AN ASSOCIATION RULE MINING BASED OFFLINE SELECTION HYPER-HEURISTIC FOR SOLVING COMPUTATIONAL SEARCH PROBLEMS. A CASE STUDY ON TRAVELLING SALESMAN PROBLEM, BIN PACKING AND BOOLEAN SATISFACTION.

BY

OMONIYI, TOLULOPE MARVELOUS
(20CG028128)

A PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES, COLLEGE OF SCIENCE AND TECHNOLOGY, COVENANT UNIVERSITY, OTA, OGUN STATE.

IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE AWARD OF THE BACHELOR OF SCIENCE (HONOURS) DEGREE IN COMPUTER SCIENCE.

JULY, 2024

CERTIFICATION

I hereby certify that this project was carried out by Tolulope Marvelous OMONIYI in the Department of Computer and Information Sciences, College of Science and Technology, Covenant University, Ogun State, Nigeria, under my supervision.

Dr. Stephen A. Adubi
Supervisor

Signature and Date

Prof. Olufunke O. Oladipupo
Head of Department

Signature and Date

DEDICATION

I dedicate this work to God, who has been my very present help in times of need during my 4-year journey in this institution. I also dedicate this work to my friends and family, who support me.

ACKNOWLEDGEMENTS

I am incredibly grateful to God for His guidance throughout my studies. My sincere gratitude to my brother Tobi Omoniyi for his constant support. I express my gratitude to Mr. Mike Omoniyi and Mrs. Dupe Omoniyi for their steadfast support. My gratitude also extends to everyone who helped with this study, including my supervisor, Dr. Stephen Adubi.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ARM: Association Rule Mining

BP: Bin Packing

CSP: Computational Search Problem

SAT: Boolean Satisfaction

TSP: Travelling Salesman Problem

EHH: Expert Hyper Heuristic

SRHH: Simple Rule-Based Hyper-Heuristic

SSHH: Sequence Selection Hyper-Heuristic

ABSTRACT

The intricacy of the problems in the field of computational search has forced the creation of novel approaches to efficient problem-solving. Conventional methods frequently rely on manually constructed heuristics, which might not be adequate when issues get more complex. In order to solve this, adaptive and automated methods that seek to find optimal solutions more quickly—like randomized search heuristics—are becoming more and more popular. Though these heuristics have great potential, they lack thorough theoretical understanding, and the performance of existing hyper-heuristic algorithms suffers from generalization issues, which affects them in computational search problems (CSPs). This study offers a sophisticated method by combining hyper-heuristics with offline learning techniques. In particular, it investigates how Association Rule Mining (ARM) can be used to extract patterns and rules from effective CSP solutions and then apply these understandings to a more straightforward rule-based hyper-heuristic framework. By directly applying learnt rules from expert hyper-heuristics, this unique combination seeks to lower the learning curve and improve the performance of hyper-heuristics. The research entails creating and assessing an offline selection hyper-heuristic that extracts knowledge using ARM. Developing an expert hyper-heuristic, designing a basic rule-based hyper-heuristic, extracting heuristic selection rules via the Frequent Pattern Growth algorithm, and assessing performance with conventional metrics are among the main goals. The Boolean Satisfaction Problem, Bin Packing Problem, and Traveling Salesman Problem are examples of tests.

The results show that generalization and efficiency in CSPs can be greatly enhanced by combining ARM with hyper-heuristics. This methodology offers a viable way to develop problem-solving methodologies despite possible limits related to data availability and quality. It has significant implications for both academic research and industrial applications.

CHAPTER ONE
INTRODUCTION

1.1. Background Information

In the realm of computational search problems, researchers are constantly seeking innovative approaches to solve them efficiently and effectively. Traditional methods rely heavily on manually crafted heuristics or algorithms tailored for specific problem scenarios. However, as computational problems grow increasingly complex, there's a demand for more adaptive and automated techniques for choosing the best problem-solving strategies. Randomized search heuristics emerge as a promising solution, aiming to find good solutions quickly. While there's a wealth of experimental data on these heuristics, theoretical insights are limited. Hence, the adoption of adaptive and automated techniques, like randomized search heuristics, addresses the challenges posed by the escalating complexity of computational search problems.

In recent years, the utilization of Association Rule Mining (ARM) has gained substantial popularity in various domains. ARM involves the identification of frequent patterns in data, enabling informed decisions regarding the selection of appropriate search algorithms. Some recent applications of ARM include a study which applied ARM to explore the comorbidity of Attention-Deficit Hyperactivity Disorder (ADHD) in children using Korean National Health Insurance data. The study identified the most prevalent comorbid psychiatric disorder of ADHD youths and produced nine association rules using ARM (Kim & Myoung, 2018). ARM was used to investigate the causal relationship of design changes and error in new apartment housing projects. The study identified the associated relationship between design change factors that can significantly affect the productivity and performance of projects (Kim, Lee, and Kim 2022). ARM was also employed to analyze COVID-19 patient data in order to identify common patterns in symptoms. The study revealed various frequent symptoms and their associations, providing valuable insights for medical attention and management (Singh *et al*., 2022). A study presented a data mining technique that utilizes ARM analysis to optimize chiller systems. The ARM analysis identified operational parameters with strong association rules, which, when simulated, successfully saved energy consumption (Nisa, Kuan, & Lai 2021). ARM was applied to extract the rules for frequently occurring crash/near-crash events in

naturalistic driving studies. The analysis provided insights into the factors contributing to these events, such as road segments, driver behaviour, and environmental factors (Qu, Li, Liu, Pan & Zhang, 2022).

Historically, hyper-heuristics have been used to increase the generality of existing solvers, particularly in the context of combinatorial optimization problems (Gárate-Escamilla *et al*. 2022). However, the recent growth in popularity of hyper-heuristics has led to an increase in the amount of unstructured text in the related literature. Traditional systematic literature review studies have limitations, including high time consumption, lack of replicability, and subjectivity of the results. For this reason, text mining has become essential for researchers in recent years (Gárate-Escamilla, Amaya, Cruz-Duarte, Terashima-Marín & carlos Ortíz-Bayliss, 2022). A mini literature review shows that ARM has been used in various contexts, such as analyzing outbound tourism in Hong Kong (Oh, Chan, and Mehraliyev 2018) and e-commerce website usability analysis (Kumar *et al*. 2022). Another study introduced two algorithms to decrease side effects such as hiding failure, losing non-sensitive rules, making new rules, and hiding sensitive rules without any restriction in the number of items in the left and right hand (Ghalehsefidi & Dehkordi 2016). However, there is still a need for research on the use of ARM in offline selection hyper-heuristics for solving computational search problems. Online learning relies on data generated during the hyper-heuristic's execution on a single problem. It focuses on optimizing that specific problem. While offline learning, on the other hand, utilizes a pre-built database of heuristic selections and their corresponding results from running the hyper-heuristic on various benchmark problems (Yates & Keedwell 2019). This approach aims to learn generalizable rules that improve performance on unseen test problems. In simpler terms, online learning tailors itself to one problem at a time, while offline learning aims to learn broadly from multiple problems to tackle new ones effectively.

The significance of this study lies in the potential to optimize the search process and improve the efficiency of solving computational search problems. To achieve the goal for this study an expert hyper heuristic will be used to extract solution data on computational search problems specifically Travelling Salesman Problem(TSP), Bin Packing Problem, and Boolean Satisfiability(SAT), then an ARM technique will be used to identify frequent patterns in that data. These patterns will then be used to in our simple rule-based offline selection hyper-

heuristic. The Frequent Pattern(FP) Growth algorithm is a data mining technique that can be used to identify frequent patterns in data. It has been used in various contexts, such as e-commerce website usability analysis (Oh *et al*. 2018) and discovering demand and supply patterns based on Thai social media data (Tanantong & Ramjan 2021). The algorithm has also been used in privacy preserving data mining to decrease side effects such as hiding failure, losing non-sensitive rules, making new rules, and hiding sensitive rules without any restriction in the number of items in the left and right hand (Ghalehsefidi & Dehkordi 2016). The use of the Frequent Pattern Growth algorithm in this study is significant because it can help optimize the search process and improve the efficiency of solving computational search problems. By identifying frequent patterns in the data, the offline selection hyper-heuristic can make informed decisions a, leading to more efficient and effective solutions to computational search problems.

1.2. Statement of Problem

Existing hyper-heuristic algorithms struggle with generalization, affecting their performance in Computational Search Problems (CSPs). To enhance hyper-heuristic efficiency and overcome these generalization challenges, we propose integrating offline learning strategies. By utilizing Association Rule Mining (ARM) to extract rules and patterns from proven CSP solutions, we can incorporate these insights into the hyper-heuristic framework. This approach offers a significant advantage by reducing the algorithm's learning curve within the problem domain, enabling the direct application of rules from successful CSP solutions. The unexplored combination of offline learning with ARM presents a unique method to enhance hyper-heuristic performance by revealing hidden patterns in heuristic sequences.

1.3. Aim and Objectives of the Study

This research aims to design and evaluate an offline selection hyper-heuristic that leverages ARM to extract knowledge from expert hyper-heuristic performance on benchmark problems, ultimately integrating this knowledge into a simpler hyper-heuristic for improved problem-solving efficiency.

The research is guided by the following key objectives:

(i) To design and implement an expert hyper-heuristic that will be used to gather training instances.

(ii) To implement a simple rule-based offline selection hyper-heuristic that will leverage on mined rules from an existing expert hyper-heuristic.

(iii) To run the expert hyper-heuristic on test instances of some benchmark problems from the HyFlex(Hyper-heuristics Flexible framework) with the aim of extracting heuristic selection rules.

(iv) To extract hidden patterns from the training data gathered from the expert hyper-heuristic in objective three.

(v) To evaluate the performance of offline rule-based hyper-heuristic using standard evaluation metrics of hyper-heuristics.

1.4. Methodology

(i) Review and implement a modified SSHH algorithm, as detailed by Kheiri and Keedwell (2015), tailored to the specific needs of an expert hyper-heuristic, using Java within the HyFlex framework.

(ii) Develop and integrate a rule-based selection algorithm into a hyper-heuristic framework, defining how extracted rules will guide heuristic selection at each decision point to dynamically choose the most appropriate heuristics during the search process.

(iii) Execute the expert hyper-heuristic on Travelling Salesman Problem (TSP), Bin Packing Problem, and Boolean Satisfiability (SAT) within the HyFlex Java framework, gathering data on heuristic selections and performance outcomes for rule extraction.

(iv) Convert the data into a structured dataset, use the Frequent Pattern Growth algorithm to extract hidden patterns, and analyze these patterns to identify relevant rules for heuristic selection.

(v) Evaluate the rule-based offline selection hyper-heuristic using standard metrics, measuring performance in terms of solution quality and computational efficiency, and compare its performance against our Expert Hyper Heuristic.

Table 1.1 Objectives-Methodology Mapping Table

S/N	OBJECTIVES	METHODOLOGY
1.	To design and implement an expert hyper-heuristic that will be used to gather training instances	i. Review of relevant papers detailing a similar hyper-heuristic algorithm. ii. Use of a modification of the sequence-based selection hyper heuristic (SSHH) algorithm (Kheiri & Keedwell, 2015) to suit the specific requirements of the expert hyper-heuristic. iii. Implement the modified algorithm using Java within the HyFlex framework
2.	To implement a simple rule-based offline selection hyper-heuristic that will leverage on mined rules from an existing expert hyper-heuristic.	i. Develop a rule-based selection algorithm that uses the extracted rules to guide heuristic selection. ii. Define how the rules will be applied to choose the most appropriate heuristic at each decision point. iii. Integrate the rule-based selection algorithm into a hyper-heuristic framework. iv. Ensure that the hyper-heuristic can dynamically select heuristics based on the extracted rules during the search process.

3.	To run the expert hyper-heuristic on test instances of some benchmark problems from the HyFlex framework with the aim of extracting heuristic selection rules.	i. **Benchmark Problems:** Tackle the Travelling Salesman Problem (TSP), Bin Packing Problem, and Boolean Satisfiability (SAT). ii. Execute the expert hyper-heuristic on benchmark problems within the HyFlex Java framework. iii. Gather data on heuristic selections and performance outcomes for rule extraction.
4.	To extract hidden patterns from the training data gathered from the expert hyper-heuristic in objective three.	i. Convert the data into a structured format such as a dataset with rows and columns. ii. Utilize the Frequent Pattern Growth algorithm to extract hidden patterns from the converted data. iii. Analyze and identify relevant rules for heuristic selection from the extracted patterns.
5.	To evaluate the performance of offline rule-based hyper-heuristic using standard evaluation metrics of hyper-heuristics.	i. Evaluate the rule-based offline selection hyper-heuristic using standard hyper-heuristic evaluation metrics. ii. Measure performance in terms of solution quality and computational efficiency.

		iii. Compare the hyper-heuristic's performance against our Expert Hyper-Heuristic.

1.5. Significance of the Study

Hyper-heuristic implementers might find this project helpful for comparing own results with other approaches for solving CSPs. Software developers who make use of hyper heuristics for solving everyday real world problem will be able to used this approach of integrating ARM and offline learning in their hyper heuristic implementation to possible solve problem more efficiently and effectively.

1.6. Limitations of the Study

The major limitation of this study revolves around the availability and quality of data obtained from expert hyper-heuristics. This data serves as the foundation for the ARM technique to generate rules for the simple selection hyper-heuristic. If the data obtained is of low quality or insufficient in quantity, it can significantly impact the output of the simple selection hyper-heuristic. Therefore, the reliability and adequacy of the data sourced from expert hyper-heuristics play a crucial role in determining the effectiveness and accuracy of the ARM-generated rules, ultimately affecting the performance of the simple selection hyper-heuristic.

1.7. Chapter Outline

Chapter One: This chapter provides an overview of hyper-heuristics and association rule mining (ARM), exploring their potential to tackle complex computational search problems. The chapter also presents the problem statement, the specific objectives, the implementation method, and the scope and significance of this research. Chapter Two: Literature Review provides an in-depth exploration of existing literature on hyper-heuristics, association rule mining, and their applications in computational search problems. Chapter Three: Methodology outlines the research methodology, detailing the development of the proposed hyper-heuristic framework and the integration of association rule mining. Chapter Four: Research Design, Results and Discussion presents the experimental setup, datasets used, interprets the findings,

discusses implications, and the results of applying the developed approach to computational search problems. Chapter Five: Covers the summary, conclusion, and recommendation.

CHAPTER TWO
LITERATURE REVIEW

2.1. Introduction

The literature on hyper-heuristics, association rule mining (ARM), and their uses in computational search problems (CSPs) is examined in this chapter. It seeks to give readers a thorough grasp of the state-of-the-art in various fields, point out any research gaps, and emphasize how integrating them could improve readers' ability to solve problems.

2.2. Hyper-heuristics

In the realm of computational search problems, hyper-heuristics are a promising method because they offer a high-level framework for automatically choosing and using low-level heuristics to solve challenging optimization issues (Cruz-Reyes *et al*. 2012).

2.2.1. Definition and Key Concepts

The basic idea of hyper-heuristics is the design of a high-level heuristic that solves Computational Search Problems (CSPs) by automatically selecting and applying low-level heuristics. Hyper-heuristics are the family of meta-heuristic algorithms. Rather than focusing on specific problem instances as traditional heuristics do, hyper-heuristics work on an abstract level -- which means that the distributed selection of individual low-level (problem-dependent) heuristics should be done dynamically. Itis flexible and can be used for addressing solutions to many problems or situations.

Key concepts in hyper-heuristics include:

(i) Heuristic Pool: Hyper-heuristics maintain a diverse pool of low-level heuristics, each designed to tackle specific aspects of a problem. This pool includes a range of tactics or guidelines that can be used to alter a solution, offering adaptability in handling various CSP aspects (de Carvalho, Özcan, & Sichman 2021).

(ii) Selection Mechanisms: These are the algorithms or techniques used to choose the best-performing heuristic or its configuration from the heuristic pool based on the problem's characteristics and current solution quality (Pukhkaiev *et al*.

2020). This selection process is often driven by a strategy or algorithm that evaluates the current state of the problem and dynamically chooses the most suitable heuristic. Common selection criteria include the historical performance of heuristics, problem features, or a combination of both.

(iii) Adaptation Strategies: These are the methods used to adapt the selected heuristic or its parameters to the specific problem or situation, improving its performance (Pukhkaiev, Semendiak, Götz & Assmann, 2020). These strategies involve learning from past experiences and adjusting the heuristic pool or selection mechanisms accordingly. By adapting to the characteristics of the problem at hand, hyper-heuristics enhance their ability to find effective solutions across a variety of CSP instances.

Hyper-heuristics have a number of benefits, including the ability to reduce the need for laborious trial and error procedures and to simplify and improve algorithm configuration or selection for optimization problems (de Carvalho, Özcan & Sichman, 2021). To solve challenging optimization issues, they can be utilized in combination with a variety of low-level heuristics, including Multi-Objective Evolutionary Algorithms (MOEAs) (de Carvalho *et al*., 2021)

By automatically choosing the best-performing method and its configuration at runtime, hyper-heuristics combine the advantages of parameter control and hyper-heuristics to produce promising outcomes in tackling real-world multi-objective optimization problems (Pukhkaiev *et al*., 2020). Because it enables a more flexible and effective solution process, this method is especially helpful in situations where the best-performing heuristic is unknown.

2.2.2 Types of Hyper-heuristics

Selection Hyper-heuristics are a subclass of hyper-heuristics that focus on selecting the best heuristic from a predetermined set. This choice is based on an assessment of the situation as it is right now or is guided by prior knowledge. The principal aim of selection hyper-heuristics is to adjust dynamically to the attributes of the issue instance and optimize the entire problem-solving process's effectiveness. (Qian, Tang & Zhou, 2016).

Within the domain of hyper-heuristics, generation hyper-heuristics are unique in that they concentrate on constantly generating new heuristics. Generation hyper-heuristics use a more creative approach by actively creating new heuristics through combinations or modifications of existing ones, as opposed to only depending on a predetermined pool of preexisting heuristics. The main goal is to investigate and develop efficient problem-solving techniques that might not have been stated clearly in the first collection of heuristics (Burke *et al*., 2017;Ochoa, 2012).

Generation hyper-heuristics can be further divided into two subcategories (Adriaensen, Brys & Nowe, 2014a):

(i) Generation constructive hyper-heuristics: These hyper-heuristics generate new heuristics by constructing them from scratch, combining existing heuristics, or adapting existing heuristics to the specific problem at hand (Singh & Pillay 2023).

(ii) Generation perturbative hyper-heuristics: These hyper-heuristics generate new heuristics by perturbing or modifying existing heuristics, such as through parameter adjustments or minor changes to the heuristic algorithm (Singh & Pillay 2023).

