct characterization is the key to select the best solution method [31]. Consequently, this affects the Hyper-heuristic design, which in our work is focused on off-line learning. We used two different well-known domains Graph colouring and Capacitated Vehicle Routing Problem, which have different restrictions and variable numbers. We choose the meta-learning approach since, in industrial topics and the academic area, it is important to generate knowledge based on the previous information stored to predict future actions or making decisions.
- Finally, to do a better design or choice of Hyper-heuristics and predict which is the best algorithm according to the previous instance classification we propose to use statistical analysis of the heuristics, which will allow improve these points. Our proposal also provides information that allows us to understand the performance of heuristics and hyper-heuristic in the problem of interest.

The remaining content of this article contains a description of related work in Section 2, which covers a review on heuristics, hyper-heuristics, and meta-learning. Problem definition and theory related to heuristics are reported in Section 3. Section 4 and 5 present the proposed Methodology. The found results and finding including performance comparison are described in Section 6. Finally, concluding remarks are presented in Section 7.

2 Related Work

In this section, we will give a view of the Heuristic and Hyper-heuristic algorithms. Moreover, we will define some basic concepts of meta-learning. Finally, we will give a discussion of the pros and cons of the presented methods.

2.1 Heuristics.

K-flip or *K-opt* heuristic was proposed by Lin, S., and B. Kernighan in [25] for the Travel Salesman Problem (TSP). This heuristic was based on the general interchange transformation, i.e. a city must change its position with another city on the same tour. Besides, this heuristic is one of the most popular for TSP [20], it has been applicated in other problems as Planar Graphs, Unconstrained Binary Quadratic Programming, and the study of its

complexity in SAT and MAX-SAT. The two points Perturbation is a case of *k-flip*, and we give a detailed description and algorithms of these heuristics in the following sections.

The *k-swap* heuristic is a similar heuristic and frequently confused with *k-flip*. The *K-swap* heuristic improves its performance as a perturbation moves when it uses two or three movements [7].

The Move to less conflict heuristic also known as Minimizing Conflicts was proposed by Minton et al. [26]. The Minimizing Conflicts heuristic has been applied to different areas in Computer Science as hyper-heuristics, Graph Coloring Problem, Pickup-and-Delivery Problems, and Scheduling problem. The Move to less conflict heuristic is a variant of first fit the only difference is that the first one takes a random variable and changes its value for another that generates the least cost.

First- fit heuristic was proved by Brenda S. Baker [6] for the Bin-Packing problem. On the other hand, in recent decades this heuristic was applied to best-known problems as bin packing, virtual machine relocation problem, and cutting stock. A remarkable variety of heuristic is *Worst-fit* which was studied by Brenda S. Baker [6] and J. Csirik [12], in particular, its application on Bin packing Problem.

Soria et al. [36] proposed three heuristics for University Course Timetabling Best Single Perturbation (BSP), Statical Dynamic Perturbation (SDP), and Double dynamic Perturbation as part of the pool low-level heuristics for Hyper-heuristic. Moreover, these heuristics were applied to the VRP in later research [38].

2.2 Hyper-heuristics

We focused on offline learning Hyper-heuristics selection with perturbation heuristics, whose aim is to gather knowledge in the form of rules or programs, from training set instances. With this training-set, our aim is to find a generalized methodology that can solve unclassified instances. Usually, the offline selection hyper-heuristics belongs to machine learning methods, which are trained to create a tuned methodology for a problem domain [42]. Yates and Keedwell [41] demonstrate that sub-sequences of heuristics were found in the offline learning database that is effective for some problem domains. They used the Elman network to compute sequences of heuristics which was evaluated on unseen *HyFlex* example problems, and the results obtained are capable of intra-domain learning and generalization with 99% confidence.

One of the crucial issues in Hyper-heuristics design is the quality and size of the heuristic pool [37]. Soria et al. [37] proposed a methodology using non-parametric statistics and fitness landscape measurements for Hyper-heuristics design. This methodology was tested on course timetabling and vehicle routing problems; their hyper-heuristic proposal had a compact heuristic pool and competed with some traditional methods in course timetabling. In the course timetabling problem, they obtained five best-known solutions of 24 PATAT instances.

Finally, a recent report by Amaya et al. [2] documented a model for creating selection hyper-heuristics with constructive heuristics. The effectiveness of the model proposed by Amaya depends on the delta's values used, which is useful with higher deltas.

2.3 Meta-learning

The importance of meta-learning, matching learning, and optimization has been studied by Song et al. [35]. The meta-learning aim concern with accumulating and adapting experiences on the performance of multiple applications of a learning system. The meta-learning field also known as "*learning to learn*" [8] and It brings systems that can help by searching patterns across different tasks to control the process of exploiting cumulative expertise. The meta-learning concept has been present in the field of heuristics and Meta-heuristics for TSP [17], the Quadratic Assignment Problem, and Hyper-heuristics.

The TSP was chosen as a benchmark problem for several applications of meta-learning. First, Kanda et al. [21] proposed an approach based on the quality of the meta-features that describes the instances. Besides, they used four different sets of features based on properties of TSP instances as edge and vertex measures, complex network measures, properties of meta-heuristics, and sub-sampling land markers properties. Their principal contribution was the methodology to address the computational issues of two of the approaches to compute the meta-features and to combine them with the sub-sampling landmarks approach.

On the other hand, Gutierrez-Rodriguez et al. [17] uses VRP with time windows and proposed a methodology based on meta-learning to select the best meta-heuristic for each instance. Besides, their proposal shared and exploited an off-line scheme for the instant solutions of academy and industry. Their main contributions were to propose a set of features for characterizing VRPTW instances and design a classification process that predicts the most suitable meta-heuristic for each instance. Nevertheless, they assumed that the solutions of the set instances could be stored, shared, and exploited in an off-line scheme for predicting good solvers for new unseen instances.

The aim of this paper is not to present a survey on heuristics or hyper-heuristic, our proposal is slightly different. Our proposal considers some vital aspects of the researches, including the ones from Yates and Keedwell [41]. We took the Offline Hyper-heuristic approach from Soria et al. [37] and the statistical approach to selecting a pool heuristic from Kanda et al. [21]. The Offline Hyper-heuristic approach is an effective and popular method on the Matching learning area [35]. On the other hand, the statistical approach to selecting a pool heuristic is a useful and reliable method, because it takes statistical information from the input data

3 Combinatorial Problems

Our methodology is a general approach with competitive performance across several classes of problems. Thus, we used two problem domains: Graph Colouring and Vehicle Routing Problem. In the following sections, we will review the formal definition of each of these problems as well as their benchmark instances.

3.1 Graph Colouring

Graph Coloring Problem (GCP) consists of labeling each vertex of a given graph G with q colors. The problem must consider that the restriction of the two adjacent vertexes must be labeled with a different color. The Graph Coloring problem demonstration as NP-Hard problem was proposed by Karp, R.M. [22]. The benchmark instances can be found in [1]. In this problem,

[1] http://mat.gsia.cmu.edu/COLOR/instances.htmlmat.gsia.cmu.edu

the only hard constraint is not repeat two adjacent nodes with the same color. Let a graph G with V a set of nodes $v_1, \dots, v_n$ and $E = e_1, \dots, e_m$ edges. The cost o a problem solution is the sum of the nodes connected with the same color as:

$$F(G) = \sum_{i=0}^{n} \sum_{i=0}^{n} f(m) \quad (1)$$

$$f(m) = \begin{cases} if(v_i, v_j \epsilon E) \cap (q_{vi} == q_{vj}) & 1 \\ othercase & 0 \end{cases} \quad (2)$$

3.2 Capacitated Vehicle Routing Problem

The Capacitated Vehicle Routing Problem (CVRP) is a variant of VRP [13]. In this problem we have an undirected graph G, m vehicles, Q capacity and a set of Cities $C = \{c_0, c_1, \dots, c_n\}$. Formally the City c_0 is the $depot$, each vehicle this must visit these cities starting from the $depot$ and coming back to this.

Alba and Dorronsoro [1] define a distance or travel time matrix $D = (d_{I,j})$ between cities c_i and c_j. Each city c_i has a demand of things q_i. We denote it as a route $R = \{\vec{r}_0, \vec{r}_1, \dots, \vec{r}_m\}$, and $\vec{r}_i$ is a permutation of the cities , starting and finishing at the depot c_0. The cost of a problem solution is the sum of the costs of each route of R as:

$$F(R) = \sum_{i=1}^{m} Cost(\vec{r}_i) \quad (3)$$

$$Cost(\vec{r}_i) = \sum_{j=0}^{k} d_{j,j+1} \quad (4)$$

This problem aims to determine for each vehicle the lowest cost (see Eq. 3 and 4) tour or distance or travel time, considering the max capacity. Nota bene the hard constraints are the capacity of each vehicle.

4 Methodology of Design

In this section, we describe the methodology used for the experimentation for two Combinatorial problems based on the Methodology API-Carpio. We integrated methodology API-Carpio [10] and methodology of the design proposed by Soria et al. [4].

Table 1. MMA Matrix

City/Node	N_1	N_2	...	N_{n-1}	N_n
N_1	**10**	99	91	97	137
N_2	99	**101**	88	80	96
...	91	88	**58**	33	32
N_{n-1}	97	80	33	**35**	10
N_n	137	96	32	10	**68**

4.1 Methodology API-Carpio

This Methodology was born to solve the *University Course Timetabling* and it considers three factors: Students, Teachers, and institutions (infrastructure). This methodology uses

several structures for the equations previously described. One of the most important structures of this work is: MMA. This matrix is constructed with information of the cities or nodes. For Graph Coloring, we use the information of adjacency matrix while for CVRP it is considered the cost Matrix. Table 1 shows an example of an MMA matrix.

4.2 Methodology of design

This methodology was extended from the proposal by Carpio [10] and their formal definition was proposed in Soria et al. [5]. The methodology of design by Soria allows us to consider the objectives of *Course-Timetabling* and to satisfy the different restrictions, by converting these to lists of time and space restrictions, it is seeking to minimize student conflict. To use this methodology, two structures are used to consider restrictions and variables: MMA Matrix, and LPH. The LPH has information about the possible time slot of each subject can be assign. An example of this list can be found in table 2. The list shows in each row the number hour, i.e. the subject N_2 can be assigned in time-slots 1,2, 4, 5, 7 or 8, but not in 6. The algorithm for generating artificial instances of LPH can be found in Ortiz-Aguilar [28].

Table 2: LPH List. $N_1 \ldots N_n$ represents the subjects

Node or City	Timeslots							
N_1	1	2	3	4	5	6	7	8
N_2	1	2	3	4	5		7	8
N_{n-1}	1	2	3	4	5	6	7	8
N_n	1	2	3	4	5	6	7	8

5 Methodology for Selecting and determining a subset of heuristics

In this section, we propose a new approach for selecting and determining a subset of a heuristics to solve GCP and CVRP instances. We describe our methodology in next steps and the graphical representation of our methodology is shown in figure 1.

1. **Variables and Problem Restrictions Identification**. First, the variables and restrictions of the problem are identified according to the problem aims or objectives. To model the GCP, the values in the MMA matrix represent the weights of the edges of nodes. If there is a zero in a certain position (x, y) in the matrix, this represents no connection between those nodes. For graph coloring, each one node is colored considering that the adjacent nodes do not have the same color. CVRP is aligned with our methodology due to its aim seeks to get sub-routes in which the tour-cost (sub-group) must be the minimum or the cheapest.
2. **Problem's restrictions modeling.** In both problems, we must design a partition of nodes or cities. First, it is necessary to model all the restrictions for each variable in an LPH, e.g. a node cannot be colored by a specific color or a restricted city for a tour. Then it must design the MMA which represents the edge or connection weight between nodes or cities. On GCP the adjacency matrix corresponding to MMA and in CVRP the MMA matrix will be the matrix that has the distances of the node to node. For GC, our LPH is constructing based on the number of colors in which the nodes can be labeled. In case the problem restricts colors to five, the list will be like the one shown in table 2. Similarly, this list will be built for the CVRP, where the number of trucks is the number of *time-slots* that should be represented on the list (see table 2). For the problems used in this work, it was not

necessary to elaborate additional structures for soft restrictions. Besides, for an extensive review and how to model additional restriction, the research proposed by Ortiz [27] details all the possible cases and different features.

3. **Given a set of instances denoted as I. For each of the instances I_i apply a number k of times the heuristic H_j**. After modeling the inputs for each problem, the next step is to apply the heuristics. In this stage, each instance is pre-processing to obtain more information about the problem. This lets to improve the designing and testing the Hyper-Heuristic algorithm. The algorithms description used will be seen in section 5.1.1. The goal in this stage is to determine which heuristics can have a better performance individually for all the problem instances. This improves the next part in which the Hyper−heuristic must choose the sequence application each heuristic and it uses a minimal pool of heuristics which is a fundamental part of it [29]. For all instances, it will be applied k times for each heuristic. it is possible after this step, that instances can be solved to the best solution due to their complexity. This means that it is possible to avoid executing a complete and expensive computation process when solving problems with the application of a simple heuristic.
4. **Apply the nonparametric statistical tests to the instances, by problem domain. Example: Friedman, Friedman alienated and Quade for all instances of graph coloring and separately the same 3 tests for the CVRP**. For the $inter-feature$, it is necessary to obtain the ranks (based on the performance) of each heuristic per problem.
5. **Determine the number c** . To determinate the subclasses with Sturges Rule [33] with:

$$c = 1 + \frac{log(T)}{log(2)} \quad (5)$$

Where T is the total number of data**.**

6. **Generate the pattern based on the *inner* and *basic features*, per instance. The *basic features* will be the problem information and the *inner features* will be the fitness obtained by the heuristics applied k times to the problem.** The description and generation of characteristics which permit differentiate into at least two groups of instances within the same problem class. We used the term of basic-feature and $inter-feature$ based on the proposal conducted by Gutierrez-Rodríguez et al. [17]. As basic-features, these are given by the problem e.g. the number of nodes, colors, trucks, so on, depending on each problem information. For both classes of problems the number of different $basic-features$ is summarized in table 3.

Table 3: Basic-Features for CVRP and Graph Coloring

Problem	Min Partition	Max Partition	Variables	Edges
VRP	Trucks	Trucks	Cities	Cities connections
Graph Coloring	Lower Bound	Upper Bound	nodes	Edges

The fitness performance values of all heuristics are the inter-feature key. Finally, the pattern per instance is *Basic-feature+inter-feature*. The final pattern is shown in table 4. For example, instance 1 has pattern (3,3,8,50,3,2,1,4), the number of $inter-feature$ is according to the pool heuristics (eight features for the given example).

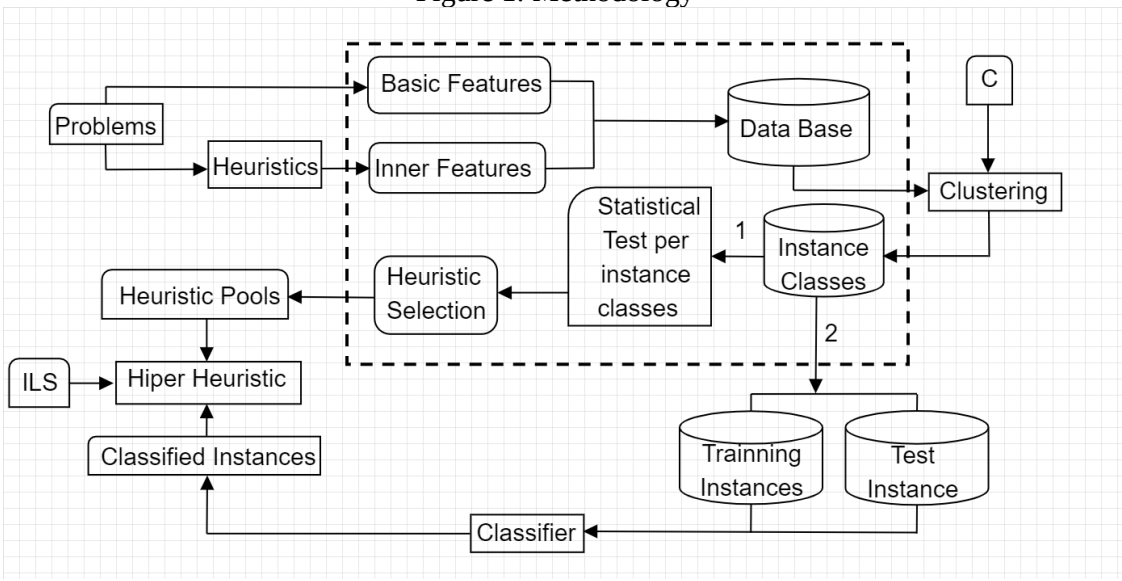
Table 4: Characteristics for instances

Instance	Lower Bound	Upper Bound	Nodes	Edges	H0	H1	H2	H3
1	3	3	8	50	3	2	1	4
2	5	10	6	36	4	3	1.5	1.5
3	3	8	25	80	3	1	2	4

4	10	10	50	80	2	4	3	1

7. **Apply the $k-means$ algorithm to group all instances into c groups**. Clustering the instances with the k-means algorithm. The corresponding cluster number of each instance is the number of classes.
8. **Label each pattern according to the group number in which each pattern (instance) was classified.**
9. **Apply again the three statistical tests to the results of the heuristics (step 3), but now by problem class (those obtained from the grouping of $k-means$, step 7 and 8).** The heuristic with the lowest rank value is which had better performance [37].
10. **According to the r_j of each of the non-parametric tests as a reference, a cut-off is established for each test: FC_{rj} for Friedman, FAC_{rj} for Aligned Friedman and QC_{rj}.** According to the experimental behavior of the heuristics.
11. **Choose only the heuristics that are below the tolerance established in the three tests per class.** In the end these heuristics will the minimum set per class.
12. **Separate the patterns (step 6) into a training and test set to proceed to the classification phase.** It is important to consider at least one pattern of each class for the test set.
13. **Use the classifier with the training set to make necessary adjustments to it**. After describing and getting all pattern characteristics per instance, the next step is training and testing all instances by a classifier. For our approach, we prefer to use a simple classifier as Bayesian, because, our objective was not to compare the performance between classification algorithms or to design ad-hoc classifiers to our research. The NBC simplifies learning by assuming per class that all features are independent [24]. In our methodology, we assume that each heuristics performance is independent because we applied each heuristic on independent experiments. In the previous stage, each experiment must be run with only one heuristic, thus we did not apply two or more heuristics at the time. Finally, all the features in the created data-set Instances were normalized before applying the classifier.
14. **Finally, the set of test instances will use with the classifier to assign a "class" and solve it with its corresponding set of heuristics**.

