IJOA.

International Journal on Optimization and Applications

VOL 06 - ISSUE 01
2026

Editor in Chief

Prof. Dr. Hanaa HACHIMI

ISSN : 2737-8314

International Journal on Optimization and Applications

Vol 06 – Issue 01

2026

Editor in Chief

Prof. Dr. Hanaa HACHIMI

ISSN : 2737 – 8314

FOREWORD

The International Journal on Optimization and Applications (IJOA) is an open access, double blind peer-reviewed online journal aiming at publishing high-quality research in all areas of : Applied mathematics, Engineering science, Artificial intelligence, Numerical Methods, Embedded Systems, Electric, Electronic engineering, Telecommunication Engineering... the IJOA begins its publication from 2021. This journal is enriched by very important special manuscripts that deal with problems using the latest methods of optimization. It aims to develop new ideas and collaborations, to be aware of the latest search trends in the optimization techniques and their applications in the various fields..

Finally, I would like to thank all participants who have contributed to the achievement of this journal and in particular the authors who have greatly enriched it with their performing articles.

Prof. Dr. Hanaa HACHIMI Editor in Chief

Full Professor in Applied Mathematics & Computer Science National School of Applied Sciences,
Ibn Tofail University, Kenitra , Morocco

TABLE OF CONTENT

International Journal on Optimization and Applications
IJOA. Vol. 06, Issue No. 01, Year 2026, www.ijoa-journal.com

AI Workload Management: A Comparative Analysis of Process Scheduling Algorithms Evaluating Scheduling Algorithms for AI Optimization

Lucci Ania L. J. Fagela
University of the Cordilleras
ljf7925@students.uc-bcf-edu.ph

Jenny Ann T. Guyong
University of the Cordilleras
jtg6745@students.uc-bcf-edu.ph

***Abstract*—Efficient workload management is critical for optimizing the performance of AI-based systems, especially in environments with diverse and dynamic processing requirements. This study presents an extensive examination of five CPU scheduling algorithms with regard to their performance in managing AI workloads, focusing on First-Come, First-Served (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Priority Scheduling, and Round Robin (RR). Through a scenario-based approach, each algorithm was analyzed based on its waiting time and turnaround time. To gather the data, a simulation tool was used with random arrival times and burst times, which provided insights into their adaptability for various AI processes. The results revealed SRTF to be the superior one among the five despite its frequent context switching. FCFS and RR excel in simplicity and fairness but face inefficiencies such as the convoy effect and high context-switching overhead. On the other hand, SJF and Priority Scheduling perform well in predictable or hierarchical workloads, minimizing turnaround and waiting times, but encounter challenges with starvation and dynamic task management. The study concludes that no single algorithm universally outperforms the others, as their effectiveness depends on workload characteristics and system requirements. This highlights the importance of selecting scheduling algorithms aligned with specific AI workload demands to maximize system efficiency and performance.**

***Index Terms*—Keywords—CPU scheduling algorithms; Artificial intelligence optimization; Process management; Scheduling performance metrics; Modeling and simulation.**

I. Introduction

A. Background of the Study

Technology paved the way for the development of computers and mobile devices that enabled people to manage all aspects of daily life, such as shopping, online learning, booking appointments, and chatting. At the core of this technological shift are computers and mobile devices, which have become indispensable tools in modern life. These devices are able to seamlessly function with the advent of operating systems, a type of system software that manages the resources of the computer and facilitates the system's management and control.

When a user uses a computer and performs a task, such as opening a document, there are often background jobs for sending or receiving email. In these instances, the numerous processes need to be handled by the Central Processing Unit (CPU), which is responsible for processing computer operations. However, only a single process is allocated to the CPU at a time. This is called scheduling, which is the allocation of computer resources to tasks in order to make full use of the CPU [1]. Scheduling algorithms are developed to help the system efficiently manage process execution even while other processes are on hold.

In recent years, artificial intelligence (AI) has emerged as a transformative technology that enables machines to perform tasks that typically require human intelligence . As evident, modern processors, such as the Intel Core Ultra, now integrate AI into their operating systems to support more advanced software and meet the growing demands of AI applications. These systems rely on powerful computational resources to process complex data; hence, the choice of scheduling algorithm for managing AI workloads in these systems becomes critical to ensuring the smooth execution of multiple and complex processes.

1) Research gaps: While scheduling algorithms have been extensively studied, limited investigations have examined their performance specifically within the context of managing AI workloads. Although existing studies have evaluated various scheduling algorithms [2]–[4],

they have not fully explored how well these algorithms meet AI-specific demands.

In 2018, OpenAI reported that the computational power required to train the largest AI models had been doubling every 3.4 months since 2012, debunking the predictions of Moore's Law, where computing power doubles every two years [5]. This situation calls for an urgent need to deeply understand how advanced systems must be implemented to effectively manage AI workloads.

As AI continues to advance and integrate into various fields, determining which scheduling algorithms are best suited for managing AI workloads becomes essential. This ensures that systems are built with algorithms capable of managing resources effectively even under advanced processing power. Furthermore, only a few studies have conducted a comprehensive comparison of the five selected scheduling algorithms considered in this study, highlighting a notable gap in the current research landscape.

2) Current body of knowledge: Artificial intelligence (AI), along with machine learning (ML), is one of the defining technological trends of the modern era. Technologies ranging from computers to mobile devices increasingly incorporate AI capabilities. This trend continues to evolve, impacting various fields including business, industry, and everyday life.

The foundation of AI lies in analyzing, categorizing, and predicting outcomes based on user input data, thereby facilitating improved decision-making processes. AI aims to replicate human thought and behavior, focusing on aspects such as perception, reasoning, learning, and prediction. Xu [6] suggests that by understanding these human concepts, society may progressively replace certain forms of human labor and resources.

As AI continues to evolve, its applications expand across diverse sectors including healthcare, finance, transportation, and many others. These applications rely heavily on AI's ability to process large volumes of data, adapt to new inputs, and generate predictions through complex algorithms, which in turn leads to highly resource-intensive workloads.

AI workloads refer to tasks performed by AI systems that often involve processing vast amounts of data and executing complex computations [7]. These tasks include predictive analysis forecasting, Natural Language Processing (NLP), anomaly detection, image or video recognition, and recommendation algorithms [8].

Such workloads are highly resource-intensive, requiring substantial computational power, memory, and storage to process complex computations and analyze extensive datasets [9]. As Sekar [10] explains, AI workloads are characterized by their significant computational requirements during both the training and inference phases of machine learning models. Training may involve millions of iterations of algorithm optimization, while inference requires rapid computations to deliver real-time results.

Consequently, efficient handling of data movement, computational resources, and algorithm optimization is required to achieve effective outcomes [7]. Despite their advantages, AI workloads present several challenges that complicate their implementation. According to Vohra [9], resource demand remains a primary concern, with workloads requiring high-performance hardware such as GPUs, TPUs, or ASICs to support computational intensity.

Data movement also poses a significant challenge, as AI tasks require efficient handling of large datasets across distributed computing systems. Furthermore, the complexity of deploying multiple interdependent components—including data preprocessing, model training, and monitoring—necessitates advanced management techniques.

