# COMPARATIVE EVALUATION OF HARMONY SEARCH ALGORITHM AND TABU SEARCH ALGORITHM ON SHELF SPACE ALLOCATION PROBLEM (A CASE STUDY OF SUPREME SUPERMARKET FATE ROAD, ILORIN)

**BY**

**BALOGUN, AYOMIDE AYOYINKA**

**HND /23/COM/FT/0322**

**BEING A PROJECT SUBMITTED TO:**

**THE DEPARTMENT OF COMPUTER SCIENCE**

**INSTITUTE OF INFORMATION, COMMUNICATION AND TECHNOLOGY**

**KWARA STATE POLYTECHNIC ILORIN, KWARA STATE**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF HIGHER NATIONAL DIPLOMA (HND) IN COMPUTER SCIENCE**

**SUPERVISOR:**

**DR. AGBOOLA, O.M.**

**APRIL, 2025**

## CERTIFICATION

This is to certify that this research study was conducted by **BALOGUN AYOMIDE AYOYINKA** with the matric number **HND/23/COM/FT/0322** and had been read and approved as meeting therequirements for the award of Higher National Diploma (HND) in the Department of ComputerScience, Institute of Information Communication Technology (IICT). Kwara State PolytechnicIlorin

______________________________ ______________________________

**DR. AGBOOLA, O. M.** Date

*(Project supervisor)*

______________________________ ______________________________

**MR. OYEDEPO, F. S** Date

*(Head of Department)*

______________________________ ______________________________

**EXTERNAL** Date

## DEDICATION

This research project is dedicated to the Almighty God that preserved me throughout my programme at the Kwara State Polytechnic, Ilorin. For his mercy, favor and unending grace that have always been with me and to my beloved parents who have stood by my sides all the time.

## ACKNOWLEDGEMENTS

Firstly, I acknowledge the Almighty God, who has always been precious in strengthening me and giving me the inspiration to keep me working and able to complete this research. This report cannot be completed and will be impossible without a thorough supervision and proper guidance. In that regard, I sincerely acknowledge the effort of my supervisor and my mentor, **DR. AGBOOLA, O. M**, who despite his tight schedule as the Director of the IICT and other activities, found time to make sure this research is carried out properly and well documented, and ensure that it makes a brilliant contribution to my academic knowledge.

My appreciation also goes to the **Head of the Department MR. OYEDEPO, F.S and all the member of staff in the great department of Computer Science, Kwara State Polytechnic, Ilorin**, for their Contribution towards the success of my programme.

It is a great pleasure to express my special thanks to my parents, **Apostle S.A Balogun and Mrs C.O Balogun and my siblings** for giving me their spiritual, moral and financial supports. May almighty God grant them long life and prosperity, and to my family at large. To my project partner and a very close friend **Amos Peter Adewunmi** thanks for your contribution and your understanding during the course of this research.

It is also with love and tender heart to remember myfriends, I pray we will not labor in vain; I thank all my friends and course mates especially **Badmus Oluwatobiloba,Taiwo Hassanat Kehinde** and those who have support my academic right from the beginning and I appreciate your cooperation towards my success in the polytechnic, may Almighty God reward you abundantly.

# TABLE OF CONTENTS

## ABSTRACT

*Shelf space allocation plays a crucial role in retail optimization, influencing sales performance, customer satisfaction, and inventory management. This research compares the effectiveness of The Harmony Search Algorithm (HSA) and Tabu Search Algorithm (TSA) in solving the shelf space allocation problem at Supreme Supermarket, Fate Road, Ilorin. The primary objective is to evaluate these metaheuristic algorithms in terms of solution quality, constraint satisfaction, and execution time. HSA mimics the improvisation process of musicians, adjusting solutions iteratively based on memory consideration and pitch adjustment. Conversely, TSA uses a dynamic memory structures, the Tabu list to prevent cycling and enhance local search capabilities. The study assesses algorithm performance based on the penalty points incurred due to constraint violations and execution time. Experimental results indicate that HSA provides better constraint satisfaction with a lower penalty point (39) compared to TSA (51 penalty points) in its best iteration. However, TSA exhibits slightly faster execution time (1.25 sec) than HSA (134 sec). The findings highlight a trade-off between accuracy and speed, reinforcing the No Free Lunch theorem, which states that no single algorithm is universally superior. The significance of this study lies in its practical implications for retailers seeking optimized product arrangement strategies that maximize space utilization, improve customer experience, and enhance supply chain efficiency. Future research could explore integrating machine learning techniques to dynamically adjust shelf configurations based on demand patterns.*

*Keywords: Harmony Search Algorithm. Tabu Search Algorithm, retail optimization,metaheuristic algorithms,no free lunch theorem.*

# CHAPTER ONE

## 1.0 INTRODUCTION

Shelf space allocation, refers to the process of allocating, assigning and managing the physical space on shelves in a shop, store, warehouse or a depot. The goal of shelves space allocation is to maximize the use of available shelf space, optimize product placement and improve the overall shopping experience for customers,Therefore a proper usage of shelves needs to consider some factors, which are:

Product demand: A particular, product that is being demanded regularly should be allocated more space unlike the one with slower demands,Product categories: Arranging products that are similar will make it easier for customers to find what they are looking for,Product sizes and shape: Ensuring that products fit comfortably on the shelves and are easy to access,Labeling and Signage:Clearly labeling shelves and products will help customers to find their ways in the store, Inventory management: Ensuring that shelves are stocked with the right quantities of protects to meet customers' demands.

To allocate shelves space manually has a lot of drawbacks that inspired me into working on this research, in other to compare and evaluate Harmony search algorithm and Tabu search algorithm to allocate shelves space in a super market.

Firstly, Harmony Search Algorithms and Tabu Search Algorithms are Metaheuristic Algorithms that use heuristic algorithm to reach global optimal. Harmony Search Algorithms is a population-based algorithms (metaheuristic) inspired by the process of musicians improvising music. It was first proposed by Zong Wot Geem in 2001.

Tabu Search Algorithm is also a meta-heuristic algorithms that uses local search methods to iterate from one potential solution to an improved solution. It was first proposed by Fred Glover in 1986.

Therefore, this research work is focused on comparing and evaluating the Harmony search Algorithms and Tabu search Algorithms to effectively allocate shelves space.

## 1.1 STATEMENT OF PROBLEM

Manually, allocating shelves space in a supermarket can lead to the following problems: **Inefficient of Space**: manually allocating disorganized space can lead to poor utilization space leading to wasted space and reduced product facings,Stock out and overstocking: improper manual allocation can lead to stock acts or overstocks of products, Difficulty in managing products categories: manual allocation can make it challenging to manage products categories leading to a  store layout, Increased Labor costs: manual allocation requires significant labor hours to plan, implement and maintain shelf layouts, Limited Data Analysis: manual allocation relies on manual data collection and analysis which can be time consuming, etc.

My Research work based on solving these above problems by optimizing the shelves space allocation by comparing and evaluating Harmony Search algorithms and Tabu search Algorithms for shelves space allocation.

## 1.2 AIM AND OBJECTIVES

This research work is aimed at solving shelves space allocation using Harmony search

Algorithms and Tabu Search algorithm. The implementation was done using Python programming language.

Objectives are;

i. adaptation of Harmony Search Algorithms and Tabu Search Algorithm;
ii. implementation of the Adapted Algorithms, using Python and MySql as database; and
iii. comparative evaluation of the Harmony search algorithms and Tabu search algorithms to shelve space allocation.

## 1.3 SIGNIFICANCE OF THE STUDY

This research work offers the following advantages:

To improve space utilization, enhanced product visibility, better stock management, increased sales, reduced labor costs, increased efficiency, improved supply chain efficiency, enhanced customer experience, data driven decision making with scalability and flexibility.

## 1.4 SCOPE OF THE STUDY

The scope of the sturdy covers the operation of Supreme Supermarket located along Fate Road, opposite Kwara Mall, Ilorin, Kwara State.

## 1.5 THE ORGANISATION OF THE STUDY

This study is categorized into five chapters. Chapter one consists of Introduction to the study, Statement of the problem, Aim and Objectives, Significance of the study, Scope of the study, organization of the study, and Definition of terms. Chapter Two discusses the review of literature on harmony search algorithms and Tabu search algorithms. Chapter Three presents the methodology used in the study, including parameter settings and performance metrics. Chapter Four discusses the results and analysis of the study. Chapter Five presents the summary, conclusion, and recommendations.

## 1.6 DEFINITION TERMS

- **Comparative**: This is an act of comparing two or move things, ideas, entities in order to identify their similarities, difference and relationships.
- **Evaluation**: This is a systematic process of assessing, analyzing and interpreting the quality, effectiveness and impact of a program project a service.
- **Shelves**: are structures used to store and display objects, products, or materials in a variety of settings
- **Space**: This refers to the physical area or volume within a store, ware house or other facility where products, inventory sale are stored, displayed or used.
- **Algorithm**: these are well defined procedures that take some input and produce a corresponding output. They are essential in computer science.
- **Allocation**: refers to the process of assigning a distributing resource.
- **Heuristic:** is a problem-solving approach that uses practical methods or shortcuts to find a solution that is good enough, but not necessarily optimal. Heuristics are often used when the problem is complex or the optimal solution is difficult to find.
- **Metaheuristic**: is a higher-level heuristic that guides the search process to find good solutions to complex optimization problems. Metaheuristics are often inspired by natural phenomena, such as evolution, swarm behavior, or physical processes.

Examples include genetic algorithms, simulated annealing, and ant colony optimization.

- **Optimization**: is the process of finding the best solution to a problem, often by minimizing or maximizing a objective function. Optimization problems can be found in various fields, such as engineering, economics, and computer science.
- **Global optimum**: is the best possible solution to an optimization problem, considering all possible solutions. It is the optimal solution that achieves the minimum or maximum value of the objective function.
- **Elasticity**: refers to the ability of a system or material to stretch or deform and then return to its original shape. In optimization and operations research, elasticity can refer to the flexibility of a system or solution to adapt to changes or uncertainties.

# CHAPTER TWO

## 2.0 LITERATURE

Gwang and Ukyeong (2021), worked on “Integrated planning for product selection shelf space allocation and replenish decision with elasticity and positioning effects”. The introduction emphasizes the importance of shelf space allocation, product selection and replenishment decision in retail. These processes affect customer demand, cost and profit. Effective systems are needed to maximize profits while addressing challenges like limited shelf space, product diversity and customer demand elasticity. This research work aimed at developing an integrated model for optimizing shelf space allocation, product selection and replenishment decisions, with the objective of considering real-world factor such as space and cross space elasticities, product positioning effects and two-dimensional display spaces. The methodologies processes are;

Development of a mixed-integer non-linear programming (MINLP) model, two heuristic solution methodologies hybrid Tabu search and hybrid genetic algorithms and Comparative experiments using small and large datasets.

The researcher encounters various problems such as; Complexity of decision-making processes involving multiple interdependent factors. Challenges in integrating spaces elasticities, product positioning and inventory replenishment into a single optimization model, limitations of existing algorithms in solving large dataset effectively.

The researchers find out that the proposed hybrid algorithms effectively solves small datasets with high accuracy, they also find out that Algorithm 1 (tabu- search) and performed

Algorithm 2 (genetic algorithms) in large datasets in terms of computation time and solution quality. The study also highlights the significance of considering elasticity and positioning effects in retail optimization. The effective use of algorithms can help retailers overcome challenges in optimizing complex retail operations. Retailers should adopt integrated approaches to product selection, shelf-space allocation and replenishment to maximize profits. In conclusion, future researchers should focus on addressing uncertainties in demand and using advanced machine learning techniques for better parameter estimation and demand forecasting.

Carolina, Janeiro (2014), worked on Optimization Algorithms for the shelf Space Allocation problem. The dissertation emphasizes the importance of shelf space management in retail due to its limited nature. It explores how factors like the number of facings, product position and price affect consumer behaviors, while existing software and literature tackle the Shelf Space Allocation Problem (SSAP), they often lack real-world applicability. Inspired by a Portuguese supermarket chain, the study proposed novel biased, Random key Genetic Algorithms to optimize shelf allocation. The research work aimed at exploration of the use of Meta heuristics specifically Biased. Random-key Genetic Algorithms (BRKGA) to solve the SSAP. With the objectives of bridging the gap between theoretical models and practical applications, testing the algorithms on real-world data. With the methodologies of the following processes;

Development of two BRKGA-based algorithms tailored for SSAP, Testing algorithms using real data from a Portuguese retail company and Analysis of algorithms, performance under practical constraints and execution time.

The gaps or problems encounter are; Limitations on existing software which requires extensive human interaction, Existing SSAP models lack practical constraints and have hard to estimate parameters, challenges in integrating theoretical models with real world retail scenarios. During the research work the researcher finds out that the proposed BRKGAs proved effective and practical far solving SSAP in real-world cases and they also find out that both algorithms managed to allocates products efficiently, considering families and constraints while boosting sales.

In Summary, Retailers should adopt advanced algorithms to improve shelf space management, it is also recommended that practical applicability should guide algorithm development to align with real-world constraints. Future researchers should explore integrating other retail dimensions such as inventory and demand forecasting.

Seyed, Sajadi and Ahmadi (2022), conducted a research titled "An integrated optimization model and meta heuristics for assortment planning Shelf space Allocation and Inventory management of perishable products". The study focuses on optimizing assortment or planning shelf space allocates and inventory management for perishable products in retail. These decisions are crucial to maximize sales and profit under constraints like shelf space, budget and perishability. The paper proposes on integrated mathematical model and Meta heuristic approaches to addresses these challenges effectively. This study is aimed to development of mathematical model for assortment planning Shelf space allocation and inventory management for and perishable products. With the objectives to: design meta heuristic algorithms (genetic algorithm and vibration damping optimization) for solving the model, especially for large-scale problem and to validate the proposed methods through a real-world case study and computational experiments. With the following methodologies; Development of a mixed integer non-linear programming model considering perishability, space elasticity and substitution behavior; Implementation of genetic algorithms (GA) and vibration damping optimization (VDO) for solving large scale instances; Validation using real-world data from a retail case in Iran.

The researchers faced some challenges while carrying act the research which are (i) Complexity to integrating assortment planning shelf space allocation and inventory management (ii) High computational demands for solving the proposed non-linear model especially for large datasets (iii) challenges in balancing costs (e.g. perishable storage, ordering)with sales revenue. During the research, the researchers find out that the proposed Genetic Algorithms and Vibration damping optimization algorithms provided near optimal solutions efficiently for small and large datasets. They also find out that the integrated approach increased profit by optimizing space usage, inventory levels and supplier selection, sensitivity analyses reveal the significant impact of demand, price and shelf space on profitability. In summary, Retailers should adopt integrated optimization models to improve decision-making for perishable products and also metaheuristic algorithms like GA and VDO and effective to solving complex retail optimization problems. It is recommended that future studies should explore the incorporations of dynamic pricing and demand forecasting into the model.

Kateryna Zerniachowsky (2021) worked on A Genetic Algorithm for the Retail shelf Space Allocation problem with Virtual Segments. Shelf space allocation is a critical decision-making process for retailers influencing profitability, client satisfaction and stock management. This research focuses on addressing challenges in small retail stores with limited shelf space, proposing a model to combine practical merchandising strategies and optimization techniques. The research work is aimed at creating a practical model for retail Shelf Space allocation; with objectives of (a) To maximize profit by considering visual merchandising practices and constraints (b) To integrate and improve the allocation using genetic algorithms. The methodologies used in this research are: (i) A mathematical model using genetic algorithms was developed (ii) constraints like shelfs dimensions, product characteristics and merchandising tactics were considered (iii) Computational experiments were conducted on with the CPLEX solver. The researchers faced different problems such as (i) Small retailers face challenges in optimizing scarce shelf space. (ii) Existing models do not adequately reflect complex category management rules or practical constraints. The researchers find out that the proposed genetic algorithms effectively optimize Shelf space allocation and they also find out that results from computational experiments showed the model works well for both small and large product numbers. The study presents a robust frame work combining theoretical and practical insights for retail merchandising and also the proposed solution can improve profitability and efficiency in retail shelf space management.