Figure 1: Methodology



5.1 Design and testing the Hyper-heuristic offline learning with K −folds

To choose the minimal set of heuristics and design the Hyper-heuristic for each class in more detail, our methodology considered the Hyper-Heuristics with Off-Line training due to it has demonstrated good results for constraint satisfaction problems in terms of generality solution [42]. A random constructive heuristic was used to generate solutions to our problems of GCP and for CVRP a greedy algorithm. A selection hyper-heuristic algorithm has three components: the pool of operators (low-level heuristics), a high-level search strategy, and a control mechanism to select the operator will be applied at each search step.

5.1.1 Low-level heuristics

The heuristics were applied to the two problems and their respective instances were:

1) ***k-flip* / Simple Random Perturbation (SRP) (H1)**. This heuristic change the value of one or more variables (in some cases k) for another feasible value. The GCP aim is to change the color of a certain node by another [40]. Finally, for CVRP the movement implies changing a city to another specific truck [20].
2) ***K-Swap*/ Kempe Chain Neighbourhood/ S-chain neighborhood (H2).** It must be selected two or more varieties and then interchange their values among them when possible, otherwise, the change is not made. We exchange the color between nodes previously selected on GCP. This heuristic is using in works related to TSP or CVRP, also it called k-interchange [34] [14].
3) **Best Single Perturbation (BSP) (H3)** . This heuristic chooses a variable according to the List of hard restrictions (LPH) and changes its value. This exchange produces a better cost or in the worst case, the same cost [36]. Next time this heuristic is going to apply, the next variable will be chosen according to the next position of the last variable which was modified. The next node which has to change color will be selected according to the last variable chosen, for the Graph Coloring problem. The CVRP must be changing the city of Truck to another truck.
4) **Statical Dynamic Perturbation (SDP) (H4).** It is also known as Statically Dynamic Perturbation (SDP). It is based on the variable selection on a probability distribution of the frequency in the last k iterations. This heuristic chooses a variable and changes its value randomly [36]. The variables with fewer changes will have a higher probability to be selected. Applied to GCP, it would be a node with fewer color changes and for the CVRP the city that has moved a few times from Truck.
5) **Two Points Perturbation (2pp) (H5).** It is also known as k-opt, it is a particular case of the K-Swap, with a value of $k = 2$.
6) **Double dynamic Perturbation (DDP)(H6)**. This heuristic is based on the SDP, this receives a solution and it modifies the value of a variable concerning a probability distribution. The difference is that a copy of the initial solution is kept and in the end, the best of the two solutions is returned [36].
7) **Move to less conflict (MLC)(H7).** This selects a random variable and it assigns to a part of the value which generates the least cost [38]. In GCP, the color changes according to another which improves the fitness, and for CVRP the city is moved to another truck where the total distance of the route is minimized.
8) **Min-Conflicts (H8).** The heuristic selects a random variable and it assigns to a part in which generates the cheapest cost [38]. On GCP the heuristic must change color to the selected node with another which improves the result. For CVRP the selected city in which the total distance of the route is minimized.

9) **First-Fit (H9)** It changes the value of a variable to another, which is the least repeated in other variables [38], i.e. on CVRP the heuristic will take a city and it will change it to the truck that has fewer cities in its route. For the GCP, it will select a node and it will assign the color that is least repeated.
10) **Worse-fit (H10).** It assigns the most repeated value if possible, without violating the hard constraints to a randomly selected variable [18]. For GCP and CVRP, we assign a node or city to the most repeated timeslot, color, or truck.
11) **Burke-Abdullah (BA) (H11).** This heuristic was proposed by Abdullah et al. [32], in which it chooses a variable applying Fail-First or Brelaz Heuristic [30] and its value changes according to the one that has obtained better performance by applying the following algorithms: Minimum conflict, Randomly, Sequential selection and least Constrained.

5.1.2 High-level search strategy

The iterated local search algorithm was used as a High-level search strategy. This Meta-heuristic was proposed by Lourenço et al. [19] and it is constructing a sequence of solutions generated by an embedded heuristic. The generated solutions could be better if they were only constructed randomly. The essence of this algorithm is to intensify an initial solution, exploring neighboring solutions to it. The algorithm is shown in 5.1.2, was taken from Talbi El-Ghazali [15].

Algorithm 1 High Level Iterated Local Search (ILS)

```
s0 = GenerateInitialSolution
s* = ImprovementStage(s0)
while !StopCondition() do
    s' = SimpleRandomPerturbation(s*)
    s*' = ImprovementStage(s')
    if f(s*') < f(s*) then
        s* = s*'
    end if
end while
return s*
```

Algorithm 2 Improvement Stage

```
ls ← IncumbentSolution
while! LocalStopCriteria() do
   hi = Perturbate()
   ls* = apply(hi, ls)
   if f(ls*) < f(ls) then
      ls = ls*'
   end if
end while
return ls
```

5.1.3 Selection operator

In the perturbation phase (step 4 in the algorithm 2) it is necessary to choose a variable following a probability distribution based on the frequency of variable selection in the last k iterations. This simple heuristic allows us to modify the methodology solution.

We used the same Hyper-heuristic Framework and according to each class, we gave a different pool of low-level heuristics. For example, if for class 1 the best heuristics were H1, H8, H10, and H5, these were our pool for the Hyper-heuristic. With a similar procedure, the design was done for each class. After this process, we trained each hyper-heuristic with their respective classes, for the next stages.

6 Experimental results

This section describes our experiments in detail for Graph coloring and CVRP benchmarks used in this paper. We give the configuration for the implementation of the Iterated Local search Hyper-heuristic. Finally, we described the statistical tests that we used to compare our results of experimental methodology.

Our approach was implemented in JAVA language with JDK 1.8 using the IDE NetBeans IDE 8.2. The experiments were executed on a computer with processor Intel i7-7700U, 2.6 GHz, 16 GB DDR3 RAM, and operational system Windows 10 Home. The tests presented in this work were executed in a common notebook, with a single processor, it is showing the effectiveness of the exposed methodology.

For each heuristic, a limit of 100,000 function calls was given in each test run, for all instances. We applied the Shapiro Wilks test to check if the data results were normal or not, hence choose a better representative (average or Medium). If the data behavior is according to a normal distribution, the average was taken as representative and otherwise the median.

6.1 Heuristics results to Graph Colouring and CVRP

6.1.1 Graph Colouring

We used the benchmark proposed for the second DIMACS challenge on Graph Coloring [23] and this is tested with 41 runs. In tables 5 and 6 we shown our results. We denote the best results with a bold face, only *myciel2* instance was solved with the application of individual heuristic in their optimum.

We applied a non-parametric test to prove that exist differences between the performance of each heuristic. The table 7 shows the ranges obtained in the three statistical tests for Graph Coloring instances. The three omnibus tests indicated there are significant differences between the heuristics. The heuristics which obtained the highest ranks in the three tests were H6 and H11 i.e. these have the worse performance. The H3 and H8 were the best results for the problems, this is because in all omnibus tests those heuristics obtained the lowest ranks (see table 7 marked in bold).

6.2 Capacitated Vehicle Routing Problem (CVRP)

Three sets of the state of the art were used and tested with 41 runs:

1. Augerat et al. (SET A) 9 instances, proposed by [3]
2. Christofides, Mingozzi and Toth (CMT) 14 instances, proposed by [11].
3. Golden, Wasil, Kelly and Chao (GWKC) 20 instances, proposed by [16]
4. Uchoa et al. 9 instances, proposed by [39]

Table 5: Heuristics Results for Graph Coloring Instances

#	Name	Nodes	Colors	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
1	1-Insertions_4	67	3	12	18	11	15	10	66	61	13	25	28	75
2	2-Insertions_3	37	3	4	7	4	5	4	24	15	4	11	10	29
3	2-Insertions_4	149	3	33	41	16	32	32	148	158	29	45	47	155
4	3-Insertions_3	56	3	5	9	4	4	5	33	24	4	9	9	40
5	4-Insertions_3	79	2	14	19	16	13	11	61	66	13	25	22	71
6	anna	138	10	17	17	14	17	16	59	45	13	35	30	68
7	ash331GPIA	662	3	99	76	85	96	82	1067	724	89	164	156	1082
8	ash608GPIA	1216	3	1497	1448	729	1541	1370	2003	1120	1083	1513	1485	2066
9	ash958GPIA	1916	3	2618	2553	1900	2621	2462	3161	1999	2039	2616	2624	3206
10	david	87	10	16	18	14	16	16	49	48	15	35	37	58
11	DSJC1000.1	1000	19	1847	1801	610	1837	1781	2565	1019	942	1789	1821	2634
12	DSJC1000.5	1000	82	2432	2513	838	2425	2379	3130	1126	1230	2327	2327	3234
13	DSJC1000.9	1000	214	1856	2073	828	1850	1824	2354	1204	1062	2375	1768	2414
14	DSJC125.1	125	4	48	59	36	48	54	161	97	37	56	66	173
15	DSJC125.5	125	15	113	156	72	114	124	275	133	68	250	112	298
16	DSJC125.9	125	42	125	173	77	124	131	210	132	72	208	112	221
17	DSJC250.5	250	25	350	415	146	349	368	658	278	157	339	326	681
18	DSJC250.9	250	70	311	388	152	315	319	477	234	153	487	294	485
19	DSJC500.1	500	11	629	646	197	634	598	1085	532	279	623	614	1114
20	DSJC500.5	500	47	927	1007	290	911	905	1381	471	368	859	853	1412
21	DSJC500.9	500	122	771	927	326	759	763	1044	487	382	1051	731	1129
22	DSJR500.1	500	11	124	130	33	125	119	334	128	46	129	127	343
23	DSJR500.1c	500	84	1257	1439	203	1262	1259	1566	1162	475	1539	1210	1589
24	DSJR500.5	500	121	376	425	201	372	369	619	299	211	627	352	662
25	fpsol2.i.1	496	64	141	183	82	139	122	263	169	91	255	262	267
26	fpsol2.i.2	451	29	52	81	59	51	52	337	281	52	204	215	348
27	fpsol2.i.3	425	29	51	82	55	51	54	364	272	54	211	232	368
28	games120	120	8	16	24	11	17	17	82	34	10	23	23	88
29	homer	561	12	23	22	22	24	23	165	139	24	53	44	176
30	huck	74	10	13	13	12	13	13	46	38	12	27	24	47
31	inithx.i.1	864	53	78	95	81	80	80	423	320	115	264	263	444
32	inithx.i.2	645	30	55	100	62	56	59	525	459	85	350	324	531
33	inithx.i.3	621	30	59	108	65	56	63	510	462	78	338	331	539
34	jean	80	9	13	12	13	14	12	47	31	13	27	26	46
35	le450_15a	450	14	292	293	103	291	291	580	410	137	292	289	597
36	le450_15b	450	14	289	297	106	290	285	589	427	125	300	288	624
37	le450_15c	450	14	716	762	358	713	713	1187	887	424	734	707	1179
38	le450_15d	450	14	723	751	371	718	733	1163	799	418	728	725	1197
39	le450_25a	450	24	160	162	50	158	153	372	229	63	164	162	395
40	le450_25b	450	24	156	161	51	164	160	377	188	58	156	156	404
41	le450_25c	450	24	407	417	150	400	393	736	510	180	396	394	780
42	le450_25d	450	24	400	422	150	400	403	736	461	179	397	394	791
43	le450_5a	450	4	780	772	484	755	738	1195	966	551	753	771	1209
44	le450_5b	450	4	769	768	498	759	738	1192	886	565	764	769	1280

In the table 8 we showed the fitness values for the instances and the lowest city cost tour is indicated in bold; where n is the number of nodes, Q is the capacity of each truck and k is the number of trucks (colors in the case of Graph Coloring).

We applied the same procedure with the statistical tests of Friedman (FT), Alienated Friedman (AFT), and Quade (Qt) to distinguish the behavior of heuristics set. We established $\alpha = 0.05$ and $h_0 =$ as there are no differences between the performance of the heuristics and as h_a as there are differences between the performance of the heuristics. Table 9 shows the ranks obtained in the three statistical tests.

Table 6: Heuristics Results for Graph Coloring Instances

#	Name	Nodes	Colors	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
45	le450_5c	450	4	1459	1476	909	1422	1394	2024	1692	1083	1428	1464	2073
46	le450_5d	450	4	1464	1442	843	1449	1385	1981	1712	1060	1431	1436	2004
47	miles1000	128	41	52	58	49	51	49	133	75	49	129	70	137
48	miles1500	128	73	81	107	82	82	81	149	94	79	123	100	169
49	miles250	128	7	14	14	15	15	13	73	37	16	24	28	75
50	miles500	128	19	26	24	25	25	24	87	48	24	81	43	90
51	miles750	128	30	38	41	38	38	37	102	64	37	101	53	119
52	mugg100_1	100	3	5	12	4	6	5	55	19	5	12	12	57
53	mugg100_25	100	3	5	12	4	6	6	49	25	4	12	14	55
54	mugg88_1	88	3	5	10	4	5	5	43	19	5	9	10	50
55	mugg88_25	88	3	5	10	4	5	6	46	20	5	11	11	61
56	mulsol.i.1	197	48	60	68	60	60	57	142	76	59	131	80	153
57	mulsol.i.2	188	30	49	56	48	50	47	172	118	48	143	95	186
58	mulsol.i.3	184	30	50	56	48	49	47	177	117	48	160	99	176
59	mulsol.i.4	185	30	49	53	50	49	48	165	119	48	142	94	192
60	mulsol.i.5	186	30	50	62	48	50	50	171	114	48	107	100	191
61	myciel3	11	3	4	6	4	3	3	10	6	4	10	4	12
62	myciel4	23	4	5	9	6	5	5	24	12	5	15	9	27
63	myciel5	47	5	8	13	8	9	8	53	32	8	20	17	56
64	myciel6	95	6	18	24	12	19	19	135	105	13	37	51	143
65	myciel7	191	7	84	96	18	85	78	347	303	23	107	136	328
66	qg.order30	900	29	554	544	152	554	525	912	223	227	533	531	953
67	qg.order40	1600	39	1174	1105	627	1160	1078	1626	378	558	1131	1128	1671
68	qg.order60	3600	59	3076	2907	2425	3079	2897	3692	831	2234	3070	3008	3738
69	queen10_10	100	10	41	58	37	38	35	155	65	37	60	71	163
70	queen11_11	121	10	57	76	54	57	50	209	101	53	87	82	216
71	queen12_12	144	12	51	74	52	54	49	224	93	48	81	74	233
72	queen13_13	169	12	70	107	67	74	68	281	112	69	108	117	299
73	queen14_14	196	15	58	83	54	56	53	288	117	53	93	101	317
74	queen15_15	225	15	75	111	71	81	74	370	138	74	124	112	368
75	queen16_16	256	16	85	108	83	90	80	407	159	79	125	117	438
76	queen5_5	25	4	17	38	16	17	15	45	25	17	25	23	50
77	queen6_6	36	6	19	27	17	17	18	49	28	16	47	27	55
78	queen7_7	49	6	29	51	27	30	33	90	42	30	36	39	87
79	queen8_12	96	11	29	52	30	32	25	138	60	28	48	48	138
80	queen8_8	64	8	32	56	25	31	33	100	45	27	80	42	104
81	queen9_9	81	9	31	48	33	30	27	124	55	34	49	58	129
82	school1	385	13	997	1008	626	1009	987	1423	1330	691	1076	1037	1440
83	school1_nsh	352	13	723	742	445	735	720	1100	1048	490	734	781	1135
84	will199GPIA	701	3	438	418	438	453	430	1762	1542	442	547	555	1788
85	zeroin.i.1	211	48	62	72	62	61	60	147	80	60	126	89	159
86	zeroin.i.2	211	29	42	55	41	41	41	165	113	45	163	89	169
87	zeroin.i.3	206	29	40	55	43	40	39	179	111	40	162	91	164

Table 7: Statistic test results for Graph Coloring instances for Friedman (F), Friedman Aligned (FA) and Quade (Q)

	statisc	p-value	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
F	255.72	0	4.64	6.40	2.18	4.68	3.33	10.01	7.52	2.48	7.40	6.47	10.89
FA	77.86	0	374.88	461.35	209.30	370.94	340.22	820.66	575.50	212.84	568.01	505.86	829.45
Q	99.04	0	5.27	6.49	1.91	5.24	3.68	9.99	6.84	2.44	7.11	6.13	10.90

In this case, the heuristic H5 has the lowest ranks for all the tests and H6 has the second-lowest ranks for QT and FT.