Ant algorithms, for example, have been successfully employed in generation constructive and generation perturbative hyper-heuristics, showing potential in terms of generality, optimality, and computational effort (Singh & Pillay 2022). The performance of these hyper-heuristics can be further enhanced by analysing the impact of different parameters and pheromone map types on their effectiveness (Singh & Pillay 2022).

Learning hyper-heuristics form a specialized category within hyper-heuristics that incorporates machine learning techniques to dynamically adapt the heuristic selection or generation process over time. The integration of machine learning adds a layer of intelligence to hyper-heuristic frameworks, allowing them to learn from past experiences, optimize their decision-making processes, and improve performance based on feedback from problem-solving instances (Udomkasemsub, Sirinaovakul, & Achalakul 2023).

2.2.3. Advantages and Limitations of Existing Approaches

Hyper-heuristics, in contrast to conventional heuristics, provide: Decreased laboriousness: Hyper-heuristics remove the requirement for manual tuning by automating the heuristic selection procedure for each task. This lessens the amount of manual labor needed to resolve challenging optimization issues (Ochoa, 2012).

Enhanced adaptability: By choosing the right heuristics from a large pool, hyper-heuristics may handle a variety of CSPs with effectiveness. Because of their flexibility, hyper-heuristics can be used to tackle a variety of challenging optimization issues (Ochoa, 2012).

Enhanced learning potential: Some hyper-heuristics can learn from past experiences and adapt their strategies over time. This learning potential allows hyper-heuristics to optimize their decision-making processes and improve performance based on feedback from problem-solving instances (Burke *et al*. 2013).

However, limitations remain: Dependence of performance on heuristic pool: The quality and variety of heuristics that are available have a significant impact on how successful hyper-heuristics are. The limited or insufficient set of heuristics can impede hyper-heuristics' capacity to efficiently address intricate optimization issues (Burke *et al*., 2013).

Limited problem understanding: Most hyper-heuristics lack a deep understanding of specific CSPs, potentially hindering the selection of optimal heuristics. This limitation can lead to suboptimal performance, especially in problem domains where a profound understanding of the problem is crucial for selecting the most effective heuristics (Burke *et al*. 2013).

Lack of adaptation in non-stationary environments: Traditional hyper-heuristics may struggle to adapt to rapidly changing problem landscapes. In non-stationary environments, where the characteristics of the problem change over time, hyper-heuristics may not be able to dynamically adjust their strategies, leading to decreased performance and efficiency (Misir 2022).

2.2.4. Open Challenges and Research Gaps in Hyper-heuristics

Integrating domain knowledge into hyper-heuristics is a crucial open challenge. Hyper-heuristics are better able to comprehend the particular issue domains they are designed to tackle

when they include domain knowledge. In the end, this can enhance the process of addressing problems by resulting in better-informed and efficient heuristic selection or generation. Research in this field could concentrate on creating methods for integrating domain knowledge—such as information unique to an issue or expert heuristics—into hyper-heuristic frameworks in an efficient manner. This integration could be achieved through methods like knowledge representation, ontologies, or machine learning approaches that leverage domain-specific data to enhance problem understanding and optimize heuristic selection (Ochoa, 2012).

One important area of unmet research need in the realm of hyper-heuristics is the creation of more efficient learning mechanisms. Over time, hyper-heuristics can achieve ever-better results by augmenting their learning capacity. This entails integrating cutting-edge machine learning techniques, such reinforcement learning, to allow hyper-heuristics to adjust and refine their approaches in response to feedback from problem-solving instances and historical events. In order to enable hyper-heuristics to effectively generalize from prior problem instances and make defensible conclusions in novel and unseen problem contexts, research in this area could concentrate on developing robust learning processes. Furthermore, a crucial area for future research is examining how transfer learning and meta-learning strategies might support ongoing performance enhancement in hyper-heuristics (Udomkasemsub, Sirinaovakul & Achalakul, 2023).

The design of robust hyper-heuristics capable of effectively operating in dynamic and non-stationary environments is an open challenge. Traditional hyper-heuristics may struggle to adapt to rapidly changing problem landscapes, leading to decreased performance and efficiency. Addressing this challenge involves developing adaptive hyper-heuristic frameworks that can dynamically adjust their strategies in response to changes in the problem environment. This could be achieved through the integration of techniques such as online learning, self-adaptation, and dynamic algorithm configuration, enabling hyper-heuristics to maintain high performance in the face of evolving problem dynamics (Burke *et al*., 2017;Ochoa, 2012). Additionally, to develop strong frameworks for solving problems that can deal with non-stationary environments, researchers in this field could investigate the possibilities of hybrid approaches that blend hyper-heuristics with other adaptive optimization methods like swarm

intelligence or evolutionary algorithms (Udomkasemsub *et al*., 2023). Enhancing problem-solving skills through the synergistic combination of hyper-heuristics with other optimization approaches is a potential approach. Through the integration of distinct optimization methodologies, including machine learning algorithms, precise methods, and metaheuristics, scholars can create hybrid frameworks that exhibit enhanced resilience and performance. Examining this integration entails figuring out how to successfully combine several optimization strategies to address difficult optimization problems and examining how complementary they are (Misir, 2022).

This could include the development of novel hybrid algorithms, the identification of suitable problem domains for synergistic integration, and the empirical evaluation of hybrid frameworks across a diverse set of benchmark problems. Additionally, research in this area could focus on developing standardized frameworks for the synergistic integration of optimization techniques, enabling researchers and practitioners to systematically leverage the benefits of hybrid approaches in their problem-solving endeavours (Udomkasemsub *et al*. 2023).

2.3. Association Rule Mining (ARM)

Association rule mining is an active research area in data mining and has led to the development of improved algorithms to handle the increasing complexity and size of modern datasets (Abdel-Basset *et al*. 2018; Zhao *et al*. 2021).

2.3.1. Definition and Key Concepts

Association rule mining is a data mining technique used to discover interesting relationships and patterns hidden in large datasets. It involves identifying associations or correlations between items in a dataset. The most well-known algorithm for association rule mining is the Apriori algorithm, which is widely used for this purpose. The technique is applied in various fields to extract potentially useful information and knowledge from big datasets. It is particularly useful for market basket analysis, where the goal is to find associations between products purchased by customers.

Association rule mining involves several key concepts, including frequent itemsets, association rules, support, and confidence measures.

(i) Frequent Itemsets: These are sets of items that frequently appear together in a transactional dataset. The Apriori algorithm is commonly used to identify frequent itemsets by iteratively finding the sets of items that meet a minimum support threshold (Liu, Hsu, & Ma 1998).

(ii) Association Rules: Once frequent itemsets are identified, association rules are generated to represent the relationships between items. An association rule is typically in the form of "if {antecedent} then {consequent}," indicating a strong relationship between the items in the antecedent and consequent (Liu, Hsu & Ma, 1998).

(iii) Support: Support is a measure of how frequently an itemset or rule appears in the dataset. It is calculated as the proportion of transactions that contain the itemset or satisfy the rule. High support indicates that the itemset or rule is common in the dataset (Liu *et al*., 1998).

(iv) Confidence: Confidence is a measure of the reliability of the association rule. It is calculated as the proportion of transactions that contain the antecedent and also contain the consequent. High confidence indicates that the consequent is highly likely to appear in transactions that contain the antecedent (Liu *et al*., 1998).

2.3.2. Popular ARM Algorithms

There are several commonly used algorithms for Association Rule Mining (ARM), including Apriori, ECLAT (Equivalence Class Transformation), FP-Growth, PrefixSpan and CARMA (Classification and Regression on Association Rule Mining) just to name a few. Each algorithm with its strengths and weaknesses.

A. Apriori Algorithm

The Apriori algorithm is a foundational approach in association rule mining, designed to efficiently discover frequent itemsets and generate association rules based on these itemsets. It follows a systematic process outlined as follows: First, a minimum support threshold is defined to filter out insignificant patterns, setting the minimum number of times an itemset must appear in the dataset to be considered frequent. The algorithm then begins by identifying frequent 1-itemsets through scanning the entire dataset and counting the occurrence of each individual

item. Next, it employs a level-wise approach to generate candidate itemsets of increasing size, using a process of joining and pruning based on the Apriori property, which states that no superset of an infrequent itemset can be frequent (Toivonen 2017). This property helps to efficiently prune out potentially infrequent candidates. Once candidate itemsets are generated for a level, the algorithm scans the dataset again to count occurrences and identify frequent itemsets (Al-Maolegi & Arkok, 2014). This iterative process continues until no more frequent itemsets can be generated, at which point the algorithm proceeds to generate association rules based on the discovered frequent itemsets. These rules follow a specific format indicating antecedents and consequents with associated confidence levels. For instance, in a grocery transaction dataset, the algorithm might generate rules such as "if (Bread) then (Milk)" with a certain confidence level. While Apriori has limitations such as requiring multiple scans of the dataset and facing challenges with candidate explosion, it remains a fundamental concept in association rule mining due to its simplicity and ease of understanding.

Some recent areas this algorithm has been used include “Association Rule Generation for Student Performance Analysis using Apriori Algorithm” by Prasad (2022). The author of this paper believes that data mining techniques, specifically the Apriori algorithm, can be a valuable tool for educational institutions. Their goal is to use student data to predict academic performance and ultimately improve educational quality. The Apriori algorithm plays a central role in this approach. It analyzes a large dataset of student information, likely including factors like assignment completion, internal assessment scores, and attendance. By sifting through this data, Apriori identifies patterns and relationships between these factors. These patterns are then translated into rules. For instance, a rule might state that "if a student has a high attendance rate and consistently completes assignments, then they are likely to perform well on exams." The author aims to use these Apriori-generated rules to classify students based on their academic performance. By identifying students who are struggling (e.g., those with low attendance or incomplete assignments), the institution can intervene and provide targeted support to improve their performance. This can lead to better overall academic results for the institution. Additionally, the paper suggests using these student profiles to match them with suitable job opportunities after graduation. In essence, the Apriori algorithm acts as a tool for uncovering hidden patterns in student data. These patterns are then used to predict performance, personalize education, and potentially enhance student outcomes.

In another study by Santoso (2021). The Apriori algorithm was used to analyze sales patterns at Indomaret, a retail store in Indonesia. The aim of the research was to increase product sales by finding association rules from product purchase transactions. The Apriori algorithm was used to determine consumer buying patterns and then form a combination of items that are often sold to be used as sales packages or bundling. The algorithm was applied to the transaction data from Indomaret, and the results showed that customers often buy toothpaste and detergents that have met the minimum confidence value. This information was then used to improve sales strategies by providing insights into customer buying patterns and preferences. The paper also discussed the importance of data mining in the retail industry, as it helps to extract useful information from large datasets and provides insights for decision-making purposes. In summary, the Apriori algorithm was used in this paper to analyze sales patterns at Indomaret, with the aim of improving sales strategies by finding association rules from product purchase transactions. The results showed that customers often buy toothpaste and detergents, which can be used to create sales packages or bundling. The paper also highlighted the importance of data mining in the retail industry and the usefulness of the Apriori algorithm for analyzing customer buying patterns.

In another paper by Ilayaraja and Meyyappan (2013), the Apriori algorithm plays a crucial role in analyzing medical data to identify frequent diseases. The primary function of the Apriori algorithm in this context is to discover patterns and associations within a large dataset of medical records, specifically focusing on patient profiles, disease occurrences, and geographical locations. By applying the Apriori algorithm, the researchers aimed to extract valuable insights from healthcare data that are often underutilized. The algorithm was used to identify frequent diseases in specific geographical areas during given time periods, providing healthcare administrators with essential information to enhance decision-making processes and improve the quality of service. The Apriori algorithm was implemented in a step-by-step manner in the paper. Initially, the algorithm was used to find frequent itemsets in the medical data, such as diseases that occur frequently in patients. Through multiple passes over the dataset, the algorithm iteratively explored the relationships between different medical conditions and patient profiles. By employing a breadth-first search approach, the Apriori algorithm identified large itemsets that met the minimum support threshold, indicating the frequent occurrence of specific diseases. This process allowed the researchers to extract

meaningful patterns and associations from the medical data, enabling them to identify the most common diseases affecting patients in different geographical locations over specific time periods.

B. ECLAT (Equivalence Class Transformation) Algorithm

The ECLAT (Equivalence Class Transformation) algorithm is a powerful tool in association rule mining, providing an efficient alternative to traditional methods like Apriori, particularly for handling large datasets. Here's an analogy to understand ECLAT:

Imagine a library with books categorized by genre (e.g., Fiction, Science Fiction). ECLAT would first identify frequent genres (e.g., Fiction). Then, it would create separate sections within each genre for books with additional shared characteristics (e.g., "Fiction with Mystery" or "Science Fiction with Space Travel"). Finally, it would analyze these sections to identify relationships between genres and specific book titles. The ECLAT algorithm operates in several distinct steps, beginning with the definition of a minimum support threshold and the creation of Transaction ID Sets (TIDsets) for each item in the dataset. These TIDsets represent sets of transaction IDs where a specific item appears. ECLAT then proceeds to build initial Equivalence Classes, with each class representing a single item and its associated TIDset. Through an iterative process of Class Merging, equivalence classes are merged based on similarities in their TIDsets, effectively reducing redundancy and improving efficiency. The algorithm checks the resulting merged classes for frequent itemsets, with those meeting the minimum support threshold considered significant. Unlike Apriori, ECLAT employs a specific technique to generate candidate itemsets, reducing unnecessary candidate generation steps and enhancing scalability. Ultimately, ECLAT leverages its focus on TIDsets and class merging to streamline the association rule mining process, offering advantages such as reduced candidate generation and efficient handling of transaction-based data. However, it may pose challenges in terms of memory requirements and efficiency with very short frequent itemsets. Nonetheless, ECLAT stands as a valuable alternative approach for association rule mining, especially in scenarios involving large datasets where scalability and efficiency are paramount concerns.

Some recent study by Haikal, Chrisnanto, and Abdillah (2023). The ECLAT algorithm plays a crucial role in analyzing transactional data from Aufco Clothing to determine product layout strategies. The ECLAT algorithm is used to identify patterns in customer purchases and to generate association rules that can help optimize product displays for increased sales. Specifically, the ECLAT algorithm is applied to the transactional data, which includes variables such as order number and items sold. The data is processed using JavaScript with a minimum support of 0.2 and a minimum confidence of 0.7. This processing results in the generation of 16 rules with confidence levels ranging from 70% to 100%. These rules represent combinations of items that are frequently purchased together by customers. By utilizing the ECLAT algorithm, the study aims to analyze customer buying habits and preferences to create effective product layouts that can enhance customer satisfaction and purchase intent. The generated association rules provide insights into which products are commonly purchased together, enabling retailers to strategically place these items in proximity to each other to encourage additional sales.

In another paper by Devika, Koushik, and Subramaniyaswamy (2018). ECLAT is used in this paper to mine frequent item sets from the Twitter data, which are then compared to detect events. The ECLAT algorithm is applied in conjunction with TRCM to analyze the Twitter data and identify significant events. The study aims to improve the efficiency and runtime complexity of event detection compared to using the Apriori algorithm. By leveraging the ECLAT algorithm, the researchers were able to automate the process of identifying important events from Twitter data, particularly focusing on the sports domain. The results showed that ECLAT provided better runtime complexity and efficiency in detecting events compared to the Apriori algorithm. Overall, the ECLAT algorithm was instrumental in efficiently mining frequent item sets from Twitter data and detecting events in the sports domain, showcasing its effectiveness in event detection applications on social media platforms like Twitter.

C. FP-Growth (Frequent Pattern-growth) Algorithm

The FP-Growth (Frequent Pattern-growth) algorithm is a widely used and efficient method for association rule mining, offering improvements over traditional approaches like Apriori. It focuses on discovering frequent itemsets, which are groups of items that frequently appear together in a dataset, forming the basis for generating association rules. FP-Growth employs a

unique data structure called the FP-Tree to efficiently store information about these itemsets. This tree is built by scanning the data to identify frequent 1-itemsets and then transforming transactions into sorted lists based on item frequency. Subsequently, conditional FP-Trees are constructed to focus on finding frequent itemsets conditional on specific items. Using a divide-and-conquer strategy, FP-Growth utilizes recursion to efficiently mine frequent itemsets from the FP-Tree and its conditional FP-Trees. Finally, association rules are generated based on the discovered frequent itemsets, considering support and confidence levels. FP-Growth offers several advantages, including reduced data scans, avoidance of candidate generation, and efficient handling of large datasets. However, it may require higher memory usage and relies on a minimum support threshold, potentially overlooking less frequent but interesting patterns.

In a paper by Wahana, Maylawati, Irfan, and Effendy (2018), the FP-Growth algorithm plays a crucial role in optimizing the distribution of drugs from the Department of Health to Public Health Centers (Puskesmas) in Indonesia. The research utilizes the FP-Growth algorithm as a method for frequent pattern mining, which generates frequent item sets without the need to generate feature candidates. Unlike the Apriori algorithm that requires multiple scans of the database, FP-Growth only needs one database scan, making it more efficient. Firstly, the algorithm performs a database scan to identify the support for each item and eliminates non-frequent items. Subsequently, the frequent items are sorted based on their support, starting from the highest to the lowest. Transactions are then read and saved in an FP-Tree structure, where each node contains information about the item name, count, and node-link. The FP-Growth algorithm extracts frequent item sets from the FP-Tree by reading from leaves to the root in a bottom-up approach, following a divide and conquer strategy. Through the successful implementation of the FP-Growth algorithm, the research demonstrates the ability to generate frequent item sets of drugs that align with the needs of each Public Health Centers (Puskesmas) based on existing transaction data. The algorithm's consistency in producing frequent patterns, even with variations in input parameters such as minimum support and minimum length, showcases its effectiveness in optimizing the distribution of medicines. Overall, the FP-Growth algorithm serves as a powerful tool in enhancing the efficiency and accuracy of drug distribution in the supply chain management process outlined in the study.

In another paper by Dharmarajan, and Dorairangaswamy (2016). The FP-Growth algorithm is utilized to extract frequent access patterns from web log data, offering valuable insights into user interests and behaviour. By employing the FP-growth algorithm, researchers can identify common user behaviours, preferences, and trends from the vast amount of web access data available. This information can then be leveraged to enhance web design, personalize services, optimize browsing experiences, and even predict user behaviour for various applications. In this study, the FP-growth algorithm is implemented to navigate through the FP-Tree data structure in a bottom-up approach. By creating a compact representation of the database in the form of an FP-tree, the algorithm efficiently discovers frequent data sets. The process involves two main steps: first, constructing the FP-tree by making two passes over the dataset, and second, extracting frequent item sets by traversing the FP-tree. Through this method, the FP-growth algorithm effectively identifies patterns in web log data that reflect user behaviour and interests, ultimately contributing to a deeper understanding of user navigation patterns on websites.