Muchen and ChiaPing (2007), a data mining approach to product assortment and shelf space allocation. Retailers face challenges in managing limited shelf space for competing products while meeting customers' demands and responding to market competition. This paper explores product assortment and shelf space allocation using a data mining approach focusing on multi-level association rule mining to uncover relationships between products and improve retail management's strategies. This research work aimed to develop a systematic approach for product assortment and shelf space allocation in retail stores. The objectives were to leverage multi-level association rule mining for analyzing transactional data and optimizing shelf management and to enhance profitability and customer satisfaction through effective retail space usage. Data mining approach employing multi-level association rule, mining is used to replace traditional space elasticity models in the study, Mathematical model for product assortment and Shelf allocation were also constructed, with implementation tested on a retail database. The problems of the previous research work is that traditional methods requires estimating numerous parameters, learning to errors and inefficiencies and also space elasticity models are complex and lack of flexibility in responding to dynamic retail environments. The researchers find out that multi-level association rules provided reliable insights for shelf management during the research, and also, they find out that the proposal approach reduced parameter estimation errors and adapted quickly to market charges. Retailers should adopt data mining to improve shelf space management. In conclusion the proposed method ensures effective product assortment, highlights basic product to maintain store image and optimally utilizes shelf space for added products, it also robust data driven framework to enhance retail performances and customer satisfaction.

Geismar,Dewande,Murthi and Sriskandarajah (2015).Conducted a sturdy titled Maximizing Revenue through two-dimensional Shelf space Allocation. This study addresses the problem of optimally allocating contiguous rectangular presentation spaces for retail shelf space to maximize revenues. It considers both horizontal and vertical display dimensions and emphasizes the impact of display location on sales. The model provides solution for in store presentations, features advertising displays and webpage layouts, emphasizing flexibility and effectiveness in revenue generation. This research work aimed to maximize revenue by developing a novel approach to two-dimensional shelf-space allocation, with objectives of including factors such as vertical location, tallness and multiple shelf displays on the optimization model and also to improve revenue generation by efficiently arranging products displays. The methodologies that is the techniques used in solving the problem of the previous research work is as follows: The problem is modeled as a maximum weight independent set problem on a network ISO, two sub problems are solved assigning products to cabinets and arranging units within cabinets, lastly the study uses a combination of integer programming and heuristics approaches, validated through computational studies on both real-world and simulated data. Some of the gaps the authors tried to cover during their research work are as follows: (i) Retailers faces limited shelf space and increasing, competition, making effective space utilization crucial (ii) Current software solutions lack the capability to globally optimize displays or account for two-dimensional arrangements. The researchers found out some information during their research which are as follows: The proposed method generates revenue within 1% of the optimal for real data and 5% for stimulated data; it significantly improves the efficiency of shelf space-allocation compared to existing heuristics (iii) The approach is robust and adaptable for various retail environments, including DVD stores and supermarkets. It is recommended that retailers should adopt two-dimensional display optimization technique to enhance revenue. Future researchers could extend the model to include cross-product elasticities, different product dimensions and applications in digital advertising. In conclusion, the proposed methodology provides flexibility and efficiency in product arrangements, catering to the unique characteristics of different products.

Manuel, Tobias,Alexander (2022), Worked on A model and solution approach for store-wide shelf space allocation. The document discusses the limited sales area in retail stores which is a cost intensive resource. It emphasizes the hierarchical planning process for space allocation, including store, layout planning, category space, assignment, and product allocation. The importance of optimizing their processes to enhance profitability is highlighted. This research work aimed to introduce a store wide shelf space model that optimizes shelf space assignments for product categories based on profit contribution with the objective to decompose the problem into sub problems and provide a solution approach suitable for large-scale retail application, with the following methodologies (i) Development of a mathematical model and solution approach (Swiss Optimization) that integrates product allocation and store-wide space planning (ii)Decomposition of the problem into two sub problems product allocation model (SAM). (iii) Validation using a case study with a major European retailer and simulation with test data. The previous research work has different problems that lead to the development of this system such as Limited shelf space requires trade-offs in category allocation, current models fail to integrate category specific shelf types and product level data effectively and computational complexity due to the large scale and interdependencies between allocation decisions. It was found out that the proposed solution increases store profit by up to 3.2% in a case study, significant improvement in profit contributions and efficient space allocation across division and categories and also finds out the decomposition approach efficiently handles large scale problems. In Summary, The Swiss optimization approach is effective in optimizing store-wide shelf space allocation while considering practical constraints, it is also recommended that for further research include exploring additional demand factors, like substitution rates and marketing strategies. In conclusion, Retailers can enhance profitability by adopting systematic and integrated space allocation models.

Hakan (2020), Worked on Joint shelf design and shelf space allocation problem for retailers. The documents highlight the challenges in retail space management, emphasizing the importance of optimizing shelf space allocation and planogram designs to maximize revenue. The Covid-19 pandemic accelerated the need for efficient space utilization. Existing approaches often rely on intuition or assume fixed shelf dimensions, leading to suboptimal results. The research work aimed at introducing the Joint Shelf Design and Shelf Space Allocation (JSD-SSA) problem to maximize retailer revenue with objectives to develop a mathematical programming model to optimize shelf design and sky placement and to addresses computational challenges through a hybrid decomposition-based approach with methodologies (a) A mixed integer programming (MIP) model for the JSD-SSA problem (b)Decomposition into two sub problems using particles swarm optimization (PSO) and constraints programming (CP). (c) Validation through experiments and a case study. The previous or traditional method for shelf space allocation have several problems that lead to this research work such as (i) Limited retail shelf space and fixed Shelf dimensions in existing models (ii) Computational challenges in solving realistic problem sizes (iii) Lack of integration between Shelf design and skill allocation. The researchers find out the hybrid PSO-CP approach effectively solves large problem instances within reasonable timeframe, optimized Shelf designs and increase retailer profits by up to 22% in certain scenarios, shapes variations and tighter planogram Spaces significantly impact revenue. In Summary, Retailers can enhance revenue by adopting dynamic shelf designs and efficient allocations. It is recommended that further research is to integrate additional real-world constraints such as Shopper behavior and marketing. In conclusion, the proposed methodology offers practical solutions to improve retail operations in a competitive landscape.

Chase C. Murray, Debabrata, Abhijit(2010), Worked on Joint optimization of product price, Display Orientation and shelf-space Allocation in Retail Category Management. The document addresses the critical operational challenge faced by Consumer-Packaged Goods (CPG) retailers: the efficient allocation of limited shelf space. The increasing competition for shelf space, rising operational costs and growing, product assortments are identified as primary drivers necessitating advanced shelf management techniques. It emphasizes the role of integrated decision concerning pricing, display and shelf-space allocation in enhancing profitability. This research work aimed at developing a model that jointly optimize product prices, display orientations, and shelf space allocations to maximize retailer profits. The objectives of the research is: To incorporate two dimensional and three-dimensional aspects of shelf and product geometries into shelf management and To address interdependencies among pricing, product placement and consumer demand through a unified framework with the following methodologies (i) a mixed, integer Non-linear programming (MINLP) model capturing interactions between product, prices display facing areas, orientations and shelf space locations (ii) use of BONMIN algorithms (branch and bound outer approximation and hybrid methods) to solve optimization problems (iii) validation through numerical simulations and analysis of small and large scale problem instances. Existing shelf-space allocation models focus primarily on heuristic approaches and fail to capture joint optimization of pricing and spatial allocation, complexity in modeling consumer demand influenced by product placement and pricing, lack of practical tools for managing multi-dimensional shelf arrangement in retail are the problems of the existing systems of allocating Shelf space. During the research work, researchers find out that Joint optimization can significantly enhances profitability by aligning product placement, pricing and orientation strategies, also, the proposed model effectively balances product profit ability demands elasticity and spatial constraints and lastly, performance analysis of BONMIN algorithms demonstrated the feasibility of solving large scale problems within realistic time frames. In Summary, Retailers should adopt integrated decision-making significant framework for pricing, product placement and shelf space management. The study also highlights the importance of empirically estimating model parameters to align with specific market realities. It is recommended that future research could explore incorporating shopper behavior and dynamic market conditions into the model.

BurcuGencosman, Mehmet A. Beger (2021).Worked on Exact Optimization and Decomposition Approaches for 2D Shelf Space Allocation. The study focuses on retail shelf space allocation, addressing complexities in determining optimal arrangement for products on shelves it emphasizes the importance of space and location elasticities in maximizing profit and introduces a two dimensional shelf space allocation problem (2D SSAP) with new features, including rectangular arrangement constraints with the aim of optimizing the allocation of products on retail shelves to maximize profit with the objectives of developing exact solution methods like a mixed-integer Linear programming (MIP)model, logic-based benders decomposition (LBBD) and 2 stage iterative algorithms (IP1/IP2) and to provide solutions that consider rectangular arrangements, space elasticity with methodologies or techniques which are as follows;(i) Formulated a MIP model for 2DSSAP (ii) Developed LBBD and IP1/IP2 solutions methods to solve large and real-world instances (iii) Benchmarked the model performance on optimality and computational efficiency. During this research work, researchers/authors find out the problem of the existing traditional method of allocating shelves space which are:(i) Retailers face challenges in efficient shelf space utilization due to competition, product variety and dynamic customer demands (ii) Existing methods in literature often rely on heuristic, which do not guarantee optimal solutions, especially for large sale problems. The researchers find out that the proposed IP1/IP2 algorithm is the most efficient, solving large real-world instances. (up to 250 products) within minutes, providing optimal solutions (ii) Implementing these methods in a local bookstore increased profit by up to 16.56%. In Summary, the exact methods developed (LBBD and 1P1/1P2) offer robust solutions for 2DSSAP, Retailers can leverage those approaches for efficient shelf space management, enhancing profits and adapting to market demands. In conclusion, the study contributes to the literature by addressing rectangular display constraints and extending its applications to allocation problems in different sectors.

# CHAPTER THREE

## 3. 1 RESEARCH METHODOLOGY

- Harmony search Algorithm
- Tabu search Algorithm

### 3.1.1 Harmony search algorithm.

The Harmony Search Algorithm (HSA) is a population based metaheuristic optimization algorithm which is inspired by the process of musicians improvising harmonies. It was proposed byZong Woo Geem*et al* (2001), It is an algorithm that searches near global solutions by using 3 major step to search for it solution which are.

1. Harmony memory consideration rate (HMCR) this is use to consider whether to use from the harmony memory. i.e. playing known piece of music
2. Pitch adjustment rate (PAR) this control the adjusting of a solution a slight adjustment of the pitch i.e. play something similar to the famous piece of music.
3. Random selection this generate randomly i.e. random composition of new note.

### 3.1.2 Tabu search algorithm.

Tabu search algorithm is a metaheuristic optimization algorithm that escapes getting stuck in local optima by using memory to store recently visited solutions. Its focus on promising regions of solution space,exploring different regions of the solution space to avoid getting stuck in a local optimum and uses the Tabu list as a short-term memory, preventing the algorithm from revisiting solutions that have already been explored. It was proposed by Fred Glover in 1986 and formalized in 1989.

## 3.2 Analysis and Problems of the Existing System

The existing system for shelf space allocation in Traditional ways of allocating shelves and product is a manual, time-consuming and tedious process.It often-required significant effort and expertise. Additionally, manual allocationis prone to errors, inconsistencies and redundancies, which could lead to allocating conflict of product and inappropriate use of available shelf space. This problem of shelf space allocation is improved by introducing of a set of constraints, which is typically divided into two types: hard and soft constraints.

Hard constraints are those that should be met or satisfied at all times, for example, cannot be violated, while violations of the soft constraints should be avoided if possible, else there's penalty for violation. The following shows the hard and soft constraints as stated by the supreme supermarket, Ilorin, Kwara state, and gathered online regarding shelf space allocation.

**Hard Constraints**

H1: All Columns in a shelf must be assigned a product.

H2: A product must not be assigned to two columns.

H3: The weight of product in a column of a shelf must not be more than 25000g and must not be less than 10000g

**Soft Constraints**

S1: Arrange average demand product on Shelf 1 and 2 in column 1, and 2

S2: Arrange high demand product on Shelf 3, 4 and 5 in column 3, 4 and 5.

S3: Arrange low demand product on shelf 6 and 7 in column 6 and 7.

S4: Arrange product of the same type but difference sizes on the same column.

**Penalty for Soft Constraints**

Soft constraints should have a smaller penalty to encourage optimal solutions without completely discarding less ideal ones.

1. If an average demand product is placed outside column and shelf 1 and 2, add a penalty point of +1 per misplaced product.

2. If a high-demand product is placed outside column and shelf 3, 4, or 5, add a penalty point of +1 per misplaced product.

3. If an average demand product is placed outside column and shelf 6 and 7, add a penalty point of+1 per misplaced product.

4. If products of the same type but different sizes are placed in separate columns instead of together, add a penalty point of +1 per violation.

For example, if a shelf has two misplaced high- demand products and one misplaced average-demand product, the total penalty would be (1+1) + 1 = 3,the penalty point is = 3

**Final Calculation of Fitness Values**

The final fitness score is computed by calculating total penalty point and find the allocation with the minimum total penalty point and the number of penalty violated. For instance, if a shelf violated three soft constraints (1+1+1) the penalty will be 3 constraints violated with penalty point of 3 point.

This method ensures that solutions remain competitive while still discouraging violations, soft constraint should not be violated but if, there should be a penalty for it.Soft constraint penalties allow some flexibility but still push for better arrangements.

## 3.3 Analysis of the Proposed System

This section entails step by step solutions of the existing problem of shelf scheduling by using the proposed algorithms; the experiment will be carried out by using dataset of the Shelves and beverages products obtain at supreme supermarket along Fate road Ilorin, Kwara State.

**STEPS OF HARMONY SEARCH ALGORITHM**

**STEP 1** Initialize the problem and HSA parameters

1. Input data instance of the allocation problem

2. set the HSA parameters (HMCR, PAR, NI, and HMS).

**STEP 2** Initialize the harmony memory

1 Generate Solution of the harmony memory

2. Identify the worst solution in HM

**STEP 3** Improvise a new harmony

1. HMCR (Memory Consideration) with probability HMCR, solution is selected from

existing harmonies. Ensures good patterns are reused.

2. PAR (Pitch Adjustment) The probability of adjusting the selected solution

3. Random Selection If the value is not chosen from memory (based on HMCR), a completely random solution is generate

**STEP 4** Update the harmony memory

1. Is the current solution better than the worst existing one? If yes

2. Include current solution to the HM.

3. Exclude worst solution from HM.

**STEP 5** Check the stop criterion

1. Is the number of improvisation complete? If no

2. Repeat STEP3 and STEP4

## ADAPTATION OF HARMONY SEARCH ALGORITHM TO SHELF SCHEDULING PROBLEM

### THE STEPS ARE AS FOLLOWS

**STEP 1:** Initialize the problem and HSA parameters

1. State the input data used for the optimization problem, including numbers of shelves, numbers of a column in each shelf and Number of product

-Number of Shelves: 7

-Number of Column in each shelf: 7

-Number of Product: 84

2. Set the HSA parameters such as HMCR (Harmony Memory Consideration Rate), PAR (Pitch Adjustment Rate), NI (Number of Improvisations), and HMS (Harmony Memory Size).

**STEP 2:**Initialize the harmony memory

1. Create the allocation of the harmony memory, HM = {a1, a2, ……., aHMS}, where each allocation represents a possible solution to the allocation problem.

2. Recognize the worst allocation in HM, worst ∈ {a1, a2, ..., aHMS}, which will be updated if a better solution is found.