Scalability represents another major challenge. Increasing data volumes and algorithmic complexity demand infrastructures that can scale seamlessly [7]. Traditional computing systems often struggle to meet these requirements, leading to performance limitations.

Efficient management of AI workloads requires strategies that optimize both resource utilization and operational efficiency. Pacheco [8] highlights the importance of high computational power that enables parallel processing and supports AI workloads effectively. Specialized hardware configurations are essential for AI systems to operate efficiently and achieve high performance.

Key hardware components include Central Processing Units (CPUs), Graphics Processing Units (GPUs), Tensor Processing Units (TPUs), Field-Programmable Gate Arrays (FPGAs), and Application-Specific Integrated Circuits (ASICs), each serving distinct roles in optimizing AI functionality.

CPUs provide essential processing power and system control, managing complex AI tasks at a fundamental level. GPUs, with their strong parallel processing capabilities, are particularly suitable for the intensive computations associated with AI operations, especially neural network training.

TPUs are specifically designed to accelerate AI computations, significantly improving performance for deep learning tasks. FPGAs offer a combination of performance and flexibility through their reconfigurability, enabling implementation of customized algorithms for evolving AI applications.

Finally, ASICs are custom-designed hardware solutions optimized for specific tasks, maximizing efficiency by tailoring hardware to dedicated AI functions and achieving peak performance for specialized applications [11]. Leveraging these hardware accelerators can significantly enhance the performance of AI workloads [8]. However, when such hardware resources are limited, another key factor that can support the management of AI workloads is parallelization and distributed computing. Parallelization and distributed computing play a crucial role in handling AI workloads by breaking tasks into smaller and manageable units that can be processed simultaneously across multiple systems [7]. Sekar [10] further supports this by demonstrating that dynamic resource allocation and containerization can improve resource utilization by up to 85%. This approach not only accelerates data processing and model training but also ensures scalability, enabling AI applications to manage larger datasets and increasingly complex algorithms without being constrained by hardware limitations.

Efficient AI workload management has become a critical component of modern computing, allowing organizations to maximize the potential of artificial intelligence while optimizing resource utilization and operational costs. According to Vohra [9], efficient workload management reduces latency by ensuring that AI applications receive the required resources at the appropriate time, thereby preventing performance bottlenecks. Additionally, scalability and flexibility are significant advantages, as they allow systems to dynamically adapt to fluctuations in workloads. This adaptability ensures consistent performance even during varying computational demands.

The growing complexity and diversity of computing environments have emphasized the importance of efficient scheduling algorithms in AI and broader computing systems. As explained by Mohammadjafari and Khajouie [12], early computing systems relied on simple batch processing techniques to execute tasks sequentially. However, the evolution of computing environments now requires more sophisticated scheduling approaches. Process scheduling is fundamental to modern computing systems, as it governs how CPU time is allocated among multiple processes. Scheduling mechanisms ensure fair resource distribution while minimizing idle time, thereby improving system performance and maintaining operational stability.

Recent developments in computing architectures have introduced heterogeneous processing units such as CPUs, GPUs, and ASICs, which provide unprecedented computational capabilities. Nevertheless, Mohammadjafari and Khajouie highlight that effectively utilizing these resources requires scheduling strategies specifically designed for heterogeneous architectures. Effective schedulers must consider the diverse computational resources available within a system and analyze thread behavior patterns to optimize their performance across different cores, as emphasized by Nemirovsky and others [13].

Algorithms serve as the foundation of artificial intelligence by enabling machines to mimic human intelligence and execute complex computational tasks. These algorithms process data to derive meaningful insights and continuously adapt to new information. The effectiveness of AI systems largely depends on the selection and implementation of appropriate algorithms. A wide range of algorithms contributes to enhancing AI capabilities in data analysis, pattern recognition, search optimization, and performance improvement [11]. Common AI algorithms include search and optimization algorithms, supervised and unsupervised learning algorithms, neural networks, reinforcement learning algorithms, computer vision algorithms, and natural language processing (NLP) algorithms. Exploring diverse algorithmic strategies is highly recommended for improving current and future AI systems and expanding their functional capabilities [14].

Among these computational approaches, scheduling algorithms play a crucial role in determining the performance of AI workloads, as they directly influence resource allocation, system latency, and computational throughput. The efficiency of AI systems strongly depends on how computational resources are distributed and how tasks are prioritized. However, a major challenge lies in adapting these algorithms to the specific requirements of AI workloads.

This study investigates and compares five scheduling algorithms in order to determine which algorithm is the most efficient for managing AI workloads. The algorithms examined in this study include: (a) First-Come, First-Served (FCFS), (b) Shortest Job First (SJF), (c) Shortest Remaining Time First (SRTF), (d) Priority Scheduling, and (e) Round Robin.

The First-Come, First-Served (FCFS) algorithm is considered the simplest scheduling approach [15], [16], as it processes tasks in the order in which they arrive without preemption. In CPU scheduling, preemptive scheduling occurs when the currently executing process is forced to release the CPU because a higher-priority process arrives in the queue, such as system interrupts or system calls [17]. The simplicity of FCFS ensures fairness because all processes are eventually executed, thereby eliminating starvation [18]. Starvation occurs when a process is deprived of CPU resources because other processes continuously utilize them.

Despite its fairness, FCFS suffers from several performance limitations. One major drawback is the *convoy effect*, where processes with long CPU burst times significantly increase the waiting time and turnaround time for other processes [19]. Waiting time refers to the total time a process spends in the ready queue before execution, while turnaround time represents the total time required to complete a process from submission to completion [15]. Goel and Garg [18] highlight that the absence of prioritization in FCFS can lead to reduced system throughput, as long processes may monopolize the CPU. Consequently, although FCFS ensures simplicity and fairness, it often leads to inefficient resource utilization and poor response times in systems with diverse process burst durations.

The Shortest Job First (SJF) algorithm improves upon FCFS by addressing the convoy effect. SJF prioritizes tasks with the shortest execution time, allowing smaller tasks to complete earlier. Specifically, SJF selects the process with the smallest burst time, which significantly reduces average waiting time and turnaround time [15]. However, SJF also presents several practical challenges. One major limitation is the difficulty of accurately estimating the burst time of processes in advance [16], [18]. Furthermore, although SJF improves overall efficiency, it introduces the possibility of starvation for processes with longer burst times, particularly in heavily loaded systems. This trade-off between efficiency and fairness makes SJF suitable for optimizing throughput but less appropriate for systems requiring equal attention to all processes. Nevertheless, the SJF algorithm remains an effective strategy for reducing queue waiting time and improving overall The Shortest Remaining Time First (SRTF) is a preemptive variant of SJF that processes the task that is closest to the completion of its execution. This dynamically reallocates the CPU to processes with shorter remaining burst times upon their arrival, as described by Gonzalez-Rodriguez [16]. This approach offers further improvements in turnaround and waiting times over SJF but at the cost of increased context switching [19]. Context switching refers to the process of storing and restoring the state of a CPU so that it can switch from executing one process to another. Whenever a new process arrives with a shorter remaining time than the currently running process, the algorithm preempts the current process, leading to frequent switches. Each context switch introduces overhead as the system must save and restore the state of processes, consuming valuable CPU cycles. CPU overhead refers to the additional time and resources consumed by the CPU that are not directly related to executing the actual tasks or processes. It is the extra "cost" brought by managing tasks rather than performing them. In environments with numerous processes arriving in quick succession, this overhead accumulates, which reduces overall efficiency and resource utilization. While SRTF is more efficient for systems with frequently varying process bursts, it suffers from similar starvation issues as SJF, particularly for processes with long burst times in environments with continuous short processes [19]. This adaptability makes SRTF suitable for systems prioritizing quick response but raises concerns about fairness and overhead. In comparison with Shortest Job First, SJF has a much slower execution than the SRTF. This algorithm allows for easier management of updates and replacements and enables efficient memory usage.