In a paper by Siahaan, Ikhwan and Aryza (2018), the FP-Growth algorithm is used to mine frequent itemsets from a dataset related to student attributes such as last education, home address, department, and choice of program. By employing the FP-Growth algorithm, the researchers were able to generate rules that help in understanding student preferences and behaviours in selecting educational programs. This algorithm works by constructing an FP-Tree data structure, which aids in extracting frequent itemsets directly from the tree. The FP-Growth method involves three main stages: generating a conditional pattern base, building the FP-Tree, and searching for frequent itemsets. Through this process, the algorithm effectively identifies patterns and associations within the dataset, enabling educational institutions to tailor their promotional strategies based on student characteristics and preferences.

D. PrefixSpan Algorithm

PrefixSpan is a pattern-growth algorithm tailored for sequential pattern mining, distinguishing itself from Apriori and FP-Growth by focusing on frequent sequential patterns, where items appear in a specific order within sequences of data. It operates on sequence data, such as customer transactions or web browsing clickstreams, where the order of items matters. PrefixSpan identifies frequent sequential patterns known as prefixes, which are sub-sequences

that are common starting portions of longer sequences in the dataset (Pei *et al*. 2004). Using a divide-and-conquer strategy, the algorithm employs prefix projection to generate candidate extensions for each frequent prefix recursively. These extensions are validated against a minimum support threshold, and the process continues until no more frequent extensions can be found (Saraf, Sedamkar, & Rathi 2015). PrefixSpan is efficient for sequential data, avoids unnecessary processing of irrelevant item orderings, and offers flexible pattern representation for variable-length sequential patterns. However, its recursive nature can lead to overhead, especially for large datasets with long sequences, and it relies on a minimum support threshold, potentially overlooking less frequent yet interesting patterns.

The paper by Aloysius and Binu (2013) proposes a method for optimizing product placement in supermarkets using the PrefixSpan algorithm. The authors focus on understanding customer buying patterns, specifically the order in which customers purchase different product categories. PrefixSpan is ideally suited for this task because it's designed to identify frequent sequential patterns. By analyzing customer purchase data, PrefixSpan can uncover sequences of product categories that customers frequently buy together. The paper outlines a two-stage process that leverages PrefixSpan: In the first stage, PrefixSpan analyzes sequences of entire product categories. This helps identify the order in which customers tend to purchase different categories. For instance, PrefixSpan might reveal that many customers buy "Breakfast Cereals" followed by "Milk." Based on these frequent sequential patterns, the authors propose arranging product categories on shelves in a way that aligns with these buying tendencies. Once categories are positioned, the second stage focuses on individual products within each category. Here again, PrefixSpan is used to analyze purchase data, but this time looking at sequences of specific products within a category. This can reveal frequently purchased combinations of products within a category. For example, PrefixSpan might identify that "Corn Flakes" is often bought after "Milk" within the "Breakfast Cereals" category. This information can be used to arrange specific products within each category on the shelves, potentially influencing customers to purchase complementary items.

E. Carma (closed artual minimum antecedent) algorithm

CARMA (Classification and Regression on Association Rule Mining) is an innovative algorithm that builds upon association rule mining (ARM) techniques, such as Apriori, to

extend its functionality to classification and regression tasks. It leverages ARM concepts to discover closed itemsets, which are frequent itemsets with no superset sharing the same frequency, effectively reducing redundancy in patterns. Additionally, CARMA identifies minimal generators within closed itemsets, highlighting core elements contributing to the pattern. For classification, CARMA transforms the data into a transactional format, mines closed itemsets using techniques like the CHARM(Closed Association Rule Mining; the 'H' is gratuitous) algorithm, and generates classification rules in the format "if (antecedent) then (class label)" based on the minimal generators. These rules are then used to build a classification model capable of predicting class labels for new instances. Similarly, for regression tasks, CARMA generates rules to predict continuous values based on the antecedent features. While CARMA offers advantages such as reduced redundancy, improved classification/regression, and versatility in handling both tasks within a single framework, it may face challenges in complexity and scalability for exceptionally large datasets. An analogy illustrating CARMA's functionality in analyzing borrowing patterns in a library helps grasp its approach to identifying patterns and predicting outcomes based on core elements.

The paper by Hela, Amel, and Badran (2018) utilizes the CARMA (Classification and Regression on Association Rule Mining) algorithm, but not for its typical classification or regression purposes. The paper focuses on detecting potential anomalies (unusual events) in a smart home environment inhabited by elderly individuals. While traditional anomaly detection methods might simply identify deviations from normal behaviour, this paper suggests using CARMA to extract "causal association rules." By analyzing sensor data from the smart home, CARMA could help identify patterns that suggest potential causes for anomalies. Imagine a rule like "if (bathroom door remains closed for extended period) then (increased risk of fall)". This wouldn't directly classify an event as an anomaly, but it would highlight a situation (closed bathroom door) that might lead to an anomaly (fall). The authors mention that existing methods often focus solely on user activity for anomaly detection. CARMA, by analyzing sensor data, could help incorporate environmental factors into the process. For instance, a rule like "if (kitchen motion sensor inactive for extended period AND stove remains on) then (increased risk of fire)" could combine user activity (kitchen inactivity) with an environmental sensor (stove status) to identify potential hazards. Overall, while the paper doesn't use CARMA for traditional classification or regression, it appears to leverage CARMA's ability to identify

causal relationships within sensor data. This allows the system to not only detect anomalies but also potentially understand the underlying causes, leading to more targeted interventions and improved anomaly prevention in the smart home environment.

Advanced Association Rule Mining (ARM) techniques include sequential pattern mining and clustering-based approaches. These techniques aim to improve the efficiency and effectiveness of ARM algorithms in discovering hidden patterns and relationships in large datasets.

(i) Sequential Pattern Mining: This technique focuses on discovering patterns that occur in a specific order or sequence. It is particularly useful for analyzing time-series data or customer behaviour, where the order of events is important. Sequential pattern mining algorithms, such as GSP (Generalized Sequential Pattern mining) and PSP (PrefixSpan), can efficiently discover sequential patterns in large datasets (Saxena & Rajpoot 2021).

(ii) Clustering-based Approaches: Clustering-based approaches combine ARM with clustering techniques to identify groups of similar items or transactions. This can help to reduce the dimensionality of the dataset and improve the efficiency of ARM algorithms. Clustering-based approaches, such as DBSCAN (Density-Based Spatial Clustering of Applications with Noise) and HDBSCAN (Hierarchical DBSCAN), can be used in conjunction with ARM algorithms to improve the quality of the discovered patterns (Saxena & Rajpoot 2021).

These advanced ARM techniques offer several benefits, such as improved efficiency, better accuracy, and the ability to handle more complex datasets. However, they also come with their own challenges, such as increased computational complexity and the need for more sophisticated algorithms. The choice of technique depends on the specific requirements and constraints of the problem at hand.

2.3.3 Applications of ARM in Various Domains

Association rule mining (ARM) has found successful applications in diverse fields, including market basket analysis, fraud detection, and medical diagnosis.

(i) Market Basket Analysis: ARM has been widely used in market basket analysis to identify associations between products purchased by customers. By analyzing transactional data, retailers can discover patterns such as "if a customer buys product A, they are likely to buy product B," which can inform product placement, promotions, and inventory management (Han, Pei, & Yin 2000).

(ii) Fraud Detection: In the field of fraud detection, ARM has been applied to identify suspicious patterns and relationships in financial transactions. By analyzing large volumes of transaction data, ARM can help detect potentially fraudulent activities, such as unusual purchasing patterns or connections between seemingly unrelated entities (Han *et al*. 2000).

(iii) Medical Diagnosis: ARM has also been utilized in medical diagnosis to uncover associations between symptoms, diseases, and patient characteristics. By analyzing electronic health records and clinical data, ARM can assist in identifying potential risk factors, co-occurring conditions, and effective treatment pathways, contributing to improved patient care and healthcare management (Han *et al*. 2000).

These applications demonstrate the versatility and utility of ARM in extracting valuable insights and knowledge from large datasets across various domains. The technique's ability to reveal hidden patterns and relationships makes it a valuable tool for decision-making and problem-solving in diverse fields.

Association rule mining (ARM) offers several potential benefits for extracting actionable insights from data and improving decision-making processes. Some of these benefits include:

(i) Identifying Patterns and Relationships: ARM can uncover hidden patterns and relationships within large datasets, enabling organizations to gain a deeper understanding of customer behavior, market trends, or complex systems (Saxena & Rajpoot 2021).

(ii) Informing Strategic Decision-Making: By revealing associations between variables, ARM can provide valuable information for strategic decision-

making, such as product placement, targeted marketing, or resource allocation (Saxena & Rajpoot 2021).

(iii) Enhancing Predictive Analytics: The insights derived from ARM can be used to develop predictive models, allowing organizations to forecast future trends, customer preferences, or potential outcomes with greater accuracy (Saxena & Rajpoot 2021).

(iv) Improving Operational Efficiency: ARM can help organizations optimize processes and resource utilization by identifying efficient item combinations, process sequences, or system configurations (Saxena & Rajpoot 2021).

(v) Enabling Personalized Recommendations: In fields like e-commerce and content delivery, ARM can power personalized recommendation systems, enhancing customer experience and engagement (Saxena & Rajpoot 2021).

(vi) Supporting Fraud Detection and Risk Management: By identifying unusual patterns or associations, ARM can aid in fraud detection, risk assessment, and anomaly detection in various domains, including finance and cybersecurity (Saxena & Rajpoot 2021).

2.3.4 Limitations and Challenges of ARM in Problem-solving Contexts

Association rule mining (ARM) has some potential limitations that should be considered when applying the technique. Some of these limitations include:

(i) Data Quality Requirements: ARM requires high-quality data to produce meaningful results. Poor data quality, such as missing values or inconsistent data, can lead to inaccurate or irrelevant association rules (Ait-Mlouk, Agouti, & Gharnati 2017).

(ii) Scalability Issues: ARM can be computationally expensive, especially when dealing with large datasets. As the size of the dataset increases, the time and resources required to mine association rules also increase (Ait-Mlouk *et al*. 2017).

(iii) Need for Domain Expertise: Interpreting the results of ARM requires domain expertise to understand the context and relevance of the discovered association

rules. Without this expertise, the rules may be misinterpreted or misapplied (Kamsu-Foguem, Rigal, & Mauget 2013).

(iv) Limited to Finding Correlations: ARM can only identify correlations between variables and cannot establish causality. Therefore, the discovered association rules should be interpreted with caution and verified through further analysis (Ait-Mlouk *et al*. 2017).

(v) Limited to Binary Data: Traditional ARM algorithms are limited to binary data, where an item either exists or does not exist in a transaction. This can limit the applicability of ARM in domains where quantitative attributes are important (Ait-Mlouk *et al*. 2017).

2.4. Integration of Hyper-heuristics and ARM

Hyper-heuristics are thought as smart problem-solving tools. They do not focus on specific problems but instead help choose the right approach for any problem. Imagine having a toolbox with various tools. A hyper-heuristics is like a tool that picks the best tool from your toolbox for each job. They're useful when you face different problems and need a flexible solution.

Association Rule Mining (ARM) is like a detective looking for patterns in data. It finds connections between things. For example, in a supermarket, ARM can tell you which items are often bought together (like chips and soda). It's used in business, healthcare, and more to discover hidden relationships. In short, hyper-heuristics help you choose the right approach, while ARM finds hidden patterns in data.

2.4.1. Existing Research on Combining Hyper-heuristics and ARM

In a paper titled "Heuristics for interesting class association rule mining a colorectal cancer database" by José *et al*. (2020), a methodology is proposed to find "interesting" association rules in a colorectal cancer database. The methodology includes four heuristic operators:

(i) Class-relevance filtering (CRF): This prioritizes rules strongly associated with the target class, like complications or recurrences, filtering out irrelevant ones.

(ii) Confidence thresholding (CT): It selects rules with confidence levels above a predetermined threshold to ensure reliability.

(iii) Support thresholding (ST): This focuses on rules with enough support to avoid overfitting and ensure applicability.

(iv) Interestingness heuristic (IH): It introduces a new measure considering both support and confidence, penalizing redundant rules.

Applying these operators improved the quality of discovered rules, reducing uninteresting ones, and making the discovered rules more relevant, confident, and applicable. This methodology shows promise in extracting valuable knowledge from colorectal cancer data, potentially aiding in diagnosis, prognosis, and treatment decisions.

2.4.2. Potential Benefits and Challenges of this Integration

From the paper we see that integrating hyper-heuristics (high-level strategies for selecting or generating low-level heuristics) with ARM (association rule mining) could offer Improved heuristic selection: Hyper-heuristics can dynamically choose or adapt low-level heuristics like CRF, CT, and ST based on the specific data and problem characteristics. This could lead to a more efficient and effective search for interesting rules, avoiding pitfalls of static heuristic settings. Hyper-heuristics can learn from previous problem-solving experiences to automatically adjust rule evaluation criteria (like IH) and potentially discover new, domain-specific heuristics. This could automate knowledge extraction from various data sources without manual intervention. By learning and adapting, hyper-heuristics could make ARM more flexible and applicable to different domains beyond colorectal cancer analysis. This would broaden the potential applications of ARM in various fields.

Some potential challenges faced by similar data mining techniques like association rule mining (ARM): Processing large volumes of text data can be computationally expensive, especially for advanced analysis techniques like frequent association rule learning. Scalable algorithms and efficient implementations are crucial for practical application. Complex patterns extracted from text data require clear interpretation and translation into actionable insights for researchers and practitioners. Developing user-friendly visualization and explanation tools is essential for bridging the gap between data mining results and their real-world application.

2.5. Review of Related Works

This section contains paper closely related to my study.

2.5.1. An analysis of heuristic subsequences for offline hyper-heuristic learning

This paper by Yates and Keedwell (2019) explores the analysis of heuristic subsequences to enhance offline hyper-heuristic learning, focusing on automated methods for selecting or configuring lower-level heuristics to solve optimization problems. By introducing the concept of heuristic subsequences, which are ordered sequences of heuristic selections, the study aims to identify effective patterns for successful problem-solving. Utilizing a selection hyper-heuristic on benchmark problems, the authors create a database of heuristic subsequences and differentiate between effective and disruptive subsequences using logarithmic return. Effective subsequences are then used to parameterize a new hyper-heuristic, showcasing significant improvements in solution quality across problem domains. The approach shows promise in identifying transferable patterns and enhancing hyper-heuristic designs, although further research on generalizability is needed.

2.5.2. Offline Learning for Selection Hyper-heuristics with Elman Networks

The paper by Yates and Keedwell (2018) explores the use of Elman networks for offline learning in selection hyper-heuristics. The study aims to assess the network's ability to generalize within and across problem domains by analyzing sequences of heuristic selections. In their methodology, the authors detail the creation of an offline learning database, the architecture of Elman networks, and the selection of training sets. They evaluate the trained networks using HyFlex benchmark problems like 1D bin packing, permutation flow shop, and boolean satisfiability to gauge performance across diverse domains. Results show that the Elman network exhibits strong intra-domain generalization but faces challenges with inter-domain generalization. While it surpasses the performance of trained sequences within the same domain, transferring knowledge across different domains proves difficult. These findings have implications for designing customized algorithms tailored to specific domains and highlight the importance of domain-specific learning strategies. The study suggests further research to explore alternative methodologies for enhancing inter-domain generalization, such as identifying domains with similar sequences. Overall, the research contributes valuable insights into utilizing Elman networks for offline learning in selection hyper-heuristics, advancing machine learning techniques for complex optimization problems.

2.5.3. Offline learning with a Selection Hyper-Heuristic: Using the Baum–Welch learning algorithm

This paper by Yates and Keedwell (2021) explores the optimization of water distribution networks using a sequence-based selection hyper-heuristic (SSHH) with both online and offline learning capabilities. The study compares the performance of the SSHH hyper-heuristic with five multi objective evolutionary algorithms (MOEAs) across 12 water distribution network (WDN) problems of varying sizes. The SSHH hyper-heuristic employs online learning and can be trained offline using the Baum–Welch learning algorithm and a set of heuristic subsequences selected from an offline learning database. Through statistical methodology, the offline learning process analyzes results to improve the overall learning strategy. The paper also investigates scalable learning, where knowledge from solving small WDN problems is transferred to larger ones. The experimental setup includes selecting WDN problems, defining performance metrics, and establishing comparison criteria. Results show that the hyper-heuristic with online learning is competitive with state-of-the-art MOEAs across the 12 WDN problems. Offline learning enhances the hyper-heuristic's performance, highlighting improved optimization strategies. The study underscores the dynamic nature of heuristic effectiveness during WDN optimization and the transferability of knowledge across problem sizes. Conclusively, the paper suggests that offline learning significantly improves hyper-heuristic performance in WDN optimization, enabling knowledge transfer from small to large problems and paving the way for efficient optimization algorithms applicable to real-world engineering problems.

2.6. Summary and Conclusion

The literature review suggests that combining two methods, Association Rule Mining (ARM) and hyper-heuristics, could help solve computational problems better. One study looked at using special techniques to find important patterns in cancer data, showing promising results. These special techniques prioritized relevant rules, selected based on confidence and support thresholds, and introduced a measure of interestingness. Results showed significant improvements in discovering relevant and clinically significant rules. Combining ARM with hyper-heuristics could make it easier to pick the best strategies for solving problems and find useful information automatically. But there are challenges, like it being complicated and costly.

Future research could focus on making better methods and understanding how these methods can be used in different situations. Overall, this project aims to use insights from these studies to create smarter problem-solving tools that can adapt to different problems and find solutions more effectively.

CHAPTER THREE

SYSTEM ANALYSIS AND DESIGN

3.1. Introduction

This chapter presents a detailed description of the analysis and design of the proposed SSHH algorithm. It presents the Benchmark problems that we will be focusing on and the Hyflex framework being used to access and test those problems, the metrics used to evaluate the algorithm and the ARM techniques used to scale the expert hyper heuristics.

3.2. Supporting Libraries

This section introduces two algorithm development libraries: HyFlex and Mlxtend. HyFlex is a cross-domain hyper-heuristic development library written in Java language and supports cross-platform development. Mlxtend is a Python library that provides a wide range of tools and algorithms for data science and machine learning tasks, including the implementation of the Frequent Pattern Growth (FP-Growth) algorithm.