**STEP 3:** Improvise a new harmony

1. Generate a new allocation, a0, representing a potential product allocation solution.

2. For each product in a0, allocate a shelf and a column in a shelf based on the following conditions:

**Hard Constraints**

H1: All Columns in a shelf must be assigned a product.

H2: A product must not be assigned to two columns.

H3: The weight of product in a column of a shelf must not be more than 25000g and must not be less than 10000g

**Soft Constraints**

S1: Arrange average demand product on Shelf 1 and 2 in column 1, and 2

S2: Arrange high demand product on Shelf 3, 4 and 5 in column 3, 4 and 5.

S3: Arrange low demand product on shelf 6 and 7 in column 6 and 7.

S4: Arrange product of the same type but difference sizes on the same column.

3. If (U(0,1) ≤ HMCR) then choose shelf allocation based on the memory consideration.

4. If (U(0,1) ≤ PAR) thenapply pitch adjustment for shelf schedulingbased on the best and worst values in the shelf

5. If If (U(0,1) > HMCR and U(0,1) > PAR) then select a random value for allocating a productto shelf.

6. Repeat steps 2-5 for all product in a0

**STEP 4:** Update the harmony memory

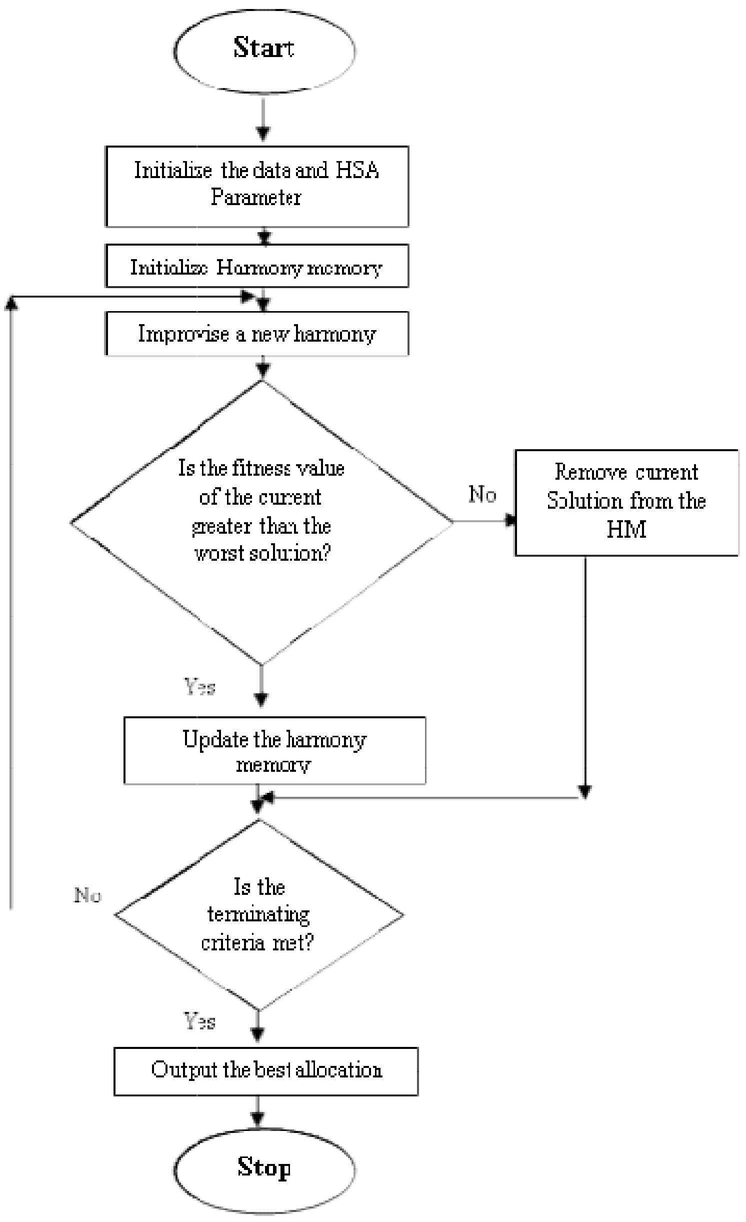
1. Evaluate the fitness value for the New Harmony solution, a0. i.e summation of the total penalty points

2. If (a0) >(worst) then replace worst with a0 in HM.

**STEP 5:** Check the stop criterion

1. Repeat Steps 3 and 4 until the termination criterion is met, which is specified by NI (Number of Iteration).

2. Return the best shelf scheduling solution found in the harmony memory



Start
Initialize the data and HSA Parameter
Initialize Harmony memory
Improvise a new harmony
Is the fitness value of the current greater than the worst solution?
No
Remove current Solution from the HM
Yes
Update the harmony memory
No
Is the terminating criteria met?
Yes
Output the best allocation
Stop

***Fig 1.0 Flowchart of the Harmony Search Algorithm***

By applying the Harmony Search Algorithm to this shelf space allocation problem, we can systematically generate allocation of products to the available shelves which satisfy both hard and soft constraints.

**Harmony Search Algorithm parameter setting**

HMCR (Harmony Memory Consideration Rate) = 0.8

PAR (Pitch Adjustment Rate) = 0.4

HM (Harmony Memory Size) = 50

NI (Number of Iteration) = 50

### 3.1.2 Tabu search algorithm.

**STEPS OF TABU SEARCH ALGORITHM**

**Step 1:** Initialize the Problem and TS Parameters

1. Input Data:

2. Define Parameters: Initial Solution: A feasible product allocation that satisfies hard constraints.

Tabu List Size (TSize): Defines how many previous moves are stored.

Max Iterations (NI): Number of iterations to perform.

Aspiration Criteria: Allows overriding the Tabu status if a move results in the best solution so far.

**Step 2:** Generate an Initial Neighboring Solution

Considering the hard and soft constraint.

**Step 3:** Define Neighborhood Search

1. Generate Neighboring Solutions.

2. Evaluate Each Neighbor Solution

Use a fitness value to evaluate solutions based on constraint satisfaction.

**Step 4:**Tabu Check

1. Check Tabu List to prevent recently explored moves from being revisited for Tabu Size iterations.

2. Aspiration Criteria: If a move is in the Tabu list but results to the best solution so far, it isaccepted.

**Step 5:** Update the Best Solution If a new solution is better than the current best solution, update it. Continue searching until NI (max iterations) is reached.

**Step 6:** Stopping Criteria the algorithm stops when:

1. Maximum iterations (NI) are reached.

2. No better solution is found after a certain number of iterations.

## ADAPTATION OF TABU SEARCH ALGORITHM TO SHELF SPACE ALLOCATION PROBLEM

To adapt shelf space allocation into each of the Tabu search algorithm steps is as follows:

**Step 1:** Initialize the Data and TS Parameters

1. Input Data:

Number of Shelves: 7

Number of Columns per Shelf: 7

Number of Products: 85

2. Define Parameters: Initial Solution: A feasible product allocation that satisfies hard constraints.

Tabu List Size (TSize): Defines how many previous moves are stored. Max Iterations (NI): Number of iterations to perform.Aspiration Criteria: Allows overriding the tabu status if a move results in the best solution so far.

**Step 2:** Generate an Initial Neighboring Solution

Allocate a shelf and a column in a based on the following conditions: :

**Hard Constraints**

H1: All Columns in a shelf must be assigned a product.

H2: A product must not be assigned to two columns.

H3: The weight of product in a column of a shelf must not be more than 25000g and must not be less than 10000g

**Soft Constraints**

S1: Arrange average demand product on Shelf 1 and 2 in column 1, and 2

S2: Arrange high demand product on Shelf 3, 4 and 5 in column 3, 4 and 5.

S3: Arrange low demand product on shelf 6 and 7 in column 6 and 7.

S4: Arrange product of the same type but difference sizes on the same column.

**Step 3:** Define Neighborhood Search

1. Generate Neighboring Solutions: Swap products between two columns within the same shelf. Move products between two different shelves while maintaining weight constraints. Shift a product to an empty space if possible.

2. Evaluate Each Neighbor Solution:

Use a fitness value to evaluate solutions based on constraint satisfaction. Prioritize solutions that satisfy more soft constraints while maintaining hard constraints.

**Step 4:** Apply Tabu Restrictions to Maintain a Tabu List to prevent recently explored moves from being revisited for TSize iterations.

Aspiration Criteria: If a move is in the tabu list but results to the best solution so far, it isaccepted.

**Step 5:** Update the Best Solution If a new solution is better than the current best solution, update it. Continue searching until NI (max iterations) is reached.

**Step 6:** Stopping Criteria: The algorithm stops when:

1. Maximum iterations (NI) are reached.

2. No better solution is found after a certain number of iterations.

**Tabu Search Algorithm Parameters Settings**

TSize (Tabu List Size): = 50

NI (Max Iterations): = 50

No better solution is found after = 7 iteration

Aspiration Criterion: = Override tabu restriction if a new solution is better than the best-known solution

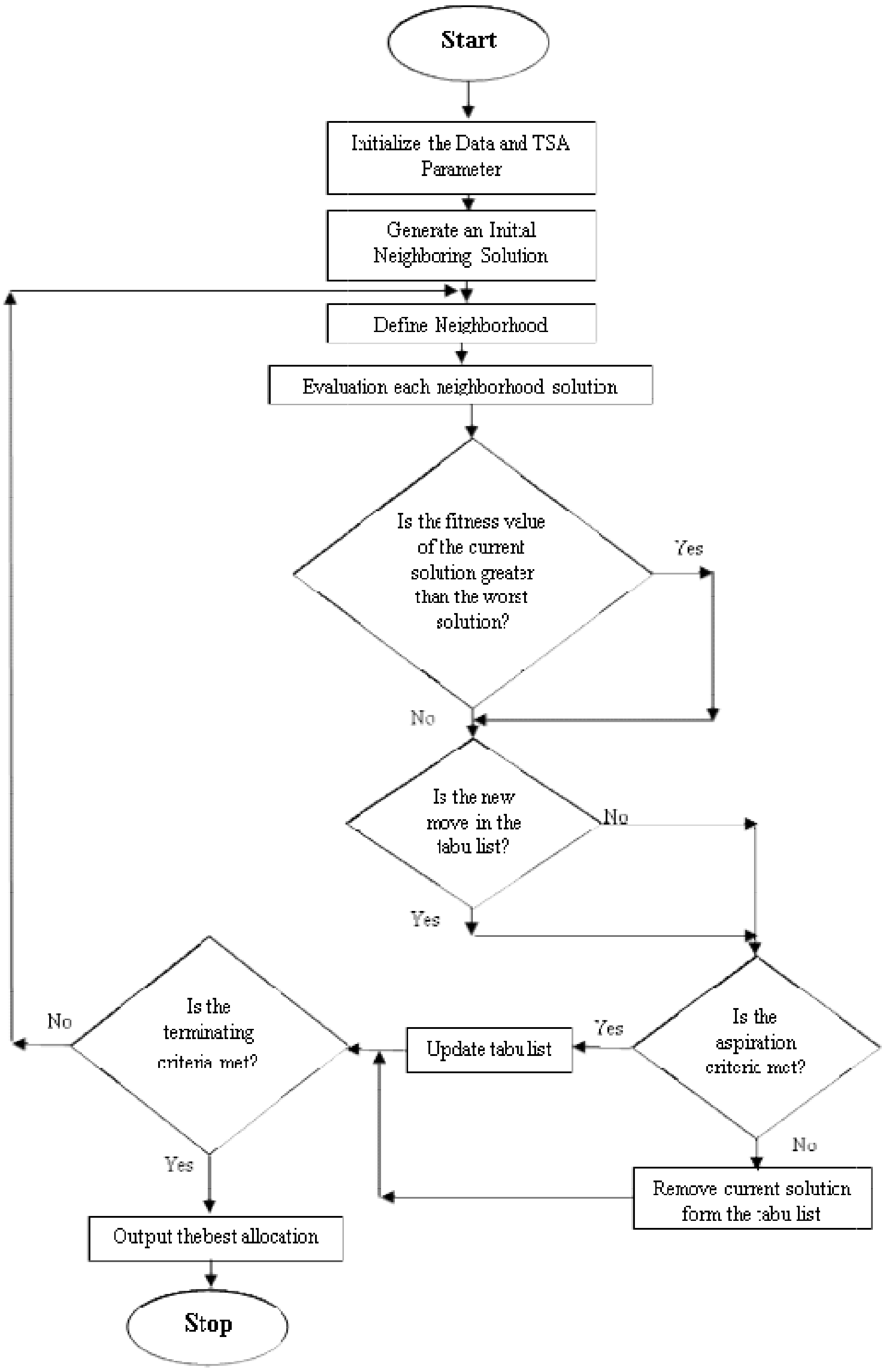

Start
Initialize the Data and TSA Parameter
Generate an Initial Neighboring Solution
Define Neighborhood
Evaluation each neighborhood solution
Is the fitness value of the current solution greater than the worst solution?
Yes
No
Is the new move in the tabu list?
No
Yes
Is the aspiration criteria met?
Yes
No
Update tabu list
Remove current solution form the tabu list
Is the terminating criteria met?
No
Yes
Output the best allocation
Stop

***Fig 1.1 Flowchart of the Tabu Search Algorithm***

# CHAPTER FOUR

# IMPLEMENTATION, RESULT AND DISCUSSION

## 4.1 Introduction

This chapter provides an overview of the implementation details, presents the results obtained,and discusses the findings of the shelf space allocation problemby a comparative evaluation of harmony Search (HS)algorithm and Tabu Search (TS) algorithm in python. The system aims at evaluating and implementing harmony search algorithm and tabu search algorithm on shelf space allocation system. These algorithms is use to solve the shelf space allocation problem effectively.

## 4.2 System Implementation

The implementation of the shelf space allocation was carried out usingHarmony Search algorithm, Tabu search algorithm and Python within the Django framework, where data is managed through the Product and ShelfAllocation models. The performance of the two algorithms is evaluated and compared based on their ability to minimize a fitness function, which measures constraint violations itinvolves the steps below:

1. Data Preparation

a. The system retrieves product data (including id, name, weight, and demand_percent) from the Product model in the database.
b. The demand percentage is converted into a demand value for further processing.
c. The shelf layout consists of 7 shelves and 7 columns, with 85 products allocated.
d. Parameters are defined for both Harmony Search and Tabu Search algorithms:

Harmony Search:

- Harmony Memory considering Rate (HMCR) = 0.8
- Pitch Adjustment Rate (PAR) = 0.4
- Harmony Memory Size (HMS) = 50
- Number of Iterations (NI) = 50

Tabu Search:

- Tabu list size = 50
- Number of Iterations (NI) = 50

2. Fitness Evaluation and Constraints

The core of the optimization is a fitness function (calculate_fitness), which penalizes allocations that violate any of the following soft constraints:

- High demand products (demand > 70%) should be placed in Shelves 3–5.
- Average demand products (60% ≤ demand ≤ 70%) should be placed in Shelves 1–2.
- Low demand products (demand < 60%) should be placed in Shelves 6–7.

The fitness function returns a penalty score for each allocation, with the aim to minimize this score through the optimization algorithms. A lower fitness score indicates a more optimal allocation.

**3.** Harmony Search Algorithm (HSA)

The Harmony Search Algorithm (HSA) is used to explore the solution space and generate potential shelf-space allocations. The steps involved in the HSA implementation are as follows:

- Harmony Memory Initialization: A population of random allocations is generated. Each harmony represents a product-to-shelf allocation with random placements of products across shelves and columns.
- For each iteration, a new harmony is generated based on memory and randomization. Specifically:
    - ✓ With probability HMCR, a product is placed in a column based on values from the memory (i.e., existing solutions).
    - ✓ With probability PAR, pitch adjustment is applied, adjusting the column value slightly to explore local solutions.
    - ✓ The new allocation is evaluated using the fitness function, and if it improves upon the worst solution in memory, it replaces it.
- The process is repeated for NI iterations, and the best allocation is selected.