Priority Scheduling is a method that assigns each task a priority level, and tasks with higher priority values are processed first. Shakor [15] and Goel and Garg [18] emphasize that this method ensures responsiveness for high-priority tasks, and meeting deadlines effectively. However, the algorithm's drawback lies in the potential starvation of lower-priority processes, a recurring issue that can be mitigated through the aging technique, where the priority of a process increases as it waits in the ready queue [19]. Additionally, equal-priority processes may experience increasing waiting times due to sequential scheduling [18]. Priority scheduling's emphasis on responsiveness and deadline adherence makes it valuable for critical systems, though careful management is required to prevent starvation and imbalance. Priority Scheduling is often used in scenarios where certain tasks are more critical than others, ensuring that high-priority tasks receive the necessary resources before lower-priority ones. Round Robin (RR) is a widely implemented scheduling algorithm that aims to distribute resources evenly among tasks by assigning a fixed time quantum to each. This is the period of time a process is permitted to run in a preemptive multitasking system. Round Robin is designed for fairness and responsiveness, utilizing time quantum to cyclically allocate CPU resources to processes. It has the ability to mitigate FCFS's drawbacks by balancing process execution [15], [16]. However, Pemasinghe and Rajapaksha [19] caution that the effectiveness of RR depends on the quantum size; too small a quantum increases context switches, while a large quantum becomes similar to FCFS. As discussed by Goel and Garg [18], a shorter time quantum can cause many context switches that lower the CPU efficiency while a longer time quantum leads to poor response time. They further note that due to high waiting times, deadlines are rarely met in this method. Despite these challenges, RR is well-suited for time-sharing systems requiring

equitable CPU distribution. This approach ensures that all tasks receive a fair share of processing time, making it suitable for time-sharing systems where responsiveness is essential.

3) Applications of CPU Scheduling Algorithms: CPU scheduling is fundamental to AI for optimizing resource use, improving system performance, and ensuring efficient execution of computationally intensive tasks. Effective CPU scheduling enables maximum CPU utilization, which is crucial for managing AI workloads that demand heavy computations and translation processes. By strategically allocating CPU time, scheduling maximizes throughput, allowing AI systems to handle tasks with high efficiency and avoid resource wastage. According to Mehta and Mehta [4], throughput refers to the number of processes that are successfully executed per time unit.

Given the variability and intensity of AI workloads, which often require dynamic resource allocation, CPU scheduling helps maintain efficiency under changing demands. It optimizes task execution in order to reduce latency, thus decreasing waiting and response times, the key factors for real-time AI applications. Additionally, by ensuring multiple tasks are processed smoothly, scheduling enhances throughput, which facilitates the completion of more tasks within a given period.

Priority management is another significant benefit CPU scheduling brings to AI systems. It ensures equitable resource distribution among various users and processes, preventing system overload and supporting collaborative environments. Priority handling allows AI tasks to be ranked based on urgency and importance, enabling systems to handle high-priority tasks promptly without disrupting others.

Lastly, CPU scheduling supports scalability, allowing AI systems to process large datasets and manage complex models without compromising performance. This scalability is essential as AI applications continue to grow in complexity and scale. Overall, CPU scheduling in AI contributes to optimized resource utilization, improved efficiency, and sustained performance, even with demanding and variable workloads [20]. FCFS processes tasks in the exact order they arrive. This is particularly useful in AI task scheduling applications, as AI systems often process numerous tasks or operations. By executing tasks based on arrival order, FCFS facilitates efficient resource allocation and shared resource access, which is crucial for generating timely responses. FCFS also offers a systematic method for data handling, where data preprocessing tasks are queued to ensure sequential processing. Furthermore, FCFS effectively models real-world applications, aiding in the analysis and optimization of service workflows. Its primary benefit lies in its predictability, as the straightforward, sequential execution order makes it highly reliable [21].

Similar to FCFS, SJF is beneficial in AI for task scheduling, as it optimizes execution order by prioritizing shorter tasks, thereby enhancing overall efficiency. In batch processing systems, SJF minimizes throughput time by completing shorter tasks first, freeing up resources for subsequent processes. Given AI's demand for timely task processing, SJF aids in improving task completion efficiency with reduced wait times. Additionally, SJF models queuing behavior and resource allocation effectively, providing performance enhancements across varying workloads [22].

Meanwhile, SRTF selects the process with the least remaining execution. This is often implemented on modern operating systems for managing CPU scheduling, often in applications where quick responses are utilized. Moreover, SRJF is often applied to real-time systems and cloud computing systems, whereas this algorithm can be used to ensure that critical tasks are prioritized effectively and able to run multiple virtual applications concurrently. It allows for better optimization of resource allocation by ensuring that shorter tasks are completed quickly. Overall, SRJF helps in managing workload effectively and minimizes latency [23].

The Priority Scheduling algorithm assigns priority levels to tasks, ensuring that higher-priority tasks are executed before lower-priority ones. In AI, this algorithm is essential for real-time processing, as it allows urgent tasks to be completed without delay, enhancing responsiveness. It optimizes batch processing by prioritizing resource allocation to higher-priority tasks, enabling the system to handle processes effectively based on urgency and importance. This leads to improved efficiency and responsiveness, making Priority Scheduling theoretically beneficial for AI applications requiring timely task execution [24].

The Round Robin algorithm allocates CPU time to each process in a rotating, cyclic order, ensuring each process receives an equal share of CPU resources. This approach is advantageous in AI for managing CPU resources, as it enables AI workflows to handle multiple tasks concurrently. Round Robin supports real-time processing in AI applications, allowing critical and complex tasks to run without significant delays. Additionally, it provides valuable insights into system performance under varying workloads and is often employed in simulations to model effective task management and resource allocation strategies [25].

B. Objectives of the Study

From the outlined research gaps, the main objective of this study is to perform a comparative analysis of the five algorithms: First-Come, First-Served (FCFS), Shortest Job First (SJF), Shortest Remaining Job First (SRJF), Priority Scheduling, and Round Robin, and identify the most suitable algorithm for managing AI workloads within AI-based systems. The specific objectives and the research questions that this study aims to address are as follows:

- To determine the advantages and disadvantages of the scheduling algorithms based on the performance metrics.
- To examine and assess the capability of the scheduling algorithms in managing AI workloads.
- To identify which scheduling algorithm is the most optimal for use as a primary method in AI data handling tasks.