Table 8: Results for CVRP-Capacitated Instances

	Instance	Cities	Trucks	OP	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
88	CMT1	50	5	524.61	1583	1161	1569	1568	1117	1141	1664	1576	1664	1664	1664
89	CMT2	75	10	835.26	2353	1591	2351	2349	1728	1898	2577	2547	2577	2577	2577
90	CMT3	100	8	826.14	1904	1821	1939	2017	1699	1743	3293	2249	3293	3293	3293
91	CMT4	150	12	1028.42	2578	2674	2520	2519	2347	2462	5096	3115	5096	5096	5096
92	CMT5	199	17	1291.29	3159	4469	6490	3144	3008	3155	6742	6707	6742	6692	6742
93	CMT6	50	6	555.43	1027	1160	1569	1028	941	933	1664	1573	1664	1571	1664
94	CMT7	75	11	909.68	1492	1616	2351	1465	1357	1407	2577	2517	2577	2531	2577
95	CMT8	100	9	865.95	1743	1890	1799	1906	1598	1636	3293	2203	3293	3195	3293
96	CMT9	150	14	1162.55	2308	2768	2489	2325	2185	2256	5096	2971	5096	4918	5096
97	CMT10	199	18	1395.85	4607	4607	2852	2852	2791	2870	6742	6707	6742	6513	6742
98	CMT11	120	7	1042.12	2077	2077	2372	2372	2046	2110	2608	2449	2608	2608	2608
99	CMT12	100	10	819.56	1692	1692	1733	1733	1472	1528	2306	2144	2287	2300	2306
100	CMT13	120	11	1541.14	2040	2040	2041	2041	1827	1906	2608	2476	2608	2577	2608
101	CMT14	100	11	866.37	1717	1717	1609	1609	1444	1517	2306	2171	2306	2270	2306
102	GWKC1	240	9	5623.47	9497	9155	8607	9261	7350	8565	13312	10546	13312	13312	13312
103	GWKC2	320	10	8404.61	14571	16631	15161	13399	12320	10903	20961	16801	20787	20735	21080
104	GWKC3	400	9	11036.2	19197	21734	24161	20070	17167	18793	26117	22304	26117	26117	26117
105	GWKC4	480	10	13590	27046	30204	32382	26643	27133	26504	38732	31793	38892	38892	38892
106	GWKC5	200	5	6460.98	14224	15677	13009	14207	12800	13435	19558	15763	21378	21378	21378
107	GWKC6	280	7	8412.9	15407	21043	14925	15701	14022	14539	26699	17688	26699	26699	26699
108	GWKC7	360	8	10102.7	25909	23841	25909	25909	24611	25718	25909	25909	25909	25909	25909
109	GWKC8	440	10	11635.3	21500	23969	20817	20483	18560	18677	31506	26204	31506	31506	31506
110	GWKC9	255	14	579.71	1102	989	2323	1264	954	916	2581	2319	2581	2581	2581
111	GWKC10	323	16	735.66	1344	1312	3476	1277	1222	1229	3589	3451	3589	3589	3589
112	GWKC11	399	17	912.03	1751	1919	4758	1640	1572	1485	5040	4478	5039	5039	5040
113	GWKC12	483	19	1101.5	2295	1979	6738	2462	2192	2363	5416	6520	6750	6750	6750
114	GWKC13	252	26	857.19	2010	1301	2062	2031	1475	1615	2117	2078	2158	2151	2158
115	GWKC14	320	29	1080.55	2351	1637	2637	2322	2221	2389	2748	2665	2745	2744	2748
116	GWKC15	396	33	1337.87	3262	2062	3322	3207	2793	3025	3366	3308	3366	3364	3366
117	GWKC16	480	36	1611.56	3978	2596	4107	3820	3492	3783	4198	4092	4217	4212	4217
118	GWKC17	240	22	707.76	1754	1518	2163	1729	1611	1669	2376	2165	2376	2376	2376
119	GWKC18	300	27	995.13	2130	2165	3443	2150	2073	2155	3443	3443	3443	3425	3443
120	GWKC19	360	33	1365.6	3238	2808	4540	3024	3226	3575	4713	4466	4713	4713	4713
121	GWKC20	420	38	1817.59	3519	3746	6190	3511	3189	3327	6186	5992	6191	6084	6191
122	A-n32-k5	32	5	784	1538	1476	1431	1474	1265	1255	2230	1662	2229	2229	2230
123	A-n45-k6	45	6	1733.36	1733	1733	1733	1733	1273	1156	1733	1733	1733	1733	1733
124	A-n55-k9	55	9	1073	1660	1413	1453	1642	1439	1328	2376	1901	2376	2376	2376
125	A-n61-k9	61	9	1034	2500	1814	2338	2464	1731	1838	2592	2517	2592	2592	2592
126	A-n62-k8	62	8	1288	1772	1724	2039	1792	1551	1615	3194	2219	3194	3194	3194
127	A-n63-k9	63	9	1314	2480	1879	2452	2396	1813	1835	2677	2564	2677	2677	2677
128	A-n64-k9	65	9	1401	1935	1893	2169	1948	1744	1715	2613	2203	2613	2613	2613
129	A-n65-k9	65	9	1174	2609	1940	2600	2546	1820	1892	2949	2808	2949	2949	2949
130	A-n69-k9	69	9	1159	2149	1803	2080	1979	1626	1687	3220	2477	3220	3220	3220
131	X-n148-k46	148	46	43448	71826	54940	75062	71665	68924	71179	77623	78715	79116	79116	79116
132	X-n153-k22	153	22	21220	36700	39843	52741	36155	34116	34337	71801	56820	71799	71793	71801
133	X-n157-k13	157	13	16876	36364	24507	36364	36364	27917	29210	36364	36364	36364	36364	36364
134	X-n162-k11	162	11	14138	39232	37225	36381	38162	32918	34059	79973	60657	79969	79963	79973
135	X-n367-k17	367	17	22814	71209	85326	97502	72482	68648	70027	154244	112919	154244	154244	154244
136	X-n393-k38	393	39	38260	141432	74629	152513	143398	130939	148728	158058	155203	158058	158058	158058
137	X-n401-k29	401	29	66187	140100	106590	163158	141582	141056	145043	197323	163230	197323	197323	197323
138	X-n411-k19	411	19	19718	67963	79642	81034	67090	63038	66173	173269	88152	173269	173269	173269
139	X-n420-k130	420	130	10798	182376	132745	207213	183053	203757	200726	217483	216871	217483	217483	217483

Table 9: Statistic test results for Capacitated Vehicle Routing Problem instances for Friedman (F), Friedman Aligned (FA) and Quade (Q)

	statisc	p-value	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10	H11
F	420.84	0	4.49	8.79	9.69	3.53	5.59	4.16	1.60	2.46	9.32	6.92	9.45
FA	47.3	0	170.82	441.25	443.96	143.53	280.22	169.32	127.19	147.94	441.70	342.00	443.57
Q	79.97	0	4.29	8.87	9.72	3.68	5.59	3.96	1.67	2.57	9.26	6.91	9.47

6.3 Selection of features and classes by statistical tests

According to the steps mentioned in section 5.1, we must determine first the number o clusters or classes to split all our test instances. In this case $T = 139$ and $c = 8$.

We considered 8 classes and used k-means clusters and we expected uniformly distributed instances into the clusters. The k-means algorithm was applied with a maximum

number of $Iterations = 500$, initially random starting points. To consider the uniform distribution of classes into clusters, we used the Manhattan distance obtained after experimental work, with the best results.

Table 10 contains the classes details, number instances per cluster/class, number of GCP (3rd column), or CVRP (4th column) per class, min and max nodes, min and max number of color or nodes. In this experimentation for clusters 1,5,6 and 7, they were only for GCP instances; clusters 3 and 4 for CVRP, and only cluster 2 has both problems domain.

Table 10: Summary Cluster/classes with k −means algorithm

Cluster	# Instances	%	GC	CVRP	Min Nodes	Max Nodes	Min colors/trucks	max Colors/trucks
1	18	12.9	18	0	11	149	3	6
2	10	7.2	8	2	420	3600	3	214
3	11	7.9	0	11	148	480	5	46
4	5	3.6	0	5	367	420	17	130
5	19	13.7	19	0	64	256	7	30
6	31	22.3	31	0	125	900	11	122
7	11	7.9	11	0	352	701	3	14
8	34	24.5	0	34	32	483	5	33

6.4 Training and test the classifier for the Instance's classes

After the heuristic pool design phase for the Hyper-heuristics, we split our data-set into training and test. The training data set was created by 125 instances with 15 features (basic+ inter) and the unseen instances were made by 15 instances. The results of the classification with Naive Bayes was reported on table 11.

Table 11: Classification value results

Correctly classified	114	91.20%
Incorrectly classified	11	8.80%
Weighted Avg		
TP Rate	0.912	
FP Rate	0.022	
Precision	0.918	
Recall	0.912	

Table 12 contains the confusion matrix of the process classification. We observed that for some classes as 3,4, 7, and 8 all the patterns were classified correctly. The rest of the classes have some patterns classified incorrectly, but e.g. for the 3 patters of class 1 classified into 5 and 6, we used the same pool of low-level heuristics and this does not represent an issue for the next step.

6.5 Design and testing the Hyper-heuristic offline learning with K −folds

In the next step, the statistical tests were applied to heuristics aforementioned and will form the characteristics of our instances Graph coloring and CVRP per class. In this phase, we choose according to the rankings the heuristics which have $x < (minrank + maxrank)/2$. If for example, the Aligned Friedman has a min rank= 776.5 and max rank= 3604.5 the limit value for considering a heuristic must be 2190.5. Because heuristic 11 has the worst

performance for both data sets, we did not consider this heuristic for this experimentation phase.

Table 12: Confusion Matrix, TP Rate (TPR) and FP Rate (FPR) and Precision (P) for each class

Class	1	2	3	4	5	6	7	8	TPR	FPR	P
1	13	0	0	0	2	1	0	0	0.813	0	1
2	0	7	0	0	0	0	0	2	0.778	0	1
3	0	0	10	0	0	0	0	0	1	0	1
4	0	0	0	4	0	0	0	0	1	0	1
5	0	0	0	0	12	5	0	0	0.706	0.019	0.857
6	0	0	0	0	0	27	1	0	0.964	0.062	0.818
7	0	0	0	0	0	0	10	0	1	0.009	0.909
8	0	0	0	0	0	0	0	31	1	0.021	0.939

Table 13 shows the ranks and the heuristic fitness for each class. E.g. the class 1 for QT and AFT has the same heuristics H5, H6, H4, H1 and H2, and H3, while FT does not consider H3. We only consider H5, H6, H4, H1 and H2, and H3 as a minimum set.

Table 13: heuristics ranks per Classes, Friedman (Fr), Aligned Friedman (A-Fr) and Quade (Qu)

	Class 1						Class 2					
*Fr	H3	H8	H5	H1	H4	H2	H3	H8	H7	H5	H2	H10
	1.50	1.69	3.78	4.03	4.08	6.69	2.8	3	3.4	4	5.7	6.25
*A-Fr	H3	H8	H1	H5	H4	H2	H3	H8	H7	H5	H2	H4
	627	638	1089	1091	1102	1760.5	293	320	335	428	508	540
*Qu	H3	H8	H5	H4	H1	H2	H3	H8	H7	H5	H2	H10
	1.55	1.59	3.77	4.04	4.15	6.81	2.24	2.73	2.98	4.18	5.51	6.39
	Class 3						Class 4					
*Fr	H5	H6	H2	H4	H1	H3	H5	H2	H1	H4	H6	H3
	1.64	2.18	4.27	4.27	4.82	5.18	2.4	2.6	2.8	3.6	3.6	6
*A-Fr	H5	H6	H4	H2	H1	H3	H2	H1	H5	H4	H6	H3
	242	283	403	418	419	568	33	74	78	79	91	128
*Qu	H5	H6	H2	H4	H1	H3	H2	H1	H5	H4	H6	H3
	1.71	2.26	4.20	4.52	4.88	5.38	2.07	2.67	2.67	3.67	3.93	6
	Class 5						Class 6					
*Fr	H8	H3	H5	H1	H4	H2	H3	H8	H5	H4	H1	H10
	1.45	1.55	3.53	4.37	4.68	6.03	1.32	1.69	4.35	4.92	5.05	5.92
*A-Fr	H8	H3	H5	H1	H4	H2	H3	H8	H5	H4	H1	H2
	646	651	1144.5	1197	1225.5	2010.5	1306	1345	4112	4211	4227.50	5058
*Qu	H8	H3	H5	H1	H4	H2	H3	H8	H5	H4	H1	H2
	1.47	1.53	3.67	4.44	4.69	5.78	1.48	1.53	3.90	4.55	4.60	6.47
	Class 7						Class 8					
*Fr	H3	H8	H5	H4	H1	H9	H5	H6	H2	H4	H1	H3
	1.32	2.45	3.45	5.32	5.86	5.91	1.47	2.38	3.44	4.21	4.59	5.46
*A-Fr	H3	H8	H5	H4	H1	H9	H5	H6	H2	H4	H1	H3
	108.50	171	477.50	578.50	637.50	642	2328	2691	2989	3847	4070	6169.50
*Qu	H3	H8	H5	H4	H9	H1	H5	H6	H2	H4	H1	H3
	1.12	2.17	3.65	5.09	5.74	5.82	1.50	2.28	3.27	4.46	4.70	5.43

The selected heuristics for each class can be summarized into five groups:

H3, H8, H5, H4 and H1.

H3, H8, H7, H5 and H2.

H5, H6, H2, H4, H1 and H3.

H2, H1, H5, H4 and H6.
H3, H8, H5 and H4.

Table 14: Hyper-heuristics results for Graph Coloring and CVRP. The best results are highlighted in bold

#	C	Median	DE	Average	#	C	Median	DE	Average	#	C	Median	DE	Average
4	1	3	2	4	30	5	10	4	12	7	7	22	284	162
1	1	4	39	30	34	5	10	5	12	19	7	160	212	490
3	1	4	12	9	49	5	8	10	15	37	7	326	185	623
4	1	3	5	5	50	5	20	11	26	38	7	321	183	620
5	1	3	10	7	51	5	32	14	40	43	7	407	160	674
14	1	18	28	40	65	5	7	73	72	44	7	402	168	672
53	1	3	8	6	69	5	22	25	36	45	7	294	398	1147
54	1	3	5	5	70	5	37	33	56	46	7	293	390	1156
55	1	3	6	5	71	5	32	42	55	83	7	340	171	658
61	1	3	0	3	72	5	52	55	79	84	7	320	413	548
62	1	4	2	5	73	5	35	58	64	123	8	1821	261	1524
63	1	5	6	7	75	5	60	80	101	124	8	1147	517	1598
64	1	6	24	19	80	5	15	15	25	125	8	1366	468	2082
76	1	4	7	8	81	5	18	21	29	126	8	1410	767	2052
77	1	8	8	12	16	6	60	33	99	127	8	1582	437	2145
78	1	16	11	23	18	6	127	82	255	128	8	1510	455	1975
8	2	573	365	1176	20	6	247	285	715	129	8	1433	583	2271
9	2	1011	510	2190	21	6	285	206	628	130	8	1420	747	2120
11	2	541	535	1435	22	6	21	59	96	88	8	909	291	1339
13	2	771	478	1542	24	6	181	97	318	97	8	2469	1897	4622
23	2	153	441	962	25	6	75	43	133	98	8	1707	342	2174
117	2	2179	776	3343	26	6	42	77	98	99	8	1116	426	1605
121	2	2858	1520	4528	27	6	40	80	100	100	8	1548	418	2051
67	2	318	330	897	29	6	14	35	35	101	8	1117	420	1564
68	2	753	689	2522	31	6	68	86	131	90	8	1441	787	2105
103	3	9429	4587	13967	32	6	45	130	142	91	8	2049	1313	3099
104	3	13211	5617	19240	33	6	46	136	145	92	8	2619	1842	4702
106	3	11376	4025	14762	35	6	85	104	239	93	8	815	342	1165
107	3	12566	6118	17696	36	6	85	108	238	94	8	1188	572	1800
108	3	13221	4905	23107	39	6	37	68	126	95	8	1436	727	2024
109	3	14390	7055	21411	40	6	35	69	124	102	8	6023	3021	8874
131	3	50634	9126	68656	42	6	127	137	332	111	8	1056	1172	2201
132	3	30323	17173	47025	47	6	43	15	53	112	8	1340	1671	2996
133	3	22175	5420	32883	48	6	74	15	86	113	8	1744	2287	4038
134	3	28769	21922	45678	56	6	50	17	62	114	8	1138	422	1658
135	4	65098	59475	160617	57	6	40	28	59	115	8	1447	522	2154
136	4	67184	29544	127827	58	6	37	26	57	116	8	1811	574	2737
137	4	97074	34915	149750	60	6	40	27	58	118	8	1300	462	1810
139	4	125875	27756	192093	66	6	131	193	417	119	8	1623	837	2587
10	5	10	6	15	85	6	53	17	65	120	8	2370	1032	3519
15	5	45	48	95	86	6	34	28	55	110	8	823	803	1603
28	5	8	14	14	87	6	33	26	53					

Therefore, we design 5 algorithms for these 8 classes. This means the hyper-heuristic with the high-level search strategy was the same, but the heuristics pool set was different according to these 5 subset heuristics. We trained and tested the hyper-heuristic with the short length pools. We left 10% of instances as unseen for the hyper-heuristics and the results are shown on table 14.