The final allocation, along with its fitness score and constraint violations, is stored in the ShelfAllocation model.

**4.** Tabu Search Algorithm (TSA)

The Tabu Search Algorithm (TSA) operates as a local search optimization technique, starting with an initial solution (either from HSA or randomly generated). The TSA works as follows:

- Initial Solution: TSA starts with an initial solution and generates neighboring solutions by randomly modifying the product column assignments.
- Tabu List: A list is maintained to track visited solutions. The list prevents revisiting recently explored solutions to avoid cycling.
- If a new solution is not in the tabu list and improves upon the current best solution (lower fitness score), it becomes the new best solution.
- The tabu list is updated dynamically, and old solutions are removed once the list exceeds its maximum size (50).
- This process is repeated for NI iterations.

The final allocation after TSA is also stored in the ShelfAllocation model.

5. Performance Comparison

After running both algorithms, the performance of Harmony Search and Tabu Search is compared based on their respective fitness scores (penalty values). The system logs constraint violations and outputs a detailed analysis of each algorithm’s results. The algorithm with the lower fitness score indicates a better allocation, and this comparison serves as the basis for evaluating the effectiveness of each method.

6. Final Output

The best shelf-space allocation found by each algorithm is saved to the database, and the corresponding fitness scores are logged. Additionally, any constraint violations are printed for further analysis. The system outputs both the final allocation and a summary of how well each algorithm performed in terms of minimizing penalty violations.

## 4.3 System Requirement

We look at the system requirements from the hardware and software application used foreffective implementation of the new design.

### 4.3.1 Hardware requirement

For effective use of the new system, the minimum requirements for the hardwarecomponents are:

**Table 1: COMPONENT SPECIFICATION**

| COMPONENT | SPECIFICATION |
|---|---|
| Processor speeds | Intel(R) Core(TM) i3-4005U CPU @ 1.70GHz |
| Ram Size | 4.0 GB |
| Hard Disk | 118 GB |

**Table 2*:* Software requirements**

| SOFTWARE | SPECIFICATION |
|---|---|
| Operating System | Windows 11 Pro N, 21H2 |
| Code Editor IDE | Visual Studio Code |
| Programming Language | Python |
| Database | MySQL |

### 4.3.3 Choice of Programming Language

The system for solving the shelf space allocation problem is developed using **Python** in the **Visual Studio Code (VS Code)** environment. Python, a high-level and interpreted language, was chosen for its strong capabilities in optimization, algorithm development, and ease of use. Here are the key reasons why Python is well-suited for this project:

1. **Simplicity and Readability**

Python's clean and straightforward syntax makes the code easy to understand, modify, and maintain. This is crucial for managing complex algorithms, such as **Harmony Search** and **Tabu Search**, and for ensuring the system remains easily scalable and debuggable.

2. **Large Standard Library**

Python comes with an extensive standard library that simplifies tasks such as data manipulation, algorithm development, and optimization. These built-in modules save time and help avoid reinventing solutions to common problems, allowing developers to focus on the core functionality of the system.

3. **Availability of Optimization Libraries**

Python offers a wealth of optimization libraries, such as **SciPy, PyGMO** and Django which include pre-implemented optimization algorithms. These libraries are directly applicable to the project, providing a solid foundation for the **Harmony Search** and **Tabu Search** algorithms used to solve the shelf space allocation problem.

4. **Extensive Community and Resources**

Python has a large and active community of developers, ensuring that there are abundant resources such as tutorials, forums, and open-source codebases. This extensive support network makes it easy to troubleshoot issues, find solutions to problems, and stay updated on best practices and advancements in algorithm design.

5. **Rapid Prototyping and Iterative Development**

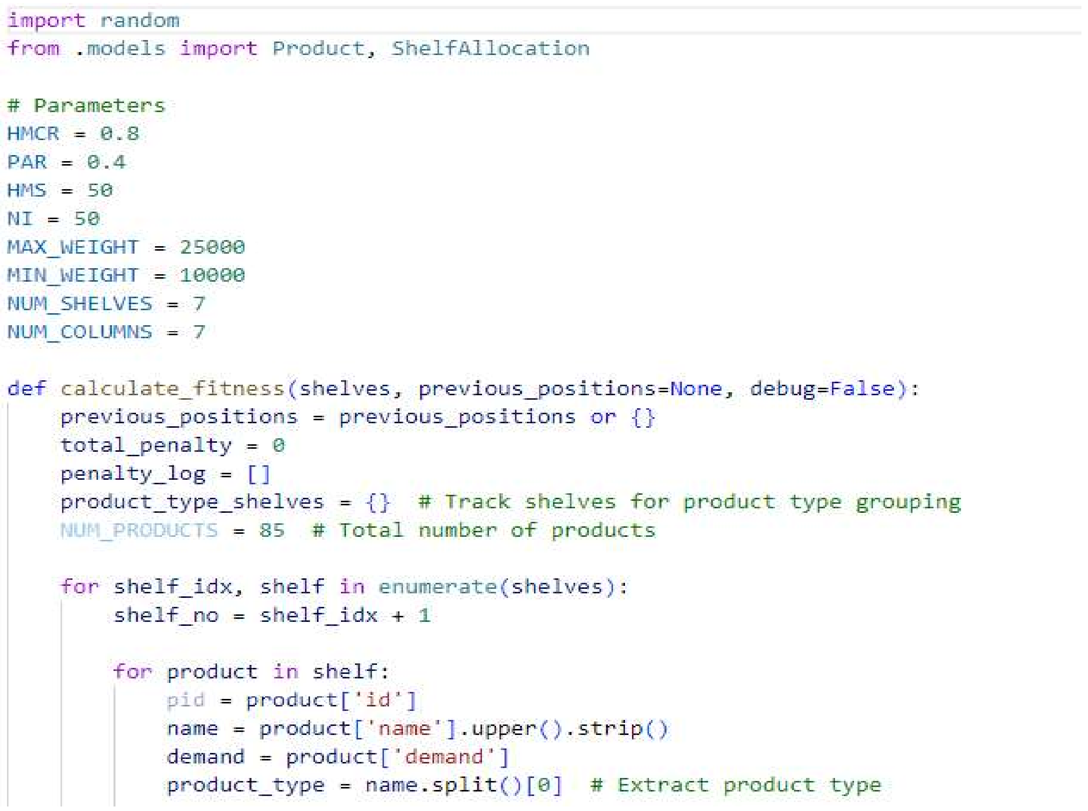
Python's interpreted nature allows for quick prototyping, making it easier to test and refine algorithms in real-time. This facilitates an iterative development process, where solutions can be continuously improved based on feedback and results from the optimization algorithms, such as **Harmony Search** and **Tabu Search**.

6. **Integration with Visual Studio Code**

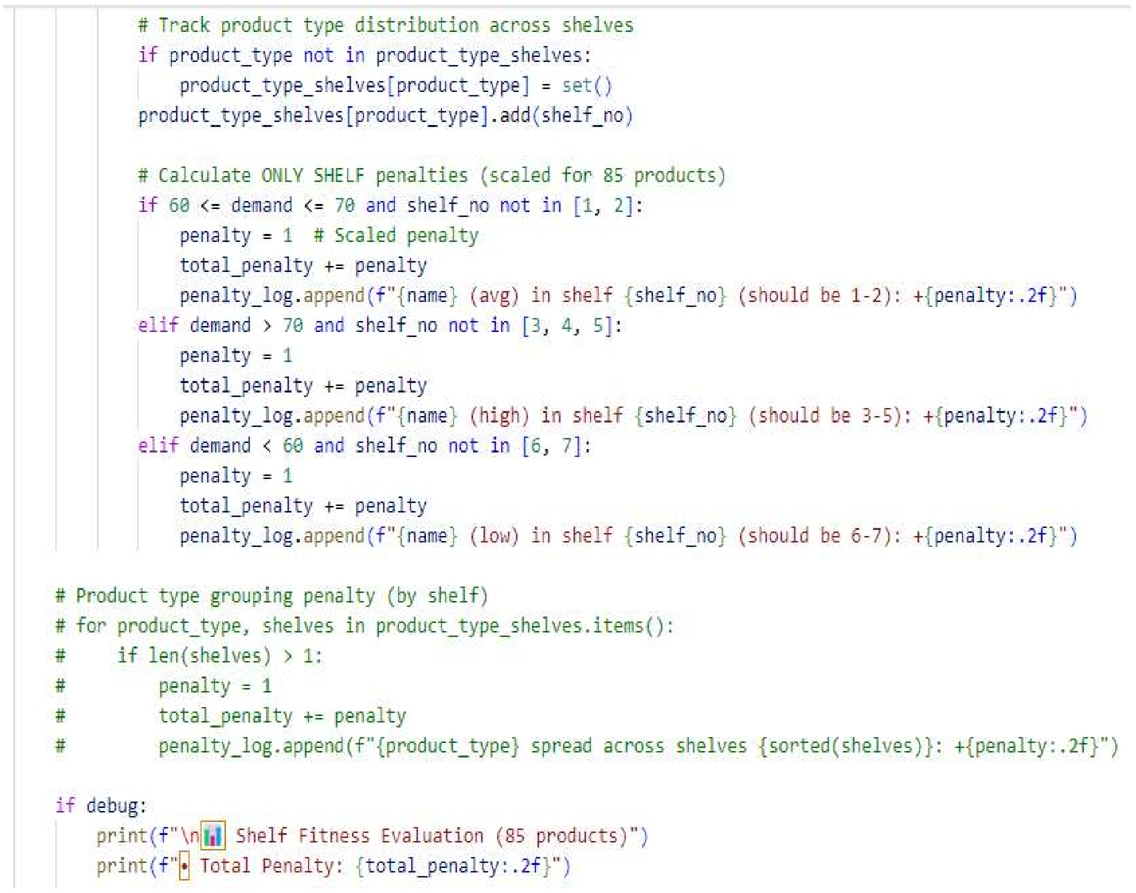
Visual Studio Code offers a lightweight yet powerful development environment for Python. It provides features like intelligent code completion, debugging tools, version control, andextensions for Python, all of which significantly enhance the development experience and increase productivity during algorithm design and implementation.

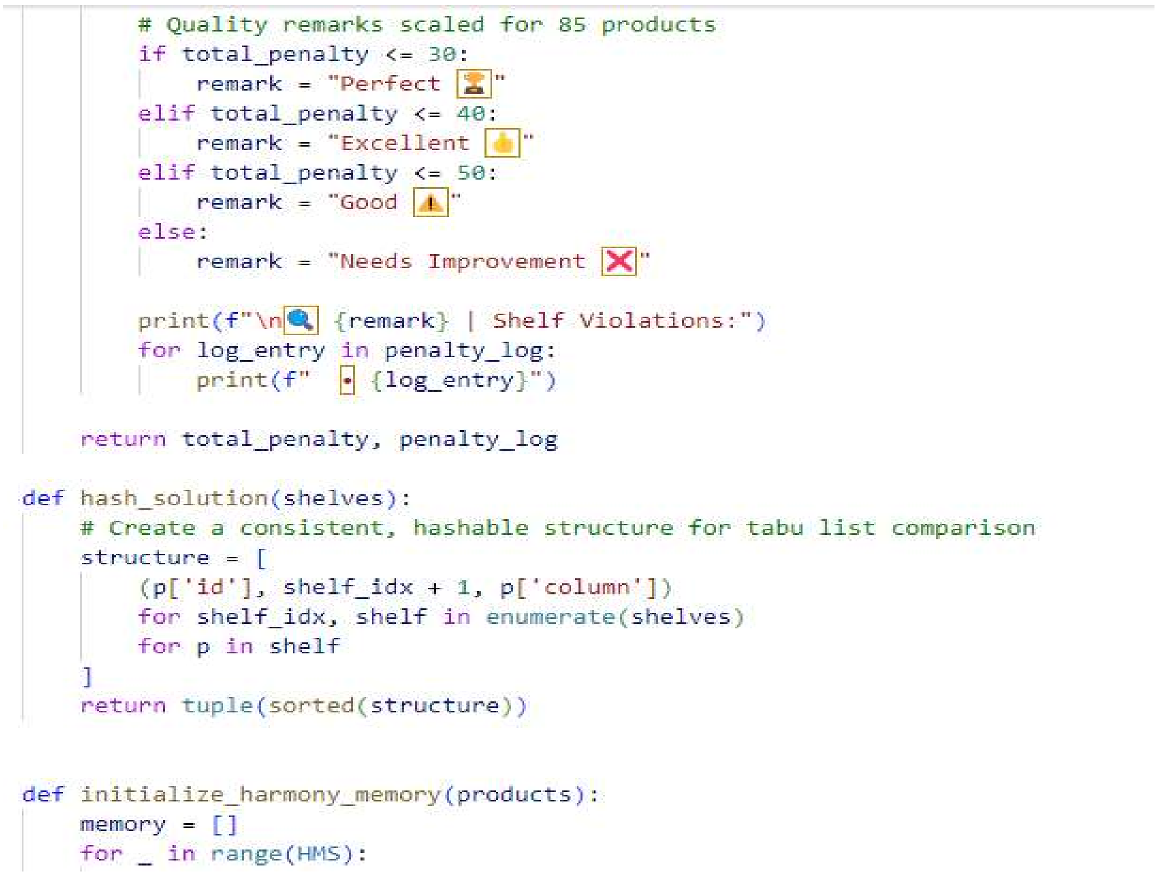
## 4.4 Program Documentation

The python program for the Shelf space allocation System to generate near optimal shelf space allocation. The algorithm takes into account both hard constraints, such as the minimum and maximum weight of product to be on a column of a shelf, and soft constraints, such as allocate product of the same type but different sizes on the same column.Here is the python Code for the algorithm below.