C. Methodology

In order to evaluate and compare the five scheduling algorithms, this study employed a quantitative comparative research design. Quantitative research utilizes statistical software, mathematical models, and computational algorithms to analyze numerical data [26]. This method significantly helped in evidence-based decision-making, which aided in determining which scheduling algorithm was the most efficient. On the other hand, a comparative research design is used to compare the performance of the scheduling algorithms across various performance metrics. In the context of the quantitative approach, this design involves comparing the values of two or more cases based on relevant variables and assessing them after [27].

D. Dataset

CPU scheduling has two key concepts that will be utilized in this research study as part of the dataset: **arrival time** and **burst time**. Arrival time indicates when a process enters the ready queue and begins waiting for execution, representing the moment a process is available to start its assigned tasks [28]. It depicts the time frame at which the process becomes available to complete the assigned job. This concept is fundamental in scheduling algorithms that prioritize processes based on their entry order, as it dictates which process is selected next. For instance, the First-Come, First-Serve (FCFS) algorithm uses arrival time to allocate CPU resources to processes in the order they arrive.

Arrival time can be determined by the formula:

$$Arrival\ Time = Completion\ Time - Turnaround\ Time$$

Burst time, also known as execution time, represents the total CPU time a process needs to complete its execution. Measured in milliseconds, burst time specifically indicates the time required by the CPU to execute a given task, excluding any I/O time. During this period, the process transitions from a running state to a completion state. Several factors influence burst time, including the complexity of the task, code efficiency, and the system's available resources.

Burst time can be determined by the formula:

$$Burst\ Time = Complete\ Time - Waiting\ Time$$

The key difference between arrival time and burst time lies in their roles within the process lifecycle. Arrival time marks the process's entry point into the ready queue, while burst time indicates the duration required by the CPU to execute the process to completion. Arrival time is determined before the process begins execution, whereas burst time is known once the process has been completed. Simply put, arrival time denotes when a process enters the queue, while burst time defines how long it will take for the process to execute [28], [29]. Incorporating both arrival time and burst time into CPU scheduling strategies enables the development of efficient algorithms that optimize overall system performance and responsiveness.

By considering these key parameters, scheduling can better manage processes within the operating system, ensuring resources are allocated in a timely and organized manner.

For this study, the data used to test the algorithms was adapted from the randomly generated arrival times and burst times simulated by Algabri et al. [30] who used a Java-based simulation framework. The current research study also used the five scenarios they detailed, which already have the arrival times and burst times needed for the study's analysis. Making use of eight processes similar to theirs, this approach allowed for a simulation of a wide array of operational conditions that reflect the dynamic and often complex nature of real-world computing environments.

TABLE I
DATASET ADAPTED FOR THE STUDY (TAKEN FROM ALGABRI ET AL. [30]).

Process	Scenario 1 (AT, BT)	Scenario 2 (AT, BT)	Scenario 3 (AT, BT)	Scenario 4 (AT, BT)	Scenario 5 (AT, BT)
1	AT: 0, BT: 10	AT: 8, BT: 4	AT: 1, BT: 10	AT: 0, BT: 4	AT: 6, BT: 1
2	AT: 3, BT: 10	AT: 10, BT: 4	AT: 1, BT: 10	AT: 2, BT: 8	AT: 8, BT: 6
3	AT: 4, BT: 1	AT: 11, BT: 5	AT: 2, BT: 9	AT: 3, BT: 5	AT: 10, BT: 10
4	AT: 9, BT: 4	AT: 13, BT: 8	AT: 2, BT: 7	AT: 5, BT: 4	AT: 11, BT: 4
5	AT: 12, BT: 4	AT: 14, BT: 1	AT: 10, BT: 2	AT: 6, BT: 5	AT: 14, BT: 2
6	AT: 16, BT: 10	AT: 16, BT: 4	AT: 14, BT: 10	AT: 6, BT: 7	AT: 15, BT: 7
7	AT: 18, BT: 7	AT: 18, BT: 5	AT: 15, BT: 8	AT: 10, BT: 6	AT: 15, BT: 10
8	AT: 18, BT: 10	AT: 20, BT: 2	AT: 16, BT: 1	AT: 15, BT: 4	AT: 17, BT: 2

TABLE II
FIVE SCENARIOS ADAPTED FOR THE STUDY (TAKEN FROM ALGABRI ET AL. [30]).

Scenario	Arrival time characteristics	Burst time characteristics	General observation
1	Steady, starting from time 0 with a noticeable initial gap	Ranges from 1–10, with several longer bursts	Potentially longer waits for processing
2	Later starts, spread out arrivals	Shorter on average, mostly under 5 units	Quicker processing, less initial congestion
3	Early and congested, most arriving within the first 2 units	Varied, from 1–10, unpredictable delays	Early congestion, varied processing times
4	Early like scenario 1, more evenly spread over time	Moderate, none exceeding 8 units	Balanced processing load
5	Later starts like scenario 2, arrivals spread out	Mostly short, between 2 and 7 units	Quick processing, less congestion

The randomness of the data reflected various scenarios, each representing varying levels of system load and operational demands. These are designed to capture different levels of complexity and unpredictability, providing a robust test environment for evaluating different scheduling algorithms. This setup facilitated a comprehensive understanding of a wide range of operational algorithms, contributing to the development of more resilient and flexible scheduling solutions capable of supporting a dynamic and robust system [30]. 2.2 Data gathering instruments The study employed simulation tools as the main instrument to analyze the performance of the five process scheduling algorithms. A simulation tool is software or a program that imitates the dynamics of a real-world process or system. The system's predefined history allows for gaining an insight into its characteristics and workings under a real-world scenario [31]. With a simulation model, the system's behavior is studied. Yin and McKay [33] defined the modeling and simulation domains. Among the modeling and simulation techniques, the current research study specifically requires a process and system simulation, which relates to the simulation of operational systems with the purpose of understanding their processes. With this in mind, the researchers looked for existing process scheduling simulation tools that are utilized in the study and deemed Deepak Aggarwal's program to be the most appropriate. The chosen tool allowed for the testing of algorithms using predefined inputs for arrival time and burst time and provided the key performance metrics needed in the study, which are waiting time and turnaround time. The tool supports the simulation of five process scheduling algorithms chosen for the study, which are First-Come, First-Served, Shortest Job First, Shortest Remaining Job First, Round Robin, and Priority Scheduling. By employing this simulation tool, the collected data on waiting time and turnaround time are used to compare how each algorithm performs across the eight distinct scenarios. This provided insights into the efficiency of the five algorithms and helped in determining the algorithms appropriate for managing AI workloads.

E. Data Gathering Procedures

The data gathering procedure for this study involved several key steps. First, the study adapted the research method from Algabri et al. [30], which utilized a Java-based simulation tool to generate random arrival times and burst times across five distinct scenarios. To maintain consistency, the arrival times and burst times from the original study were used, as they already effectively capture the values appropriate for each scenario, simulating eight processes that were also made use of in the current study (see Tables I and II).