The hyper-heuristic configuration was 10 iterations for local search and 100,000 functions calls. For some GCP instances, we got the optimum number of colors (denoted in bold on table 14). Besides, for the instances A-n45-k6, A-n55-k9, A-n62-k8, A-n64-k9, CMT13, GWKC1, GWKC2, GWKC3, and X-n148-k46 we get values near to the optimal with a maximum 20% of the distance.

6.6 Classification the test instances and Application the Hyper-heuristic to the corresponding Instance

Finally, for the 14 unseen instances, we used the Naive Bayes classifier and it determines the class for these instances. Later we applied the Hyper-heuristic with the corresponding pool heuristics according to the previous design and we obtained the results showed in tables 15 and 16.

Table 15: Confusion Matrix, TP Rate (TPR) and FP Rate (FPR) and Precision (P) for each test instances

class	1	2	3	4	5	6	7	8	TPR	FPR	P
1	2	0	0	0	0	0	0	0	1	0	1
2	0	1	0	0	0	0	0	0	1	0	1
3	0	0	1	0	0	0	0	0	1	0	1
4	0	0	0	1	0	0	0	0	1	0	1
5	0	0	0	0	0	2	0	0	0	0	-
6	0	0	0	0	0	3	0	0	1	0.182	0.6
7	0	0	0	0	0	0	1	0	1	0	1
8	0	0	0	0	0	0	0	3	1	0	1

Instance	Class	Fitness	Instance	Class	Fitness
anna	1	3	DSJC250.5	6	27
mugg100_1	1	3	le450_25c	6	26
DSJC1000.5	2	83	mulsol.i.4	6	30
GWKC4	3	14255.68	school1	7	13
X-n411-k19	4	20005.37	A-n32-k5	8	876.19
queen15_15	5	29	CMT2	8	911.37
queen8_12	5	21	CMT9	8	1258.57

Table 16: Fitness results of test instances (Hyper-heuristic)

Table 15 shows the confusion matrix, TP rate, FP Rate, and Precision of the classification test. In these results, two patterns which belong to class five were classified incorrectly, but this does not affect the Hyper-heuristic solution because this class shares the same heuristics with class 6.

Finally, we applied a Wilcoxon rank-sum test for the training data-set and T student test for test data-set and we verified the statistical significance with best know results of the state of the art. We established $\alpha = 0.05$ and $h_0 =$ there are no differences between the performance of the heuristics and as h_a there are differences between the performance of the heuristics.

The Wilcoxon rank-sum statistic is 9418.0 with a $p - value = 0.05$, with these values we can conclude that our results are near to the optimal best-known solutions. It is important to remember that our methodology is general, as we use two different problem domains. Our aim is not to get in all cases the optimal solution, our objective is to improve the selection of the heuristic pool for hyper-heuristic with offline learning.

For the test data set, the T statistic is 95.0 with a $p -$value $= 0.96$, this indicates that our results are like the optimal solutions.

7 Conclusion

In this paper, we have proposed a methodology with meta-learning and non-parametric statistics tests for selecting the best pool heuristic for Hyper-heuristic. An important task for the researchers and decision-makers is to know the best solver for a given problem instance since each problem has a different difficulty and complexity. The first step in our methodology was to divide the instances and select the best heuristics for the hyper-heuristic according to these classes or subsets. Then the training set (GCP and CVRP) was tested on the hyper-heuristic. With the last step, we can reduce our pool of heuristics according to 5 different cases, and design a better algorithm according to the characteristics and previous results of the heuristics.

For the best pool heuristics and hyper-heuristic prediction to a given instance, we used the Naive Bayes classifier using the test dataset. This process involves the previous phase of generating features (basic+inter) and finally, the Naive Bayes predict the class and the corresponding pool heuristics for each instance.

Our experimental results show that the hyper-heuristic with each different pool can effectively resolve each instance, and the classifier can predict the class for each problem type. We consider to design a methodology which can be applied in different problem domains, and avoid to design heuristics ad-hoc for each problem.

The combination of different concept areas of computer science as classifiers, meta-learning, and optimization, it can be a powerful tool for solving complex problems.

As future directions, we propose to use this approach for selecting the pool of heuristics to other Hype-heuristics approaches. As the second step, deeper research should be done to select or improve the characterization of each instance. Furthermore, this methodology can be applied to different problem domains as Timetabling problems, VRP, Bin Packing, so on. Finally, we consider it relevant to improve our approach using other classifiers and apply it to real instances.

Acknowledgments

Authors thank to Tecnológico Nacional de México/I. T. León and Universidad de Guanajuato. This work was supported by the National Council of Science and Technology of Mexico (CONACYT) via the Scholarship for Posgraduate Study 446106 (L. Ortiz) and Research Grant: CÁTEDRAS-2598 (A. Rojas)

References

[1] Alba, E. and Dorronsoro, B.: Computing nine new best-so-far solutions for capacitated vrp with a cellular genetic algorithm. Information Processing Letters 98(6), 225–230 (2006)

[2] Amaya, I., Ortiz-Bayliss, J.C., Conant-Pablos et al., H.: Hyper-heuristics reversed: Learning to combine solvers by evolving instances. In: 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE (2019). DOI 10.1109/cec.2019.8789928

[3] Augerat, P., Belenguer, J.M., Benavent, E. et al., : Computational results with a branch and cut code for the capacitated vehicle routing problem. IMAG (1995)

[4] Martin, C., Héctor, P., Hugo, T. M. et al. Methodology of design: A novel generic approach applied to the course timetabling problem. In Soft Computing Applications in Optimization, Control, and Recognition (pp. 287-319). Springer, Berlin, Heidelberg. 2013

[5] A., S.A.J., Martin, C., Hector, P. et al.: Comparison of Metaheuristic Algorithms with a Methodology of Design for the Evaluation of Hard Constraints over the Course Timetabling Problem, Studies in Computational Intelligence, vol. 451. Springer Berlin Heidelberg (2013)

[6] Baker, B.S.: A new proof for the first-fit decreasing bin-packing algorithm. Journal of Algorithms 6(1), 49–70 (1985).

[7] Blazinskas, A. and Misevicius, A.: combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem. In: 17th International Conference on Information and Software Technologies, pp. 27–29 (2011)

[8] Brazdil, P., Giraud-Carrier, C., Soares, C. et al.: Metalearning. Springer Berlin Heidelberg (2009). DOI 10.1007/978-3-540-73263-1

[9] Burke, E.K., Hyde, M.R., Kendall, G., et al.: A classification of hyper-heuristic approaches: Revisited. In: Handbook of Metaheuristics, pp. 453–477. Springer International Publishing (2018). DOI 10.1007/978-3-319-91086-4_14

[10] Carpio, M.: Modelo integral de asignación optima de carga academica usando un algoritmo heurístico. Encuentro de investigación en Ingenieria electrica (2006)

[11] Christofides, N.: The vehicle routing problem. n. christofides, a. mingozzi, p. toth, c. sandi, eds. combined optimization (1979)

[12] Csirik, J.: The parametric behavior of the first-fit decreasing bin packing algorithm. Journal of Algorithms 15(1), 1–28 (1993). DOI 10.1006/jagm.1993.1028

[13] Dantzig, G.B. and Ramser, J.H.: The truck dispatching problem. Management science 6(1), 80–91 (1959)

[14] Derbel, H., Jarboui, B. and Bhiri, R.: A skewed general variable neighbourhood search algorithm with fixed threshold for the heterogeneous fleet vehicle routing problem. Annals of Operations Research 272(1-2), 243–272 (2019)

[15] El-Ghazali, T.: Metaheuristics: From Design to Implementation. Wiley Publishing (2009)

[16] Golden, B.L., Wasil, E.A., Kelly, J.P. et al.: The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results. In: Fleet management and logistics, pp. 33–56. Springer (1998)

[17] Gutierrez-Rodríguez, A.E., Conant-Pablos, S.E., Ortiz-Bayliss, J.C. et al.: Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning. Expert Systems with Applications 118, 470–481 (2019)

[18] Habashi, S.S., Salama, C., Yousef, A.H. et al.: Adaptive diversifying hyper-heuristic based approach for timetabling problems. In: 2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 259–266. IEEE (2018)

[19] Helena, L., Olivier, M. and Thomas, S.: Iterated local search. In: F. Glover, G. Kochenberger, F.S. Hillier (eds.) Handbook of Metaheuristics, International Series in Operations Research & Management Science, vol. 57, pp. 320–353. Springer New York (2003)

[20] Helsgaun, K.: General k-opt submoves for the lin–kernighan tsp heuristic. Mathematical Programming Computation 1(2-3), 119–163 (2009)

[21] Kanda, J., de Carvalho, A., Hruschka, E. et al.: Meta-learning to select the best meta-heuristic for the traveling salesman problem: A comparison of meta-features. Neurocomputing 205, 393–406 (2016). DOI 10.1016/j.neucom.2016.04.027

[22] Karp, R.: Complexity of Computer Computations. Springer Us, Boston, MA (1972)

[23] Leighton, F.T.: A graph coloring algorithm for large scheduling problems. Journal of research of the national bureau of standards 84(6), 489–506 (1979)

[24] Lewis, D.D.: Naive (bayes) at forty: The independence assumption in information retrieval. In: Machine Learning: ECML-98, pp. 4–15. Springer Berlin Heidelberg (1998). DOI 10.1007/bfb0026666

[25] Lin, S. and Kernighan, B.W.: An effective heuristic algorithm for the traveling-salesman problem. Operations Research 21(2), 498–516 (1973). DOI 10.1287/opre.21.2.498

[26] Minton, S., Johnston, M.D., Philips, A.B. et al.: Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. Artificial Intelligence 58(1-3), 161–205 (1992). DOI 10.1016/0004-3702(92)90007-k

[27] Ortiz-Aguilar, L.d.M., Carpio, M., Puga, H. et al.: Increase methodology of design of course timetabling problem for students, classrooms, and teachers. In: Nature-Inspired Design of Hybrid Intelligent Systems, pp. 713–728. Springer (2017)

[28] Ortiz Aguilar, L.d.M.: Diseño de horarios de alumnos y maestros mediante técnicas de soft computing, para una institución educativa. Master's thesis, Instituto Tecnológico de León (2016)

[29] Pillay, N.: A review of hyper-heuristic for educational timetabling. Springer (2014)

[30] P., G.I., Ewan, M., Patrick, P. et al.: An empirical study of dynamic variable ordering heuristics for the CSP. In: E. Freuder (ed.) Principles and Practice of Constraint Programming CP96, Lecture Notes in Computer Science, vol. 1118, pp. 179–193. Springer Berlin Heidelberg (1996)

[31] Rice, J.R.: The algorithm selection problem. In: Advances in Computers, pp. 65–118. Elsevier (1976). DOI 10.1016/s0065-2458(08)60520-3

[32] Salwani, A., K., B.E. and Barry, M.: Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem. In: Metaheuristics, Operations Research Computer Science Interfaces Series, vol. 39, pp. 153–169. Springer US (2007)

[33] Scott, D.W.: Sturges' rule. Wiley Interdisciplinary Reviews: Computational Statistics 1(3), 303–306 (2009)

[34] Solomon, M.M.: On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints. Networks 16(2), 161–174 (1986)

[35] Song, H., Triguero, I. and Özcan, E.: A review on the self and dual interactions between machine learning and optimisation. Progress in Artificial Intelligence 8(2), 143–165 (2019). DOI 10.1007/s13748-019-00185-z

[36] Soria-Alcaraz, J.A., Ochoa, G., Carpio, M. et al.: Effective learning hyper-heuristics for the course timetabling problem. European Journal of Operational Research p. 10 (2014)

[37] Soria-Alcaraz, J.A., Ochoa, G., Sotelo-Figeroa, M.A. et al.: A methodology for determining an effective subset of heuristics in selection hyper-heuristics. European Journal of Operational Research 260(3), 972–983 (2017).

[38] Soria-Alcaraz, J.A., Özcan, E., Swan, J. et al.: Iterated local search using an add and delete hyper-heuristic for university course timetabling. Applied Soft Computing 40, 581–593 (2016)

[39] Uchoa, E., Pecin, D., Pessoa, A. et al.: New benchmark instances for the capacitated vehicle routing problem. European Journal of Operational Research 257(3), 845 – 858 (2017).

[40] de Werra, D.: Heuristics for graph coloring. In: Computational graph theory, pp. 191–208. Springer (1990)

[41] Yates, W.B. and Keedwell, E.C.: Offline learning for selection hyper-heuristics with elman networks. In: Lecture Notes in Computer Science, pp. 217–230. Springer International Publishing (2018).

[42] Yates, W.B. and Keedwell, E.C.: An analysis of heuristic subsequences for offline hyper-heuristic learning. Journal of Heuristics 25(3), 399–430 (2019). DOI 10.1007/s10732-018-09404-7

Increase Methodology of Design of Course Timetabling Problem for Students, Classrooms, and Teachers

Lucero de M. Ortiz-Aguilar, Martín Carpio, Héctor Puga, Jorge A. Soria-Alcaraz, Manuel Ornelas-Rodríguez and Carlos Lino

Abstract The aim of the Course Timetabling problem is to ensure that all the students take their required classes and adhere to resources that are available in the school. The set of constraints those must be considered in the design of timetabling involves students, teachers, and classrooms. In the state of the art are different methodologies of design for Course Timetabling problem, in this paper we extend the proposal from Soria in 2013, in which they consider variables of students and classrooms, with four set of generic structures. This paper uses Soria's methodology to adding two more generic structures considering teacher restriction. We show an application of some different Metaheuristics using this methodology. Finally, we apply nonparametric test Wilcoxon signed-rank with the aim to find which metaheuristic algorithm shows a better performance in terms of quality.

Keywords Course timetabling · Faculty timetabling · Cellular genetic algorithm · Metaheuristics · Iterated local search

L. de M. Ortiz-Aguilar · M. Carpio (✉) · H. Puga · M. Ornelas-Rodríguez · C. Lino
División de Estudios de Posgrado E Investigación, TNM-Instituto Tecnológico de León, León, Mexico
e-mail: juanmartin.carpio@itleon.edu.mx

L. de M. Ortiz-Aguilar
e-mail: Ldm_oa@hotmail.com

H. Puga
e-mail: pugahector@yahoo.com

M. Ornelas-Rodríguez
e-mail: mornelas67@yahoo.com.mx

C. Lino
e-mail: carloslino@itleon.edu.mx

J.A. Soria-Alcaraz
Universidad de Guanajuato, Guanajuato, Mexico
e-mail: jorge.soria@ugtomx.onmicrosoft.com

P. Melin et al. (eds.), *Nature-Inspired Design of Hybrid Intelligent Systems*,
Studies in Computational Intelligence 667, DOI 10.1007/978-3-319-47054-2_47

1 Introduction

The timetabling problem is present in different organizations such as schools, universities, hospitals, transport, etc. Universities need timetabling that included students, teachers, and others [1]. Therefore, having a good design of timetable help us optimize resources. And provide to the students necessary tools to finish on time his carrier. Therefore, this type of problem has been considered in the state of the art as NP-complete.

The design of Timetable depends directly over the specific school, university, or curricula; hence there is not universal timetable that can be applied in any case. The set of constraints that must be considered in the design of timetable involves students, teachers, and infrastructure. A key for the timetabling are the test instance, i.e., information regarding about supply and demand of resources, students, and teachers. This work generated instances that were created by artificial simulation techniques.

This work focuses on generating acceptable solution to the problem of Course timetabling, faculty timetabling and classroom assignment, using Metaheuristics. There are a diverse number of approaches that have been used to solve the problem of Course Timetabling as graph coloring [2], IP/LP (Integer programming/Linear programming) [3], genetic algorithms [4–6], Memetic Algorithms [7, 8], Tabu search [9, 10], simulated annealing [11] local search [12] Best-Worst Ant System (BUS), and Best-Worst Ant Colony System [13], in recent years Hyperheuristic approaches [14] and CSP [15] have been raised as generics and good alternatives when solving this problem.

The course timetabling problem has been classified as NP-complete [16]. In state or the art exist a lot of different problems at least class NP, which can be solved with different Metaheuristics, but as we indicated Theorem No Free Lunch [17] there is no Metaheuristics that outperforms all others for all known problems of class NP. Accordingly in this paper we show comparison between algorithm Cellular and Iterated Local Search, using nonparametric statistical tests.