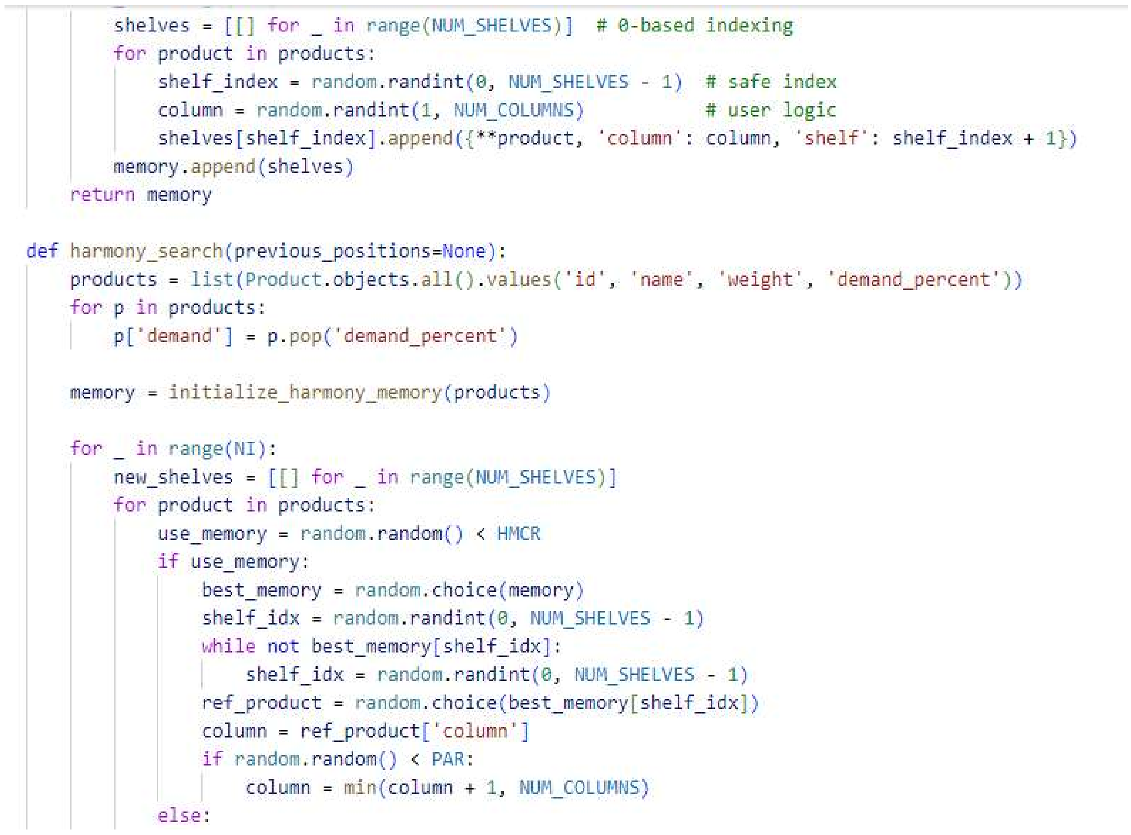


```python
import random
from .models import Product, ShelfAllocation

# Parameters
HMCR = 0.8
PAR = 0.4
HMS = 50
NI = 50
MAX_WEIGHT = 25000
MIN_WEIGHT = 10000
NUM_SHELVES = 7
NUM_COLUMNS = 7

def calculate_fitness(shelves, previous_positions=None, debug=False):
    previous_positions = previous_positions or {}
    total_penalty = 0
    penalty_log = []
    product_type_shelves = {}  # Track shelves for product type grouping
    NUM_PRODUCTS = 85  # Total number of products

    for shelf_idx, shelf in enumerate(shelves):
        shelf_no = shelf_idx + 1

        for product in shelf:
            pid = product['id']
            name = product['name'].upper().strip()
            demand = product['demand']
            product_type = name.split()[0]  # Extract product type
```

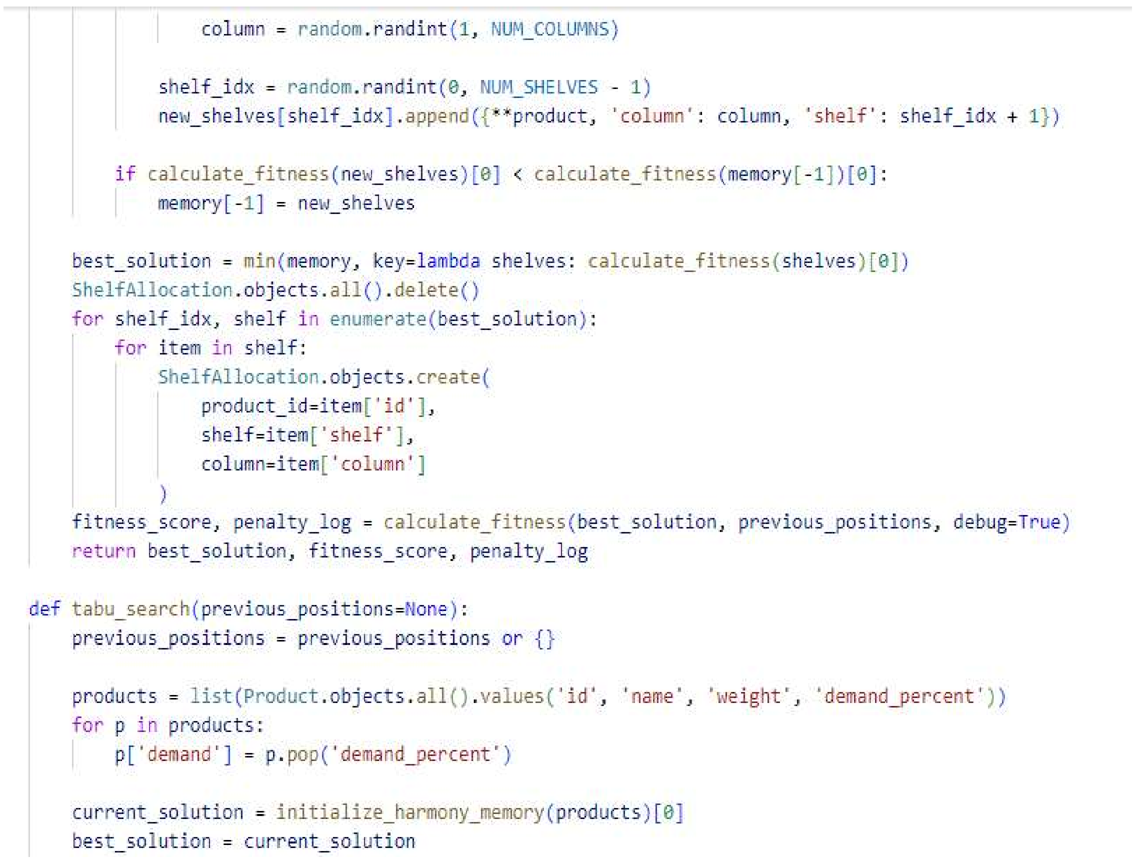

```
            # Track product type distribution across shelves
            if product_type not in product_type_shelves:
                product_type_shelves[product_type] = set()
            product_type_shelves[product_type].add(shelf_no)

            # Calculate ONLY SHELF penalties (scaled for 85 products)
            if 60 <= demand <= 70 and shelf_no not in [1, 2]:
                penalty = 1  # Scaled penalty
                total_penalty += penalty
                penalty_log.append(f"{name} (avg) in shelf {shelf_no} (should be 1-2): +{penalty:.2f}")
            elif demand > 70 and shelf_no not in [3, 4, 5]:
                penalty = 1
                total_penalty += penalty
                penalty_log.append(f"{name} (high) in shelf {shelf_no} (should be 3-5): +{penalty:.2f}")
            elif demand < 60 and shelf_no not in [6, 7]:
                penalty = 1
                total_penalty += penalty
                penalty_log.append(f"{name} (low) in shelf {shelf_no} (should be 6-7): +{penalty:.2f}")

    # Product type grouping penalty (by shelf)
    # for product_type, shelves in product_type_shelves.items():
    #     if len(shelves) > 1:
    #         penalty = 1
    #         total_penalty += penalty
    #         penalty_log.append(f"{product_type} spread across shelves {sorted(shelves)}: +{penalty:.2f}")

    if debug:
        print(f"\n📊 Shelf Fitness Evaluation (85 products)")
        print(f"• Total Penalty: {total_penalty:.2f}")
```



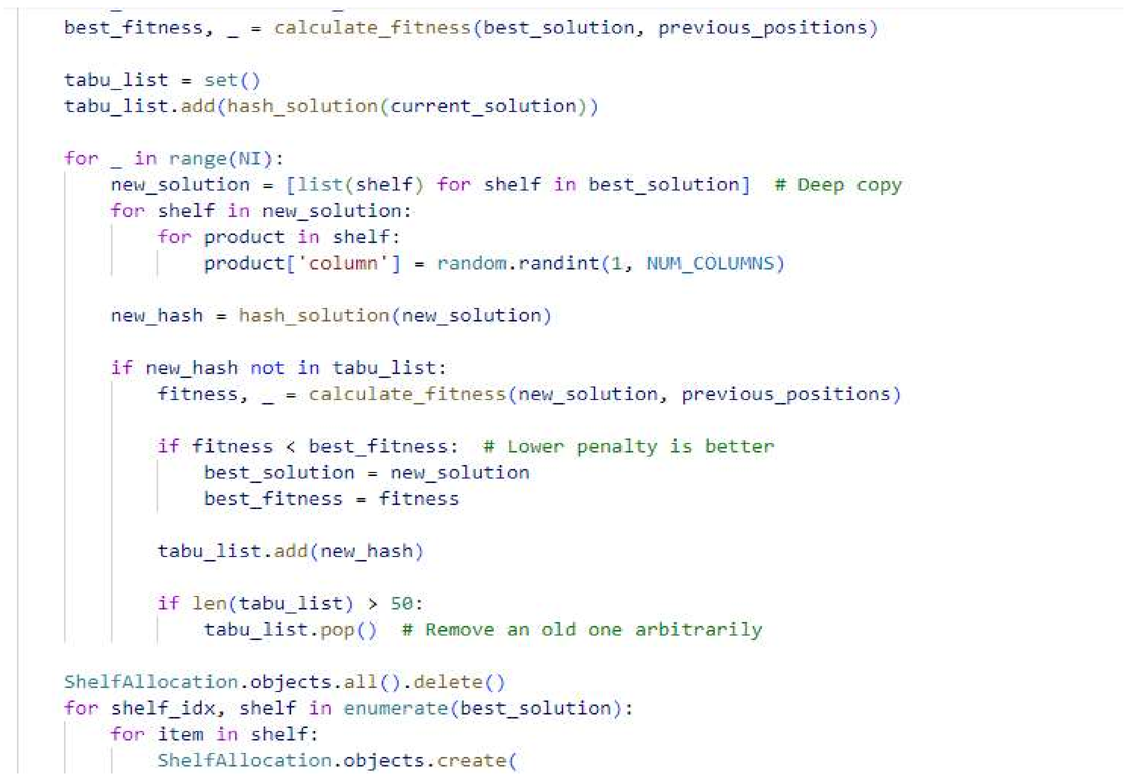
```
        # Quality remarks scaled for 85 products
        if total_penalty <= 30:
            remark = "Perfect 🏆"
        elif total_penalty <= 40:
            remark = "Excellent 👍"
        elif total_penalty <= 50:
            remark = "Good ⚠️"
        else:
            remark = "Needs Improvement ❌"

        print(f"\n🔍 {remark} | Shelf Violations:")
        for log_entry in penalty_log:
            print(f"  • {log_entry}")

    return total_penalty, penalty_log

def hash_solution(shelves):
    # Create a consistent, hashable structure for tabu list comparison
    structure = [
        (p['id'], shelf_idx + 1, p['column'])
        for shelf_idx, shelf in enumerate(shelves)
        for p in shelf
    ]
    return tuple(sorted(structure))


def initialize_harmony_memory(products):
    memory = []
    for _ in range(HMS):
```



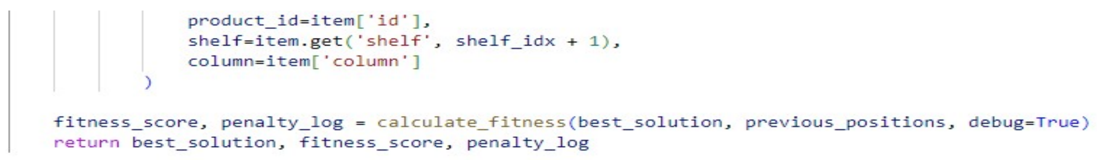
```python
        shelves = [[] for _ in range(NUM_SHELVES)]  # 0-based indexing
        for product in products:
            shelf_index = random.randint(0, NUM_SHELVES - 1)  # safe index
            column = random.randint(1, NUM_COLUMNS)           # user logic
            shelves[shelf_index].append({**product, 'column': column, 'shelf': shelf_index + 1})
        memory.append(shelves)
    return memory

def harmony_search(previous_positions=None):
    products = list(Product.objects.all().values('id', 'name', 'weight', 'demand_percent'))
    for p in products:
        p['demand'] = p.pop('demand_percent')

    memory = initialize_harmony_memory(products)

    for _ in range(NI):
        new_shelves = [[] for _ in range(NUM_SHELVES)]
        for product in products:
            use_memory = random.random() < HMCR
            if use_memory:
                best_memory = random.choice(memory)
                shelf_idx = random.randint(0, NUM_SHELVES - 1)
                while not best_memory[shelf_idx]:
                    shelf_idx = random.randint(0, NUM_SHELVES - 1)
                ref_product = random.choice(best_memory[shelf_idx])
                column = ref_product['column']
                if random.random() < PAR:
                    column = min(column + 1, NUM_COLUMNS)
            else:
```



```python
                column = random.randint(1, NUM_COLUMNS)

            shelf_idx = random.randint(0, NUM_SHELVES - 1)
            new_shelves[shelf_idx].append({**product, 'column': column, 'shelf': shelf_idx + 1})

        if calculate_fitness(new_shelves)[0] < calculate_fitness(memory[-1])[0]:
            memory[-1] = new_shelves

    best_solution = min(memory, key=lambda shelves: calculate_fitness(shelves)[0])
    ShelfAllocation.objects.all().delete()
    for shelf_idx, shelf in enumerate(best_solution):
        for item in shelf:
            ShelfAllocation.objects.create(
                product_id=item['id'],
                shelf=item['shelf'],
                column=item['column']
            )
    fitness_score, penalty_log = calculate_fitness(best_solution, previous_positions, debug=True)
    return best_solution, fitness_score, penalty_log

def tabu_search(previous_positions=None):
    previous_positions = previous_positions or {}

    products = list(Product.objects.all().values('id', 'name', 'weight', 'demand_percent'))
    for p in products:
        p['demand'] = p.pop('demand_percent')

    current_solution = initialize_harmony_memory(products)[0]
    best_solution = current_solution
```



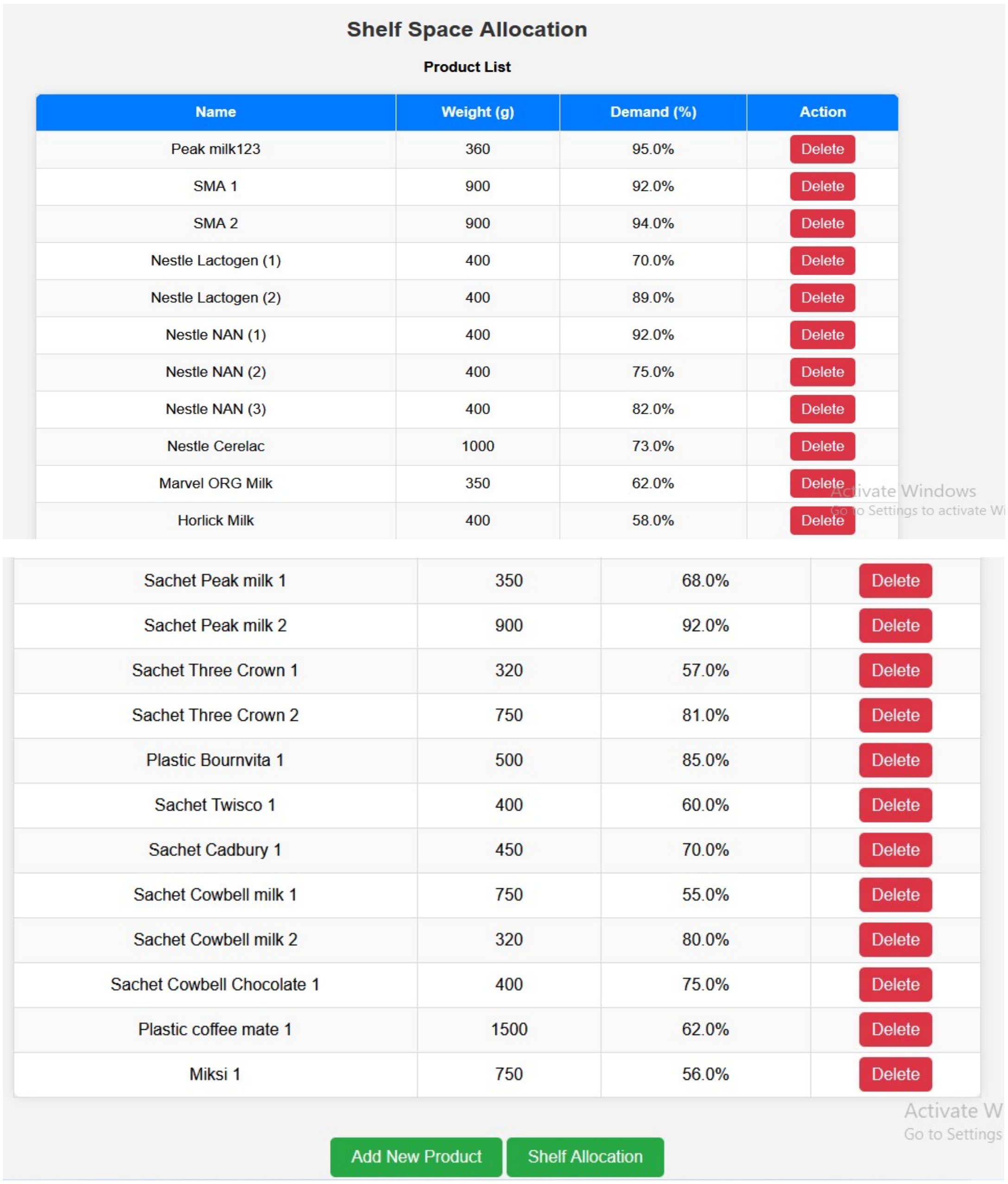
```
    best_fitness, _ = calculate_fitness(best_solution, previous_positions)

    tabu_list = set()
    tabu_list.add(hash_solution(current_solution))

    for _ in range(NI):
        new_solution = [list(shelf) for shelf in best_solution]  # Deep copy
        for shelf in new_solution:
            for product in shelf:
                product['column'] = random.randint(1, NUM_COLUMNS)

        new_hash = hash_solution(new_solution)

        if new_hash not in tabu_list:
            fitness, _ = calculate_fitness(new_solution, previous_positions)

            if fitness < best_fitness:  # Lower penalty is better
                best_solution = new_solution
                best_fitness = fitness

            tabu_list.add(new_hash)

            if len(tabu_list) > 50:
                tabu_list.pop()  # Remove an old one arbitrarily

    ShelfAllocation.objects.all().delete()
    for shelf_idx, shelf in enumerate(best_solution):
        for item in shelf:
            ShelfAllocation.objects.create(
```



```
                product_id=item['id'],
                shelf=item.get('shelf', shelf_idx + 1),
                column=item['column']
            )

    fitness_score, penalty_log = calculate_fitness(best_solution, previous_positions, debug=True)
    return best_solution, fitness_score, penalty_log
```

## 4.5 INTERFACE DESIGN

This section presents the user interface designs developed for the system. Each interface is designed with usability and functionality in mind, ensuring a smooth and intuitive user experience. The layouts aim to support efficient task completion while maintaining a clear visual structure. The images below illustrate the key screens of the system, including navigation elements, input forms, and output displays, highlighting how users interact with the application.