3.2.1. HyFlex Library

The HyFlex (Hyper-heuristics Flexible framework) library is a widely used platform for developing and evaluating hyper-heuristics for solving complex computational search problems (CSPs). It provides a standardized interface for implementing and comparing different hyper-heuristic approaches across various problem domains. HyFlex was initially introduced by (Ochoa, 2012) to facilitate the design and testing of hyper-heuristics. It consists of a set of problem domains, each with a set of low-level heuristics and problem instances. The framework abstracts away the details of the underlying problem, including solution representations, construction methods, and low-level heuristic implementations (Yates & Keedwell, 2018). Each problem within HyFlex is categorized into four broad heuristic classes as seen in Figure 3.1: 1. Parameterized Mutation (M), which randomly perturbs a solution; 2. Crossover (C), which creates a new solution by combining two or more existing solutions; 3. Parameterized Ruin and Recreate (R), which partially destroys and then rebuilds parts of a solution; and 4. Parameterized Hill Climbing or Local Search (L),

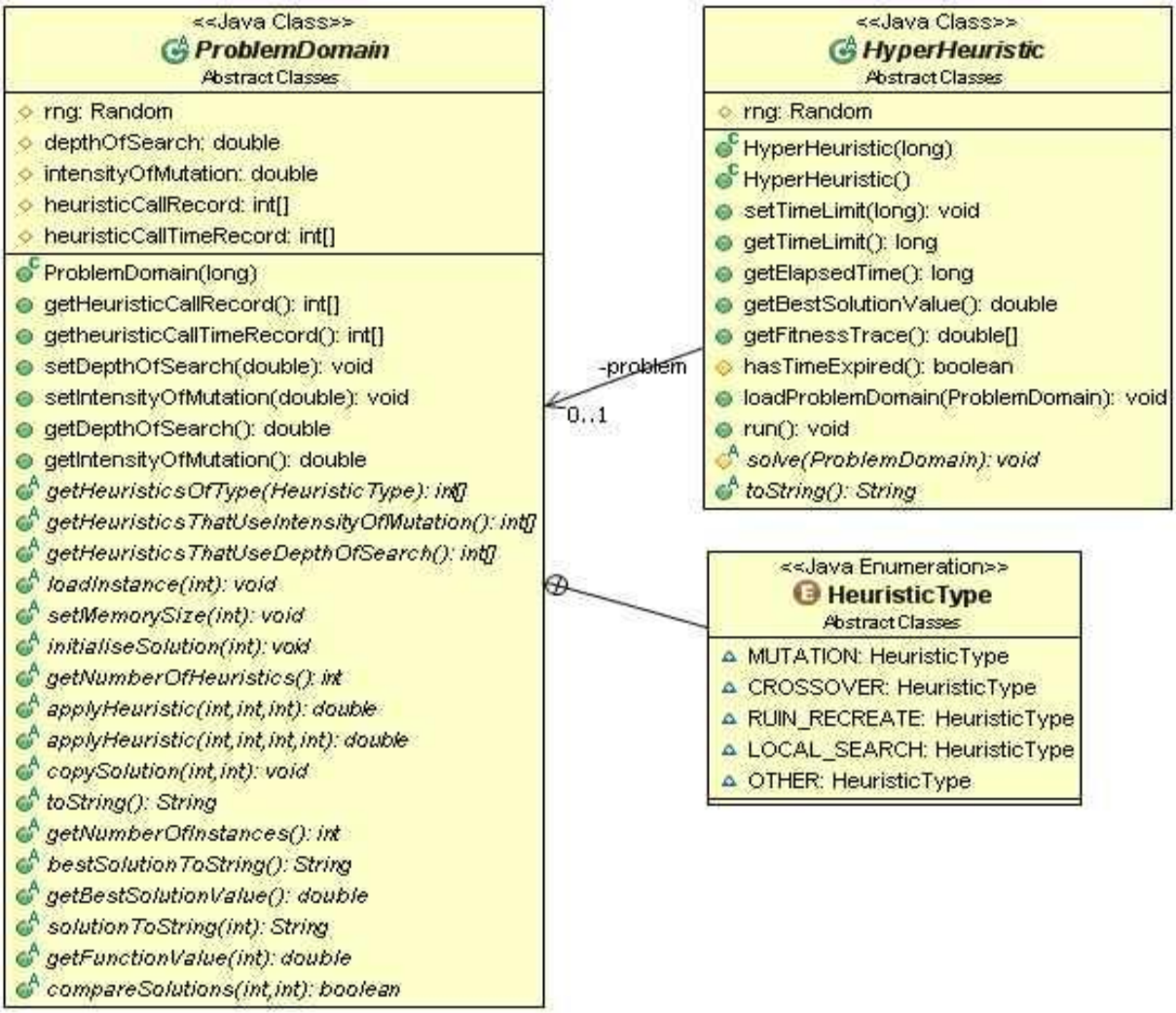

Figure 3.1 Class diagram for the HyFlex framework from Burke, Curtois, Hyde, Ochoa & Vázquez Rodríguez (2011)

which iteratively improves a solution and ensures it does not worsen. The number and specific implementations of these low-level heuristics vary across different problem domains (Yates & Keedwell, 2018).

One of the major advantages of HyFlex is its flexibility. It supports multiple problem domains, including Bin Packing, Boolean Satisfiability (SAT), Permutation Flow Shop, Traveling Salesman Problem (TSP), Personnel Scheduling, and Vehicle Routing Problem. This flexibility allows researchers to test their hyper-heuristics on a wide range of problems, ensuring their robustness and generality.

Another benefit of HyFlex is its modular design. The framework is divided into four main components: the problem domain, the solution representation, the low-level heuristics, and the

hyper-heuristic. This modular approach allows for easy integration of new problem domains, heuristics, and hyper-heuristics, making it a versatile tool for researchers and developers. The HyFlex library also provides a set of standard evaluation metrics, such as solution quality, convergence speed, and computational efficiency. These metrics help researchers assess the performance of their hyper-heuristics and compare them with other approaches. Additionally, HyFlex supports the use of different search strategies, such as iterative improvement and population-based methods, further enhancing its flexibility.

In summary, the HyFlex library is a powerful tool for developing and evaluating hyper-heuristics for solving complex computational search problems. Its flexibility, modularity, and standardized evaluation metrics make it an attractive choice for researchers and developers working in this field. By providing a common platform for testing and comparing hyper-heuristics, HyFlex has contributed to the advancement of hyper-heuristic research and its practical applications.

3.2.2. Mlxend Library

The Mlxtend (Machine Learning Extensions) library is a Python library that provides a wide range of tools and algorithms for data science and machine learning tasks, including the implementation of the Frequent Pattern Growth (FP-Growth) algorithm. The FP-Growth algorithm is a popular association rule mining technique used to efficiently discover frequent itemsets from large datasets. This library provides an implementation of the FP-Growth algorithm through the `apriori` and `association_rules` functions allowing users to easily perform association rule mining on their datasets.

The Mlxtend library's FP-Growth implementation is highly optimized and efficient, allowing for fast processing of large datasets. This efficient implementation is a significant advantage, as it enables researchers and developers to quickly and effectively perform association rule mining on their datasets. Additionally, the library provides a user-friendly API that makes it easy to perform association rule mining with minimal code, making it accessible to users of all skill levels. The flexibility of the library is another key advantage, as it allows users to customize the parameters of the FP-Growth algorithm, such as the minimum support and minimum confidence, to suit their specific needs. This flexibility is particularly useful in situations where the user needs to adapt the algorithm to a specific problem or dataset.

Furthermore, Mlxtend is designed to work seamlessly with other popular Python data science libraries, such as Pandas and Scikit-learn, making it easy to integrate into existing workflows. By leveraging the Mlxtend library's FP-Growth implementation, researchers and developers can efficiently perform association rule mining on their datasets, gaining valuable insights and patterns that can inform their decision-making processes.

3.2.3. Panda Library

This is an open-source software library built on top of Python specifically for data manipulation and analysis, Pandas offers data structure and operations for powerful, flexible, and easy-to-use data analysis and manipulation. In this project panda was used in the creating the rule extraction code, it specific functions were to read data from CSV files into a DataFrame, making it easy to handle and manipulate the data. Pandas was also used to handle missing data efficiently and to create a new DataFrame from the one-hot encoded array. This DataFrame is then used for the FP-Growth algorithm.

3.3. Benchmark Problems

The proposed offline selection hyper-heuristic is designed to be evaluated on a set of benchmark problems from the HyFlex framework. This section provides a detailed description of the chosen problems, their characteristics, and the specific instances used for evaluation.

3.3.1. Travelling Salesman Problem (TSP)

The Travelling Salesman Problem (TSP) is a classic problem in the field of computational search problems. It involves finding the shortest possible tour that visits a set of cities and returns to the starting point. The problem is NP-hard, meaning that the running time of traditional algorithms increases exponentially with the size of the input (Goyal, 2010).

Some instances of TSP are seen in Table 3.1.

Scenario: Imagine a salesman who needs to visit a set of cities to deliver goods. He wants to find the most efficient route that minimizes the total distance traveled. The salesman has a list of cities with their coordinates and needs to find the shortest possible tour that visits each city exactly once and returns to the starting point.

The Travelling Salesman Problem (TSP) challenges traditional optimization methods in several ways:

(i) NP-hardness: TSP is an NP-hard problem, meaning that the running time of traditional algorithms increases exponentially with the size of the input. This makes it difficult to find an optimal solution in a reasonable amount of time.

(ii) Local optima: TSP has many local optima, which are suboptimal solutions that are difficult to escape. Traditional optimization methods often get stuck in these local optima, making it challenging to find the global optimum.

(iii) Non-linearity: TSP is a non-linear problem, meaning that the objective function is not linear. This non-linearity makes it difficult to apply traditional optimization methods that rely on linear approximations.

Table 3.1: TSP Problem Instances and Characteristics

Problem	Instances	Characteristics
Travelling Salesman Problem (TSP)	Symmetric TSP: Distances between cities are the same in both directions.	Number of Cities (N): The total number of cities that need to be visited.
	Asymmetric TSP: Distances between cities can differ depending on the direction of travel.	Distance Matrix: A matrix representing the distances or travel costs between each pair of cities.
	Euclidean TSP: Cities are points in a plane, and distances are Euclidean(straight-line distances).	Optimality: The goal is to find the shortest possible route that visits each city exactly once and returns to the starting city.
	Generalized TSP: Each city belongs to a group, and the goal is to visit exactly one city from each group.	Computational Complexity: As N increases, the problem becomes exponentially harder to solve.

The Traveling Salesman Problem (TSP) is a classic problem in combinatorial optimization, which involves finding the shortest possible tour that visits a set of cities and returns to the starting point. This problem is widely used to model various real-world scenarios, such as logistics, transportation, and telecommunications.

The TSP can be formally defined as follows:

Problem Statement:

Given a set of n cities, find the shortest possible tour that visits each city exactly once and returns to the starting point.

Mathematical Formulation:

Let V = {1, 2, . . ., n} be the set of cities. The TSP can be formulated as an integer linear programming (ILP) problem (Velednitsky, 2017):

$$\boldsymbol{min} \sum_{\boldsymbol{i-1}}^{\boldsymbol{n-1}} \sum_{\boldsymbol{j=i+1}}^{\boldsymbol{n}} \boldsymbol{c_{ij} x_{ij}} \qquad (3.\mathbf{1})$$

subject to:

1. Flow Constraints:

$$\sum_{j=1}^{n} x_{ij} = 1 \ \ for\ all\ i \in V \qquad (3.2)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \ \ for\ all\ j \in V \qquad (3.3)$$

2. Subtour Elimination Constraints:

$$\sum_{i-1}^{n} \sum_{j=i+1}^{n} x_{ij} \ \leq \mathrm{n} - 1 \qquad (3.4)$$

3. Non-Negativity Constraints:

$$x_{ij} \geq 0 \ \ for\ all\ i,j \in V$$

4. Binary Constraints:

$$x_{ij} \in \{0,1\} \ \ for\ all\ i,j \in V$$

where c_{ij} is the cost of traveling from city i to city j, and $x_{ij}$is a binary variable indicating whether the edge (i, j) is part of the tour.

The Traveling Salesman Problem (TSP) is a fundamental issue in combinatorial optimization, with numerous applications across various fields. To solve the TSP, several methods have been

developed. One approach is the brute force algorithm, which involves generating all possible tours and selecting the shortest one. However, this method is impractical for large instances due to its exponential time complexity (Wilfahrt & Kim, 2017). Heuristics offer another solution by providing approximate algorithms that do not guarantee the optimal solution but deliver good solutions in a reasonable time. Examples of heuristics include the nearest neighbour algorithm, the 2-opt algorithm, and the Christofides algorithm. Metaheuristics, which are high-level algorithms that use heuristics as building blocks, also play a crucial role in solving the TSP. Notable examples of metaheuristics are the Ant Colony Optimization (ACO) algorithm, the Tabu Search (TS) algorithm, and the Artificial Bee Colony (ABC) algorithm (Dewantoro *et al*., 2019). Exact methods involve solving the Integer Linear Programming (ILP) formulation using specialized algorithms such as branch and bound, cutting plane methods, and branch and cut methods (Cook *et al*., 2007).

The TSP finds applications in various fields, including logistics and transportation, where it models the delivery of goods, vehicle routing, and flight scheduling. In telecommunications, the TSP is used to design communication networks and optimize node placement. Additionally, in Very-Large-Scale Integration (VLSI) design, the TSP helps optimize the placement of components on a chip. There are several variations of the TSP, including the Asymmetric TSP (ATSP), where the cost of traveling from city i to city j is not necessarily equal to the cost of traveling from city j to city i. The Metric TSP, another variation, requires that the cost of traveling from city i to city j satisfies the triangle inequality. The TSP with Time Windows adds another layer of complexity by introducing time windows during which each city must be visited.

In conclusion, the Traveling Salesman Problem is a challenging and extensively studied problem that necessitates the development of efficient algorithms and heuristics for its solution. Its importance is underscored by its wide range of applications and the various methods devised to tackle it.

3.3.2. Bin Packing Problem

The Bin Packing Problem is another well-known problem in the field of computational search problems. It involves packing a set of items of different sizes into a set of bins of fixed capacity. The goal is to minimize the number of bins used.

Scenario: Imagine a warehouse manager who needs to pack a set of items of different sizes into bins of fixed capacity. The manager wants to minimize the number of bins used to pack all the items. The items have different sizes, and the bins have a fixed capacity.

The Bin Packing Problem has several key characteristics that make it challenging to solve:

(i) Non-uniform item sizes: The items to be packed have different sizes, which makes it difficult to determine the optimal packing strategy.

(ii) Fixed bin capacity: The bins have a fixed capacity, which means that the items must be packed efficiently to minimize the number of bins used.

(iii) No empty space allowed: The items must be packed tightly, with no empty space allowed in the bins. This makes it challenging to find a packing strategy that minimizes the number of bins used.

Table 3.2: Bin Packing Problem Instances and Characteristics

Problem	Instances	Characteristics
Bin Packing Problem	1D Bin Packing: Items of different sizes must be packed into bins of fixed capacity in one dimension.	Number of Items (n): Total number of items to be packed.
	2D/3D Bin Packing: Items have two or three dimensions, adding complexity to how they can be packed.	Bin Capacity (C): Maximum capacity of each bin.
	Online Bin Packing: Items arrive sequentially, and decisions must be made without knowledge of future items.	Item Sizes: Sizes of the items to be packed, which can be integers or real numbers.
	Offline Bin Packing: All items are known in advance.	Optimality: The goal is to minimize the number of bins used.
		Heuristics and Algorithms: Various strategies like First Fit, Best Fit, and Worst Fit are used to approximate solutions.

The Bin Packing Problem (BPP) is a classic problem in combinatorial optimization that involves packing items of different sizes into a minimum number of bins of equal capacity. This problem is widely used to model various real-world scenarios, such as logistics, storage, and resource allocation.

The BPP can be formally defined as follows:

Problem Statement:

Given a set of n items of different sizes, and a set of m bins of equal capacity, find the minimum number of bins required to pack all items such that the total size of items in each bin does not

exceed the bin capacity.

Mathematical Formulation:

Let I = {1, 2, . . ., n} be the set of items, and B = {1, 2, . . ., m} be the set of bins. The BPP can be formulated as an integer linear programming (ILP) problem(Borges, Schouery & Miyazawa, 2023):

$$min \sum_{i-1}^{m} x_i \tag{3.5}$$

subject to:

1. Item Constraints:

$$\sum_{i=1}^{m} y_{ij} = 1 \ \ for\ all\ \ \in I \tag{3.6}$$

2. Bin Constraints:

$$\sum_{j=1}^{n} y_{ij} \leq c \ \ for\ all\ \ \in B \tag{3.7}$$

3. Non-Negativity Constraints:

$$x_i \geq 0 \ \ for\ all\ i \in B$$

4. Binary Constraints:

$$y_{ij} \in \{0,1\} \ \ for\ all\ i \in B, j \in I$$

where x_i is the number of bins used, y_{ij} is a binary variable indicating whether item *j* is packed in bin *i*, and *c* is the capacity of each bin.

The Bin Packing Problem (BPP) is a fundamental issue in combinatorial optimization with numerous applications across various fields. Several methods have been developed to solve the BPP, including heuristics, metaheuristics, and exact methods. Heuristics are approximate algorithms that provide good solutions in a reasonable time without guaranteeing optimality. Examples include the First-Fit Decreasing (FFD) algorithm, the Best-Fit Decreasing (BFD) algorithm, and the Next-Fit (NF) algorithm(Munien & Absalom, 2021). Metaheuristics, which

use heuristics as building blocks, include the Simulated Annealing (SA) algorithm, the Genetic Algorithm (GA), and the Ant Colony Optimization (ACO) algorithm(Munien & Absalom, 2021). Exact methods involve solving the Integer Linear Programming (ILP) formulation using specialized algorithms such as branch and bound, cutting plane methods, and branch and cut methods (Borges *et al*., 2023).

The BPP has applications in logistics and transportation, where it models the packing of goods in containers, routing of vehicles, and scheduling of flights. In supply chain management, it optimizes the storage and transportation of goods. In cutting industries, it helps in optimizing the cutting of materials like sheet metals and paper rolls. The BPP also has several variations, including the One-Dimensional BPP (1D-BPP), Two-Dimensional BPP (2D-BPP), and Three-Dimensional BPP (3D-BPP), corresponding to the dimensionality of the items and bins involved. Other variations include the Unbounded BPP, where item size is not restricted, and the Online BPP, where items are packed as they arrive.

To solve the BPP, several algorithms are used. The Greedy Algorithm packs items starting from the largest but does not guarantee optimality. The First-Fit and Best-Fit algorithms are more efficient but still do not guarantee optimal solutions. The Dynamic Programming Algorithm can provide optimal solutions but is computationally expensive for large instances.

In conclusion, the Bin Packing Problem presents a significant challenge in combinatorial optimization, necessitating the development of efficient algorithms and heuristics. Its extensive study has led to the creation of various methods tailored to its solution, highlighting its critical importance in multiple practical applications.

3.3.3. Boolean Satisfiability

The Boolean Satisfiability Problem (SAT) is a problem in the field of computational search problems that involves determining whether a given Boolean formula is satisfiable. A Boolean formula is a set of Boolean variables and their logical operations.

Scenario: Imagine a software engineer who needs to verify whether a given Boolean formula is satisfiable. The engineer wants to determine whether there exists an assignment of values to the Boolean variables that makes the formula true.

The Boolean Satisfiability Problem (SAT) differs from other computational search problems

in several ways:

(i) Boolean variables: SAT involves Boolean variables, which are variables that can take on only two values: true or false. This makes it different from other search problems that involve real-valued variables.