2 Concepts

In this Section, we focus to concepts about University Timetabling, Methodology of design, and Metaheuristics.

2.1 University Timetabling

The university Timetabling is present in the most of universities and schools. This problem was described by Adriaen et al. [18]:

- **Faculty Timetabling**: assign each teacher to courses.
- Class teacher Timetabling: assign courses with minimum conflicts between groups of students.
- **Course timetabling**: assign courses with minimum conflicts between individual students.
- Examination timetabling: assign examinations with minimum conflicts between students.
- **Classroom assignment**: after assigning classes to teachers, assign this class teacher to each classroom.

University timetabling problem can be described as a set events e, a set s time slots, and set c restrictions between events, where we have to set each event e in one timeslot s [19]. We focused into generate solutions of the mixed problem of: faculty, Course, and Classroom assignment timetabling.

2.1.1 Faculty Timetabling

In [4] consider that faculty timetabling extends from basic model Class Teacher timetabling and defined as follows: given m teachers $t_1, t_2,\ldots,It_m$, n class $c_1, c_2,\ldots,c_n$ and the set $\{1, 2, \ldots, t\}$ of periods of time, so each tuple (t, c) must be set in only one t period of time.

This case each teacher must be assigned in only one class in a specific period of time. For example of hard restriction is: one teacher cannot be scheduled in two classes at the same time; and example of soft restriction is: one teacher wants 2 or more consecutive classes.

2.1.2 Course Timetabling

The course timetabling problem can be described as a process of assign classes to resources (timeslots, classrooms, and teachers) while satisfying a set of restrictions [20]. The CTTP was defined by Conant-Pablos [21]: a set of events $E = \{e_1, e_2,\ldots,e_n\}$, a set of periods of time $T = \{t_1, t_2,\ldots,t_s\}$, a set of classrooms $P = \{p_1, p_2,\ldots,p_m\}$ and a set of students $A = \{a_1, a_2,\ldots,a_o\}$, so a quadruple (e, t, p, S) is a complete solution which satisfies the set of hard constraints defined by each university.

2.1.3 Classroom Assignment

From the point of the view of University, classroom assignment problem is the most restrictive due to the fact limited infrastructure, so the university cannot enroll more students than its capacity. Moreover in some cases, depending on the methodology of design from the university, the classroom assignment is related to the teachers, students, or subjects.

After forming the timetabling design with minimum conflicts, we should assign the subject to one classroom [22]. Each classroom has capacity, available schedule, etc.

2.2 *Methodology of Design*

The methodology of design proposed by Soria in [23] let us manage different constraints and restrictions of Course timetabling problem through conversion of all restrictions of time and space in a simple kind restriction: student conflicts. This methodology proposed 4 structures: MMA matrix, LPH list, LPH list, and LPS list. The first 3 structures represent the hard restrictions and the last represents the soft restrictions.

In our work we used three structures: MMA, LPH, and LPA. In [23] describe the structures as follows:

MMA matrix contains the number of students in conflict between classes, i.e., the number of conflicts if two classes are assigned in the same timeslots. Also this matrix shows the number of students that demand one class, this information is on the diagonal of the matrix. The Fig. 1 shows an example, for instance we have the subject GEE0925 which 172 students demand it, and it has 91 conflicts with GEF0914. The algorithm 1 shows the procedure to generate this matrix, taken from [24].

Fig. 1 MMA matrix

CLAVE	ACF0901	ACC0906	GEC0906	AEF1074	GEC0913	GEF0914	PDH	AE	ACF0902	AEB1082	GED0904
ACF0901	160	99	91	97	137	109	89	136		66	64
ACC0906		101	88	88	80	96	88	78	2	9	9
GEC0906			91	90	70	88	88	69		2	2
AEF1074				97	76	93	89	75		7	6
GEE0925					172	91	68	168	35	97	96
GEF0914						115	88	93	4	19	19
PDH							1	244	4	1	
AE								1	45	99	94
ACF0902									58	33	32
AEB1082										103	95
GED0904											96

Algorithm 1 MMA Construction

```
Require:int N Students, int[][] LD Students Demands
1: for i = 0 to N do
2:    Starr = LD[i]
3: for j = 0 to size(Starr) do
4:       for k = j + 1 to size(Starr) do
5:             MMA[Starr[j]][Starr[k]]+ = 1
6:             MMA[Starr[k]][Starr[j]]+ = 1
7:    end for
8:    end for
9: end for
10: return MMA
```

LPH this list contains information about each class and feasible time domain when can be assigned. The Fig. 2 shows an example, for instance we have the subject M4 has only 1 possible time slot where can be assigned. The algorithm 2 shows the procedure to generate this list, taken from [24].

Algorithm 2 LPH Construction

```
Require:int Nm CMaxEvent, int[] SI Initial solution
1: for i = 0 to size(SI) do
2:    intCrandom= random(Nm)
3: int [] lpht = nuevo
4:    for j = 0 to Crandom do
5:lpht[j]= randomEntre(Nm)
6:end for
7:lpht[Nm]= SI[i]
8:LPH[i]:add(lpht)
9: end for
10: return LPH
```

LPA this list contains information about feasible space domain from each class and where (classroom) can be assigned. The Fig. 3 shows an example, for instance

Fig. 2 LPH List

Event	Timeslots							
M_1	1	2	3	4	5	6	7	8
M_2	1	2	3	4	5		7	8
M_3	1	2	3	4	5	6	7	8
M_4	0							
⋮								
M_{n-1}	1	2	3	4	5	6	7	8
M_n	1	2	3	4	5		7	8

Fig. 3 LPA List

Events	*Classrooms*
M_1	$< A_1, A_2, A_3 >$
M_2	$< Lab_1 >$
M_3	$< A_1, A_2, A_3, A_4, A_5, A_6 >$
M_4	$< A_1, A_2, A_3 >$
⋮	⋮
M_{n-1}	$< A_1, A_2, A_3 >$
M_n	$< A_1, A_2, A_3 >$

we have the subject M4 can be assigned in classroom 1, 2, and 3. The algorithm 3 shows the procedure to generate this list, taken from [24].

Algorithm 3 LPA Construction

Require: *int Nm Subjects, int[][] CA Room features, int[][] DA =Subjects Demands, int[] Rms Room List*

```
1: for i = 0 to Nm do
2: for j = 0 to size(CA) do
3: for k = 0 to size(CA[j]) do
4:    if DA[i] <= CA[j][k] then
5:          LPA[i]:add(Rms[k])
6:    end if
7:   end for
8:   end for
9: end for
10: return LPA
```

2.2.1 Extending Methodology of Design of Course Timetabling Problem Through Teacher Management

In this paper we used two auxiliary structures LMS (list Subject Semester) and LPT (List Possible of Teachers). In [1] describe the structures as follows:

LMS this list contains each subject with their corresponding semester. The number of semester is given by the curriculum. This structure represents a hard constraint that must be satisfied. Figure 4 shows an example, for instance we have the subject M1 to M3 are semester first.

LPT list contains information related to each subject in terms of its available teacher. This list is built with the preferences from each teacher about schedules and subjects. Each university could have different teacher with defined working hours, academic profile, and minimum hours per week. Once having this information, we can create the LPT to work with the methodology of design. LPT list reports in its rows the possible teachers available to impart the Subject. The Fig. 5 shows an example, in this example M4 can be assigned to timeslot 1 assigning teacher seven.

Fig. 4 LMS List

Subject	*Semester*
M_1	1
M_2	1
M_3	1
M_4	2
M_5	2
⋮	⋮
M_{n-1}	9
M_n	9

Fig. 5 LPT List

Subject	$Teacher - Timeslot$
M_1	$< P_1H_2, P_2H_3, P_3H_3 >$
M_2	$< P_2H_6 >$
M_3	$< P_2H_3, P_1H_3 >$
M_4	$< P_7H_1 >$
⋮	⋮
M_{n-1}	$< P_2H_3, P_1H_3, P_2H_4 >$
M_n	$< P_9H_3 >$

This list should contain at least one teacher for each subject, due to the fact if only one subject do not have it, the timetabling design cannot be formulated, and this means that university does not have enough resources to cover all classes.

2.3 Metaheuristics

In this paper we used two Metaheuristics: cellular algorithm and Iterated Local Search. This section shows our adaptations for course timetabling problems.

2.3.1 Parallel Cellular Genetic Algorithm

The parallel cellular algorithm usually creates a conceptual population where each processor contents some individuals. Its main feature is the population structure form of grid, where each individual only relates to its neighbors [25]. In evolutionary algorithms (EA) model most widespread since its origins is called panmictic, where the evolutionary process works in a single population [26] and among other models are evolutionary algorithms Cellular (Fig. 6, taken from [1]).

Fig. 6 Kinds of population

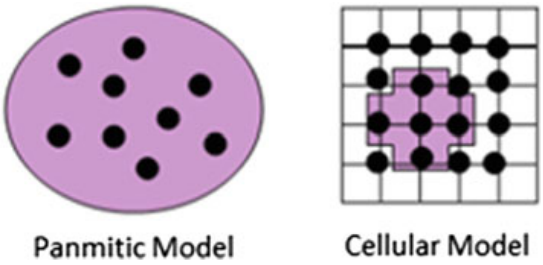
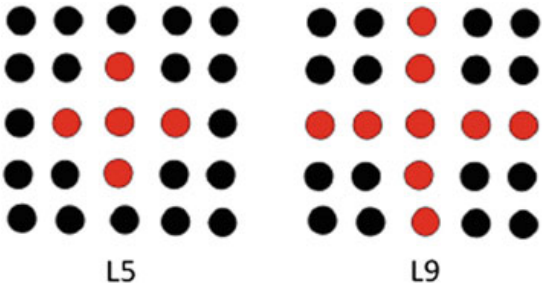


Fig. 7 Linear grid



The use of distributed parallel populations in cEAs is based on the idea that the isolation of populations allows maintain a higher genetic diversification. In some cases, these algorithms with decentralized population do a better sampling of the search space and improve behavior and the runtime as in algorithm panmixia [25].

The type of neighborhood used in this work was linear (LR), where neighbors include $r-1$ structures closest chosen on the horizontal and vertical axes (see Fig. 7, taken from [1]). The algorithm 4 shows the Parallel Algorithm, taken from [8, 27].

Algorithm 4 Pseudo-code of a canonical cGA

```
1: procEvolve(cga)
2: GenerateInitialPopulation(cga.pop)
3: Evaluation(cga.pop)
4: while !StopCondition() do
5: for individual ← 1 to cga.popsizedo
6: neighbors ← CalculateNeighborhood(cga,position(indicidual));
7:    parents ← Selection(neighbors);
8:    offspring ← Recombination(cga.Pc,parents);
9:     offspring ← Mutation(PM);
10:  Replacement(position(individual),auxiliaryPop,offspring);
11:  end for
12: cga.pop← auxiliaryPop;
13: end while
14: return Best(cga.pop)
```

2.3.2 Iterated Local Search

The core of the iterated local search metaheuristic focus on initial solution, then iteratively builds a sequence of solutions generated by embedded heuristic [28]. In [29] describe the algorithms 5 and 6. This algorithm has perturbation stage (see step 4 on algorithm 5) where it applies a simple random perturbation to the solution. This simple random Perturbation operator uniformly selects at random value to replace it with another feasible value.

Algorithm 5 High Level Iterated Local Search (ILS)

```
1: s0 = GenerateInitialSolution
2: s* = ImprovementStage(s0)
3: while !StopCondition() do
4:     s' = SimpleRandomPerturbation(s*)
5:     s*' = ImprovementStage(s')
6:     if f(s*') < f(s*) then
7:         s* = s*'
8:end if
9:end while
10:return s*
```

Algorithm 6 Improvement Stage

```
1: ls ← IncumbentSolution
2: while!LocalStopCriteria()do
3:   hi = Perturbate()
4:   ls* = apply(hi, ls)
5:   iff(ls*) < f(ls)then
6:     ls = ls*'
7:   end if
8: end while
9: returnls
```

3 Metaheuristics Adapted to the Methodology of Design

3.1 Matrix Construction

We use the methodology of design proposed by Soria in [23] and we extend this for teachers. Now we will describe the structure of solutions as follows:

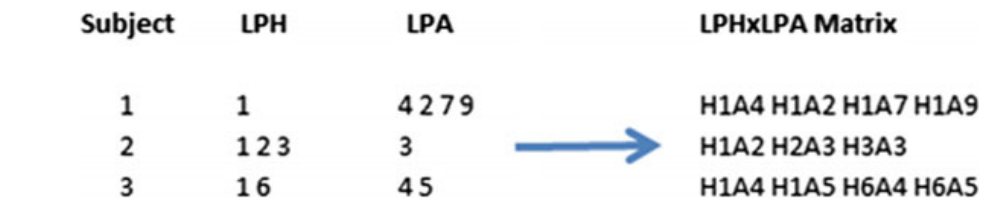


Fig. 8 Representation LPH × LPA matrix construction

In this work we use artificial instances and the list LPH, list LPA, list LPT, and matrix MMA for the solution construction. However to generate inputs for meta-heuristic, in this case we construct a matrix LPT × LPH × LPA as follows: first have LPH × LPA matrix and then make a Cartesian product between the matrix LPH × LPA and the LPT List.

The Fig. 8 shows an example of LPH × LPA matrix construction. For instance we have 3 subjects and their LPH and LPA list for each one; the subject 1 can be assigned in timeslot 1 and classroom 4, 2, 7 and 9, whereas this information the feasible list for subject 1 is: timeslot 1 and classroom 4, timeslot 1 classroom 2, timeslot 1 classroom 7 and timeslot 1 classroom 9.

Now we do Cartesian product between LPT and LPH × LPA matrix. Figure 9 shows an example of LPT × LPH × LPA construction.

For example subject 2 has teacher 2 and timeslot 4 and 1, matrix LPH × LPA has timeslot 1 classroom 2, timeslot 2 classroom 3 and timeslot 2 and classroom 3. When the Cartesian product is applied the feasible result is teacher 2 timeslot 1 and classroom 2. This matrix has inside all the hard constraints.

3.2 Solution Representation

In this case we had the LPT × LPH × LPA matrix, so for the solution we keep the column position from each subject (row). An illustrative example it shows in Fig. 10 take from [1]. In this example we can see each subject has at least one possible tuple, therefore the final solution is a feasible solution for the current instance. Each semester has each different color, and the highlighted boxes are the option that we choose. For example, the subject 1 has the teacher 10 timeslot 7 and classroom 14, but we keep only the position of this tuple. This let us use an integer codification on the Metaheuristics instead a string codification.

Fig. 9 Representation LPT × LPH × LPA matrix construction

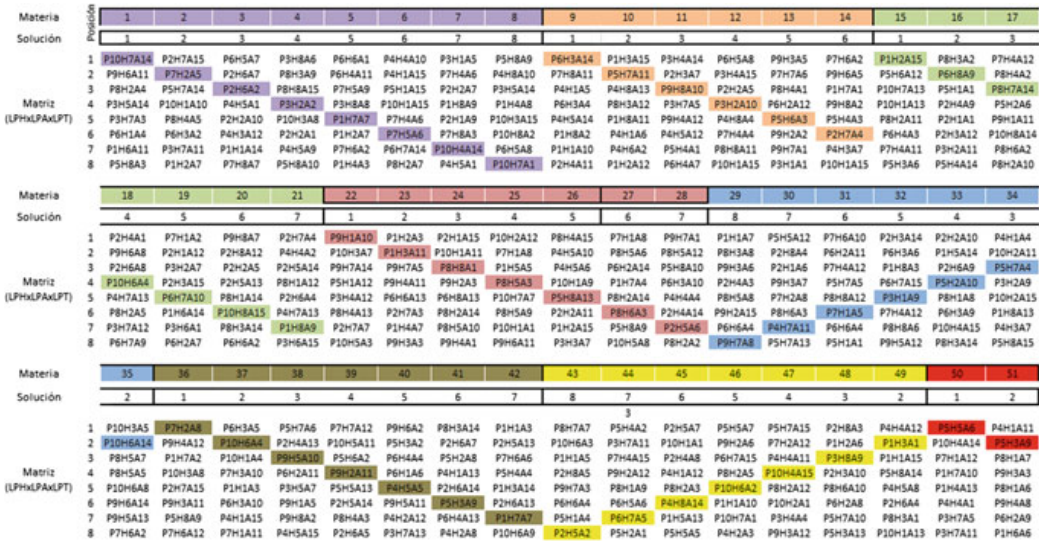


Fig. 10 Solution Representation

If not exist at least one possibility for the subject it means that not exist a solution for the timetable.