A simulation tool that supports the five chosen scheduling algorithms was to be selected to assess their performance under the varying scenarios. However, since

the original simulation tool used in the original paper was not publicly available, an alternative tool was selected instead, which is Deepak Aggarwal's program that simulates FCFS, SJF, SRTF, Priority Scheduling, and Round Robin algorithms. The simulation tool is available in GitHub, a platform that allows developers to share their work and code within the field. It is also published as a web application, which the researchers utilized due to its user-friendly interface, simplifying the testing process.

The priority ranking used for the Priority Scheduling for processes 1 to 8 is 4, 3, 5, 1, 6, 4, 1, and 2, respectively. In real-world systems, priorities are usually based on criteria such as urgency, importance, deadlines, or resource requirements. However, for theoretical or academic purposes, randomly assigned priorities can help test a scheduling algorithm, particularly across diverse and unstructured scenarios.

In evaluating the Round Robin algorithm, varying sizes of time quantum were employed. This included a time quantum of 1, a time quantum of 5, and a time quantum of 10. This ensured that the choice of time quantum, which is a key concept of Round Robin, is considered as a factor that can influence its results.

Next, the input data consisting of the arrival times and burst times was fed into the simulation tool. Each simulation was run multiple times to ensure accuracy and to minimize variability in the results. The simulation tool output the two metrics needed in the study, waiting time and turnaround time, which helped to assess the effectiveness and performance of each scheduling algorithm in the context of managing AI workloads.

The output generated by each simulation was meticulously logged in a structured format, capturing all relevant timing data for each process, including the aforementioned two metrics for every scheduling algorithm. Each scenario was then analyzed using descriptive statistical methods, focusing on calculating averages for these key performance metrics. This approach allowed for a thorough assessment of the differences in performance, showing how each algorithm handles varying loads and process characteristics.

For instance, averages for waiting time and turnaround time provided insights into how promptly and efficiently each algorithm processes tasks in each scenario. The results were then represented visually through bar charts to illustrate and compare each algorithm's performance across different scenarios. Special emphasis was placed on arrival and burst times to observe how fluctuations in these variables impact the overall efficacy of the scheduling algorithms. This graphical representation enabled clear comparisons, revealing strengths, weaknesses, and patterns in each algorithm's performance.

Finally, the study contextualized these findings by examining their relevance to AI systems and assessing the capability of each algorithm in managing AI workloads.

F. Data Analysis Metrics

The selection of the appropriate scheduler for systems is an important task as it needs to ensure an efficient utilization of resources [33]. There are several CPU scheduling criteria that must be taken into consideration when designing a scheduler. Some criteria that are commonly used are turnaround time, waiting time, throughput, response time, and CPU utilization [4], [30], [33]. These metrics help to evaluate how well an algorithm manages system resources while maximizing system performance and minimizing delays. Each algorithm has its own characteristics and preferences. For this particular reason, which parameters to be used for the analysis has a huge influence on what algorithm is considered to be the best [4].

For this research, the focus was specifically on waiting time and turnaround time. The reason why these two metrics were chosen is driven by their relevance to the nature of the phenomena being studied—workloads of AI-based systems. Both waiting time and turnaround time are key metrics that directly influence how quickly tasks are completed and how responsive the system is in processing tasks. Therefore, these two measures should be well considered when analyzing CPU scheduling algorithms in relation to system performance [34].

For Round Robin, however, since it is directly affected by the time quantum, it was evaluated with varying quantum values (1, 5, and 10). This variation allows for an assessment of how different time slice lengths impact the overall performance of the algorithm in terms of the two key metrics mentioned. With all these considerations, the capabilities of the algorithms in managing AI workloads are established. The two metrics mentioned are further discussed in the sections below.

1) Turnaround Time: The turnaround time is the total amount of time taken from when a process arrives in the systems to when it is terminated [16]. This is the duration that is required for a process to complete its procedure, which is important for evaluating a system's performance and responsiveness [30]. The turnaround time should be lower to achieve an efficient scheduling algorithm [4]. A short turnaround time would mean that processes finish faster, which is particularly essential in computational tasks implemented in AI for faster system performance. The formula for determining the turnaround time is as follows:

$$Turnaround\,Time = Completion\,Time - Arrival\,Time \tag{I.1}$$

The completion time refers to the time the process completes its execution while the arrival time is the time when the process is now in the ready state. Alternatively, if the inputs for the waiting time and burst time are available, the turnaround time can also be calculated using the formula:

$$Turnaround\ Time = Burst\ Time + Waiting\ Time \tag{I.2}$$

While this provides the total execution time, turnaround time alone does not output any details of delays that occur during the waiting period. This is why waiting time is an important additional metric to consider for a more comprehensive analysis.

2) Waiting Time: According to Gonzalez-Rodriguez et al. [16], waiting time is the total amount of time the processes spend in the ready queue before their execution. This significantly helps in assessing how efficient a scheduling algorithm is. A low waiting time is preferred for a good scheduling algorithm [4]. Evaluating the waiting time is vital for system performance because this determines how long processes are delayed before they can actually start running. For systems that need high hardware requirements such as AI software or gaming applications, longer waiting times can cause noticeable delays, which will negatively affect user experiences. The formula for determining the waiting time is as follows:

$$Waiting\ Time = Turnaround\ Time - Burst\ Time \tag{I.3}$$

II. Findings

A. Simulation Results for the Average Waiting Time

The bar chart (Fig.1) showcases the comparison of the average waiting time for CPU scheduling algorithms: First-Come, First-Serve (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Priority Scheduling, Round Robin with quantum 1 (RR1), Round Robin with quantum 5 (RR5), and Round Robin with quantum 10 (RR10) across five distinct scenarios. The analysis of average waiting times, where lower values indicate better performance, revealed clear differences among the scheduling algorithms across the five scenarios.

In Scenario 1, SRTF achieved the lowest waiting time with 10.375 ms, followed by SJF with 10.375 ms. Meanwhile, RR1 had the longest delay at 20.375 ms, followed closely by RR5. In Scenario 2, SRTF once again led with a better performance of 5.875 ms of waiting time, followed by SJF at exactly 6 ms. RR1 further obtained the highest waiting time, performing at 14.75 ms.

Scenario 3 shows SRTF and Priority Scheduling outperforming the others with the lowest waiting time within the range of 14 ms. Meanwhile, FCFS, SJF, and RR10 all tied at 23 ms. Similar to some of the previous scenarios, RR1 (29 ms) and RR5 (29.125 ms) obtained the highest waiting times compared to other algorithms.

In Scenario 4, SRTF had the lowest waiting time at 10.25 ms, followed by FCFS, SJF, and RR10, which all recorded waiting times of 13.125 ms. For this scenario, the Round Robin algorithms RR1 and RR5 had the highest waiting times.

Finally, in Scenario 5, SRTF and SJF achieved the lowest waiting times, performing at 7.125 ms and 7.25 ms, respectively. Priority scheduling recorded a waiting time of 10.125 ms, while FCFS and RR10 both had slightly higher times at 12.25 ms. RR1 and RR5 had the highest waiting times at 14.75 ms and 15.375 ms, respectively.