(ii) Logical operations: SAT involves logical operations such as AND, OR, and NOT, which makes it different from other search problems that involve arithmetic operations.

(iii) Satisfiability: SAT is a satisfiability problem, meaning that the goal is to find an assignment of values to the Boolean variables that makes the formula true. This makes it different from other search problems that involve optimization or minimization.

Table 3.3: Boolean Satisfiability Problem (SAT) Problem Instances and Characteristics

Problem	Instances	Characteristics
Boolean Satisfiability (SAT)	3-SAT: Each clause in the formula has exactly three literals.	Number of Variables (V): Total number of Boolean variables in the formula.
	k-SAT: Each clause has k literals (a generalization of 3-SAT).	Number of Clauses (C): Total number of clauses in the formula.
	CNF-SAT (Conjunctive Normal Form): The formula is expressed as a conjunction of disjunctions of literals.	Clause Length: Number of literals in each clause.
	MAX-SAT: The goal is to maximize the number of satisfiable clauses in the formula.	Satisfiability: The goal is to determine if there exists an assignment of truth values to variables that satisfies the entire formula.
		Complexity: SAT is NP-complete, and specific instances vary widely in difficulty.

The Boolean Satisfiability Problem (SAT) is a fundamental problem in computer science and mathematical logic that involves determining whether a given Boolean formula is satisfiable, i.e., whether there exists an assignment of values to its variables that makes the formula true. This problem is widely used to model various real-world scenarios, such as logic circuits, constraint satisfaction, and cryptography.

The SAT problem can be formally defined as follows:

Problem Statement:

Given a Boolean formula ϕ in conjunctive normal form (CNF), determine whether there exists an assignment of values to its variables that makes ϕ true.

Mathematical Formulation:

Let $\phi = \bigwedge_{i=1}^{m} C_i$ be a Boolean formula in CNF, where $\bigvee_{j=1}^{n_i} l_{ij}$ is a clause and l_{ij} is a literal (either a variable or its negation). The SAT problem can be formulated as :

$$\phi(x) = \bigwedge_{i=1}^{m} \bigvee_{j=1}^{n_i} l_{ij}(\mathrm{x}) = 1 \qquad (3.8)$$

where x = $(x_1, x_2, \ldots, x_n)$ is an assignment of values to the variables of ϕ(Pichugina & Matsyi, 2020).

The Boolean Satisfiability Problem (SAT) is a fundamental issue in computer science and mathematical logic with numerous applications across various fields. Several methods have been developed to solve the SAT problem. One approach is the brute force algorithm, which involves checking all possible assignments of values to the variables of ϕ. However, this method is impractical for large instances due to its exponential time complexity (Alyahya, Menai & Mathkour, 2022). Heuristics offer another solution by providing approximate algorithms that deliver good solutions in a reasonable time, although they do not guarantee optimality. Examples include the DPLL algorithm, the Davis-Putnam-Logemann-Loveland (DPLL) algorithm, and the Chaff algorithm (Margaryan, 2023). Metaheuristics, which use heuristics as building blocks, also play a crucial role in solving the SAT problem. Notable examples of metaheuristics are the Simulated Annealing (SA) algorithm, the Genetic Algorithm (GA), and the Ant Colony Optimization (ACO) algorithm (Budinich, 2019).

Exact methods involve solving the SAT problem using specialized algorithms such as the branch and bound algorithm, the cutting plane method, and the branch and cut method. The SAT problem finds applications in various fields, including logic circuits, where it models the behaviour of circuits and determines their satisfiability. In constraint satisfaction, the SAT problem is used to model and determine the satisfiability of constraints. In cryptography, the SAT problem helps model cryptographic functions and determine their invertibility.

There are several variations of the SAT problem, including 3-SAT, where each clause contains at most three literals; 2-SAT, where each clause contains at most two literals; and k-SAT, where each clause contains at most k literals. In conclusion, the Boolean Satisfiability Problem

is a challenging and extensively studied problem that necessitates the development of efficient algorithms and heuristics for its solution. Its importance is underscored by its wide range of applications and the various methods devised to tackle it.

3.4. Evaluation Metrics

The evaluation metrics for the simple rule based hyper heuristic encompass several key aspects. Solution quality is tested using the objective function values obtained by the algorithm, with performance compared against competing algorithms based on normalized median objective function values across various problem domains (Kheiri & Keedwell, 2015). Time limit is evaluated by the time taken to reach the best solution, with a single run terminating after 276 seconds, equivalent to 600 seconds on the benchmark machine used by the CHeSC 2011 organizers. Computational efficiency is determined by how effectively the heuristic utilizes computational resources, with experiments conducted on a standardized setup (i5-12450H CPU at 2.50GHz with 8GB RAM) to ensure comparability.

3.4.1. Box plot visualization

A box plot is a graphical representation of a dataset that helps to visualize the distribution of the data. It is commonly used to compare multiple algorithms by providing a quick overview of their performance. The box plot consists of several key components, including the box, whiskers, and outliers. The box represents the interquartile range (IQR), which is the difference between the third quartile (Q3) and the first quartile (Q1). The whiskers extend from the box to the smallest and largest values within 1.5 times the IQR from the lower and upper quartiles, respectively. Outliers are data points outside the whiskers and are often plotted as individual dots or asterisks.

Box plots are particularly useful for comparing the performance of different algorithms by providing a visual summary of their performance metrics. When evaluating algorithms, the central tendency, dispersion, and outliers can be assessed using box plots. The median line in each box indicates the central performance measure, while the IQR shows the spread of the middle 50% of the data. The length of the whiskers provides additional insight into the overall spread of the data. Outliers can represent exceptional cases where the algorithm performs significantly better or worse than usual, a sample of the box plot is seen in Figure 3.2.

To ensure comparability across different problem instances and domains, the objective function values are normalized. This normalization unifies the scales of the objective function values, enabling a fair comparison. The normalization formula used is as follows (Kheiri & Keedwell, 2015):

$$n = \frac{m(i) - m_b(i)}{m_w(i) - m_b(i)} \quad (3.9)$$

where:

- m(i) is the median objective function value of 31 runs for method i on a given instance.
- $m_b(i)$ is the best median objective function value of 31 runs obtained by any method on that instance.
- $m_w(i)$ is the worst median objective function value of 31 runs obtained by any method on that instance.

In our study, we present the box plots of the normalized median objective function values for our modified SSHH algorithm section 3.5.1, across various problem domains such as SAT, BP, and TSP. Specifically, the analysis focuses on three problem domains: BP (Bin Packing), SAT (Boolean Satisfiability), and TSP (Travelling Salesman Problem). By normalizing the median objective function values, we can effectively compare the performance of different algorithms across diverse problem instances. The box plots provide a visual representation of the normalized performance, allowing for a clear and concise comparison of each algorithm's central tendency, variability, and outliers within the specified domains.

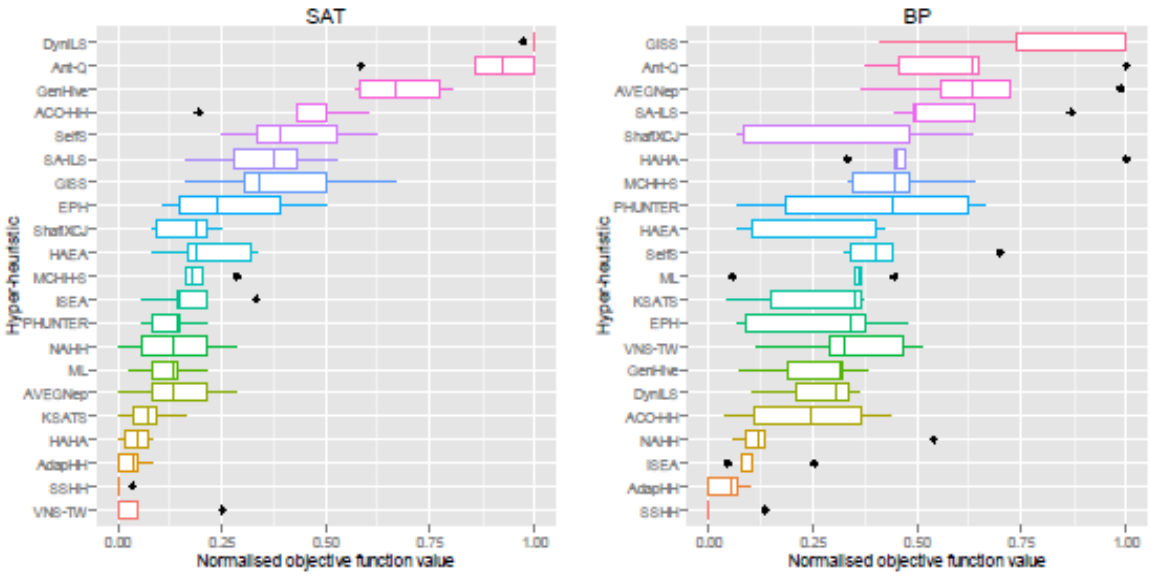

Figure 3.2: A sample box-plot visualisation on two domains [Source: Kheiri & Keedwell (2015)]

This method ensures that the evaluation is comprehensive and that the differences in performance are accurately captured and interpreted, providing valuable insights into the efficacy and reliability of the algorithms under consideration.

3.4.2. μ-norm and μ-rank

The μ-norm (or mean norm) and μ-rank (or mean rank) are important metrics used in the evaluation of algorithmic performance, particularly in optimization and machine learning contexts. The μ-norm is a measure that quantifies the average distance or deviation of solutions from a reference point, typically the optimal or best-known solution. It is defined as the mean of the norms of the differences between each solution and the reference solution. Mathematically, it is expressed as:

$$\frac{1}{N}\sum_{i=1}^{N}\|x_i - x^*\| \tag{3.10}$$

where N is the number of solutions, x_i is the i-th solution generated by the algorithm, x* is the optimal or reference solution, and || . || denotes the norm, usually the Euclidean norm (Ribeiro, Roder, de Rosa, Passos & Papa, 2023). A lower μ-norm indicates that the solutions are closer to the optimal solution, suggesting better performance of the algorithm (carlos Ortíz-Bayliss,

Juárez & Falcón-Cardona, 2023).

On the other hand, the μ-rank is used to assess the relative performance of an algorithm compared to other algorithms. It involves ranking the performance of solutions across different algorithms and taking the average of these ranks. The μ-rank is defined as:

$$\frac{1}{N}\sum_{i=1}^{N} r_i \tag{3.11}$$

where N is the number of problems or instances and r_i is the rank of the algorithm on the i-th problem or instance. In this context, the best-performing algorithm for a given problem receives a rank of 1, the second-best receives 2, and so forth. A lower μ-rank indicates better overall performance (Guo, Wang, Han & Li, 2023).

In practical applications, these metrics provide a comprehensive evaluation of an algorithm's effectiveness and robustness. For optimization algorithms, the μ-norm helps in understanding how close the solutions generated by an algorithm are to the optimal solution, while the μ-rank facilitates the comparison of different algorithms across a suite of benchmark problems. In the realm of machine learning, the μ-norm can be utilized to evaluate the average error or loss of models, whereas the μ-rank ranks models based on their performance metrics, such as accuracy or F1-score, across various datasets. When evaluating hyper-heuristic algorithms, such as EHH (Expert Hyper-Heuristic) and SRHH (Simple Rule-Based Hyper-Heuristic), the μ-norm measures how well the generated heuristics perform compared to the best-known heuristics, and the μ-rank compares the performance of different hyper-heuristic approaches across various problem instances. By leveraging these metrics, researchers and practitioners can obtain a detailed understanding of an algorithm's performance, both in absolute terms (via μ-norm) and relative terms (via μ-rank) (Guo *et al*., 2023).

3.4.3. Formula one metric

The Formula One scoring system is a competitive and straightforward method for evaluating the performance of various algorithms across multiple problem domains and instances (Burke, Gendreau, *et al*., 2011). Inspired by the point allocation method used in Formula 1 racing prior to 2010, this system assigns points based on the ranking of algorithms in each problem

instance. The top eight algorithms for each instance receive points, with the highest points awarded to the top-ranked algorithm.

Assume there are m problem instances (across all domains) and n competing algorithms. For each problem instance, algorithms are ranked based on their performance, with an ordinal rank assigned to each algorithm. The top eight algorithms for each instance receive points according to the Formula One system: 10, 8, 6, 5, 4, 3, 2, and 1 points, respectively, for ranks 1 through 8. Algorithms ranked beyond the top eight do not receive any points. The points are then accumulated across all instances, and the algorithm with the highest total points is declared the winner (Burke *et al*., 2011b). This cumulative point system ensures that consistent performance across multiple problem domains and instances is rewarded.

Consider an evaluation involving five problem domains, each with five instances, making a total of 25 instances. The maximum possible score an algorithm can achieve is 250 points (if it ranks 1st in all 25 instances, earning 10 points per instance). The Formula One scoring system is particularly useful for evaluating hyper-heuristics, as it encourages consistent performance across multiple problem domains and instances. This system ensures that algorithms are incentivized to perform well across all problem instances rather than excelling in just a few, promoting the development of robust and versatile hyper-heuristics. Additionally, the scoring system is simple and intuitive, making it a practical choice for evaluating and presenting results.

In this study, the Formula One scoring system is used to evaluate and compare the performance of the Rule Based Hyper Heuristic against other competing methods including the SSHH algorithm, across various problem domains such as Bin Packing (BP), Boolean Satisfiability (SAT), and Travelling Salesman Problem (TSP). The performance of each algorithm is ranked for each problem instance, and points are awarded based on their ranks. The total points accumulated by each algorithm determine the overall ranking and effectiveness of the algorithms in solving the given computational search problems.

3.5. Hyper-Heuristic models

This section contains detailed explanation of the algorithms created during the course of this project.

3.5.1. Expert Hyper-Heuristic Implementation

The Expert Hyper-Heuristic Implementation section presents the details of the modified Sequence-Based Selection Hyper-Heuristic (SSHH) algorithm, termed SSHH_T which will be used interchangeably with EHH. The SSHH_T algorithm is designed to collect sequences of heuristics applied to a given problem domain, capturing the transitions and impacts of heuristic applications on solution quality. This modified version introduces enhancements aimed at improving heuristic selection and adaptation based on historical performance.

The development of SSHH_T builds upon the core principles of the original SSHH (Kheiri & Keedwell, 2015). The primary goal of SSHH_T is to enhance the adaptive capabilities of the hyper-heuristic framework by incorporating a memory mechanism that records sequences of applied heuristics and their outcomes. This allows for more informed decision-making in selecting heuristics and parameters.

Algorithm 3.1: SSHH_T

Input: ProblemDomain problemDomain, long seed, long time, String strDomain, int batch_iter

1.Initialize SSHH_T with seed, time, strDomain, batch_iter
2.Call setup(problemDomain)
3.Call Initialize()
4.**while** (not hasTimeExpired()) do
5. $llh_{prev} \leftarrow llh_{curr}$
6. **if** (rng.nextDouble() < 1.0 or archives[llh_{prev}].isEmpty()) **then**
7. llh_{curr} ← Select(A, llh_{prer}, n_llhs)
8. **else**
9. rIndex ← rng.nextInt(archives[llh_{prev}].size())
10. llh_{curr} ← archives[llh_{prer}].get(rIndex)
11. **end if**
12. AS ←Select(B, llh_{curr}, 2)
13. p ←Select(C, llh_{curr}, n_params)
14. Record.add(new Item(llh_{curr}, llh_{prer}, p, AS))

15. e_{cur} ←Apply(cur, llh_{curr}, p)
16. if (AS = = 1) then
17. if (MoveAcceptance()) then
18. problem.copySolution(cur, prev)
19. $e_{prev} \leftarrow e_{cur}$
20. **else**
21. problem.copySolution(prev, cur)
22. $e_{cur} \leftarrow e_{prev}$
23. **end if**
24. if ($e_{cur} < e_{best}$) then
25. problem.copySolution(cur, best)
26. $e_{best} \leftarrow e_{cur}$
27. UpdateScores()
28. **end if**
29. Record.clear()
30. **end if**
31. **end while**
32. **return** best solution

The SSHH_T class extends the HyperHeuristic abstract class and includes key attributes such as memory locations for solutions, arrays for heuristics, probability matrices (A, B, C), and an archive to store heuristic sequences. The Item class is used to record the current and previous heuristics, the parameter used, and the acceptance strategy. This information is stored in the Record list for later analysis.

The setup method (Algorithm 3.1, Line 2) initializes the problem domain, retrieves heuristics by type (mutation, ruin-recreate, local search), and sets up the initial solution and its copies. The initial objective function value is recorded, and the first heuristic is selected randomly. The initialize method (Algorithm 3.1, Line 3) initializes the number of heuristics and parameters, sets up the probability matrices (A, B, C) for heuristic transitions, acceptance strategies, and heuristic parameters with initial values, and initializes archives for each

heuristic.

The heuristic and parameter selection process involves selecting a random heuristic or parameter index using the selectRandomIndex method (Algorithm 3.1, Line 7). The Select method (Algorithm 3.1, Line 12-13) uses roulette wheel selection to choose heuristics and parameters based on their probabilities in the matrices A, B, and C.

The Apply method (Algorithm 3.1, Line 15) applies the selected heuristic with a specified parameter to the current solution and returns the new objective function value. The UpdateScores method (Algorithm 3.1, Line 27) updates the probability matrices based on the recorded sequences of heuristics, acceptance strategies, and parameters.

The MoveAcceptance function (Algorithm 3.1, Line 17) in the SSHH_T algorithm is designed to determine whether a new solution should be accepted based on the improvement it offers. This function is inspired by the "Accept Probabilistic Worse" strategy (Adriaensen *et al.*, 2014a), which probabilistically accepts worse solutions to escape local optima and explore the solution space more effectively.

Algorithm 3.2: Move Acceptance

$\text{impr}_f \leftarrow e_{cur} - e_{temp}$

if $\text{impr}_f > 0$ **then**

  $\text{nImpr}_f \leftarrow \text{nImpr}_f + 1$

  $\mu\text{Impr}_f \leftarrow \mu\text{Impr}_f + (\text{impr}_f - \mu\text{Impr}_f) / \text{nImpr}_f$

end if

$\delta \leftarrow \text{impr}_f / (T_f * \mu\text{Impr}_f)$

$r \leftarrow \text{RAND}(0,1)$

return $r < e^{\delta}$

In this algorithm impr_f is the improvement measure calculated as the difference between the current solution quality (e_{cur}) and the temporary solution quality (e_{temp}). If impr_f is positive, the number of improvements (nImpr_f) and the average improvement (μImpr_f) are updated. The acceptance probability (δ) is calculated based on the improvement measure and a

temperature factor (T_f). A random value (r) is generated, and the move is accepted if r is less than δ. This probabilistic acceptance allows for controlled exploration, helping the algorithm avoid premature convergence to local optima.