3.3 Fitness Function

In [1] defined fitness function as follows:Given a B matrix, $P = \{p_1, p_l,...,p_r\}$ teachers where $1 \leq l \leq r$, $A = \{a_1, a_m,...,a_s\}$ classrooms where $1 \leq m \leq s$, M a set of $\{m_1, m_h,...,m_n\}$ subjects where $1 \leq h \leq t$. Each element $p \varepsilon P$ is unique and requires on classroom $a \varepsilon A$ and one subject $m \varepsilon M$, so we have tuple denote as (m, p, a). Given $V = \{v_1, v_w,\ldots, v_{u-1}\}$ timeslot, where $0 \leq w \leq u\text{-}1$, each $v_v = \{s_1, s_d, ...,s_e\}$ and $s_d = (m_h, p_l, a_m)$, therefore our aim is to minimize the following function:

$$f(x) = \sum_{i=0}^{u-1} \sum_{j=0}^{e} \sum_{k=j+1}^{e-1} B_{[m_{jh}][m_{kh}]} + FP(j,\ k) + GA(j.k), \tag{1}$$

where $FP(j,k)$ is:

$$FP(j,k) = \begin{cases} B_{[m_{jh}][m_{kh}]} + B_{[m_{jh}][m_{kh}]} & \text{if } p_{jl} = p_{kl} \\ 0 & else \end{cases}, \tag{2}$$

where $GA(j,k)$ is:

$$GA(j,k) = \begin{cases} B_{[m_{jh}][m_{kh}]} + B_{[m_{jh}][m_{kh}]} & \text{if } a_{jl} = a_{kl} \\ 0 & else \end{cases} \quad (3)$$

$GA(j,k)$ will be zero in case that does not have conflicts between classrooms and in another case we add the number of students that demand the subject. And $FP(j,k)$ will be zero in case does not have conflicts between teachers and in another case we add the number of students that demand the subject. In this work we take Eq. 1 for measure fitness.

4 Experiments and Results

In this section, we show experiments and the performance of Metaheuristics applies to the problem of Course timetabling, faculty timetabling, and classroom assignment. We also describe the most important characteristics of our set instances.

4.1 Test Instances

We generated 35 artificial instances, the Table 1 shows the information about this instances.

The instances are divided in subsets, for example: sets 1–5 have 52 subjects, 8 time slots, 15 classrooms, and 11 teachers. Each instance has different constraints, therefore not exist equals instances. In general we have instances with 52–300 subjects, 8–32 timeslots, 15–60 classrooms, and 11–50 teachers.

4.2 Experimental Design and Results

The configuration initial for Metaheuristics is on Table 2. We use call functions as stop criteria for both algorithms.

Table 1 Artificial instances

Number	Subjects	Timeslots	Classrooms	Teachers
1–5	52	8	15	11
6–10	52	16	15	11
11–15	52	8	15	11
16–20	104	16	30	25
21–30	208	32	60	50
31–35	300	32	60	50

Table 2 Configuration for Cellular Algorithm and Iterate Local Search

	Cellular	ILS
Subpopulations	4	10
Elitism	1 per subpopulation	
Crossover	0.9	
Mutation	0.1	
Individuals	16	
Local Search Iterations	N/A	10
Stop Criteria (functions Call)	5,000,000	

Table 3 has the results in terms of conflicts obtained by both Metaheuristics. We calculate the median, best fitness, worst fitness, and standard deviation for 35 instances.

The first column is the instance number; followed by median, best fitness, worst fitness, and standard deviation from both algorithms. For instance the bold number means the best result between each median, e.g., in instance 1–5 ILS has 0 conflicts, and cellular algorithm has only 0 in instance 2 and 3. The best fitness columns show that cellular algorithm has 0 conflicts in instances 1–20, and ILS has some 0 in instances like 1–5, 8, 11, 13, and 18–20. Now the worst results columns, the Cellular Algorithm has the lowest results except in first two instances. Also we show the standard deviation, because we need an algorithm with low standard deviation, and again the Cellular algorithm has the lowest result except in instances 1, 2 and 3; but this is not a significant answer, so we applied a nonparametric test.

4.3 Wilcoxon Signed-Rank Test

Now we want to know which algorithm has the best performance in terms of conflicts. First we applied Wilcoxon signed-rank test between ILS and Cellular algorithm, which say if this algorithms come from the same distribution.

Applying the Wilcoxon signed-rank test for populations $n > 30$, under the assumption that the ILS and Cellular algorithm come from populations with equal median, and $\propto = 0.05$ and $\propto = 0.01$, and $h_0 =$ There is no difference between the performance of the algorithms and $h_a =$ There are differences between the performance of the algorithms, we got following results:

$$W_{-} = 615$$
$$W_{+} = 12$$
$$Z = -4.96$$
$$P_{value} = 3.472E - 07$$

Table 3 Results experiments

	Median		Best fitness		Worst Fitness		Std deviation	
	ILS	Cellular	ILS	Cellular	ILS	Cellular	ILS	Cellular
1	**0**	5	**0**	**0**	**0**	50	**0**	5.95
2	**0**	**0**	**0**	**0**	**2**	10	**0.5**	4.4
3	**0**	**0**	**0**	**0**	76	**15**	13	**7.55**
4	**0**	3	**0**	**0**	2	**1**	**0.4**	18.04
5	**0**	2	**0**	**0**	138	**7**	58.2	**10.13**
6	732	**5**	161	**0**	1791	**42**	430.3	**10.5**
7	925	**7**	320	**0**	2526	**45**	509.5	**27.54**
8	703	**0**	**0**	**0**	1837	**8**	495.7	**1.57**
9	504	**0**	103	**0**	3904	**29**	1021.4	**3.88**
10	439	**0**	36	**0**	1993	**30**	555.4	**20.63**
11	341	**0**	**0**	**0**	1565	**21**	303.7	**6.96**
12	943	**3**	135	**0**	2030	**34**	449.8	**16.96**
13	397	**5**	**0**	**0**	1566	**15**	398	**3.08**
14	350	**16**	52	**0**	1925	**10**	519.1	**9.72**
15	1237	**56**	434	**0**	2627	**1**	506.6	**7.78**
16	9609	**12**	2357	**0**	25,846	**39**	7708.6	**12.3**
17	6497	**35**	2	**0**	20,395	**8**	5066.7	**17.14**
18	5018	**12**	**0**	**0**	21,073	**7**	5578.3	**17.17**
19	3414	**35**	**0**	**0**	20,775	**20**	4650.2	**48.5**
20	1872	**21**	**0**	**0**	11,761	**26**	2856.6	**11.83**
21	137,650	**234**	10,3701	**144**	178,027	**847**	16,513	**32.66**
22	140,174	**357**	99,970	**238**	176,874	**404**	21,146.2	**32.96**
23	165,490	**493**	129,316	**454**	196,366	**761**	15,712.9	**68.78**
24	138,172	**390**	108,764	**116**	176,308	**136**	14,397.5	**71.06**
25	157,661	**282**	124,817	**244**	178,698	**252**	13,769.1	**16.23**
26	149,412	**289**	127,638	**60**	174,593	**184**	11,879.5	**28.6**
27	130,602	**423**	105,590	**180**	162,570	**576**	13,296.5	**59.1**
28	117,600	**218**	87,222	**210**	154,111	**606**	16,285.5	**57.23**
29	161,040	**566**	120,276	**383**	202,775	**976**	14,930.2	**86.71**
30	142,257	**598**	114,886	**228**	193,008	**594**	19,459.2	**52.19**
31	539,375	**404**	451,269	**262**	661,147	**780**	45,605.1	**7.33**
32	562,611	**495**	435,852	**402**	656,118	**314**	50,244	**6.11**
33	611,247	**393**	471,273	**353**	695,651	**722**	58,068.4	**55.73**
34	544,623	**495**	453,478	**92**	622,181	**476**	45,566.5	**69.9**
35	551,297	**504**	477,894	**414**	665,014	**987**	44,513.2	**54.4**

So, we have that in case $\propto = 0.05{:}0.05 < 3.47\text{E-}07$, we have enough evidence to reject h_0 and with $\propto = 0.01{:}\ 0.01 < 3.47\text{E-}07$, also we have enough evidence to reject h_0. In other words our two algorithms had different median and the algorithm with best performance is Cellular algorithm.

4.4 Conclusions and Future Work

This paper presents an extension of the methodology of design approach for course timetabling, faculty timetabling, and classroom assignment. We used the Methodology of design proposed by Soria in [23] and extend it adding teacher-related restrictions. This improvement permits to use this approach when solving different universities.

Also we present a comparison between ILS and Cellular algorithm Metaheuristics in the context of the course timetabling, faculty timetabling, and classroom assignment problem. For this experimentation we generate artificial instances to test the methodology proposed, due to the fact in state of the art in which we searched, we did not found real or benchmark instances.

Our results show in this experimentation, that the metaheuristic with best performance was Cellular Algorithm; due to the fact this metaheuristic has neighborhoods operators and work in parallel with different solutions.

As future work we propose to implement difference metaheuristic as Memetic algorithms, Tabu Search, etc. Also we propose to use real instances for test the methodology and Metaheuristics

Acknowledgment Authors thanks the support received from the Consejo Nacional de Ciencia y Tecnologia (CONACYT) México

References

1. Ortiz-Aguilar Lucero de M.: Diseño de horarios de alumnos y maestros mediante técnicas de Soft Computing, para una Institución Educativa. Master's thesis, InstitutoTecnológico de León (2016).
2. De Werra, D.: An introduction to timetabling. European Journal of Operational Research(2), 151 – 162 (1985).
3. Obit, J. H., Landa-Silva, D., Ouelhadj, D., Khan Vun, T., & Alfred, R.: Designing a multi-agent approach system for distributed course timetabling. IEEE. (2011).
4. Asratian, A. S., de Werra, D., Luleå.: A generalized class–teacher model for some timetabling problems. University of Technology, Department of Engineering Sciences and Mathematics, Mathematical Science, & Mathematics. European Journal of Operational Re-search, **143**(3), 531-542. (2002). doi:10.1016/S0377-2217(01)00342-3.
5. Deng, X., Zhang, Y., Kang, B., Wu, J., Sun, X., Deng, Y.: An application of genetic algorithm for university course timetabling problem. 2119-2122 (2011). doi:10.1109/CCDC.2011.5968555.
6. Mahiba, A. A., Durai, C. A. D.: Genetic algorithm with search bank strategies for university course timetabling problem. Procedia Engineering, 38, 253-263. (2012). doi:10.1016/j.proeng.2012.06.033.
7. Soria-Alcaraz, J. A., Carpio, J. M.; Puga, Hé., Melin, P.; Terashima-Marn, H., Reyes, L. C., Sotelo-Figueroa, M. A. Castillo, O., Melin, P., Pedrycz, W. &Kacprzyk, J.: Generic Memetic Algorithm for Course Timetabling ITC2007 Recent Advances on Hybrid Approaches for Designing Intelligent Systems, Springer, 547, 481-492. (2014).
8. Nguyen K., Lu T., Le T., Tran N.: Memetic algorithm for a university course timetabling problem, **132**(1) 67-71. (2011). doi:10.1007/978-3-642-25899-2_10.

9. Aladag C., Hocaoglu G. A tabu search algorithm to solve a course timetabling problem. hacettepe journal of mathematics and statistics, **36**(1), 53-64. (2007).
10. Moscato, P.: "On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts: Towards Memetic Algorithms". Caltech Concurrent Computation Program (report 826). (1989).
11. Zheng, S., Wang, L., Liu, Y., & Zhang, R.: A simulated annealing algorithm for university course timetabling considering travelling distances. International Journal of Computing Science and Mathematics, **6**(2), 139-151. (2015). doi:10.1504/IJCSM.2015.069461.
12. Joudaki M., Imani M., Mazhari N. Using improved Memetic algorithm and local search to solve University Course Timetabling Problem (UCTTP). Doroud, Iran: Islamic Azad University. (2010).
13. Thepphakorn T., Pongcharoen P., Hicks C.: An ant colony based timetabling tool. International Journal of Production Economics, 149, 131-144. (2014). doi:10.1016/j.ijpe.2013.04.026.
14. Soria-Alcaraz J., Ochoa G., Swan J., Carpio M., Puga H., Burke E.: Effective learning hyper-heuristics for the course timetabling problem. European Journal of Operational Research, **238**(1), 77-86. (2014) doi:10.1016/j.ejor.2014.03.046.
15. Lewis, M. R. R.: Metaheuristics for university course timetabling. Ph.D. Thesis, Napier University. (2006).
16. Cooper, T. B. & Kingston, J. H.: The Complexity of Timetable Construction Problems. PhD thesis, The University of Sydney. (1995).
17. Wolpert, H., Macready, G.: No free lunch Theorems for Search. Technical report The Santa Fe Institute, 1 (1996).
18. Adriaen M., De Causmaecker P., Demeester P., VandenBerghe G.: Tackling the university course timetabling problem with an aggregation approach. In Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno.,**1**, 330-335. (2006).
19. Soria-Alcaraz, J.A.: Diseño de horarios con respecto al alumno mediante tecnicas de computo evolutivo. Master's thesis, Instituto Tecnologico de Leon (2010).
20. McCollum., B. University timetabling: Bridging the gap between research and practice. In Proceedings of the 6th International Conference on the Practice and Theory of Automated Timetabling, Brno, 1, 15-35. (2006).
21. Conant-Pablos, S.E., Magaa-Lozano, D.J., Terashima-Marin, H.: Pipelining memetic algorithms, constraint satisfaction, and local search for course timetabling. MICAI Mexican international conference on artificial intelligence **1**, 408–419 (2009).
22. DammakAbdelaziz, ElloumiAbdelkarim, K. H. Classroom assignment for exam timetabling. Advances in Engineering Software, **37**(10), 659-666. (2006).
23. Soria-Alcaraz Jorge, A., Carpio, M., Puga, H., Sotelo-Figueroa, M.: Method-ology of design: A novel generic approach applied to the course timetabling problem. In: P. Melin, O. Castillo (eds.) Soft Computing Applications in Op-timization, Control, and Recognition, Studies in Fuzziness and Soft Compu-ting, vol. 294, pp. 287–319. Springer Berlin Heidelberg (2013).
24. Alcaraz J. A. S.: Integración de un esquema de Diseño en Algoritmos de Dos fases con técnicas CSP para calendarización de eventos. PhD thesis, Instituto Tecnologico de Leon. (2014).
25. Alba, E.: Parallel Metaheuristics: A New Class of Algorithms. Wiley-Interscience. (2005).
26. Goldberg, D. E.: Genetic Algorithms in Search, Optimization and Machine Learning. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st edition. (1989).
27. Alba, E. & Dorronsoro, B.: The state of the art in cellular evolutionary algorithms. Cellular Genetic Algorithms, **1**, 21-34. (2008).
28. Lourenco, H., Martin, O., &Stützle, T. Iterated local search. In F. Glover, G. Kochenberger, & F. S. Hillier (Eds.), Handbook of Metaheuristics, volume 57 of International Series in Operations Research & Management Science, 320-353. Springer New York. (2003).
29. Talbi, E.-G. Metaheuristics: From Design to Implementation. Wiley Publishing. (2009).

Referencias

[1] P. Jeavons, D. Cohen, and M. C. Cooper, "Constraints, consistency and closure," *Artificial Intelligence*, vol. 101, no. 1, pp. 251–265, 1998.

[2] M. Carpio, "Modelo integral de asignación optima de carga academica usando un algoritmo heurístico," *Encuentro de investigación en Ingenieria electrica*, 2006.

[3] G. D. A. Ríos and J. M. C. Valadez, "Optimización y automatización en la asignación de tareas aplicadas en el area educativa," *ConTECSI*, 2005.

[4] J. A. Soria-Alcaraz, "Diseño de horarios con respecto al alumno mediante técnicas de cómputo evolutivo," Master's thesis, Instituto Tecnologico de León, 2010.

[5] J. A. S. Alcaraz, *Integración de un esquema de Diseno en Algoritmos de Dos fases con tecnicas CSP para calendarizacion de eventos.* PhD thesis, INSTITUTO TECNOLOGICO DE LEON, Diciembre 2014.

[6] L. d. M. Ortiz-Aguilar, "Diseño de horarios de alumnos y maestros mediante técnicas de soft computing, para una institución educativa," mathesis, Instituto Tecnologico de León, 2016.

[7] P. Crescenzi, V. Kann, and M. Halldórsson, "A compendium of np optimization problems," 1995.

[8] P. Jeavons, "On the algebraic structure of combinatorial problems," *Theoretical Computer Science*, vol. 200, no. 1, pp. 185–204, 1998.

[9] M. A. Sotelo-Figueroa, A. Hernández-Aguirre, A. Espinal, J. A. Soria-Alcaraz, and J. Ortiz-López, *Symbolic Regression by Means of Grammatical Evolution with Estimation Distribution Algorithms as Search Engine*, pp. 169–177. Cham: Springer International Publishing, 2018.

[10] A. B. Evans, "Latin squares based on groups," in *Orthogonal Latin Squares Based on Groups*, pp. 3–40, Springer, 2018.

[11] J. N. Srivastava, ed., *A Survey of Combinatorial Theory*. North-Holland, 1973.

[12] R. M. Karp, "On the computational complexity of combinatorial problems," *Networks*, vol. 5, no. 1, pp. 45–68, 1975.

[13] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1979.

[14] V. Klee, "Combinatorial optimization: What is the state of the art?," in *Information Linkage Between Applied Mathematics and Industry* (P. C. WANG, A. L. Schoenstadt, B. I. Russak, and C. Comstock, eds.), pp. 71 – 136, Academic Press, 1979.

[15] J. Pearson and P. G. Jeavons, "A survey of tractable constraint satisfaction problems," tech. rep., Technical Report CSD-TR-97-15, Royal Holloway, University of London, 1997.