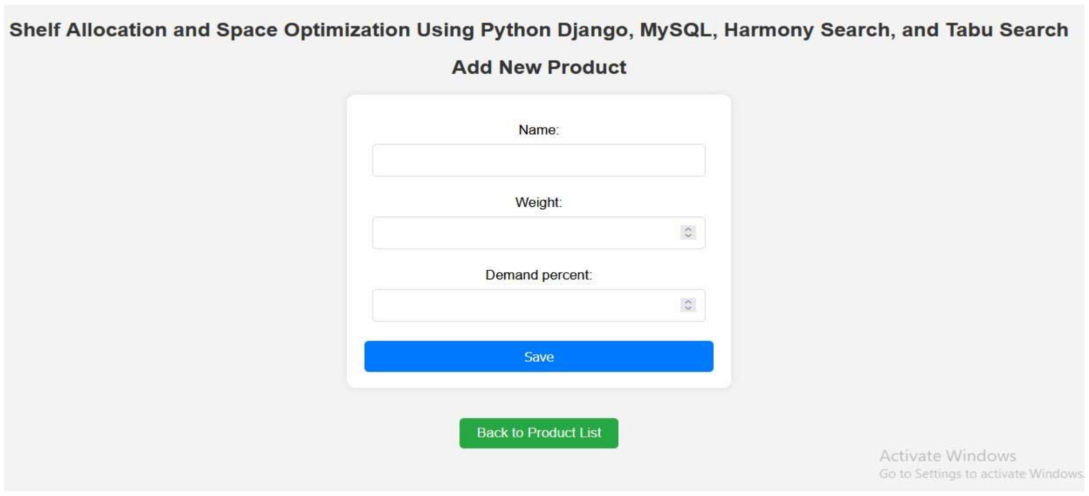


## Shelf Space Allocation

**Product List**

| Name | Weight (g) | Demand (%) | Action |
|---|---|---|---|
| Peak milk123 | 360 | 95.0% | Delete |
| SMA 1 | 900 | 92.0% | Delete |
| SMA 2 | 900 | 94.0% | Delete |
| Nestle Lactogen (1) | 400 | 70.0% | Delete |
| Nestle Lactogen (2) | 400 | 89.0% | Delete |
| Nestle NAN (1) | 400 | 92.0% | Delete |
| Nestle NAN (2) | 400 | 75.0% | Delete |
| Nestle NAN (3) | 400 | 82.0% | Delete |
| Nestle Cerelac | 1000 | 73.0% | Delete |
| Marvel ORG Milk | 350 | 62.0% | Delete |
| Horlick Milk | 400 | 58.0% | Delete |
| Sachet Peak milk 1 | 350 | 68.0% | Delete |
| Sachet Peak milk 2 | 900 | 92.0% | Delete |
| Sachet Three Crown 1 | 320 | 57.0% | Delete |
| Sachet Three Crown 2 | 750 | 81.0% | Delete |
| Plastic Bournvita 1 | 500 | 85.0% | Delete |
| Sachet Twisco 1 | 400 | 60.0% | Delete |
| Sachet Cadbury 1 | 450 | 70.0% | Delete |
| Sachet Cowbell milk 1 | 750 | 55.0% | Delete |
| Sachet Cowbell milk 2 | 320 | 80.0% | Delete |
| Sachet Cowbell Chocolate 1 | 400 | 75.0% | Delete |
| Plastic coffee mate 1 | 1500 | 62.0% | Delete |
| Miksi 1 | 750 | 56.0% | Delete |

Fig: 2.0 Product list


**Shelf Allocation and Space Optimization Using Python Django, MySQL, Harmony Search, and Tabu Search**

**Add New Product**

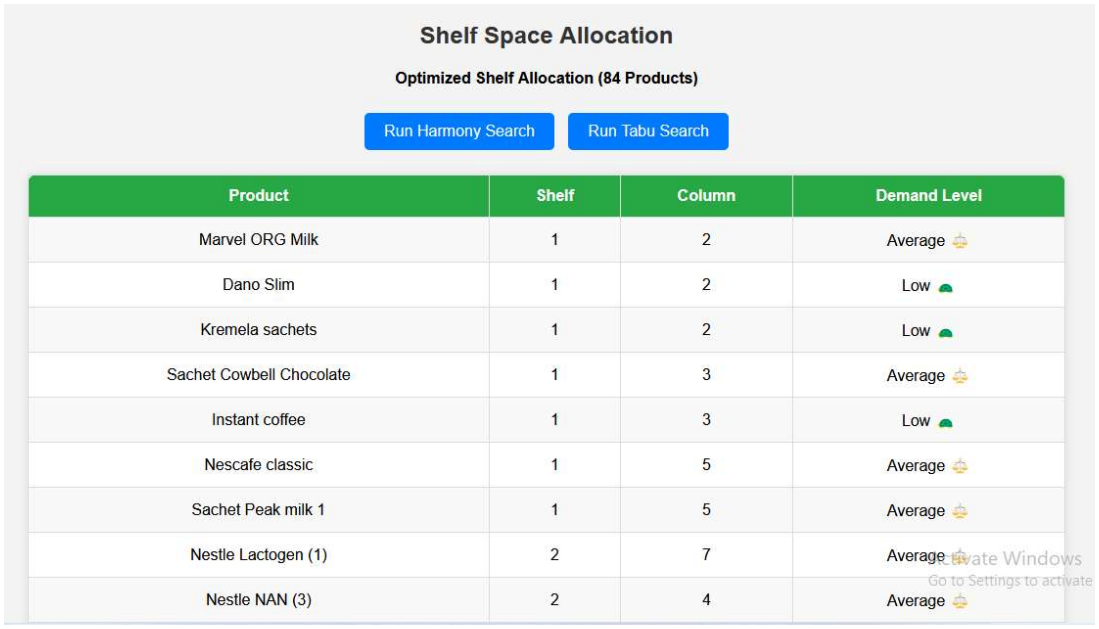
Name:

Weight:

Demand percent:

Activate Windows
Go to Settings to activate Windows.

Fig: 2.1 Add products

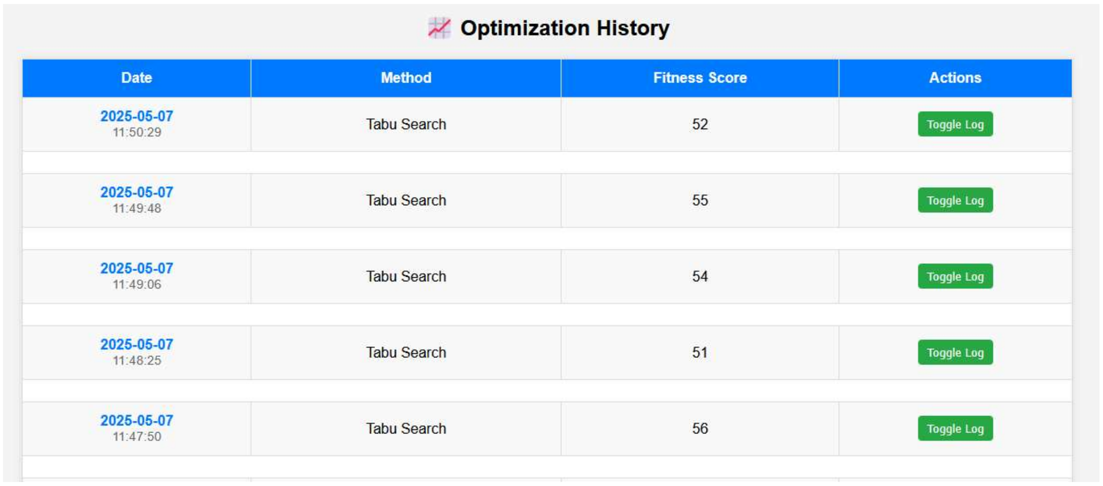

**Shelf Space Allocation**

**Optimized Shelf Allocation (84 Products)**

| Product | Shelf | Column | Demand Level |
|---|---|---|---|
| Marvel ORG Milk | 1 | 2 | Average ⚖ |
| Dano Slim | 1 | 2 | Low ● |
| Kremela sachets | 1 | 2 | Low ● |
| Sachet Cowbell Chocolate | 1 | 3 | Average ⚖ |
| Instant coffee | 1 | 3 | Low ● |
| Nescafe classic | 1 | 5 | Average ⚖ |
| Sachet Peak milk 1 | 1 | 5 | Average ⚖ |
| Nestle Lactogen (1) | 2 | 7 | Average ⚖ |
| Nestle NAN (3) | 2 | 4 | Average ⚖ |

Activate Windows
Go to Settings to activate

Fig: 2.3 Shelf Allocations



📈 Optimization History

| Date | Method | Fitness Score | Actions |
|---|---|---|---|
| 2025-05-07 11:50:29 | Tabu Search | 52 | Toggle Log |
| 2025-05-07 11:49:48 | Tabu Search | 55 | Toggle Log |
| 2025-05-07 11:49:06 | Tabu Search | 54 | Toggle Log |
| 2025-05-07 11:48:25 | Tabu Search | 51 | Toggle Log |
| 2025-05-07 11:47:50 | Tabu Search | 56 | Toggle Log |

Fig: 2.4 Shelf Allocations

## 4.6 Results and Discussion

We experimented with random shelf space allocation data to determine how well the Harmony Search algorithm and Tabu search algorithm solved the Shelf space allocation problem. The Shelf space allocation problem is dealt with by utilizing the Harmony Search algorithm and Tabu search algorithm. We take into account both soft and hard constraints, generate new solution iteratively, evaluate their scores, and update the best solution. Printing the result makes it easier to keep track of the algorithm's progress.

**Allocation for Harmony Search Algorithm**

Table 3: First Iteration

| PRODUCT NAME | SHELF | COLUMN |
|---|---|---|
| Peak milk123 | 1 | 6 |
| Marvel ORG Milk | 1 | 4 |
| Cebon milk | 1 | 5 |
| Sachet Cowbell milk | 1 | 2 |
| Cebon Café | 1 | 6 |
| Sachet Oluji pure cocoa | 1 | 3 |
| Tin Oluji pure cocoa | 1 | 5 |
| Nestle Cerelac 2 | 1 | 3 |
| Loya milk sachets 1 | 1 | 4 |
| Sachet Peak milk 1 | 1 | 5 |
| Hollandia Evap | 1 | 5 |
| Dano milk | 2 | 6 |
| Dano Cool Cow | 2 | 7 |
| Loya milk sachets | 2 | 5 |
| Tin Peak milk | 2 | 3 |
| Sachet Bournvita | 2 | 2 |
| Sachet Cowbell Chocolate | 2 | 1 |
| Smart Diko | 2 | 6 |
| Nescafe Gold | 2 | 4 |
| Superdelious tin milk | 2 | 7 |
| Dano Cool Cow 2 | 2 | 7 |
| Dano Cool Cow 3 | 2 | 7 |
| Kremela sachets 1 | 2 | 5 |
| Sachet Cowbell milk 1 | 2 | 1 |
| Peak milk456 | 2 | 6 |
| Three crown Evap | 3 | 2 |
| Ovaltine sachets | 3 | 3 |
| Kremela sachets | 3 | 3 |
| Tin Three crown | 3 | 6 |
| Sachet Cadbury | 3 | 1 |
| Bravia | 3 | 3 |
| Nestle Cerelac 1 | 3 | 7 |
| Sachet Three Crown 1 | 3 | 4 |
| Plastic Bournvita 1 | 3 | 2 |
| Sachet Cowbell Chocolate 1 | 3 | 1 |
| Plastic coffee mate 1 | 3 | 2 |
| Miksi 1 | 3 | 3 |
| Nestle Lactogen (2) | 3 | 5 |
| Nestle NAN (2) | 4 | 5 |
| Nestle NAN (3) | 4 | 4 |
| Peak Chocolate sachets | 4 | 6 |

| | | |
|---|---|---|
| Tin Cowbell | 4 | 2 |
| Sachet Cowbell Strawberry | 4 | 3 |
| Stock well | 4 | 2 |
| Nescafe classic | 4 | 5 |
| Sachet Three Crown 2 | 4 | 2 |
| Sachet Twisco 1 | 4 | 5 |
| Sachet Cowbell milk 2 | 4 | 6 |
| SMA 1 | 4 | 6 |
| SMA 2 | 5 | 6 |
| Nestle NAN (1) | 5 | 3 |
| Milo sachets | 5 | 5 |
| Dano Cool Cow (Tin) | 5 | 3 |
| Tin Peak Gold | 5 | 5 |
| Sachet Peak milk | 5 | 3 |
| Plastic Bournvita | 5 | 7 |
| Maximus Café | 5 | 6 |
| Miksi | 5 | 7 |
| Ovaltine sachets 1 | 5 | 2 |
| Tin Peak milk 2 | 5 | 2 |
| Sachet Peak milk 2 | 5 | 7 |
| Nestle Lactogen (1) | 5 | 4 |
| Nestle Cerelac | 6 | 7 |
| My Boy 0-12m | 6 | 5 |
| De-hero milk | 6 | 6 |
| Good time milk sachets | 6 | 3 |
| Sachet Three Crown | 6 | 5 |
| Sachet Twisco | 6 | 4 |
| Plastic coffee mate | 6 | 2 |
| Milo sachets 2 | 6 | 4 |
| Hollandia Evap 2 | 6 | 7 |
| Horlick Milk | 6 | 6 |
| Peak Baby 0-12m | 7 | 2 |
| Dano Slim | 7 | 4 |
| Peak milk tin | 7 | 3 |
| Sachet Cowbell Coffee | 7 | 4 |
| Instant coffee | 7 | 6 |
| Brown Gold | 7 | 7 |
| Milo sachets 1 | 7 | 6 |
| Hollandia Evap 1 | 7 | 3 |
| Dano Slim 1 | 7 | 4 |
| Dano Cool Cow 1 | 7 | 3 |
| Tin Peak milk 1 | 7 | 7 |
| Sachet Cadbury 1 | 7 | 4 |