These results highlight the differences in performance among the scheduling algorithms across various scenarios, with SJF and SRTF generally delivering the lowest average waiting times.

In the analysis of the average turnaround time (Fig.2), where a lower turnaround time indicates better performance, distinct patterns emerged across the five scenarios for the various scheduling algorithms.

In Scenario 1, RR1 achieved the lowest turnaround time at 14.75 ms, followed closely by SJF and SRTF, both at 16.75 ms. The highest turnaround time in this scenario was recorded by Round Robin with a quantum of 5 (RR5), reaching 25.5 ms.

For Scenario 2, SRTF performed the best with the lowest turnaround time of 8 ms, followed by SJF at 10.125 ms. FCFS, RR5, and RR10 obtained similar turnaround times of approximately 14 ms, while RR1 reached a higher value of 18.75 ms.

In Scenario 3, SJF and Priority Scheduling performed equally well with turnaround times of 21.375 ms and 21.15 ms, respectively. In contrast, RR1 and RR5 recorded the highest turnaround times of 36.125 ms and 36.25 ms.

For Scenario 4, SRTF again demonstrated strong performance with a low turnaround time of 15.625 ms, followed by SJF at 18.5 ms and Priority Scheduling at 18.75 ms. Meanwhile, RR1 recorded a higher value of 27.625 ms.

Finally, in Scenario 5, SJF and SRTF delivered the best results with turnaround times of 12.375 ms and 12.5 ms, respectively. In contrast, RR1 and RR5 exhibited significantly higher values within the range of 20 ms.

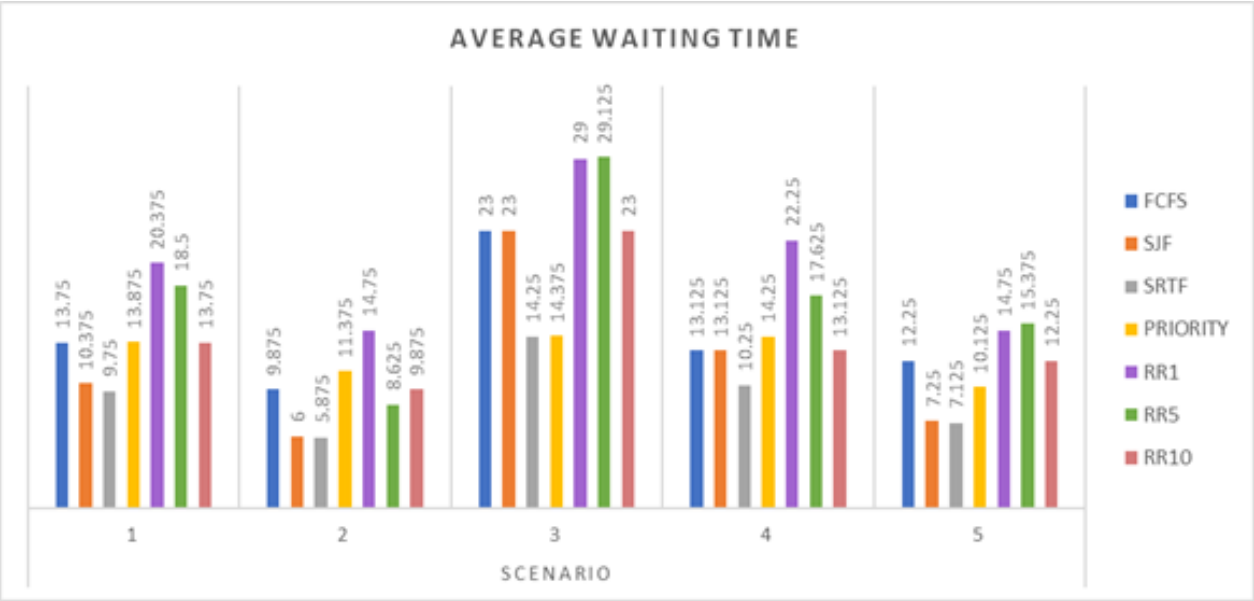

Fig. 1. Simulation results for the average waiting time.

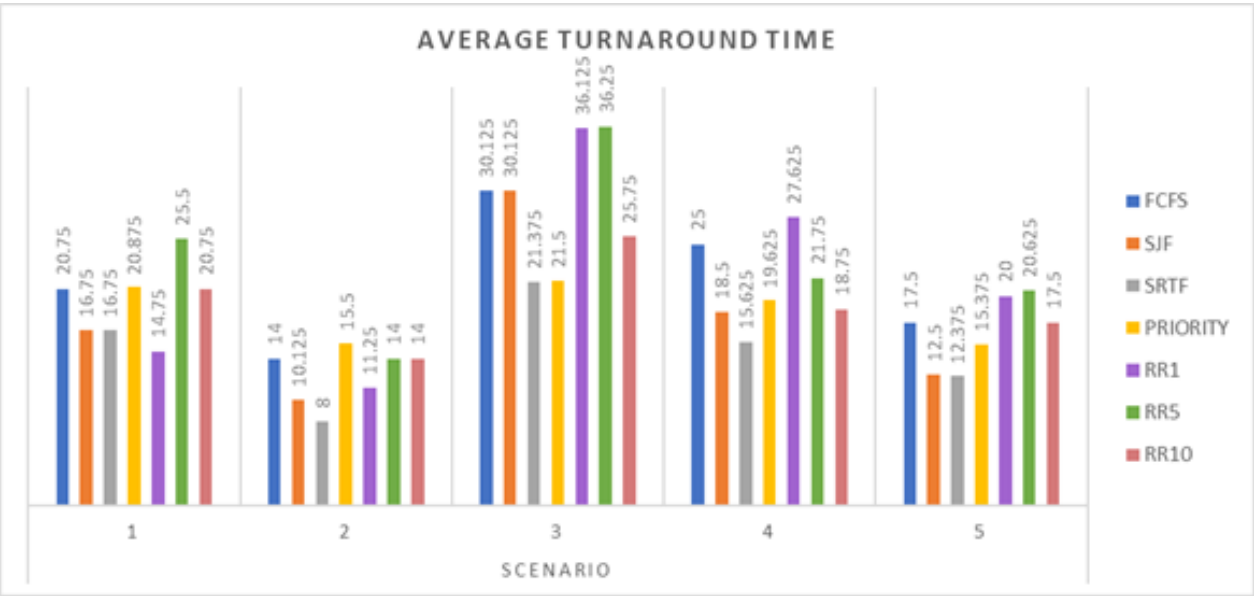

Fig. 2. Simulation results for the average turnaround time.

Overall, these results highlight that SJF and SRTF consistently provide lower turnaround times across most scenarios, indicating their efficiency in handling workloads that require faster task completion.

III. Discussion

A. Strengths and Limitations

1) First-Come, First-Served: The FCFS algorithm demonstrated simplicity in processing tasks in the order of their arrival without preemption, as highlighted by Shakor [15]. However, the results revealed that FCFS suffered from high waiting and turnaround times, particularly in scenarios involving processes with diverse burst

times. For instance, in Scenario 3, where the arrival times are early and congested while the burst times are high with unpredictable delays, FCFS recorded its highest average waiting time of 23 ms and its highest turnaround time of 30.125 ms.