[16] S. C. Brailsford, C. N. Potts, and B. M. Smith, "Constraint satisfaction problems: Algorithms and applications," *European Journal of Operational Research*, vol. 119, no. 3, pp. 557–581, 1999.

[17] D.-Z. Du and P. M. Pardalos, *Handbook of combinatorial optimization. Supplement*, vol. Volume A. Kluwer Academic Publishers, 1 ed., 1999.

[18] K. L. Hoffman, "Combinatorial optimization: Current successes and directions for the future," *Journal of Computational and Applied Mathematics*, vol. 124,

no. 1, pp. 341 – 360, 2000. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations.

[19] H. H. Hoos and T. Stützle, *Stochastic local search: Foundations and applications*. Elsevier, 1 ed., 2004.

[20] M. Zimand, "Chapter 6 - optimization problems," in *Computational Complexity: A Quantitative Perspective* (M. Zimand, ed.), vol. 196 of *North-Holland Mathematics Studies*, pp. 225 – 316, North-Holland, 2004.

[21] A. Schrijver, "On the history of combinatorial optimization (till 1960)," in *Discrete Optimization* (K. Aardal, G. Nemhauser, and R. Weismantel, eds.), vol. 12 of *Handbooks in Operations Research and Management Science*, pp. 1 – 68, Elsevier, 2005.

[22] K. Smith-Miles and L. Lopes, "Measuring instance difficulty for combinatorial optimization problems," *Computers & Operations Research*, vol. 39, no. 5, pp. 875–889, 2011.

[23] M. S. Levin, "Four-layer framework for combinatorial optimization problems domain," *Advances in Engineering Software*, vol. 42, no. 12, pp. 1089–1098, 2011.

[24] D. Ding-Zhu, M. P. Panos, and L. G. Ronald, eds., *Handbook of Combinatorial Optimization*. Springer-Verlag New York, 2 ed., 2013.

[25] R. Lahyani, M. Khemakhem, and F. Semet, "Rich vehicle routing problems: From a taxonomy to a definition," *European Journal of Operational Research*, vol. 241, no. 1, pp. 1 – 14, 2015.

[26] M. F. Gorman, "A metasurvey analysis in operations research and management science: A survey of literature reviews," *Surveys in Operations Research and Management Science*, vol. 21, no. 1, pp. 18–28, 2016.

[27] T. Bartz-Beielstein and M. Zaefferer, "Model-based methods for continuous and discrete global optimization," *Applied Soft Computing*, vol. 55, pp. 154–167, 2017.

[28] S. Lin and B. W. Kernighan, "An effective heuristic algorithm for the traveling-salesman problem," *Operations Research*, vol. 21, pp. 498–516, apr 1973.

[29] K. Helsgaun, "General k-opt submoves for the lin–kernighan tsp heuristic," *Mathematical Programming Computation*, vol. 1, no. 2-3, pp. 119–163, 2009.

[30] A. Blazinskas and A. Misevicius, "combining 2-opt, 3-opt and 4-opt with k-swap-kick perturbations for the traveling salesman problem," in *17th International Conference on Information and Software Technologies*, pp. 27–29, 2011.

[31] J. M. Thompson and K. A. Dowsland, "A robust simulated annealing based examination timetabling system," *Computers & Operations Research*, vol. 25, no. 7-8, pp. 637–648, 1998.

[32] S. Minton, M. D. Johnston, A. B. Philips, and P. Laird, "Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems," *Artificial Intelligence*, vol. 58, pp. 161–205, dec 1992.

[33] B. S. Baker, "A new proof for the first-fit decreasing bin-packing algorithm," *Journal of Algorithms*, vol. 6, pp. 49–70, mar 1985.

[34] J. Csirik, "The parametric behavior of the first-fit decreasing bin packing algorithm," *Journal of Algorithms*, vol. 15, pp. 1–28, jul 1993.

[35] J. A. Soria-Alcaraz, G. Ochoa, M. Carpio, H. Puga, and E. K. Burke, "Effective learning hyper-heuristics for the course timetabling problem," *European Journal of Operational Research*, p. 10, abril 2014.

[36] J. A. Soria-Alcaraz, E. Özcan, J. Swan, G. Kendall, and M. Carpio, "Iterated local search using an add and delete hyper-heuristic for university course timetabling," *Applied Soft Computing*, vol. 40, pp. 581–593, 2016.

[37] W. B. Yates and E. C. Keedwell, "An analysis of heuristic subsequences for offline hyper-heuristic learning," *Journal of Heuristics*, vol. 25, pp. 399–430, jan 2019.

[38] W. B. Yates and E. C. Keedwell, "Offline learning for selection hyper-heuristics with elman networks," in *Lecture Notes in Computer Science*, pp. 217–230, Springer International Publishing, 2018.

[39] J. A. Soria-Alcaraz, G. Ochoa, M. A. Sotelo-Figeroa, and E. K. Burke, "A methodology for determining an effective subset of heuristics in selection hyper-heuristics," *European Journal of Operational Research*, vol. 260, no. 3, pp. 972–983, 2017.

[40] I. Amaya, J. C. Ortiz-Bayliss, S. Conant-Pablos, and H. Terashima-Marin, "Hyper-heuristics reversed: Learning to combine solvers by evolving instances," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, jun 2019.

[41] H. Song, I. Triguero, and E. Özcan, "A review on the self and dual interactions between machine learning and optimisation," *Progress in Artificial Intelligence*, vol. 8, pp. 143–165, apr 2019.

[42] P. Brazdil, C. Giraud-Carrier, C. Soares, and R. Vilalta, *Metalearning*. Springer Berlin Heidelberg, 2009.

[43] A. E. Gutierrez-Rodríguez, S. E. Conant-Pablos, J. C. Ortiz-Bayliss, and H. Terashima-Marín, "Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning," *Expert Systems with Applications*, vol. 118, pp. 470–481, 2019.

[44] J. Kanda, A. de Carvalho, E. Hruschka, C. Soares, and P. Brazdil, "Meta-learning to select the best meta-heuristic for the traveling salesman problem: A comparison of meta-features," *Neurocomputing*, vol. 205, pp. 393–406, sep 2016.

[45] F. Comellas, J. Fabrega, and A. S. O. Serra, *Matemática discreta*. Alfaomega, 2001.

[46] K. H. Rosen, *Matemática discreta y sus aplicaciones*. McGraw-Hill Interamericana de España S.L., 2004.

[47] D. de Werra, "Heuristics for graph coloring," in *Computational graph theory*, pp. 191–208, Springer, 1990.

[48] M. M. Solomon, "On the worst-case performance of some heuristics for the vehicle routing and scheduling problem with time window constraints," *Networks*, vol. 16, no. 2, pp. 161–174, 1986.

[49] H. Derbel, B. Jarboui, and R. Bhiri, "A skewed general variable neighborhood search algorithm with fixed threshold for the heterogeneous fleet vehicle routing problem," *Annals of Operations Research*, vol. 272, no. 1-2, pp. 243–272, 2019.

[50] S. S. Habashi, C. Salama, A. H. Yousef, and H. M. Fahmy, "Adaptive diversifying hyper-heuristic based approach for timetabling problems," in *2018 IEEE 9th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pp. 259–266, IEEE, 2018.

[51] A. Salwani, B. E. K., and M. Barry, "Using a randomised iterative improvement algorithm with composite neighbourhood structures for the university course timetabling problem," in *Metaheuristics*, vol. 39 of *Operations Research Computer Science Interfaces Series*, pp. 153–169, Springer US, 2007.

[52] G. I. P., M. Ewan, P. Patrick, S. BarbaraM., and W. Toby, "An empirical study of dynamic variable ordering heuristics for the CSP," in *Principles and Practice of Constraint Programming — CP96* (E. Freuder, ed.), vol. 1118 of *Lecture Notes in Computer Science*, pp. 179–193, Springer Berlin Heidelberg, 1996.

[53] L. Helena, M. Olivier, and S. Thomas, "Iterated local search," in *Handbook of Metaheuristics* (F. Glover, G. Kochenberger, and F. S. Hillier, eds.), vol. 57 of *International Series in Operations Research & Management Science*, pp. 320–353, Springer New York, 2003.

[54] T. El-Ghazali, *Metaheuristics: From Design to Implementation.* Wiley Publishing, 2009.

[55] P. Brazdil, C. G. Carrier, C. Soares, and R. Vilalta, *Metalearning: Applications to data mining*. Springer Science & Business Media, 2008.

[56] A. A. Ghorbani, W. Lu, and M. Tavallaee, *Network intrusion detection and prevention: concepts and techniques*, vol. 47. Springer Science & Business Media, 2009.

[57] O. Maimon and L. Rokach, "Data mining and knowledge discovery handbook," 2005.

[58] M. Kubale, *Graph colorings*, vol. 28. Amer Mathematical Society, 2004.

[59] K. R.M., *Complexity of Computer Computations*. Springer Us, 1972.

[60] R. R. C, *Graph theory and computing*. Academic Press, 2014.

[61] O. Borodin, "Colorings of plane graphs: A survey," *Discrete Mathematics*, vol. 313, no. 4, pp. 517 – 539, 2013.

[62] F. M. M. Preissmann, "On the np-completeness of the k-colorability problem for triangle-free graphs," *Discrete Mathematics*, vol. 162, 1996.

[63] D. Narsingh, *Graph Theory with Applications to Engineering and Computer Science*. Dover Publications Inc., 2016.

[64] G. P. A., J. Matthew, P. Daniël, and S. Jian, "A survey on the computational complexity of coloring graphs with forbidden subgraphs," *Journal of Graph Theory*, vol. 84, no. 4, pp. 331–363, 2017.

[65] J. Adalat and M. Petra, "New integer linear programming models for the vertex coloring problem," *arXiv preprint arXiv:1706.10191*, 2017.

[66] K. A. MCA, S. Robert, and T. Martin, "A flow based pruning scheme for enumerative equitable coloring algorithms," *Annals of Operations Research*, pp. 1–26, 2017.

[67] W. Meyer, "Equitable coloring," *The American Mathematical Monthly*, vol. 80, no. 8, pp. 920–922, 1973.

[68] J. Y., H. JP, and H. JK, "Algorithms for the minimum sum coloring problem: a review," *ARTIFICIAL INTELLIGENCE REVIEW*, vol. 47, no. 3, pp. 367–394, 2017.

[69] K. Ken-Ichi and T. Mikkel, "Coloring 3-colorable graphs with less than n1/5 colors," *J. ACM*, vol. 64, pp. 4:1–4:23, Mar. 2017.

[70] G. Wei, "Three algorithms for graph locally harmonious colouring," *Journal of Difference Equations and Applications*, vol. 23, no. 1-2, pp. 8–20, 2017.

[71] P. A. V, "Interval coloring of (3, 4)-biregular bipartite graphs having large cubic subgraphs," *Journal of Graph Theory*, vol. 47, no. 2, pp. 122–128, 2004.

[72] D. W. A and Z. Xuding, "Circular colorings of weighted graphs," *Journal of Graph Theory*, vol. 23, no. 4, pp. 365–376, 1996.

[73] C. Sylvie, V.-P. Mario, G. Danièle, B. Dominique, and D. Alain, "The permutation-path coloring problem on trees," *Theoretical Computer Science*, vol. 297, no. 1-3, pp. 119–143, 2003.

[74] W. Wei and L. Xin, "List-coloring based channel allocation for open-spectrum wireless networks," in *Vehicular Technology Conference, 2005. VTC-2005-Fall. 2005 IEEE 62nd*, vol. 1, pp. 690–694, IEEE, 2005.

[75] M. R. Garey and D. S. Johnson, "Computers and intractability: a guide to the theory of np-completeness," *WH Free. Co., San Fr*, pp. 90–91, 1979.

[76] L. R. Duncan and P. A. D, "A method of matrix analysis of group structure," *Psychometrika*, vol. 14, no. 2, pp. 95–116, 1949.

[77] B. I. M, B. Marco, P. P. M, and P. Marcello, "The maximum clique problem," in *Handbook of combinatorial optimization*, pp. 1–74, Springer, 1999.

[78] P. P. M and X. Jue, "The maximum clique problem," *Journal of global Optimization*, vol. 4, no. 3, pp. 301–328, 1994.

[79] C. Randy and P. P. M, "An exact algorithm for the maximum clique problem," *Operations Research Letters*, vol. 9, no. 6, pp. 375–382, 1990.

[80] Ö. P. RJ, "A new algorithm for the maximum-weight clique problem," *Nordic Journal of Computing*, vol. 8, no. 4, pp. 424–436, 2001.

[81] E. Paul and G. Tibor, "On the minimal number of vertices representing the edges of a graph," *Publ. Math. Inst. Hungar. Acad. Sci*, vol. 6, no. 18, pp. 1–203, 1961.

[82] H. Refael and L. Asaf, "The minimum generalized vertex cover problem," in *ESA*, pp. 289–300, Springer, 2003.

[83] H. Refael and L. Asaf, "The minimum generalized vertex cover problem," *ACM Transactions on Algorithms (TALG)*, vol. 2, no. 1, pp. 66–78, 2006.

[84] G. Jiong, N. Rolf, and W. Sebastian, *Parameterized Complexity of Generalized Vertex Cover Problems*, vol. 3608, pp. 36–48. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005.

[85] H. Elad, S. Shmuel, and S. Oded, "On the complexity of approximating k-dimensional matching," in *Approximation, Randomization, and Combinatorial Optimization.. Algorithms and Techniques*, pp. 83–97, Springer, 2003.

[86] K. Viggo, "Maximum bounded 3-dimensional matching is max snp-complete," *Information Processing Letters*, vol. 37, no. 1, pp. 27–35, 1991.

[87] M. Cygan, "Improved approximation for 3-dimensional matching via bounded pathwidth local search," in *Foundations of Computer Science (FOCS), 2013 IEEE 54th Annual Symposium on*, pp. 509–518, IEEE, 2013.

[88] D. George, F. Ray, and J. Selmer, "Solution of a large-scale traveling-salesman problem," *Journal of the operations research society of America*, vol. 2, no. 4, pp. 393–410, 1954.

[89] C. William, *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press, 2012.

[90] G. M. R, J. D. S., and T. R. Endre, "The planar hamiltonian circuit problem is np-complete," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 704–714, 1976.

[91] W. David and G. J. A, "A survey: Hamiltonian cycles in cayley graphs," *Discrete Mathematics*, vol. 51, no. 3, pp. 293–304, 1984.

[92] G. R. J, "Updating the hamiltonian problem—a survey," *Journal of Graph Theory*, vol. 15, no. 2, pp. 121–157, 1991.

[93] D. A. M, L. P. Frazer, and S. Michael, "A survey of graphs hamiltonian-connected from a vertex," *Graph Theory, Combinatorics and Applications*, vol. 2, 1991.

[94] R. J. Gould, "Advances on the hamiltonian problem–a survey," *Graphs and Combinatorics*, vol. 19, no. 1, pp. 7–52, 2003.

[95] F. Harary, "Graph theory," *Historia Mathematica*, vol. 4, 1977.

[96] G. Hendry, "Graphs uniquely hamiltonian-connected from a vertex," *Discrete Mathematics*, vol. 49, 1984.

[97] K. J., S. U., and T. Jacobo, *Graph Isomorphism Problem*. Birkhauser Boston, 2012.

[98] S. A. Cook, "The complexity of theorem-proving procedures," in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, (New York, NY, USA), pp. 151–158, ACM, 1971.

[99] D. Harm, "The graph isomorphism problem and approximate categories," *Journal of Symbolic Computation*, vol. 59, 12 2013.

[100] J. R. Ullmann, "An algorithm for subgraph isomorphism," *J. ACM*, vol. 23, pp. 31–42, Jan. 1976.

[101] K. J., S. U., and T. Jacobo, *The Graph Isomorphism Problem: Its Structural Complexity*. Birkhauser Boston, 1993.

[102] V. N. Zemlyachenko, N. M. Korneenko, and R. I. Tyshkevich, "Graph isomorphism problem," *Journal of Mathematical Sciences*, vol. 29, 05 1985.

[103] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, vol. 8, pp. 128–140, 1741.

[104] B. Bollobás, "The evolution of random graphs—the giant component," in *Random Graphs*, Cambridge University Press, 2001.

[105] R. Hopcroft, John; Tarjan, "Algorithm 447: efficient algorithms for graph manipulation," *Communications of the ACM*, vol. 16, 6 1973.

[106] T. Gallai, "Transitiv orientierbare graphen," *Acta Mathematica Hungarica*, vol. 18, 1967.

[107] D. D., G. J., and P. P.M., *Satisfiability Problem: Theory and Applications : DIMACS Workshop, March 11-13, 1996*. Center for Discrete Mathematics and Theoretical Computer Science New Brunswick, NJ: DIMACS series in discrete mathematics and theoretical computer science, American Mathematical Society, 1997.

[108] W. Roman, D. Jack, M. Norbert, and W. Jerzy, *[Lecture Notes in Computer Science] Parallel Processing and Applied Mathematics Volume 3911 —— Parallel Resolution of the Satisfiability Problem (SAT) with OpenMP and MPI*, vol. 10.1007/11752578, ch. 46, pp. 380–388. Springer, 2006.