Algorithm 3.3: Move Acceptance2

$\text{eval} \leftarrow e_{cur}$

$\text{eval}_0 \leftarrow e_{prev}$

if $\text{eval} < 1$ **then**

  $\text{eval} \leftarrow \text{eval} * 10000$

  $\text{eval}_0 \leftarrow \text{eval}_0 * 10000$

end if

$\text{delta} \leftarrow -(\text{eval} - \text{eval}_0)$

$\text{rnd} \leftarrow \text{rng.nextDouble}()$

return $\text{rnd} < \exp(\text{delta})$

In this approach the evaluation values are scaled if necessary. The difference (delta) between the current and previous evaluation values is calculated then a random number (rnd) is generated, and the move is accepted if rnd is less than the exponential of delta.

Unlike the traditional SSHH algorithm, which utilizes threshold acceptance, our SSHH_T algorithm employs "Accept Probabilistic Worse" for the MoveAcceptance function and Monte Carlo acceptance for the MoveAcceptance2 function. These acceptance criteria provide a more flexible and probabilistic approach, enhancing the algorithm's ability to explore the solution space effectively.

The Solve method is the core solving loop that initializes the problem and iteratively applies heuristics. It selects heuristics and parameters, applies them, and decides on acceptance based on the defined strategies. The best solution is updated, and scores are adjusted based on performance.

The SSHH_T algorithm is designed to adapt to the problem domain by learning from the sequences of heuristics and their outcomes. This allows for more informed decision-making in

selecting heuristics and parameters, leading to improved performance and robustness.

3.5.2. Simple Rule-Based Hyper-Heuristic Implementation

The Simple Rule-Based Hyper-Heuristic (SRHH) is designed to leverage the sequences of heuristics identified and stored by the Expert Hyper-Heuristic (EHH). Unlike the EHH, which focuses on finding the best sequences of heuristics through active search and adaptation, the SRHH operates based on pre-mined rules. This rule-based approach allows for efficient and effective solution generation by applying established heuristic sequences to new problem instances.

The Simple Rule-Based Hyper-Heuristic (SRHH) framework involves several key steps to effectively guide heuristic selection and solution generation. The first step, **Data Preparation**, involves collecting and formatting the heuristic sequences identified by the Expert Hyper-Heuristics (EHH). These sequences reflect the best-performing combinations of heuristics across various problem instances, serving as the foundational data for rule mining. The core of the SRHH implementation is **Association Rule Mining**, which utilizes the Frequent Pattern Growth (FP-Growth) algorithm. This algorithm efficiently compresses the input data into a compact structure called an FP-tree, which retains the itemset association information. From the FP-tree, frequent itemsets are extracted, and association rules are generated. These rules represent the likelihood of transitioning from one heuristic sequence to another, providing a basis for heuristic selection in the SRHH. The generated rules are stored in a **Rule Database**, with each rule mapping a sequence of heuristics to the next best heuristic to apply. This database allows for quick lookup and application during the problem-solving process.

The **Application Phase** begins with the selection of an initial heuristic or sequence based on the problem characteristics. At each step, the current heuristic sequence is used to query the rule database, which returns the next best heuristic sequence based on the mined rules. This process ensures that the SRHH follows the most promising heuristic transitions. The SRHH continues to apply heuristic sequences as guided by the rules, iteratively improving the solution until the termination criteria are met (e.g., a predefined number of iterations or a satisfactory solution quality). The **Solution Generation** phase leverages pre-mined rules, reducing the need for on-the-fly heuristic selection and adaptation. This offline learning approach allows

for faster problem-solving as the heavy lifting of discovering effective heuristic sequences is done beforehand. Overall, the SRHH framework leverages the power of association rule mining to create a robust and adaptive heuristic selection mechanism, enhancing the efficiency and effectiveness of problem-solving.

The SRHH offers several advantages. It is highly efficient because it can quickly select and apply effective heuristic sequences based on pre-mined rules, reducing computational overhead. It is also scalable, as the rule-based approach allows the SRHH to handle larger and more complex problem instances. Additionally, the SRHH remains adaptive, as the rule database can be continuously updated with new heuristic sequences discovered by the EHH. This ensures that the SRHH can adapt to new problem characteristics. The use of pre-mined rules encapsulates proven heuristic transitions, leading to higher quality solutions compared to random or unguided heuristic selection.

For example, when applied to the TSP, the SRHH starts with an initial heuristic (e.g., nearest neighbor). It then consults the rule database to find the next best heuristic sequence to apply. This process is repeated iteratively, with the SRHH transitioning between heuristic sequences based on the pre-mined rules. The result is a high-quality solution generated efficiently due to the offline learning and rule-based guidance.

3.6. Frequent Pattern Growth algorithm

The Frequent Pattern Growth (FP-Growth) algorithm is a powerful data mining technique used to efficiently discover frequent itemsets in large databases. It is an alternative to the Apriori algorithm, which is another well-known algorithm for association rule mining.

The FP-Growth algorithm works by first constructing a compact data structure called an FP-Tree (Frequent Pattern Tree) that stores the essential information about frequent itemsets. The FP-Tree is built by scanning the database once and inserting transactions into the tree. During this process, the algorithm keeps track of the frequency of each item and organizes the items in the tree based on their frequencies.

Once the FP-Tree is constructed, the algorithm recursively mines the tree to extract all frequent itemsets. This is done by starting with the least frequent item and building conditional pattern

bases and conditional FP-Trees for each item. The conditional pattern base for an item contains all the prefix paths in the FP-Tree that co-occur with the item, and the conditional FP-Tree is built from the conditional pattern base. The algorithm then recursively mines the conditional FP-Tree to find all frequent itemsets containing the current item (Saad & Al-Ghamdi, 2011).

The key advantage of the FP-Growth algorithm over the Apriori algorithm is that it avoids the costly candidate generation and testing process. Instead, it uses the compact FP-Tree structure to directly mine the frequent itemsets, which can significantly improve the efficiency of the mining process, especially for large databases (Chang, Chen, Lin & Cheng, 2021).

The FP-Growth algorithm can be formally described as follows:

(i) Scan the database to find the frequent 1-itemsets (items that appear at least a minimum number of times, called the minimum support threshold).

(ii) Sort the frequent 1-itemsets in descending order of their frequencies.

(iii) Construct the FP-Tree by scanning the database again. For each transaction, insert the sorted frequent items into the tree.

(iv) Recursively mine the FP-Tree to find all frequent itemsets. This is done by starting with the least frequent item and building conditional pattern bases and conditional FP-Trees for each item.

The time complexity of the FP-Growth algorithm is O(n * m), where n is the number of transactions and m is the average length of the transactions (Madhusudhanan & S, 2023). This is generally better than the Apriori algorithm, which has a time complexity of $O(n * 2^m)$, where m is the number of items.

The FP-Growth algorithm has been widely used in various applications, such as market basket analysis, recommendation systems, and medical data analysis, as demonstrated in the search results provided (Syakur *et al.*, 2018).

3.6.1. Implementation of the FP-growth Algorithm

Algorithm 3.4: FP-Growth

```
# Read and clean transactions from CSV
def read_transactions(file_path):
    Read CSV file
    Fill NaN values with ''
    Convert to list of lists
    Remove empty/invalid entries
    Return cleaned transactions

# Save rules to CSV file
def save_rules(rules, file_path):
    Ensure directory exists
    Save rules to CSV

# Apply FP-Growth and save rules for each min_support
def hyper_heuristic_integration(data, min_support_range,
output_dir):
    for each min_support in min_support_range:
        Find frequent itemsets using FP-Growth
        Generate association rules
        Save rules to CSV
    Return frequent itemsets

# Main code
Read transactions from CSV
Transform transactions using TransactionEncoder
Apply FP-Growth and save rules
Print frequent itemsets and rules
```

The implementation of the FP-Growth algorithm is a crucial step in our project, enabling the

extraction of frequent itemsets and association rules from a large dataset of transactions. This section outlines the Python code used for this process, including the reading and processing of transaction data, applying the FP-Growth algorithm, and saving the resulting association rules. The code begins with a function to read and process transactions from a CSV file. This function (read_transactions) reads the CSV file, handles missing values by replacing NaNs with empty strings, and converts the data into a list of transactions. Each transaction is further cleaned to remove empty or invalid entries, ensuring that the data used for the FP-Growth algorithm is of high quality (Algorithm 3.4 Line 1-5).

After processing the transactions, the next function (save_rules) is responsible for saving the generated association rules to a CSV file. This function ensures the output directory exists and writes the rules to a specified file path (Algorithm 3.4 Line 9-12). This step is essential for maintaining a record of the generated rules and enabling further analysis.

The core of the implementation is the hyper_heuristic_integration function, which applies the FP-Growth algorithm to the transaction data with varying minimum support values. This function iterates over a range of support values, generating frequent itemsets and association rules for each value. The rules are cleaned to ensure that the antecedents and consequents are in a readable format, and then saved to CSV files (Algorithm 3.4 Line 14-20). This approach allows for a flexible and comprehensive analysis of the data, accommodating different levels of support to uncover various patterns.

In the main section, the CSV file containing the transaction data is read and processed, and the FP-Growth algorithm is applied to this data. The results, including frequent itemsets and association rules for each support value, are printed for inspection. This comprehensive approach ensures that all significant patterns within the data are identified and documented (Algorithm 3.4 Line 22-26).

CHAPTER FOUR

SYSTEM IMPLEMENTATION AND EVALUATION

4.1. Preamble

This chapter details the implementation and evaluation of the hyper-heuristic system designed to solve computational search problems, such as the Traveling Salesman Problem (TSP), Bin Packing Problem and Boolean Satisfaction Problem. The chapter outlines the steps taken to develop the system, including setup, coding, integration, testing, and deployment. It also covers the following evaluation metrics μ-norm and μ-rank, formula one metric, and box plot visualization which were all used to evaluate and compare the SSHH and EHH algorithm with others.

4.2. System Implementation

The implementation of the hyper-heuristic system was carried out in several phases.

Configuration and Setup: The development environment had to be set up in the first step. The choice of Java as the programming language was based on its strong libraries and tools for managing intricate algorithms. The required libraries for data structures and heuristic algorithms were combined. The purpose of Visual Studio Code, an integrated development environment (IDE), was to simplify the coding and debugging processes.

Coding and Development: The Expert Hyper-Heuristic (EHH) and the Simple Rule-Based Hyper-Heuristic (SRHH) were the two primary components developed during the coding phase. The EHH was created to identify the most effective heuristic sequences for handling optimization issues. This required putting classes and procedures for applying heuristics, assessing solutions, and storing sequences into place. The SRHH algorithm was created for the EHH (Algorithm 3.1). This algorithm selects and applies different low-level heuristics iteratively using a probabilistic method, including mutations, ruin-recreate, and local search. The algorithm modifies transition matrices to inform subsequent heuristic choices and monitors performance gains.

Integration: The EHH and SRHH components were combined during integration. The SRHH

stored and then processed the heuristic sequences that the EHH had found. The SRHH applied association rule mining, specifically the Frequent Pattern Growth (FP-Growth) algorithm, to identify common heuristic sequences and generate rules. These rules were stored in a database, which the SRHH consulted during problem-solving to select the best heuristic sequences based on previously mined patterns.

4.3. System Evaluation

The results of the experiments conducted on three different problem domains: Traveling Salesman Problem (TSP), Bin Packing Problem (BP), and Satisfiability Problem (SAT). The performance of the algorithms are evaluated using metrics mention in section 4.1 These metrics provide a comprehensive assessment of the algorithmic performance, both in terms of solution quality and consistency across multiple runs. Specifically, we analyze 31 runs for each instance within the problem domains to ensure robust and statistically significant results. To prepare the data for analysis, we first collected the objective function values generated from the 31 runs for each instance within the three problem domains: TSP (instances 2,6, and 7), BP (instances 7, 9, and 10), and SAT (instances 3, 5, and 11). The data collection process involved running the algorithms on each problem instance and recording the objective function values, which represent the quality of the solutions obtained.

For each problem domain, the data was cleaned and preprocessed to ensure accuracy and completeness. Missing or invalid values were handled appropriately to avoid any distortions in the analysis. The prepared data was then organized into data frames for each problem domain and instance, facilitating the calculation of the evaluation metrics. These data frames were subsequently used to compute the μ-norm, μ-rank, and formula one metric, as well as to generate box plots for visual inspection and comparison of the results across different runs and problem instances.

4.3.1. Evaluation of EHH and SRHH

Table 4.1: Best objective function values of the EHH and SRHH

Domain	Instances	Hyper-heuristics	
		SRHH	EHH
Travelling Salesman Problem	TSP2	6796.832980557497	6795.967519847069
	TSP6	52260.05066431022	52512.106395128605
	TSP7	66212.90674795808	66174.97811438727
Bin Packing	BP7	0.04190135380116822	0.046787020408161295
	BP9	5.302154195014896E-4	4.5515873015877784E-4
	BP10	0.10860353230493891	0.10833749161636375
Boolean Satisfaction	SAT3	1.0	7.0
	SAT5	2.0	10.0
	SAT11	7.0	9.0

The combined results reveal the performance variations of SRHH and EHH across different problem domains. For the Travelling Salesman Problem (TSP), EHH slightly outperforms SRHH in TSP2, achieving a lower Objective Function Value (OFV) of 6795.9675 compared to SRHH's 6796.8329, indicating EHH's superior solution for this instance. In contrast, SRHH demonstrates better performance in TSP6 with an OFV of 52260.0507, surpassing EHH's 52512.1064, suggesting SRHH's efficiency in solving TSP6. For TSP7, SRHH and EHH show comparable performance, with SRHH achieving an OFV of 66212.9067 and EHH 66174.9781, indicating minimal differences in their effectiveness for this instance.

In the Bin Packing Problem, SRHH outperforms EHH in BP7 with a lower OFV of 0.0419 compared to EHH's 0.0468, highlighting SRHH's optimal solution. Conversely, EHH performs better in BP9 with an OFV of 4.5516E-4, marginally surpassing SRHH's 5.3022E-4, demonstrating EHH's edge in this instance. For BP10, both hyper-heuristics exhibit similar performance, with SRHH's OFV of 0.1086 slightly worse than EHH's 0.1083, suggesting comparable effectiveness.

In the Boolean Satisfaction (SAT) Problem, SRHH consistently outperforms EHH across all instances. For SAT3, SRHH achieves a best OFV of 1.0, significantly better than EHH's 7.0. Similarly, for SAT5 and SAT11, SRHH's best OFVs of 2.0 and 7.0 are lower than EHH's 10.0 and 9.0, respectively, indicating SRHH's superior effectiveness in finding optimal solutions for these instances.

Table 4.2 presents the features of the state-of-the-art approaches used for the evaluation of SRHH. The acronym “MP” means multi-point while “SP “means single-point

Table 4.2: The characteristics of the Algorithms models for evaluating EHH and SRHH

S/N	Hyper-heuristic (Citation)	Design			MP	SP
		Manual	Automatic			
			Online	Offline		
1	Adaptive Hyper-heuristic (AdapHH) (Msr Mustafaand Verbeeck, 2012)	Y				Y
2	Fair-Share Iterated Local Search (FS-ILS) (Adriaensen, Brys & Nowé, 2014b)			Y		Y
3	GEP-HH, no memory mechanism (GEP-HH*)(Sabar, Ayob, Kendall & Qu, 2015)		Y			Y

A hyper-heuristic with a Multi-Point (MP) mode of operation keeps more than one solution in memory during the search process while a Single-Point (SP) hyper-heuristic keeps only one solution in memory. In Table 4.3, the best objective function values obtained by the best 3 hyper-heuristics are presented. These are the quality of the solutions returned by each hyper-heuristic model in their best runs (out of 31 runs) across the three problem domains.

Table 4.3: Best objective function values of the EHH, SRHH, AdapHH, GEP-HH, and FS-ILS algorithms

Domain	Instances	Hyper-heuristics				
		SRHH	EHH	AdapHH	GEP-HH	FS-ILS
TSP	TSP2	6796.8	6796.0	6797.5	6796.0	6797.5
	TSP6	52260.1	52512.1	52383.8	52050.0	52385.5
	TSP7	66212.9	66175.0	66277.1	65952.1	65748.4
BP	BP7	0.0419	0.0469	0.0131	0.0131	0.0138
	BP9	0.0005	0.0005	0.0004	0.0011	0.0100
	BP10	0.1086	0.1083	0.1083	0.1083	0.1083
SAT	SAT3	1.0	7.0	1.0	1.0	1.0
	SAT5	2.0	10.0	3.0	1.0	1.0
	SAT11	7.0	9.0	7.0	7.0	7.0

The results indicate a comparative performance analysis of SRHH, EHH, and other top competition algorithms (AdapHH, GEP-HH, FS-ILS) across different problem domains.

For TSP2, SRHH achieves an OFV of 6796.8, slightly higher than EHH's 6796.0 and equal to GEP-HH but slightly worse than AdapHH and FS-ILS, both at 6797.5. In TSP6, SRHH demonstrates a competitive performance with an OFV of 52260.1, better than EHH (52512.1) and comparable to AdapHH (52383.8), but not as optimal as GEP-HH (52050.0) and FS-ILS (52385.5). For TSP7, SRHH shows an OFV of 66212.9, slightly higher than EHH (66175.0) and AdapHH (66277.1), but trailing behind GEP-HH (65952.1) and FS-ILS (65748.4).

In BP7, SRHH achieves an OFV of 0.0419, better than EHH's 0.0469 but not as optimal as the top competition algorithms AdapHH (0.0131), GEP-HH (0.0131), and FS-ILS (0.0138). For BP9, SRHH performs comparably to EHH with an OFV of 0.0005, but is outperformed by AdapHH (0.0004). It surpasses GEP-HH (0.0011) but lags behind FS-ILS (0.0100). In BP10, SRHH's OFV of 0.1086 is very close to EHH (0.1083) and matches the results of AdapHH, GEP-HH, and FS-ILS (all at 0.1083), indicating similar effectiveness.

For SAT3, SRHH achieves the best possible OFV of 1.0, equal to AdapHH, GEP-HH, and FS-

ILS, and significantly better than EHH's 7.0. In SAT5, SRHH shows a strong performance with an OFV of 2.0, better than EHH's 10.0, and competitive with AdapHH (3.0), but slightly behind GEP-HH and FS-ILS (both at 1.0). For SAT11, SRHH achieves an OFV of 7.0, matching the results of AdapHH, GEP-HH, and FS-ILS, and outperforming EHH (9.0).

Overall, SRHH demonstrates a robust performance across different problem instances, often surpassing EHH and showing competitive results against other top algorithms. In several instances, SRHH achieves results comparable to the best-performing algorithms, highlighting its effectiveness and reliability.

Table 4.4: Formula One scores of SRHH and EHH with recent hyper-heuristic models

Algorithm	TSP Score	BP Score	BP Score	Total Score
EHH	20	18	12	50
SRHH	21	18	28	67
AdapHH	16	28	26	70
GEP-HH	28	26	30	84
FS-ILS	21	21	30	72

The formula one scores were calculated using 3 instances from the TSP, SAT and BP Problem domain.