**EXECUTION TIME: 1.38 SECONDS**

**PENALTY POINT: 40**

Table 4:Second Iteration

| PRODUCT NAME | SHELF | COLUMN |
|---|---|---|
| SMA 2 | 1 | 3 |
| De-hero milk | 1 | 4 |
| Kremela sachets | 1 | 6 |
| Good time milk sachets | 1 | 3 |
| Tin Three crown | 1 | 2 |
| Milo sachets 1 | 1 | 4 |
| Dano Slim 1 | 1 | 1 |
| Loya milk sachets 1 | 1 | 7 |
| Sachet Peak milk 2 | 1 | 2 |
| Sachet Cadbury 1 | 1 | 1 |
| Horlick Milk | 2 | 3 |
| Hollandia Evap | 2 | 4 |
| Dano milk | 2 | 6 |
| Dano Slim | 2 | 6 |
| Dano Cool Cow (Tin) | 2 | 4 |
| Dano Cool Cow | 2 | 3 |
| Instant coffee | 2 | 6 |
| Nescafe Gold | 2 | 4 |
| Superdelious tin milk | 2 | 2 |
| Sachet Three Crown 2 | 2 | 3 |
| Sachet Twisco 1 | 2 | 5 |
| Sachet Cowbell milk 2 | 2 | 7 |
| Peak Baby 0-12m | 3 | 3 |
| Cebon milk | 3 | 4 |
| Peak milk456 | 3 | 3 |
| Ovaltine sachets | 3 | 5 |
| Plastic Bournvita | 3 | 2 |
| Sachet Cowbell Coffee | 3 | 1 |
| Sachet Cowbell Strawberry | 3 | 2 |
| Plastic coffee mate | 3 | 1 |
| Smart Diko | 3 | 6 |
| Cebon Café | 3 | 6 |
| Nestle Cerelac 2 | 3 | 6 |
| Milo sachets 2 | 3 | 6 |
| Dano Cool Cow 1 | 3 | 7 |
| Ovaltine sachets 1 | 3 | 7 |
| Plastic Bournvita 1 | 3 | 1 |
| Sachet Cowbell Chocolate 1 | 3 | 4 |
| Plastic coffee mate 1 | 3 | 1 |
| SMA 1 | 4 | 7 |
| My Boy 0-12m | 4 | 6 |
| Loya milk sachets | 4 | 3 |
| Peak Chocolate sachets | 4 | 7 |
| Sachet Peak milk | 4 | 4 |
| Sachet Bournvita | 4 | 1 |

| | | |
|---|---|---|
| Bravia | 4 | 4 |
| Miksi | 4 | 4 |
| Nestle Cerelac 1 | 4 | 1 |
| Hollandia Evap 1 | 4 | 6 |
| Hollandia Evap 2 | 4 | 1 |
| Peak milk123 | 5 | 5 |
| Nestle Lactogen (1) | 5 | 4 |
| Nestle NAN (1) | 5 | 3 |
| Nestle Cerelac | 5 | 4 |
| Marvel ORG Milk | 5 | 1 |
| Milo sachets | 5 | 6 |
| Three crown Evap | 5 | 1 |
| Peak milk tin | 5 | 5 |
| Tin Peak Gold | 5 | 7 |
| Sachet Three Crown | 5 | 1 |
| Sachet Twisco | 5 | 5 |
| Sachet Cadbury | 5 | 2 |
| Sachet Cowbell milk | 5 | 4 |
| Sachet Cowbell Chocolate | 5 | 6 |
| Maximus Café | 5 | 5 |
| Dano Cool Cow 3 | 5 | 2 |
| Tin Peak milk 1 | 5 | 6 |
| Nestle NAN (3) | 6 | 2 |
| Tin Peak milk | 6 | 6 |
| Tin Oluji pure cocoa | 6 | 1 |
| Kremela sachets 1 | 6 | 3 |
| Tin Peak milk 2 | 6 | 5 |
| Sachet Peak milk 1 | 6 | 6 |
| Sachet Three Crown 1 | 6 | 1 |
| Miksi 1 | 6 | 3 |
| Nestle Lactogen (2) | 7 | 7 |
| Nestle NAN (2) | 7 | 5 |
| Tin Cowbell | 7 | 1 |
| Stock well | 7 | 4 |
| Nescafe classic | 7 | 7 |
| Sachet Oluji pure cocoa | 7 | 6 |
| Brown Gold | 7 | 7 |
| Dano Cool Cow 2 | 7 | 6 |
| Sachet Cowbell milk 1 | 7 | 2 |

**EXECUTION TIME: 1.38 SECONDS**

**PENALTY POINT: 44**

3rd **Iteration**

4th **Iteration**

5th **Iteration**

6th **Iteration**

7th **Iteration**

8th **Iteration**

9th **Iteration**

10th **Iteration**

**….**

50th **Iteration**

**Table 5:RESULT SUMMARY OF HARMONY SEARCH ALGORITHM**

| ITERATION | RESULT (SOFT CONTRAINT FITNESS VIOLATION) | TIME TAKEN (SEC) |
|---|---|---|
| 1 | 40 | 1.38 |
| 2 | 44 | 1.38 |
| 3 | 43 | 1.40 |
| 4 | 40 | 1.41 |
| 5 | 43 | 1.39 |
| **6** | **39** | **1.34** |
| 7 | 47 | 1.39 |
| 8 | 45 | 1.49 |
| 9 | 44 | 1.43 |
| 10 | 44 | 1.36 |

Harmony search result discussion: The first HMS run output starts with iteration 1 and continues to iteration 50, out of which 10 best iterations were recorded each iteration includes the allocation of 84 products to 7 shelves and each shelf contain 7 column. The best allocation as indicted in the table above is **iteration 6** with penalty point of **39** and the execution time is **1.34sec.**

**Allocation for Tabu Search algorithm**

**Table 6:First Iteration**

| PRODUCT NAME | SHELF | COLUMN |
|---|---|---|
| SMA 2 | 1 | 7 |
| Nestle Lactogen (1) | 1 | 5 |
| Nestle Lactogen (2) | 1 | 7 |
| Nestle NAN (2) | 1 | 1 |
| Nestle NAN (3) | 1 | 6 |
| Nestle Cerelac | 1 | 5 |
| Dano Cool Cow (Tin) | 1 | 6 |
| Peak Chocolate sachets | 1 | 4 |
| Tin Peak Gold | 1 | 5 |
| Tin Three crown | 1 | 7 |
| Sachet Cowbell Chocolate | 1 | 3 |
| Instant coffee | 1 | 5 |
| Nescafe classic | 1 | 1 |
| Milo sachets 1 | 1 | 1 |
| Dano Cool Cow 3 | 1 | 1 |
| Sachet Cadbury 1 | 1 | 3 |
| Sachet Peak milk | 2 | 4 |
| Milo sachets 2 | 2 | 4 |
| Dano Cool Cow 2 | 2 | 4 |
| Kremela sachets 1 | 2 | 6 |
| Tin Peak milk 1 | 2 | 2 |
| De-hero milk | 3 | 6 |
| Peak milk456 | 3 | 1 |
| Hollandia Evap | 3 | 1 |
| Loya milk sachets | 3 | 7 |
| Tin Cowbell | 3 | 5 |
| Sachet Cowbell Coffee | 3 | 4 |
| Plastic coffee mate | 3 | 7 |
| Stock well | 3 | 5 |
| Hollandia Evap 1 | 3 | 5 |
| Dano Slim 1 | 3 | 7 |
| Ovaltine sachets 1 | 3 | 1 |
| Sachet Twisco 1 | 3 | 6 |
| Sachet Cowbell milk 1 | 3 | 7 |
| Sachet Cowbell Chocolate 1 | 3 | 7 |
| My Boy 0-12m | 4 | 4 |
| Milo sachets | 4 | 1 |
| Three crown Evap | 4 | 6 |
| Tin Peak milk | 4 | 2 |
| Sachet Bournvita | 4 | 6 |
| Nescafe Gold | 4 | 6 |
| Maximus Café | 4 | 7 |
| Sachet Peak milk 2 | 4 | 4 |

| | | |
|---|---|---|
| Sachet Three Crown 2 | 4 | 6 |
| Plastic Bournvita 1 | 4 | 4 |
| Sachet Cowbell milk 2 | 4 | 4 |
| Peak milk123 | 5 | 2 |
| Marvel ORG Milk | 5 | 6 |
| Peak Baby 0-12m | 5 | 3 |
| Cebon milk | 5 | 4 |
| Dano milk | 5 | 3 |
| Dano Slim | 5 | 3 |
| Kremela sachets | 5 | 3 |
| Sachet Three Crown | 5 | 6 |
| Sachet Cowbell Strawberry | 5 | 4 |
| Cebon Café | 5 | 2 |
| Sachet Oluji pure cocoa | 5 | 7 |
| Brown Gold | 5 | 4 |
| Miksi | 5 | 7 |
| Nestle Cerelac 2 | 5 | 2 |
| Plastic coffee mate 1 | 5 | 2 |
| Miksi 1 | 5 | 5 |
| Nestle NAN (1) | 6 | 3 |
| Horlick Milk | 6 | 3 |
| Dano Cool Cow | 6 | 6 |
| Plastic Bournvita | 6 | 7 |
| Sachet Twisco | 6 | 2 |
| Sachet Cadbury | 6 | 3 |
| Sachet Cowbell milk | 6 | 2 |
| Smart Diko | 6 | 7 |
| Tin Oluji pure cocoa | 6 | 1 |
| Superdelious tin milk | 6 | 6 |
| Hollandia Evap 2 | 6 | 3 |
| Dano Cool Cow 1 | 6 | 5 |
| Loya milk sachets 1 | 6 | 2 |
| Tin Peak milk 2 | 6 | 1 |
| Sachet Peak milk 1 | 6 | 6 |
| Sachet Three Crown 1 | 6 | 1 |
| SMA 1 | 7 | 5 |
| Ovaltine sachets | 7 | 2 |
| Good time milk sachets | 7 | 6 |
| Peak milk tin | 7 | 7 |
| Bravia | 7 | 2 |
| Nestle Cerelac 1 | 7 | 3 |

**EXECUTION TIME: 1.30 SECONDS**
**PENALTY POINT: 53**

**Table 7: Second Iteration**

| PRODUCT NAME | SHELF | COLUMN |
|---|---|---|
| Horlick Milk | 1 | 7 |
| Tin Peak milk | 1 | 5 |
| Tin Three crown | 1 | 7 |
| Plastic coffee mate | 1 | 1 |
| Nescafe classic | 1 | 6 |
| Miksi | 1 | 5 |
| Sachet Three Crown 2 | 1 | 6 |
| Sachet Twisco 1 | 1 | 4 |
| SMA 2 | 1 | 5 |
| De-hero milk | 2 | 7 |
| Peak milk456 | 2 | 3 |
| Three crown Evap | 2 | 5 |
| Dano milk | 2 | 1 |
| Kremela sachets | 2 | 1 |
| Good time milk sachets | 2 | 1 |
| Tin Peak Gold | 2 | 3 |
| Cebon Café | 2 | 4 |
| Milo sachets 2 | 2 | 4 |
| Sachet Cadbury 1 | 2 | 4 |
| Nestle Lactogen (2) | 2 | 6 |
| Milo sachets | 3 | 2 |
| Sachet Cadbury | 3 | 6 |
| Tin Cowbell | 3 | 1 |
| Sachet Cowbell Coffee | 3 | 1 |
| Sachet Cowbell Strawberry | 3 | 7 |
| Instant coffee | 3 | 5 |
| Hollandia Evap 2 | 3 | 4 |
| Ovaltine sachets 1 | 3 | 7 |
| Loya milk sachets 1 | 3 | 5 |
| SMA 1 | 3 | 5 |
| Peak milk tin | 3 | 7 |
| Sachet Peak milk | 3 | 1 |
| Sachet Three Crown | 3 | 6 |
| Sachet Twisco | 3 | 7 |
| Sachet Oluji pure cocoa | 3 | 7 |
| Tin Oluji pure cocoa | 4 | 4 |
| Brown Gold | 4 | 1 |
| Hollandia Evap 1 | 4 | 6 |
| Dano Cool Cow 3 | 4 | 2 |

|  |  |  |
|---|---|---|
| Plastic Bournvita 1 | 4 | 6 |
| Plastic coffee mate 1 | 4 | 6 |
| Nestle NAN (1) | 4 | 7 |
| Marvel ORG Milk | 4 | 4 |
| Dano Slim | 4 | 6 |
| Loya milk sachets | 4 | 4 |
| Peak Chocolate sachets | 4 | 4 |
| Sachet Cowbell Chocolate | 4 | 2 |
| Maximus Café | 5 | 6 |
| Dano Cool Cow 1 | 5 | 3 |
| Dano Cool Cow 2 | 5 | 4 |
| Kremela sachets 1 | 5 | 3 |
| Tin Peak milk 1 | 5 | 3 |
| Tin Peak milk 2 | 5 | 3 |
| Sachet Peak milk 2 | 5 | 6 |
| Sachet Three Crown 1 | 5 | 4 |
| Miksi 1 | 5 | 2 |
| Nestle Lactogen (1) | 5 | 7 |
| Nestle NAN (3) | 5 | 4 |
| Nestle Cerelac | 5 | 7 |
| Cebon milk | 5 | 2 |
| Hollandia Evap | 5 | 2 |
| Ovaltine sachets | 5 | 5 |
| Plastic Bournvita | 6 | 3 |
| Sachet Bournvita | 6 | 3 |
| Sachet Cowbell milk | 6 | 6 |
| Stock well | 6 | 7 |
| Nescafe Gold | 6 | 2 |
| Bravia | 6 | 3 |
| Nestle Cerelac 2 | 6 | 2 |
| Sachet Peak milk 1 | 6 | 7 |
| Sachet Cowbell milk 1 | 6 | 1 |
| Sachet Cowbell milk 2 | 6 | 6 |
| Peak milk123 | 6 | 3 |
| Nestle NAN (2) | 6 | 5 |
| Peak Baby 0-12m | 7 | 2 |
| My Boy 0-12m | 7 | 1 |
| Dano Cool Cow (Tin) | 7 | 6 |
| Dano Cool Cow | 7 | 1 |
| Smart Diko | 7 | 5 |
| Superdelious tin milk | 7 | 2 |
| Nestle Cerelac 1 | 7 | 6 |

| | | |
|---|---|---|
| Milo sachets 1 | 7 | 7 |
| Dano Slim 1 | 7 | 2 |
| Sachet Cowbell Chocolate 1 | 7 | 3 |

**EXECUTION TIME: 1.36 SECONDS**
**PENALTY POINT: 54**

3rd**Iteration**

4th**Iteration**

5th**Iteration**

6th**Iteration**

7th**Iteration**

8th**Iteration**

9th**Iteration**

10th**Iteration**

**….**

50th**Iteration**

**Table 8: RESULT SUMMARY FOR TABU SEARCH ALGORITHM**

| ITERATION | RESULT (SOFT CONTRAINT FITNESS VIOLATION) | TIME TAKEN (SEC) |
|---|---|---|
| 1 | 53 | 1.30 |
| 2 | 54 | 1.36 |
| 3 | 59 | 1.28 |
| **4** | **51** | **1.25** |
| 5 | 58 | 1.30 |
| 6 | 58 | 1.28 |
| 7 | 58 | 1.37 |
| 8 | 53 | 1.23 |
| 9 | 56 | 1.31 |
| 10 | 53 | 1.25 |

Tabu search result discussion: The first TSA run output starts with iteration 1 and continues to iteration 50, out of which 10 best iterations were recorded each iteration includes the allocation of 84 products to 7 shelves and each shelf contain 7 column. The best allocation as indicted in the table above is **iteration 4** with penalty point of **51** and the execution time is **1.25sec**

**Analysis of harmony search result for the best 10 iterations of shelf space allocation**, in the first iteration, the algorithm returns a result with a penalty pointof 40 and total execution time of 1.38sec. The algorithm proceeds through the iterations, it continues to search for near global solution, aiming to reduce the penalty point and execution timewith an objective to satisfy more soft constraint.The Second iteration returns a result with penalty point 44 andtotal execution time of1.38sec, the Third (3rd) iteration returns a result with penalty point 43 and total execution time of1.40sec, the fourth (4th) iteration returns a result with penalty point 40 and total execution time of1.41sec, the fifth (5th) iteration returns a result with penalty point 43 and total execution time of1.39sec, the sixth (6th) iteration returns a result with penalty point 39 and total execution time of1.34sec, the seventh (7th) iteration returns a result with penalty point 47 and total execution time of1.39sec, the eighth (8th) iteration returns a result with penalty point 45 and total execution time of1.49sec, the ninth (9th) iteration returns a result with penalty point 44 and total execution time of1.43sec, while the tenth (10th) iteration returns a result with penalty point 44 and total execution time of1.36sec. in these 10 iterations it shows that the 6th iteration is the best allocation with a result that has penalty point 39 and total execution time of1.34sec.