[109] D. Ding-Zhu and P. P. M., *Handbook of Combinatorial Optimization , Algorithms for the Satisfiability (SAT) Problem*, vol. 10.1007/978-1-4757-3023-4, ch. 7, pp. 379–572. Springer Us, 1999.

[110] M. Mézard, T. Mora, and R. Zecchina, "Clustering of solutions in the random satisfiability problem," *Physical Review Letters*, vol. 94, no. 19, p. 197205, 2005.

[111] M. Marc and Z. Riccardo, "The random k-satisfiability problem: From an analytic solution to an efficient algorithm," *Physical Review E*, vol. 66, no. 5, p. 056126, 2002.

[112] A. A. V. and H. J. E., *The Design and Analysis of Computer Algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1st ed., 1974.

[113] A. Biere, A. Biere, M. Heule, H. van Maaren, and T. Walsh, *Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications*. Amsterdam, The Netherlands, The Netherlands: IOS Press, 2009.

[114] H. Buning, M. Karpinski, and A. Flogel, "Resolution for quantified boolean formulas," *Inf. Comput.*, vol. 117, pp. 12–18, Feb. 1995.

[115] G. Andersson and L. Engebretsen, "Better approximation algorithms for set splitting and not-all-equal sat," *Information Processing Letters*, vol. 65, no. 6, pp. 305–311, 1998.

[116] T. J. Schaefer, "The complexity of satisfiability problems," in *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, (New York, NY, USA), pp. 216–226, ACM, 1978.

[117] C. Moore and S. Mertens, *The Nature of Computation*. Oxford University Press, 2011.

[118] U. Montanari, "Networks of constraints: Fundamental properties and applications to picture processing," *Information sciences*, vol. 7, pp. 95–132, 1974.

[119] A. K. Mackworth, "Constraint satisfaction problems," *Encyclopedia of AI*, vol. 285, p. 293, 1992.

[120] M. C. Cooper and S. Zivny, "Hybrid Tractable Classes of Constraint Problems," in *The Constraint Satisfaction Problem: Complexity and Approximability* (A. Krokhin and S. Zivny, eds.), vol. 7 of *Dagstuhl Follow-Ups*, pp. 113–135, Dagstuhl, Germany: Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[121] M. Bodirsky, V. Dalmau, B. Martin, A. Mottet, and M. Pinsker, "Distance constraint satisfaction problems," *Information and Computation*, vol. 247, no. Supplement C, pp. 87 – 105, 2016.

[122] A. Krokhin and S. Zivny, "Dfu, volume 7, the constraint satisfaction problem: Complexity and approximability, complete volume," in *Dagstuhl Follow-Ups*, vol. 7, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[123] H. M. Salkin and C. A. De Kluyver, "The knapsack problem: A survey," *Naval Research Logistics Quarterly*, vol. 22, no. 1, pp. 127–144, 1975.

[124] U.-E. Lukata and J. Teghem, "Solving multi-objective knapsack problem by a branch-and-bound procedure," in *Multicriteria analysis*, pp. 269–278, Springer, 1997.

[125] P. Chu and J. Beasley, "A genetic algorithm for the multidimensional knapsack problem," *Journal of Heuristics*, vol. 4, 06 1998.

[126] M. Dawande, J. Kalagnanam, P. Keskinocak, F. Salman, and R. Ravi, "Approximation algorithms for the multiple knapsack problem with assignment restrictions," *Journal of Combinatorial Optimization*, vol. 4, 06 2000.

[127] D. Pisinger, "The quadratic knapsack problem—a survey," *Discrete applied mathematics*, vol. 155, no. 5, pp. 623–648, 2007.

[128] L. F. Escudero, S. Martello, and P. Toth, "On tightening 0-1 programs based on extensions ofpure 0-1 knapsack and subset-sum problems," *Annals of Operations Research*, vol. 81, 06 1998.

[129] M. L. Brandeau and S. S. Chiu, "An overview of representative problems in location research," *Manage. Sci.*, vol. 35, pp. 645–674, June 1989.

[130] V. Verter, "Uncapacitated and capacitated facility location problems," in *Foundations of location analysis*, pp. 25–37, Springer, 2011.

[131] G. Cornuéjols, G. L. Nemhauser, and L. A. Wolsey, "The uncapacitated facility location problem," tech. rep., Carnegie-mellon univ pittsburgh pa management sciences research group, 1983.

[132] R. L. Dearing, P. M.; Francis, "A minimax location problem on a network," *Transportation Science*, vol. 8, 11 1974.

[133] H. Nagamochi, T. Ishii, and H. Ito, "Minimum cost source location problem with vertex-connectivity requirements in digraphs," *Information Processing Letters*, vol. 80, no. 6, pp. 287–293, 2001.

[134] B. B. Bhattacharya and S. C. Nandy, "New variations of the maximum coverage facility location problem," *European Journal of Operational Research*, vol. 224, no. 3, pp. 477–485, 2013.

[135] B. Dasarathy and L. J. White, "A maxmin location problem," *Operations Research*, vol. 28, no. 6, pp. 1385–1401, 1980.

[136] D. Shishebori, M. Jabalameli, and A. Jabbarzadeh, "Facility location-network design problem: Reliability and investment budget constraint," *JOURNAL OF URBAN PLANNING AND DEVELOPMENT*, vol. 140, no. 3, 2014.

[137] Y. Chen, Q. Zhao, L. Wang, and M. Dessouky, "The regional cooperation-based warehouse location problem for relief supplies," *COMPUTERS & INDUSTRIAL ENGINEERING*, vol. 102, pp. 259–267, 2016.

[138] V. L. Beresnev and A. A. Melnikov, "A capacitated competitive facility location problem," *Journal of Applied and Industrial Mathematics*, vol. 10, no. 1, pp. 61–68, 2016.

[139] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of discrete mathematics*, vol. 5, pp. 287–326, 1979.

[140] P. Brucker, A. Drexl, R. MÃ¶hring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, models, and methods," *European Journal of Operational Research*, vol. 112, no. 1, pp. 3 – 41, 1999.

[141] P. Senthilkumar and S. Narayanan, "Literature review of single machine scheduling problem with uniform parallel machines," *Intelligent Information Management*, vol. 2, no. 08, p. 457, 2010.

[142] M. R. Garey and D. S. Johnson, "Two-processor scheduling with start-times and deadlines," *SIAM Journal on Computing*, vol. 6, no. 3, pp. 416–426, 1977.

[143] U. Dorndorf, A. Drexl, Y. Nikulin, and E. Pesch, "Flight gate scheduling: State-of-the-art and recent developments," *Omega*, vol. 35, no. 3, pp. 326–334, 2007.

[144] P. A. M. Duque, I. S. Dolinskaya, and K. Sörensen, "Network repair crew scheduling and routing for emergency relief distribution problem," *European Journal of Operational Research*, vol. 248, no. 1, pp. 272–285, 2016.

[145] J. B. Ghosh and J. N. Gupta, "Batch scheduling to minimize maximum lateness," *Operations Research Letters*, vol. 21, no. 2, pp. 77 – 80, 1997.

[146] I. M. Alharkan, "Algorithms for sequencing and scheduling," *Industrial Engineering Department, King Saud University, Riyadh, Saudi Arabia*, 2005.

[147] A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier, "Staff scheduling and rostering: A review of applications, methods and models," *European Journal of Operational Research*, vol. 153, no. 1, pp. 3 – 27, 2004. Timetabling and Rostering.

[148] B. Cheang, H. Li, A. Lim, and B. Rodrigues, "Nurse rostering problem a bibliographic survey," *European Journal of Operational Research*, vol. 151, no. 3, pp. 447 – 460, 2003.

[149] J. V. den Bergh, J. BeliÃ«n, P. D. Bruecker, E. Demeulemeester, and L. D. Boeck, "Personnel scheduling: A literature review," *European Journal of Operational Research*, vol. 226, no. 3, pp. 367 – 385, 2013.

[150] D. de Werra, "An introduction to timetabling," *European Journal of Operational Research*, vol. 19, no. 2, pp. 151 – 162, 1985.

[151] R. Hana, M. Tomas, and M. Keith, "Complex university course timetabling," *Journal of Scheduling*, vol. 14, pp. 187–207, 2011. 10.1007/s10951-010-0171-3.

[152] N. A. Ismayilova, M. Sagir, and R. N. Gasimov, "A multiobjective faculty course time slot assignment problem with preferences," *Mathematical and Computer Modelling*, pp. 1017 – 1029, Marzo 2007.

[153] S.-T. Shih, C.-Y. Chao, and C.-M. Hsu, "An effective and efficient class-course-faculty timetabling assignment for an educational institute," *Life Sciences*, p. 9, 2012.

[154] A. Asratian and D. de Werra, "A generalized class-teacher model for some timetabling problems," *European Journal of Operational Research*, pp. 531–542, Septiembre 2002.

[155] C. M. W, L. Gilbert, and L. S. Yan, "Examination timetabling: Algorithmic strategies and applications," *Journal of the Operational Research Society*, pp. 373–383, 1996.

[156] C. Cheng, B. Feiring, and T. Cheng, "The cutting stock problem—a survey," *International Journal of Production Economics*, vol. 36, no. 3, pp. 291–305, 1994.

[157] H. Ben Amor and J. Valério de Carvalho, "Cutting stock problems," *Column generation*, pp. 131–161, 2005.

[158] T. Kis, "On the complexity of the car sequencing problem," *Operations Research Letters*, vol. 32, no. 4, pp. 331–335, 2004.

[159] C. Solnon, V. D. Cung, A. Nguyen, and C. Artigues, "The car sequencing problem: Overview of state-of-the-art methods and industrial case-study of the roadef'2005 challenge problem," *European Journal of Operational Research*, vol. 191, no. 3, pp. 912 – 927, 2008.

[160] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[161] M. Biro, M. Hujter, and Z. Tuza, "Precoloring extension. i. interval graphs," *Discrete Mathematics*, vol. 100, no. 1-3, pp. 267–279, 1992.

[162] A. Kosowski and K. Manuszewski, "Classical coloring of graphs," *Contemporary Mathematics*, vol. 352, pp. 1–20, 2004.

[163] D. Marx, "Np−completeness of list coloring and precoloring extension on the edges of planar graphs," *Journal of Graph Theory*, vol. 49, no. 4, pp. 313–324, 2005.

[164] L. d. M. Ortiz-Aguilar, M. Carpio, H. Puga, J. A. Soria-Alcaraz, M. Ornelas-Rodríguez, and C. Lino, "Increase methodology of design of course timetabling problem for students, classrooms, and teachers," in *Nature-Inspired Design of Hybrid Intelligent Systems*, pp. 713–728, Springer, 2017.

[165] N. Pillay, "A review of hyper-heuristic for educational timetabling," *Springer*, 2014.

[166] D. W. Scott, "Sturges' rule," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, no. 3, pp. 303–306, 2009.

[167] D. D. Lewis, "Naive (bayes) at forty: The independence assumption in information retrieval," in *Machine Learning: ECML-98*, pp. 4–15, Springer Berlin Heidelberg, 1998.

[168] E. Burke, T. Curtois, M. Hyde, G. Kendall, G. Ochoa, S. Petrovic, J. A. Vázquez-Rodríguez, and M. Gendreau, "Iterated local search vs. hyper-heuristics: Towards general-purpose search algorithms," in *IEEE congress on evolutionary computation*, pp. 1–8, IEEE, 2010.

[169] S. S. Choong, L.-P. Wong, and C. P. Lim, "Automatic design of hyper-heuristic based on reinforcement learning," *Information Sciences*, vol. 436, pp. 89–107, 2018.

[170] J. A. Soria-Alcaraz, M. A. Sotelo-Figueroa, and A. Espinal, "Statistical comparative between selection rules for adaptive operator selection in vehicle routing and multi-knapsack problems," in *Fuzzy Logic Augmentation of Neural and Optimization Algorithms: Theoretical Aspects and Real Applications*, pp. 389–400, Springer, 2018.

[171] P. Augerat, J. M. Belenguer, E. Benavent, A. Corberán, D. Naddef, and G. Rinaldi, *Computational results with a branch and cut code for the capacitated vehicle routing problem*. IMAG, 1995.

[172] N. Christofides, "The vehicle routing problem. n. christofides, a. mingozzi, p. toth, c. sandi, eds. combined optimization," 1979.

[173] B. L. Golden, E. A. Wasil, J. P. Kelly, and I.-M. Chao, "The impact of metaheuristics on solving the vehicle routing problem: algorithms, problem sets, and computational results," in *Fleet management and logistics*, pp. 33–56, Springer, 1998.

[174] E. Uchoa, D. Pecin, A. Pessoa, M. Poggi, T. Vidal, and A. Subramanian, "New benchmark instances for the capacitated vehicle routing problem," *European Journal of Operational Research*, vol. 257, no. 3, pp. 845 – 858, 2017.

[175] Z. Hanusz, J. Tarasinska, and W. Zielinski, "Shapiro-wilk test with known mean," *REVSTAT-Statistical Journal*, vol. 14, no. 1, pp. 89–100, 2016.

[176] G. V. Glass, "Testing homogeneity of variances," *American Educational Research Journal*, vol. 3, no. 3, pp. 187–190, 1966.

[177] P. Berman and A. Pelc, "Distributed probabilistic fault diagnosis for multiprocessor systems," in *Fault-Tolerant Computing, 1990. FTCS-20. Digest of Papers., 20th International Symposium*, pp. 340–346, IEEE, 1990.

[178] J. C. Lagarias and P. W. Shor, "Keller's cube tiling conjecture is false in high dimensions," *Bulletin of the American Mathematical Society*, vol. 27, no. 2, pp. 279–283, 1992.

[179] M. Gendreau, P. Soriano, and L. Salvail, "Solving the maximum clique problem using a tabu search approach," *Annals of Operations Research*, vol. 41, no. 4, pp. 385–403, 1993.

[180] L. A. Sanchis, "Test case construction for the vertex cover," in *Computational Support for Discrete Mathematics: DIMACS Workshop, March 12-14, 1992*, vol. 15, p. 315, American Mathematical Soc., 1994.

[181] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon, "Optimization by simulated annealing: an experimental evaluation; part i, graph partitioning," *Operations research*, vol. 37, no. 6, pp. 865–892, 1989.

[182] F. T. Leighton, "A graph coloring algorithm for large scheduling problems," *Journal of research of the national bureau of standards*, vol. 84, no. 6, pp. 489–506, 1979.

[183] D. L. Kreher and D. R. Stinson, *Combinatorial algorithms: generation, enumeration, and search*, vol. 7. CRC press, 1998.

[184] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc., 1990.

[185] K. Holmberg, M. Rönnqvist, and D. Yuan, "An exact algorithm for the capacitated facility location problems with single sourcing," *European Journal of Operational Research*, vol. 113, 1999.

[186] J. E. Beasley, "Or-library: Distributing test problems by electronic mail," *Journal of the Operational Research Society*, vol. 41, 11 1990.

[187] R. Kolisch, C. Schwindt, and A. Sprecher, "Benchmark instances for project scheduling problems," in *Project Scheduling: Recent Models, Algorithms and Applications* (J. Weglarz, ed.), pp. 197–212, Boston, MA: Springer US, 1999.

[188] D. Neuen and P. Schweitzer, "Benchmark graphs for practical graph isomorphism," *CoRR*, vol. abs/1705.03686, 2017.

[189] B. D. McKay and A. Piperno, "Practical graph isomorphism, {II}," *Journal of Symbolic Computation*, vol. 60, no. 0, pp. 94 – 112, 2014.

[190] G. Scheithauer, J. Terno, A. Müller, and G. Belov, "Solving one-dimensional cutting stock problems exactly with a cutting plane algorithm," *Journal of the Operational Research Society*, vol. 52, pp. 1390–1401, Dec 2001.

[191] G. Belov and G. Scheithauer, "A cutting plane algorithm for the one-dimensional cutting stock problem with multiple stock lengths," *European Journal of Operational Research*, vol. 141, 2002.

[192] G. Belov and G. Scheithauer, "A branch-and-cut-and-price algorithm for one-dimensional stock cutting and two-dimensional two-stage cutting," *European Journal of Operational Research*, vol. 171, no. 1, pp. 85 – 106, 2006.

[193] G. Belov and G. Scheithauer, "Setup and open-stacks minimization in one-dimensional stock cutting," *INFORMS J. on Computing*, vol. 19, pp. 27–35, Jan. 2007.

[194] G. Below and G. Scheithauer, "Lower-dimensional bounds and a new model for higher-dimensional orthogonal packing," 2008.

[195] G. Belov, H. Rohling, and G. Scheithauer, "Lp-based heuristics for conservative scales in orthogonal packing," *Preprint MATH-NM-06-2009, Technische Universität Dresden*, 2009.

[196] M. Mesyagutov, G. Scheithauer, and G. Belov, "Lp bounds in various constraint programming approaches for orthogonal packing," *Computers & Operations Research*, vol. 39, no. 10, pp. 2425 – 2438, 2012.

[197] G. Belov and H. Rohling, "Lp bounds in an interval-graph algorithm for orthogonal-packing feasibility," *Operations Research*, vol. 61, no. 2, pp. 483–497, 2013.