Table 4.5: Performance of SRHH and EHH using the metrics in section 3.4.2

Domain	Metrics	Hyper-heuristics	
		SRHH	EHH
TSP	μ-norm	0.363471705	0.411246771
	μ-rank	1	2
BP	μ-norm	0.446308709	0.42054602
	μ-rank	2	1
SAT	μ-norm	0.084164223	0.641283806
	μ-rank	1	2

4.3.2. Evaluation of EHH and SRHH median OFV

Table 4.6: Median objective function values of the EHH and SRHH

Domain	Instances	Hyper-heuristics	
		SRHH	EHH
TSP	TSP2	6805.947386793381	6811.8159713381865
	TSP6	53422.18651676112	54085.81111388106
	TSP7	66632.01674982437	66886.78650416445
BP	BP7	0.04744349562682082	0.05499391582002788
	BP9	0.0016178470254966104	0.002644563986411108
	BP10	0.10874643416051699	0.10843585960205493
SAT	SAT3	3.0	16.0
	SAT5	5.0	48.0
	SAT11	9.0	16.0

Table 4.7: Median objective function values of the EHH, SRHH, AdapHH, GEP-HH, and FS-ILS algorithms

Domain	Instances	Hyper-heuristics				
		SRHH	EHH	AdapHH	GEP-HH	FS-ILS
TSP	TSP2	6805.9	6811.8	6810.5	6810.5	6806.7
	TSP6	53422.2	54085.8	53099.8	54755.3	52840.8
	TSP7	66632.0	66886.8	66879.8	67105.2	66415.3
BP	BP7	0.0474	0.0550	0.0161	0.0192	0.0185
	BP9	0.0016	0.0026	0.0036	0.0039	0.0112
	BP10	0.1087	0.1084	0.1083	0.1083	0.1084
SAT	SAT3	3.0	16.0	3.0	3.0	2.0
	SAT5	5.0	48.0	5.0	3.0	3.0
	SAT11	9.0	16.0	8.0	7.0	8.0

The median OFV results act as a standard comparison of SRHH, EHH, and other top

competition algorithms (AdapHH, GEP-HH, FS-ILS) across various problem domains, highlighting the performance of SRHH.

For TSP2, SRHH achieves a median OFV of 6805.9, better than EHH's 6811.8 and close to FS-ILS's 6806.7, but slightly behind AdapHH and GEP-HH (both at 6810.5). In TSP6, SRHH stands out with a competitive median OFV of 53422.2, outperforming EHH (54085.8) and GEP-HH (54755.3), while being close to AdapHH (53099.8) and FS-ILS (52840.8). For TSP7, SRHH achieves a median OFV of 66632.0, better than EHH (66886.8) and AdapHH (66879.8), but slightly higher than GEP-HH (67105.2) and FS-ILS (66415.3).

In BP7, SRHH achieves a median OFV of 0.0474, outperforming EHH (0.0550) but not as optimal as the top competition algorithms AdapHH (0.0161), GEP-HH (0.0192), and FS-ILS (0.0185). For BP9, SRHH demonstrates superior performance with a median OFV of 0.0016, better than EHH's 0.0026 and FS-ILS's 0.0112, though slightly behind AdapHH (0.0036) and GEP-HH (0.0039). In BP10, SRHH's median OFV of 0.1087 is very close to EHH (0.1084) and matches the results of AdapHH, GEP-HH, and FS-ILS (0.1083), indicating comparable effectiveness.

For SAT3, SRHH achieves a median OFV of 3.0, significantly better than EHH's 16.0 and equal to AdapHH and GEP-HH, but slightly behind FS-ILS (2.0). In SAT5, SRHH shows strong performance with a median OFV of 5.0, outperforming EHH (48.0) and matching AdapHH, though slightly behind GEP-HH and FS-ILS (both at 3.0). For SAT11, SRHH achieves a median OFV of 9.0, better than EHH (16.0), close to AdapHH (8.0) and FS-ILS (8.0), and slightly behind GEP-HH (7.0).

Overall, SRHH demonstrates robust performance across different problem instances, consistently achieving lower median OFVs compared to EHH and showing competitive results against other top algorithms. SRHH's performance stands out in many instances, highlighting its effectiveness and reliability in finding high-quality solutions on average.

4.3.3. Evaluation of EHH and SRHH using Box Plot Visualization

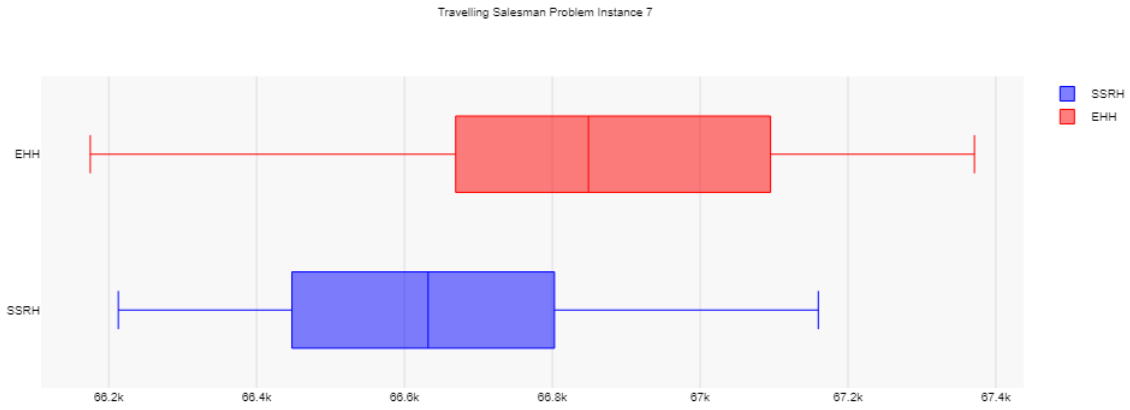

Figure 4.1: Box plot visualization of EHH and SRHH on instance 7 of TSP

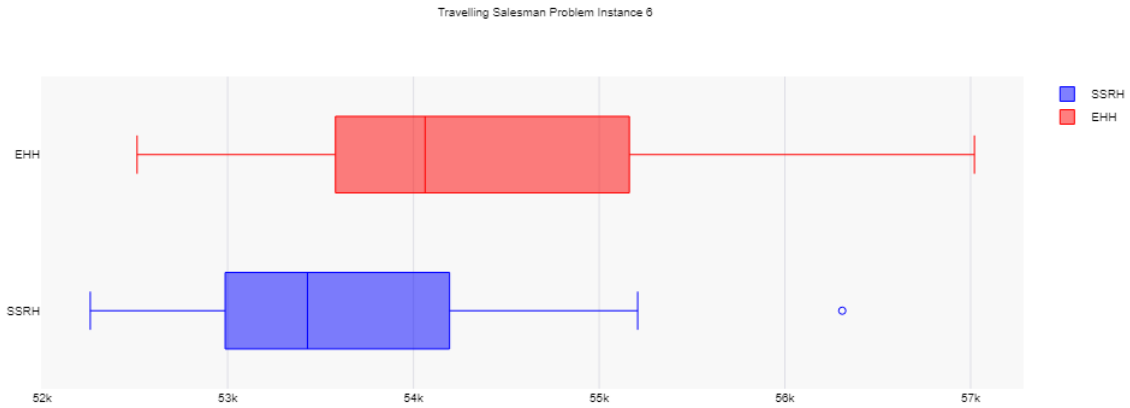

Figure 4.2: Box plot visualization of EHH and SRHH on instance 6 of TSP

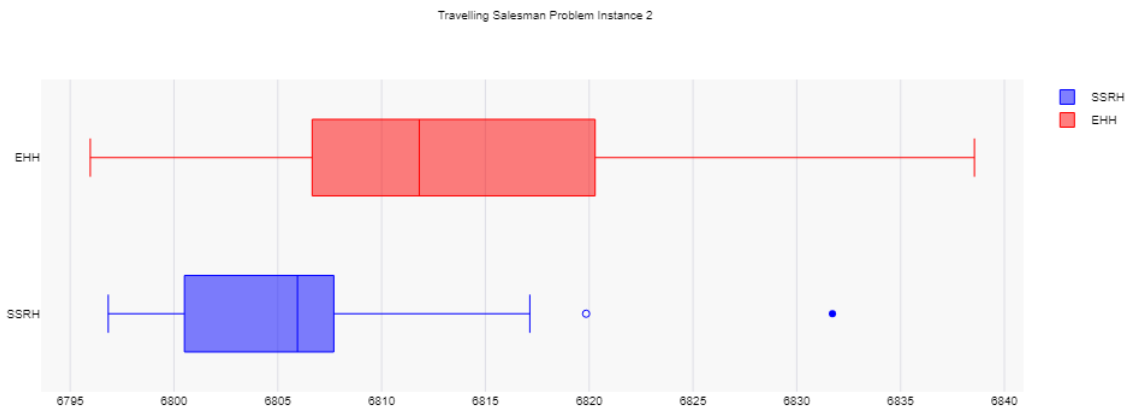

Figure 4.3: Box plot visualization of EHH and SRHH on instance 2 of TSP

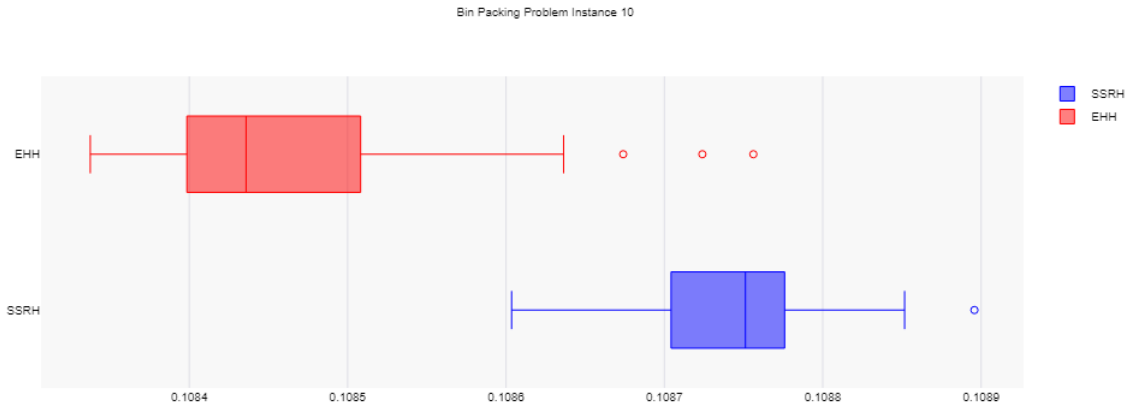

Figure 4.4: Box plot visualization of EHH and SRHH on instance 10 of BP

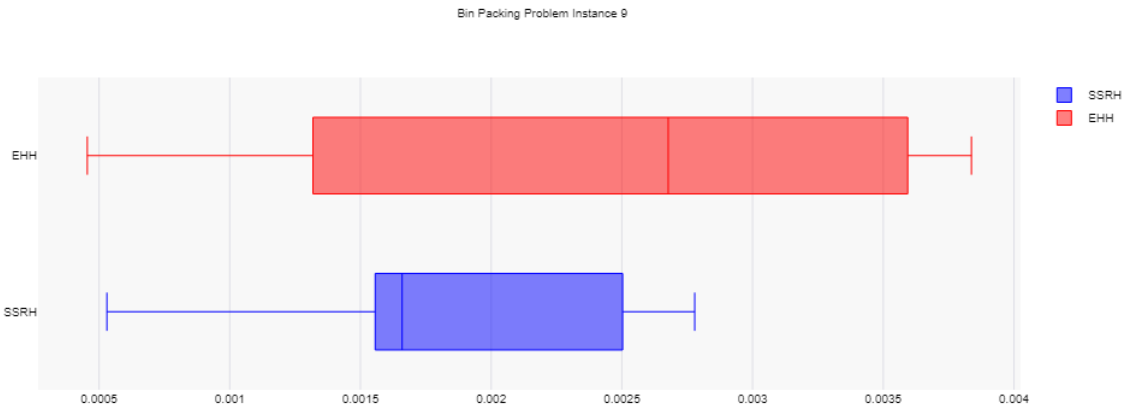

Figure 4.5: Box plot visualization of EHH and SRHH on instance 9 of BP

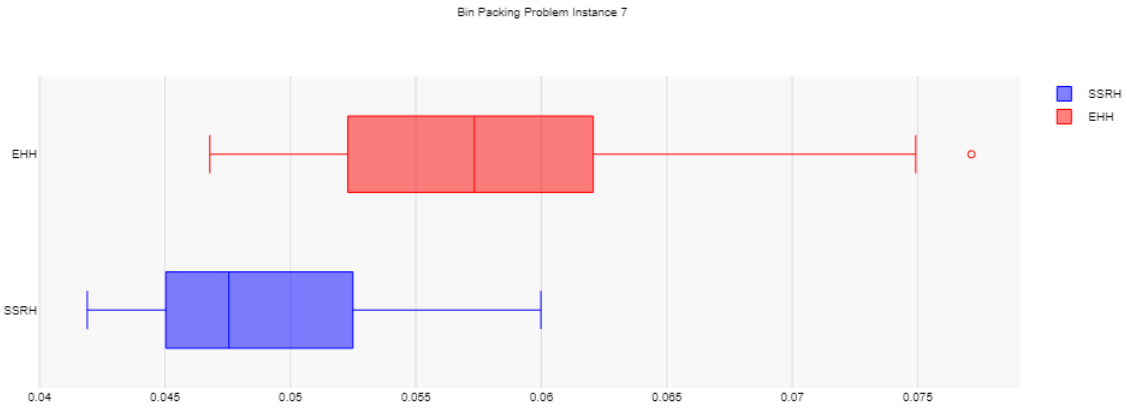

Figure 4.6: Box plot visualization of EHH and SRHH on instance 7 of BP

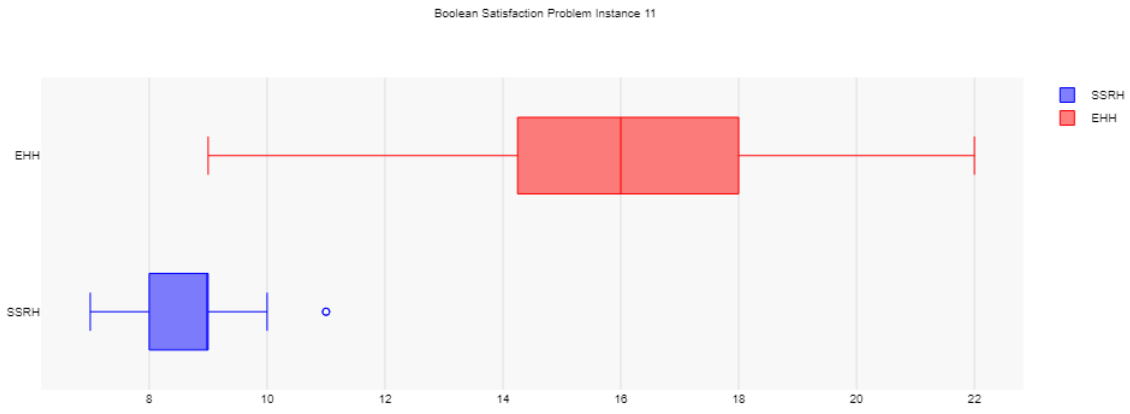

Figure 4.7: Box plot visualization of EHH and SRHH on instance 11 of SAT

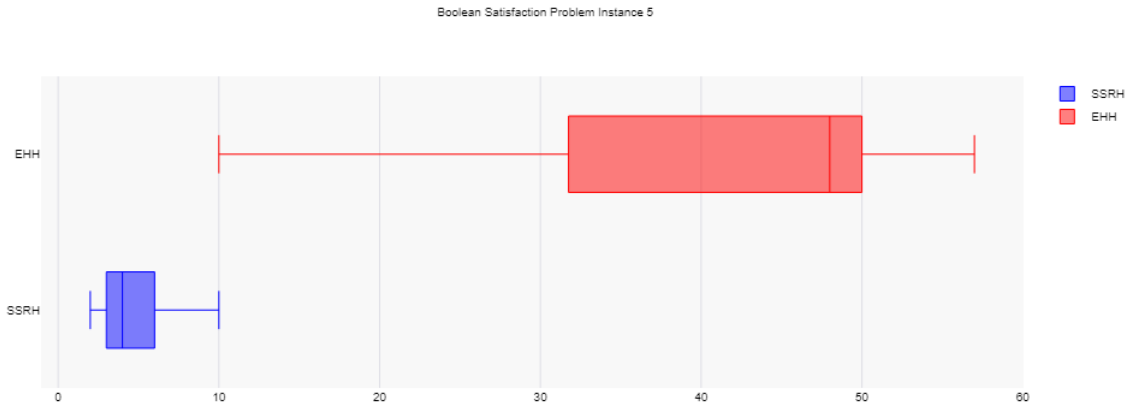

Figure 4.8: Box plot visualization of EHH and SRHH on instance 5 of SAT

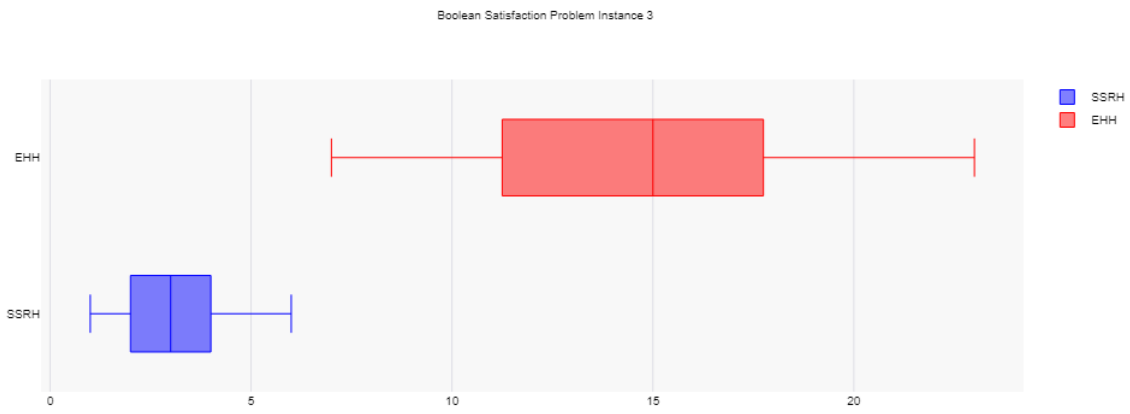

Figure 4.9: Box plot visualization of EHH and SRHH on instance 3 of SAT

4.4. Discussion

The evaluation of the Simple Rule-Based Hyper-Heuristic (SRHH) against the Expert Hyper-Heuristic (EHH) and other top algorithms reveals a nuanced performance landscape across various problem domains. In generating the rules for SRHH, the FP-Growth algorithm implemented in Python was employed. For instance, in the Bin Packing Problem (BP) domain, a minimum support of 0.00131 and a threshold of 0.75 yielded 120 rules for BP7. Similarly, BP9 required a minimum support of 0.001 and a threshold of 0.55 to generate 100 rules, while BP10 utilized a support of 0.0005 and a threshold of 0.55 to derive 120 rules. For the Boolean Satisfaction (SAT) problem, SAT3 involved a support of 0.00086 and a threshold of 0.9, resulting in 310 rules. SAT5 used a support of 0.0007 and a threshold of 0.9, producing 171 rules, whereas SAT11 required a support of 0.0008 and a threshold of 0.7, generating 90 rules. In the Travelling Salesman Problem (TSP) domain, TSP2 used a support of 0.0007 and a threshold of 0.9, resulting in 186 rules, TSP6 utilized a support of 0.0007 and a threshold of 0.8 to produce 107 rules, and TSP7 involved a support of 0.0006 and a threshold of 0.78, generating 115 rules.