This reflects the "convoy effect" described by Goel and Garg [18]. This phenomenon, where long burst-time processes delay shorter tasks, leads to increased inefficiencies and poor throughput, as the algorithm lacks prioritization mechanisms. In contrast, Scenario 2 showed the opposite results since the burst times are shorter on average (mostly under 5 units), allowing FCFS to attain its lowest waiting time of 9.875 ms and lowest turnaround time of 14 ms.

While FCFS eliminates starvation due to its fairness in eventually processing all tasks [18], the lack of optimization in waiting and turnaround times makes it less suitable for dynamic and resource-intensive environments. These results reaffirm the theoretical limitation of FCFS: its inability to prioritize or adapt, resulting in poor responsiveness in workloads with diverse burst times.

2) Shortest Job First: The SJF algorithm took advantage of its ability to prioritize shorter tasks, consistently achieving low waiting and turnaround times across most scenarios. In Scenario 2, characterized by short burst times mostly under 5 units, SJF delivered its lowest waiting time of 6 ms and its lowest turnaround time of 10.125 ms, reflecting its ability to address the convoy effect by prioritizing tasks with the shortest burst times.

This finding is consistent with Shakor [15], who stated that SJF prioritizes processes with the least burst times, significantly improving the average waiting time and average turnaround time. However, in Scenario 3, where burst times are high, SJF recorded its highest waiting time and highest turnaround time, tied with FCFS in both metrics. As noted by Goel and Garg [18], SJF can cause starvation for longer processes in saturated systems with frequent short tasks.

In addition, the practical implementation of SJF is hindered by the difficulty of accurately estimating process burst times, a limitation noted by Gonzalez-Rodriguez [16] and Goel and Garg [18]. This implies that while SJF is effective in reducing waiting times and turnaround times, its real-world application may require enhancements, such as burst-time prediction mechanisms, to mitigate its fairness issues.

3) Shortest Remaining Time First: The SRTF algorithm, being a preemptive variant of SJF, consistently delivered the lowest waiting and turnaround times across most scenarios. In Scenario 5, where processes had highly varied arrival times, SRTF outperformed every other algorithm, showcasing its adaptability in dynamic environments. This performance underscores its capability to dynamically reallocate CPU resources, as highlighted by Gonzalez-Rodriguez [16].

However, like SJF, it suffers from starvation issues [19]. This is reflected in Scenario 3, where SRTF recorded its highest waiting time and turnaround time. In this scenario, arrival times are early and congested, mostly starting within the first two time units. When processes arrive in quick succession, the scheduler frequently checks for the shortest or highest-priority task. In algorithms such as SRTF, this behavior leads to rapid preemptions and frequent context switching, which likely contributed to its relatively higher turnaround and waiting times compared to other scenarios.

In Scenario 2, however, SRTF achieved the lowest waiting time and turnaround time because the arrival times were more spread out and began later. When processes arrive with larger intervals, the scheduler has sufficient time to complete tasks before new ones enter the queue, thereby reducing context switching. This observation supports the findings in the literature [19], which indicate that although SRTF is highly efficient, it introduces overhead due to frequent process preemptions.

These results imply that SRTF is particularly suitable for environments requiring quick responses but may encounter efficiency challenges in systems with excessive context-switching occurrences. Despite these challenges, SRTF remains an effective choice for systems prioritizing quick response and low latency, as it outperformed most algorithms in scenarios characterized by rapidly changing workloads.

4) Priority Scheduling: The performance of Priority Scheduling in the study highlights its distinct advantages and limitations. Priority Scheduling assigns processes a priority value and schedules them based on these rankings, with lower numerical values representing higher priority. In the context of the given priority rankings for processes 1 to 8—4, 3, 5, 1, 6, 4, 1, and 2, respectively—the algorithm prioritized processes with higher ranks (lower numbers) over those with lower ranks (higher numbers).

In Scenario 3, where arrival times are congested with most processes arriving within the first two units, the algorithm struggled, resulting in high waiting times and high turnaround times. This poor performance stems from the simultaneous arrival of multiple processes, coupled with their assigned priority rankings. In this case, higher-priority processes, such as those with priorities 1, 2, and 3, monopolized the CPU, which forced lower-priority processes (e.g., with priorities 5 and 6) to wait for extended periods.

Shakor [15] and Goel and Garg [18] highlight that

Priority Scheduling heavily depends on the priority values, meaning that processes with lower rankings are deprioritized, leading to starvation or excessive waiting times [18], especially in congested scenarios like Scenario 3. The diversity in burst times corresponding to the priority rankings also added to the delay, as some higher-priority processes required longer CPU bursts, further increasing the waiting times for lower-priority tasks.

In contrast, Scenario 5 provided a more favorable environment for Priority Scheduling, with later and more spread-out arrivals as well as shorter and evenly distributed burst times. The algorithm achieved its lowest waiting time and lowest turnaround time in this scenario, demonstrating its efficiency when handling balanced workloads. Additionally, the shorter burst times reduced the likelihood of monopolization by any single process, enabling smoother transitions between tasks. Priority Scheduling's emphasis on responsiveness makes it suitable for systems with critical deadlines but still necessitates careful management to avoid starvation and imbalance.

5) Round Robin: The performance of Round Robin scheduling across scenarios reflects the direct influence of the quantum size and the characteristics of the workload. The findings reveal that RR with a quantum of 1 (RR1) consistently performed poorly, with the highest waiting times across scenarios, particularly in Scenario 3 (29 ms) and Scenario 4 (22.25 ms).

Similarly, RR with a quantum of 5 (RR5) exhibited high waiting times, suggesting inefficiencies when processes require significant CPU bursts. Pemasinghe and Rajapaksha [19] explained that shorter quantum sizes, such as 1 or 5, may lead to incomplete processing of tasks before another context switch occurs, adding delays in the waiting queue.

RR5 performed slightly better than RR1 but still lagged behind other algorithms due to similar inefficiencies in handling diverse workloads. In contrast, the larger quantum size of Round Robin with a quantum of 10 (RR10) provided a balance between minimizing context switches and ensuring fairness in CPU allocation, resulting in moderate performance across scenarios.

While RR10 did not achieve the best results with regard to the evaluated metrics, its performance was closer to FCFS, avoiding the drawbacks of excessive context switching associated with smaller quantum sizes. This aligns with Pemasinghe and Rajapaksha's [19] assertion that larger quantum sizes approximate FCFS behavior. Despite its fairness in distributing CPU time, the findings reveal that careful selection of the quantum size is crucial to optimize performance.

B. Process Scheduling on AI Workloads

The findings from the literature and the comparisons of various scheduling algorithms provide a comprehensive understanding of how they can manage AI workloads. FCFS is the simplest scheduling algorithm, where processes are executed in the order they arrive. While this algorithm is straightforward, its simplicity does not necessarily translate into efficiency when managing AI workloads.