**Analysis of Tabu search result for 10 iteration of shelf space allocation,**in the first iteration, the algorithm returns a result with a penalty pointof 53 and total execution time of 1.30sec. The algorithm proceeds through the iterations, it continues to search for near global solution, aiming to reduce the penalty point and execution timewith an objective to satisfy more soft constraint.The Second iteration returns a result with penalty point 54 and total execution time of1.36sec, the Third (3rd) iteration returns a result with penalty point 59 and total execution time of1.28sec, the fourth (4th) iteration returns a result with penalty point 51 and total execution time of1.25sec, the fifth (5th) iteration returns a result with penalty point 58 and total execution time of1.30sec, the sixth (6th) iteration returns a result with penalty point 58 and total execution time of1.28sec, the seventh (7th) iteration returns a result with penalty point 58 and total execution time of1.37sec, the eighth (8th) iteration returns a result with penalty point 53 and total execution time of1.23sec, the ninth (9th) iteration returns a result with penalty point 56 and total execution time of1.31sec, while the tenth (10th) iteration returns a result with penalty point 53 and total execution time of1.25sec. In these 10 iterations it shows that the 4th iteration is the best allocation with a result that has penalty point 51 and total execution time of1.25sec.

**Table 9: COMPARISON OF HARMONY SEARCH ALGORITHM AND TABU SEARCH ALGORITHM**

| HARMONY SEARCH ALGORITHM | PENALTY POINT (SOFT CONSTRAINT VIOLATION) | EXECUTION TIME (SEC) | TABU SEARCH ALGORITHM | PENALTY POINT (SOFT CONSTRAINT VIOLATION) | EXECUTION TIME (SEC) |
|---|---|---|---|---|---|
| 1 | 40 | 1.38 | 1 | 53 | 1.30 |
| 2 | 44 | 1.38 | 2 | 54 | 1.36 |
| 3 | 43 | 1.40 | 3 | 59 | 1.28 |
| 4 | 40 | 1.41 | **4** | **51** | **1.25** |
| 5 | 43 | 1.39 | 5 | 58 | 1.30 |
| **6** | **39** | **1.34** | 6 | 58 | 1.28 |
| 7 | 47 | 1.39 | 7 | 58 | 1.37 |
| 8 | 45 | 1.49 | 8 | 53 | 1.23 |
| 9 | 44 | 1.43 | 9 | 56 | 1.31 |
| 10 | 44 | 1.36 | 10 | 53 | 1.25 |

The comparison between the Harmony Search Algorithm and the Tabu Search Algorithm reveals differences in performance based on two key criteria: penalty points for soft constraint violations and execution time. The Harmony Search Algorithm consistently produces lower penalty points compared to the Tabu Search Algorithm, indicating that it is more effective in minimizing violations of soft constraints. The lowest penalty recorded for Harmony Search is 39, whereas for Tabu Search it is 51, and the highest penalty for Harmony Search is 45, while Tabu Search reaches 59. This suggests that Harmony Search is better at finding solutions that adhere more closely to constraints in this work. However, when it comes to execution time, the Tabu Search Algorithm operates slightly faster than the Harmony Search Algorithm. The fastest execution time for Harmony Search is 1.34 seconds, while Tabu Search achieves 1.23 seconds. The worst execution for Harmony Search is 1.49 seconds, compared to 1.37 seconds for Tabu Search. This implies that Tabu Search may be more time-efficient. In conclusion, the Harmony Search Algorithm is preferable when the priority is minimizing constraint violations, while the Tabu Search Algorithm is better suited for applications where speed is a crucial factor.

**Table 10:BEST ITERATION SUMMARY**

| ALGORITHM | BEST ITERATION | PENALTY POINT (SOFT CONSTRAINT VIOLATION) | EXECUTION TIME (SEC) |
|---|---|---|---|
| Harmony Search Algorithm | 6th | 39 | 1.34 |
| Tabu Search Algorithm | 4th | 51 | 1.25 |

## PERFORMANCE COMPARISON

**Soft Constraint Satisfaction (Penalty Point)**:

**Harmony Search wins**: It achieves a lower penalty point (**39**) compared to Tabu Search (**51**), meaning it satisfies soft constraints better.

**Execution Time**:

**Tabu Search is slightly faster** in its best iteration (**1.25 sec**) than Harmony Search (**1.34 sec**), but the difference is minimal (**0.09 sec**).

No Free Lunch theorem suggests no algorithm is universally best (**Wolpert & Macready, 1997**). The results of the performance of the algorithms reflect the No Free Lunch (**NFL**) theorem in the optimization, which states that no algorithm outperforms others across all possible problems. In this case, Harmony search performs better while satisfying soft constraints by achieving a lower penalty point (**39 vs. 51**), while Tabu search is better in terms of execution time (**1.25 sec vs. 1.34 sec**). This illustrates that each algorithm has strengths in different aspects, and there is no universally superior method for all evaluation criteria supporting the core idea of the NFL theorem.

# CHAPTER FIVE

# SUMMARY, CONCLUSION AND RECOMMENDATIONS

## 5.1 Summary

The system begins by collecting input data, including number of shelves and products, product name, weight and demand. To address the shelf space allocation problem, two metaheuristic algorithms, Harmony Search and Tabu Search,are implemented and compared. Each algorithm independently generates allocation of shelf while attempting to satisfy constraints such as average demand product to Shelf 1 and 2, high demand product to column 3, 4 and 5, and low product to column 6 and 7. The Harmony Search algorithm initializes a set of potential solutions, known as "harmonies," and uses improvisation-inspired operators memory consideration, pitch adjustment, and randomization to explore the solution space. Its strength lies in generating diverse solutions early in the search process, promoting broad exploration. In contrast, the Tabu Search algorithm starts with an initial solution and iteratively improves it by exploring neighboring solutions while avoiding cycles using a tabu list a short-term memory structure that stores recently visited solutions. This algorithm focuses on intensifying the search around promising areas of the solution space and escaping local optima. Both algorithms are evaluated based on solution quality, convergence speed, and their ability to meet the shelf space allocation constraints. The results provide insight into the strengths and weaknesses of each approach in solving complex optimization tasks.

## 5.2 Conclusion

The comparative analysis of the Harmony Search and Tabu Search algorithms provides valuable insights into optimizing shelf space allocation. By examining the performance, convergence behavior, and solution quality of each method, retailers can identify the most suitable algorithm for aligning product placement with demand patterns and business objectives. Selecting the appropriate approach can lead to more efficient space utilization, increased product visibility, and ultimately, improved sales performance and customer satisfaction.

In the adaptation and implementation of these algorithms, it was observed that the solutions were able to satisfy the specified constraints, which significantly shows the effectiveness of the Harmony Search algorithm and Tabu search algorithmfor solving the shelf space allocation problem.

The **overall better Algorithm** is **Harmony Search** even though it's marginally slower, its significantly **lower penalty point (39 vs 51)** means it **better satisfies soft constraints**, which is usually the main goal in shelf space allocation problems. The algorithm's performance was evaluated based on the result obtained from the 10 iteration of each algorithm and the computational resources utilized.

## 5.3 Recommendations

Based on the comparative evaluation of harmony search algorithm and tabu search algorithm on shelf space allocation system, the following recommendations can be made

1. **Harmony search algorithm Suitability:** The Harmony Search algorithm is more suitable for scenarios where the primary objective is achieving higher solution quality through better satisfaction of soft constraints, even if it comes at the cost of slightly increased execution time.

2. **Tabu search algorithm Suitability:** Conversely, Tabu Search is more suitable in scenarios where execution speed is a priority, and near-optimal solutions are acceptable. Its ability to produce reasonable solutions quickly makes it a strong candidate for time-sensitive applications.

3. **Dynamic Parameter Setting:**Dynamic parameter setting refers to the technique of adjusting algorithm control parameters such as the Harmony Memory Considering Rate (HMCR) and Pitch Adjustment Rate (PAR) during the execution of an optimization algorithm, rather than keeping them fixed or relying on randomness. In the context of the Harmony Search algorithm, this approach allows the algorithm to adaptively balance between **exploration** (searching new areas of the solution space) and **exploitation** (refining known good solutions).

4. **Feedback:**Collecting detailed feedback from the store regarding the impact of the changes on both sales performance and customer satisfaction is essential. This feedback will provide valuable insights into what is working well and highlight specific areas where further improvements can be made to enhance the overall shopping experience and drive better sales outcomes.

5. **Directions for Further Study:** It is recommended that future research explore the implementation of the Tabu Search algorithm with and without the aspiration criterion. A

comparative analysis between the two approaches could provide deeper insights into their impact on solution quality, convergence behavior, and the ability to escape local optima particularly in complex optimization problems such as shelf space allocation. This investigation would help determine whether incorporating the aspiration criterion consistently yields superior performance or if its benefits are problem-dependent.

## REFERENCES

Weyland, D. (2010). A rigorous analysis of the harmony search algorithm: How the research community can be misled by a "novel" methodology. *International Journal of Applied Metaheuristic Computing, 1*(2), 50-60.

Gao, X. Z., Govindasamy, V., Xu, H., Wang, X., & Zenger, K. (2015). Harmony search method: Theory and applications. *Computational Intelligence and Neuroscience, 2015*, Article ID 258491. https://doi.org/10.1155/2015/258491

Kim, G., & Moon, I. (2021). Integrated planning for product selection, shelf-space allocation, and replenishment decision with elasticity and positioning effects. *Journal of Retailing and Consumer Services, 58*, 102274. https://doi.org/10.1016/j.jretconser.2020.102274

Ngoc, C. M., Goh, S. L., Sze, S. N., Sabah, N. R., Abdullah, S., & Kendall, G. (2022). A survey of the nurse rostering solution methodologies: The state-of-the-art and emerging trends. *IEEE Access, 10*, 56504-56524. https://doi.org/10.1109/ACCESS.2022.3177208

Turek, J., Wolf, J. L., Pattipati, K. R., & Yu, P. S. (1992). Scheduling parallelizable tasks: Putting it all on the shelf. *Performance Evaluation Review, 20*(1), 225-236. https://doi.org/10.1145/133057.133111

Saleem, E. A., Dao, T.-M., & Liu, Z. (2018). Multiple-objective optimization and design of series-parallel systems using novel hybrid genetic algorithm meta-heuristic approach. *World Journal of Engineering and Technology, 6*(3), 532-555. https://doi.org/10.4236/wjet.2018.63032

Kusuma, P. D., & Prasasti, A. L. (2023). Multiple interaction-dual leader optimizer: A novel metaheuristic to solve high-dimension problem. *IAENG International Journal of Computer Science, 50*(2), IJCS_50_2_13.

Lesch, V., Müller, D. P. B. M., Krämer, M., Hadry, M., Kounev, S., & Krupitzer, C. (2023). Optimizing storage assignment, order picking, and their interaction in mezzanine warehouses. *Applied Intelligence, 53*, 18605-18629. https://doi.org/10.1007/s10489-022-04443-x

Tian, Z., & Zhang, C. (2018). An improved harmony search algorithm and its application in function optimization. *Journal of Information Processing Systems, 14*(5), 1237-1253. https://doi.org/10.3745/JIPS.04.0090

Tuo, S., Liu, H., & Chen, H. (2020). Multipopulation harmony search algorithm for the detection of high-order SNP interactions. *Bioinformatics, 36*(16), 4389-4398. https://doi.org/10.1093/bioinformatics/btaa215

Janeiro, A. C. R. (2014). Optimization algorithms for the shelf space allocation problem (Master's thesis). Faculdade de Engenharia da Universidade do Porto.

Sajadi, S. J., & Ahmadi, A. (2022). An integrated optimization model and metaheuristics for assortment planning, shelf space allocation, and inventory management of perishable products: A real application. *PLoS ONE, 17*(3), e0264186. https://doi.org/10.1371/journal.pone.0264186

Czerniakowska, K. (2021). A genetic algorithm for the retail shelf space allocation problem with virtual segments. *OPSEARCH*. https://doi.org/10.1007/s12597-021-00551-3

Genosman, B. C., & Begen, M. A. (2021). Exact optimization and decomposition approaches for 2D shelf space allocation. *Journal of Retail Analytics*, 10(4), 125-145.

Geismar, H. N., Dewande, M., Murthi, B. P. S., & Sriskandarajah, C. (2015). Maximizing revenue through two-dimensional shelf space allocation. *European Journal of Operational Research, 245*(1), 234-248. https://doi.org/10.1016/j.ejor.2015.03.012

Janeiro, A. C. R. (2014). Optimization algorithms for the shelf space allocation problem (Master's thesis). *Faculdade de Engenharia da Universidade do Porto.*

Kim, G., & Moon, I. (2021). Integrated planning for product selection, shelf-space allocation, and replenishment decision with elasticity and positioning effects. *Journal of Retailing and Consumer Services, 58*, 102274. https://doi.org/10.1016/j.jretconser.2020.102274

Sajadi, S. J., & Ahmadi, A. (2022). An integrated optimization model and metaheuristics for assortment planning, shelf-space allocation, and inventory management of perishable products. *PLoS ONE, 17*(3), e0264186. https://doi.org/10.1371/journal.pone.0264186

Czerniakowska, K. (2021). A genetic algorithm for the retail shelf space allocation problem with virtual segments. *OPSEARCH*. https://doi.org/10.1007/s12597-021-00551-3

Gecili, H. (2020). Joint shelf design and shelf space allocation problem for retailers. *Annals of Operations Research, 295*(1), 47-72. https://doi.org/10.1007/s10479-020-03578-7

Ostermeier, M., Dusterhoft, T., & Hubner, A. (2022). A model and solution approach for store-wide shelf space allocation. *European Journal of Operational Research, 302*(1), 155-173. https://doi.org/10.1016/j.ejor.2022.04.015

Murray, C. C., Talukdar, D., & Gosavi, A. (2010). Joint optimization of product price, display orientation, and shelf-space allocation in retail category management. *Operations Research, 58*(2), 396-411. https://doi.org/10.1287/opre.1090.0742

Gencosman, B. C., & Begen, M. A. (2021). Exact optimization and decomposition approaches for 2D shelf space allocation. *Journal of Retail Analytics, 10*(4), 125-145.

Chen, M., & Lin, C. P. (2007). A data mining approach to product assortment and shelf space allocation. *Decision Support Systems,* *44*(4), 1035-1052. https://doi.org/10.1016/j.dss.2007.01.001