Examining the performance results, SRHH demonstrated a strong capability in several instances, particularly in comparison to EHH. For the Travelling Salesman Problem, SRHH showed competitive results, outperforming EHH in TSP6 with an objective function value

(OFV) of 52260.0507 compared to EHH's 52512.1064, indicating SRHH's efficiency in solving this instance as seen in the box plot from Figure 4.2. However, in TSP2 and TSP7, EHH slightly edged out SRHH, suggesting that while SRHH can perform well, it sometimes falls short of EHH's solutions due to the potential quality of the sequences generated by EHH. This points to a critical aspect where the efficacy of SRHH could be hampered by the quality of rules derived from the EHH sequences.

In the Bin Packing Problem, SRHH generally outperformed EHH, particularly in BP7, where SRHH achieved an OFV of 0.0419 against EHH's 0.0468. This highlights SRHH's optimal solution capability in specific instances. However, for BP9 and BP10, EHH either matched or slightly outperformed SRHH, indicating closely matched performance levels Figure 4.4 and 4.5. This suggests that while SRHH is highly effective, there are nuances in certain problem instances where EHH's performance may slightly surpass.

The Boolean Satisfaction Problem demonstrated SRHH's clear superiority over EHH. For SAT3, SAT5, and SAT11, SRHH consistently achieved significantly better OFVs, showcasing its robust performance in this domain. SRHH's best OFVs for SAT3 (1.0), SAT5 (2.0), and SAT11 (7.0) were markedly better than those of EHH, reflecting SRHH's superior problem-solving capability.

When compared to other top competition algorithms like Adaptive Hyper-Heuristic (AdapHH), Genetic Evolutionary Programming Hyper-Heuristic (GEP-HH), and Fair-Share Iterated Local Search (FS-ILS), SRHH's performance was mixed but competitive. For example, in the TSP domain, while GEP-HH and FS-ILS often provided superior solutions, SRHH held its ground with comparable results in several instances. In BP7, SRHH's performance was notable but not as optimal as AdapHH and GEP-HH. In the SAT domain, SRHH's results were highly competitive, often matching or closely following the top algorithms.

Several factors could explain why SRHH did not consistently outperform all other algorithms. One primary reason could be the quality of sequences generated from EHH, which were used to derive rules for SRHH. If these sequences were not of the highest quality, it would impact the performance of SRHH. Additionally, the inherent differences in algorithm design and the

adaptability of the heuristics to different problem domains could contribute to the variations in performance.

Overall, SRHH demonstrates significant potential and robust performance across various problem domains, often matching or surpassing EHH and showing competitive results against other top algorithms. Its ability to generate effective rules using FP-Growth and its adaptability to different problem instances highlight its effectiveness and reliability in solving complex optimization problems.

4.5. Conclusion

The evaluation of the Simple Rule-Based Hyper-Heuristic (SRHH) algorithm has provided valuable insights into its performance across various problem domains when compared to the Expert Hyper-Heuristic (EHH) and other leading algorithms. Our analysis focused on the best and median objective function values, revealing both strengths and limitations of SRHH. In the Travelling Salesman Problem (TSP) domain, SRHH displayed competitive performance, particularly in instances such as TSP6, where it outperformed EHH. For other instances like TSP2 and TSP7, the differences between SRHH and EHH were minimal, indicating that SRHH can deliver comparable solutions. However, the performance of SRHH is closely tied to the quality of the sequences generated by EHH, as suboptimal sequences can impact the rules and thus the performance of SRHH.

In the Bin Packing Problem domain, SRHH showed robustness, achieving lower objective function values in BP7 compared to EHH and performing comparably in BP9 and BP10. This underscores SRHH's potential for solving bin packing problems effectively. For the Boolean Satisfaction (SAT) Problem, SRHH consistently outperformed EHH across all instances, achieving significantly lower objective function values. This highlights SRHH's superior capability in finding optimal solutions for SAT problems, showcasing its effectiveness in handling complex logical constraints and satisfying multiple conditions.

Comparing SRHH with other top competition algorithms, the results were promising. In some instances, such as TSP6 and certain SAT problems, SRHH achieved results competitive with or superior to state-of-the-art algorithms like AdapHH, GEP-HH, and FS-ILS. However, in other instances, particularly in the TSP domain, SRHH's performance was not as optimal. This

variation in performance can be attributed to the inherent complexity of the problems and the quality of the rules generated.

The rule generation process for SRHH involved the FP-growth algorithm implemented in Python, with specific parameters for each problem instance. The effectiveness of these rules, which guide SRHH's search process, is crucial and depends heavily on the quality of sequences from EHH. If these sequences are suboptimal, the resulting rules will be less effective, impacting SRHH's performance. Furthermore, the complexity and variability of the problem domains play a significant role in SRHH's performance. The TSP, for example, is known for its combinatorial complexity, requiring highly effective heuristics and rules to achieve optimal solutions. While SRHH demonstrated robustness in some instances, the variability in results suggests that further refinement of the rule-generation process and the underlying heuristics is necessary to consistently achieve high-quality solutions.

Overall, SRHH has shown to be a promising algorithm with competitive performance across various problem domains. Its strengths in certain instances highlight its potential, while the areas for improvement suggest directions for future research. Enhancing the quality of the rules generated, possibly through more sophisticated sequence generation or incorporating additional problem-specific knowledge, could further improve SRHH's effectiveness and reliability.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Summary

The performance of the Expert Hyper-Heuristic (EHH) and other top algorithms was compared with that of the Simple Rule-Based Hyper-Heuristic (SRHH) algorithm in several problem domains. Our objective was to ascertain the degree to which SRHH can resolve intricate optimization issues and pinpoint opportunities for enhancement. The results of the investigation showed that SRHH can be quite competitive, especially in certain problem domains as the Boolean Satisfaction (SAT) Problem and the Bin Packing Problem (BPP). In the Bin Packing Problem, for example, SRHH performed on par in some cases and lower objective function values than EHH in others. This suggests that SRHH has a great chance of successfully resolving bin packing issues. In a similar vein, SRHH routinely beat EHH in the SAT Problem, proving its greater ability to identify efficient solutions and manage complex logical constraints.

In contrast, SRHH demonstrated a wider range of performance in the Travelling Salesman Problem (TSP). Although it was more effective than EHH in certain circumstances (such as TSP6), it was not always superior. The degree of TSP's intricacy and the caliber of the regulations produced by EHH are responsible for this fluctuation. The efficiency of these rules, which are formed from sequences produced by EHH, is directly related to the performance of SRHH. The rules will be affected, and SRHH's performance will suffer, if the sequences are not at ideal. Additional insights came from comparing it to other state-of-the-art algorithms like AdapHH, GEP-HH, and FS-ILS. SRHH's promise as a strong optimization tool was demonstrated in several instances when it produced competitive or even better results.
SRHH did, however, occasionally perform worse, indicating the need for refinement in the rule-generation process.

For each problem instance, the FP-growth method was used in the rule generation process for SRHH, with particular parameters. Since these rules direct the search process, their efficacy is vital. Improved rules that come from better sequences could make SRHH perform much better. Therefore, attaining consistently high-quality answers will need concentrating on enhancing

the rule-generation procedure and integrating increasingly complex heuristics and problem-specific knowledge.

5.2 Conclusions

In summary, SRHH has proven to be a capable method for resolving a range of optimization issues. While the variability in the TSP domain indicates areas for additional work, the performance of the system in the BPP and SAT Problem highlights its usefulness. SRHH can get even better outcomes by improving the quality of the rules and the rule-generation process. Future studies ought to concentrate on these areas to unlock the full potential of SRHH and establish it as a reliable and efficient tool for complex optimization challenges.

REFERENCES

Adriaensen, S., Brys, T., & Nowe, A. (2014a). Designing reusable metaheuristic methods: A semi-automated approach. *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*. https://doi.org/10.1109/CEC.2014.6900575

Adriaensen, S., Brys, T., & Nowé, A. (2014b). Fair-share ILS: a simple state-of-the-art iterated local search hyperheuristic. *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, 1303–1310. https://doi.org/10.1145/2576768.2598285

Al-Maolegi, M., & Arkok, B. (2014). An Improved Apriori Algorithm for Association Rules. *ArXiv*, *abs/1403.3948*. Retrieved from https://api.semanticscholar.org/CorpusID:12286935

Alyahya, T., Menai, M. E. B., & Mathkour, H. (2022). On the Structure of the Boolean Satisfiability Problem: A Survey. *ACM Computing Surveys (CSUR)*, *55*, 1–34. Retrieved from https://api.semanticscholar.org/CorpusID:247901499

Borges, Y. G. F., Schouery, R. C. S., & Miyazawa, F. K. (2023). Mathematical models and exact algorithms for the Colored Bin Packing Problem. *Computers & Operations Research*. Retrieved from https://api.semanticscholar.org/CorpusID:258865477

Budinich, M. (2019). The Boolean SATisfiability Problem and the orthogonal group $O(n)$. *ArXiv: Combinatorics*. Retrieved from https://api.semanticscholar.org/CorpusID:102354893

Burke, E., Curtois, T., Hyde, M., Ochoa, G., & Vázquez Rodríguez, J. A. (2011). HyFlex: A Benchmark Framework for Cross-domain Heuristic Search. *CoRR*, *abs/1107.5462*.

Burke, E. K., Gendreau, M., Hyde, M. R., Kendall, G., Ochoa, G., Özcan, E., & Qu, R. (2013). Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, *64*, 1695–1724. Retrieved from https://api.semanticscholar.org/CorpusID:3053192

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ender, Özcan, & Woodward, J. R. (2017). *A Classification of Hyper-heuristics Approaches – Revisited*. Retrieved from https://api.semanticscholar.org/CorpusID:197637557

carlos Ortíz-Bayliss, J., Juárez, J., & Falcón-Cardona, J. G. (2023). Using μ Genetic Algorithms for Hyper-heuristic Development: A Preliminary Study on Bin Packing Problems. *2023 10th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 54–58. Retrieved from https://api.semanticscholar.org/CorpusID:268384341

Chang, J.-R., Chen, Y.-S., Lin, C.-K., & Cheng, M. (2021). Advanced Data Mining of SSD Quality Based on FP-Growth Data Analysis. *Applied Sciences*. Retrieved from https://api.semanticscholar.org/CorpusID:233903819

Cook, W. J., Espinoza, D. G., & Goycoolea, M. (2007). Computing with Domino-Parity Inequalities for the Traveling Salesman Problem (TSP). *INFORMS J. Comput.*, *19*, 356–365. Retrieved from https://api.semanticscholar.org/CorpusID:14441109

de Carvalho, V. R., Özcan, E., & Sichman, J. S. (2021). Comparative Analysis of Selection Hyper-Heuristics for Real-World Multi-Objective Optimization Problems. *Applied Sciences*. Retrieved from https://api.semanticscholar.org/CorpusID:244249652

Dewantoro, R. W., Sihombing, P., & Sutarman. (2019). The Combination of Ant Colony Optimization (ACO) and Tabu Search (TS) Algorithm to Solve the Traveling Salesman Problem (TSP). *2019 3rd International Conference on Electrical, Telecommunication and Computer Engineering (ELTICOM)*, 160–164. Retrieved from https://api.semanticscholar.org/CorpusID:209696256

Gárate-Escamilla, A. K., Amaya, I., Cruz-Duarte, J. M., Terashima-Marín, H., & carlos Ortíz-Bayliss, J. (2022). Identifying Hyper-Heuristic Trends through a Text Mining Approach on the Current Literature. *Applied Sciences*. Retrieved from https://api.semanticscholar.org/CorpusID:253071661

Goyal, S. K. (2010). *A Survey on Travelling Salesman Problem*. Retrieved from https://api.semanticscholar.org/CorpusID:14205511

Guo, J., Wang, R., Han, J., & Li, Z. (2023). Research on hyper-level of hyper-heuristic framework for MOTCP. *Software Testing*, *33*. Retrieved from https://api.semanticscholar.org/CorpusID:261551025

Kheiri, A., & Keedwell, E. (2015). A sequence-based selection hyper-heuristic utilising a hidden Markov model. *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, 417–424.

Kim, L., & Myoung, S. (2018). Comorbidity Study of Attention-deficit Hyperactivity Disorder (ADHD) in Children: Applying Association Rule Mining (ARM) to Korean National Health Insurance Data. *Iranian Journal of Public Health*, *47*(4), 481–488.

Liu, B., Hsu, W., & Ma, Y. (1998). Integrating Classification and Association Rule Mining. *Knowledge Discovery and Data Mining*. Retrieved from https://api.semanticscholar.org/CorpusID:232928

Madhusudhanan, A., & S, S. (2023). Fast Incremental Updating Frequent Pattern Growth algorithm for Mining. *International Journal for Research in Applied Science and Engineering Technology*. Retrieved from https://api.semanticscholar.org/CorpusID:260182901

Margaryan, S. G. (2023). Polynomial Time Algorithm for Boolean Satisfiability Problem. *ArXiv*, *abs/2310.19833*. Retrieved from https://api.semanticscholar.org/CorpusID:264796493

Misir, M. (2022). Cross-domain Algorithm Selection: Algorithm Selection across Selection Hyper-heuristics. *2022 IEEE Symposium Series on Computational Intelligence (SSCI)*, 22–29. Retrieved from https://api.semanticscholar.org/CorpusID:256448536

Msr Mustafa and Verbeeck, K. and D. C. P. and V. B. G. (2012). An Intelligent Hyper-Heuristic Framework for CHeSC 2011. In M. Hamadi Youssef and Schoenauer (Ed.), *Learning and Intelligent Optimization* (pp. 461–466). Berlin, Heidelberg: Springer Berlin Heidelberg.

Munien, C., & Absalom, E. E. (2021). Metaheuristic algorithms for one-dimensional bin-packing problems: A survey of recent advances and applications. *Journal of Intelligent Systems*, *30*, 636–663. Retrieved from https://api.semanticscholar.org/CorpusID:233391045

Ochoa, G. (2012). Hyper-heuristics and cross-domain optimization. *Annual Conference on Genetic and Evolutionary Computation*. Retrieved from https://api.semanticscholar.org/CorpusID:20935259

Pichugina, O., & Matsyi, O. (2020). Boolean Satisfiability Problem: Discrete and Continuous Reformulations with Applications. *2020 IEEE 15th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, 623–627. Retrieved from https://api.semanticscholar.org/CorpusID:218564644

Pukhkaiev, D., Semendiak, Y., Götz, S., & Assmann, U. (2020). Combined Selection and Parameter Control of Meta-heuristics. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 3125–3132. Retrieved from https://api.semanticscholar.org/CorpusID:230999483

Qian, C., Tang, K., & Zhou, Z.-H. (2016). Selection Hyper-heuristics Can Provably Be Helpful in Evolutionary Multi-objective Optimization. *Parallel Problem Solving from Nature*. Retrieved from https://api.semanticscholar.org/CorpusID:11215972

Qu, Y., Li, Z., Liu, Q., Pan, M., & Zhang, Z. (2022). Crash/Near-Crash Analysis of Naturalistic Driving Data Using Association Rule Mining. *Journal of Advanced Transportation*. Retrieved from https://api.semanticscholar.org/CorpusID:252740665

Ribeiro, L. C. F., Roder, M., de Rosa, G., Passos, L. A., & Papa, J. P. (2023). Enhancing Hyper-to-Real Space Projections Through Euclidean Norm Meta-heuristic Optimization. *ArXiv*, *abs/2301.13671*. Retrieved from https://api.semanticscholar.org/CorpusID:245958742

Saad, A., & Al-Ghamdi, A. (2011). *Efficient Implementation of FP Growth Algorithm-Data Mining on Medical Data*. Retrieved from https://api.semanticscholar.org/CorpusID:16707116

Sabar, N. R., Ayob, M., Kendall, G., & Qu, R. (2015). Automatic Design of a Hyper-Heuristic Framework With Gene Expression Programming for Combinatorial Optimization Problems. *IEEE Transactions on Evolutionary Computation*, *19*(3), 309–325. https://doi.org/10.1109/TEVC.2014.2319051

Singh, A., Singh, D., Upreti, K., Sharma, V., Rathore, B. S., & Raikwal, J. (2022). Investigating New Patterns in Symptoms of COVID-19 Patients by Association Rule

Mining (ARM). *J. Mobile Multimedia*, *19*, 1–28. Retrieved from https://api.semanticscholar.org/CorpusID:251965940

Syakur, M. A., Khusnul, B., Khotimah, Mala, E., Rochman, S., & Satoto, B. D. (2018). *USING K-MEANS ALGORITHM AND FP-GROWTH BASE ON FP-TREE STRUCTURE FOR RECOMMENDATION CUSTOMER SME*. Retrieved from https://api.semanticscholar.org/CorpusID:219616792

Udomkasemsub, O., Sirinaovakul, B., & Achalakul, T. (2023). PHH: Policy-Based Hyper-Heuristic With Reinforcement Learning. *IEEE Access*, *11*, 52026–52049. Retrieved from https://api.semanticscholar.org/CorpusID:258818158

Velednitsky, M. (2017). Short combinatorial proof that the DFJ polytope is contained in the MTZ polytope for the Asymmetric Traveling Salesman Problem. *Operations Research Letters*, *45*(4), 323–324. https://doi.org/https://doi.org/10.1016/j.orl.2017.04.010

Wilfahrt, R., & Kim, S. (2017). Traveling Salesman Problem (TSP). *ACM SIGSPATIAL International Workshop on Advances in Geographic Information Systems*. Retrieved from https://api.semanticscholar.org/CorpusID:263869809

Yates, W. B., & Keedwell, E. C. (2018). Offline learning for selection hyper-heuristics with Elman networks. *Artificial Evolution: 13th International Conference, Évolution Artificielle, EA 2017, Paris, France, October 25–27, 2017, Revised Selected Papers 13*, 217–230.