Its vulnerability to the convoy effect can lead to inefficiencies in waiting and turnaround times, particularly when dealing with tasks of varying burst lengths [18]. The findings highlight that long tasks that arrive first block shorter tasks, leading to excessive waiting and poor system performance. This is particularly problematic in AI workloads that involve mixed tasks, such as training machine learning models while simultaneously performing real-time inference [8]. Moreover, as highlighted by Mohammadjafari and Khajouie [12], these delays in process execution can lead to poor performance and reduced user satisfaction.

FCFS is also unable to dynamically prioritize tasks based on urgency or computational requirements, which further limits its usefulness in AI systems that demand rapid processing and responsiveness.

In contrast, SJF addresses these inefficiencies by prioritizing tasks with the shortest burst times, thereby improving average waiting and turnaround times. In environments where AI workloads are predictable or involve computationally lighter tasks, such as natural language processing (NLP) or simpler predictive analytics, SJF can be an effective algorithm.

However, SJF faces challenges when tasks vary significantly in burst times, as it can lead to starvation of longer tasks (as observed in the findings), especially in AI systems that involve mixed workloads. Furthermore, it relies on accurate estimation of burst times, which is often difficult to obtain in real-world environments. Its tendency to cause starvation in heavily loaded systems limits its applicability in dynamic AI environments [15].

SRTF, as the findings suggest, adapts dynamically across different scenarios, which is critical in AI systems where computational loads are dynamic and change rapidly. This dynamic adjustment to task arrival and execution times makes SRTF highly effective for AI workloads with unpredictable or fluctuating resource demands.

As discussed by Vohra [9], AI workload management often involves unpredictable task patterns, which SRTF can handle effectively, as demonstrated in the results. SRTF also minimizes waiting time by processing shorter tasks as soon as they arrive, which is crucial for AI

systems relying on real-time data processing and immediate decision-making, such as anomaly detection or recommendation systems.

However, the downside of SRTF lies in the potential overhead caused by frequent context switching, especially when tasks have similar execution lengths. In such cases, the overhead can reduce the overall performance gains obtained through reduced waiting times.

Priority Scheduling provides an effective mechanism for managing tasks with varying levels of importance, which is common in AI applications involving critical and time-sensitive tasks, such as autonomous driving or healthcare diagnostics. However, its strict prioritization can lead to starvation of lower-priority tasks, particularly in environments where high-priority tasks continuously arrive.

This aligns with observations by Vohra [9], who notes that AI systems often require balancing multiple processes with different priorities and computational requirements. Consequently, the inflexibility of Priority Scheduling may reduce its effectiveness compared to adaptive algorithms such as SRTF.

Round Robin is particularly suitable for AI workloads operating in multi-user or multitasking environments, such as cloud-based AI platforms or shared computing infrastructures. Its time-sharing mechanism ensures fairness by allocating equal CPU time to all processes, preventing resource monopolization.

This property supports parallel task management [35], which is beneficial for scalable AI systems [10]. However, Round Robin can introduce high overhead due to frequent context switching. The results show that this leads to increased waiting times, which may negatively affect AI workloads requiring low latency and rapid response times.

C. Optimal Scheduling for AI Systems

Each of the algorithms analyzed presents unique strengths and limitations, but not all are equally suited to the demands of modern AI systems. Algorithms such as FCFS and Round Robin excel in simplicity and fairness, with FCFS ensuring tasks are processed in the order they arrive and RR providing equal CPU time allocation. However, these benefits are overshadowed by significant inefficiencies.

FCFS suffers from the "convoy effect," leading to high waiting times and poor responsiveness. In contrast, Round Robin's frequent context switching, especially with smaller quantum sizes, introduces a high overhead that hampers performance when implemented in intensive AI workloads.

SJF and Priority Scheduling perform relatively better in terms of minimizing waiting and turnaround times, particularly in scenarios with predictable or hierarchical workloads. SJF excels when tasks are short and burst times are predictable, while Priority Scheduling offers flexibility in prioritizing critical tasks. However, both algorithms face challenges in dynamic settings, with SJF prone to starving longer tasks and Priority Scheduling struggling with shifting task priorities, which may lead to delays for lower-priority tasks.

Among the five algorithms, SRTF consistently outperformed the others in most scenarios, making it stand out as the most effective algorithm for AI workloads due to its efficiency. SRTF demonstrated low waiting and turnaround times, which are critical metrics for AI applications such as real-time data processing, anomaly detection, and recommendation systems.

A key strength of SRTF lies in its responsiveness. Unlike non-preemptive algorithms such as SJF, SRTF dynamically preempts ongoing tasks when a shorter task arrives, reallocating CPU resources in real time. This adaptability allows it to outperform other algorithms in scenarios with rapidly changing workloads, ensuring optimal resource utilization and minimizing delays. For example, in scenarios with highly varied arrival times, SRTF demonstrated the best performance by efficiently handling all processes without being overly constrained by the arrival sequence. However, SRTF also has its drawbacks. The frequent context switching innate in its preemptive design can result in CPU overhead. This drawback can reduce overall system efficiency, making SRTF less ideal for workloads that involve sustained, long-running computations, such as training large machine learning models or processing massive datasets in parallel. Additionally, the potential for starvation of longer tasks—while mitigated compared to SJF—remains a concern in busy systems. These limitations indicate that while SRTF showed low average waiting times and low average turnaround times, it is only best suited to specific AI use cases that prioritize low latency and rapid response times. Nonetheless, SRTF consistently demonstrated superior performance across all of the scenarios, gaining its position as the most efficient scheduling algorithm among the five. Its dynamic nature aligns well with the demands of managing AI workloads requiring high adaptability and responsiveness, making it an indispensable choice for many modern applications.

IV. Conclusion

With the rise of artificial intelligence (AI), there is an increasing demand for complex applications across

various fields. To ensure that this technology delivers high performance while providing users with efficient applications, different computational approaches must be explored. This research focused on analyzing five CPU scheduling algorithms—First-Come, First-Served (FCFS), Shortest Job First (SJF), Shortest Remaining Time First (SRTF), Priority Scheduling, and Round Robin—in the context of AI workload management.

By examining the advantages and disadvantages of these algorithms and evaluating their performance metrics, particularly turnaround time and waiting time, the study highlighted their suitability for varying workloads. Using randomly generated arrival time and burst time data, the algorithms were tested across five distinct scenarios. The simulation results illustrated differences in turnaround times and waiting times, providing insights into how each scheduling algorithm manages varying computational loads. The results revealed the strengths and limitations inherent in each scheduling method.

FCFS, while simple and fair in processing tasks sequentially, is hindered by inefficiencies such as the "convoy effect," particularly in environments with diverse task burst times. This limitation reduces its effectiveness for AI systems requiring high responsiveness and efficiency. In contrast, SJF excels in minimizing waiting and turnaround times by prioritizing shorter tasks. However, its effectiveness depends on accurate burst-time predictions, and its tendency to starve longer tasks makes it less suitable for heterogeneous workloads.

SRTF, a dynamic and preemptive alternative, demonstrated superior adaptability across the tested scenarios, making it particularly effective for real-time and latency-sensitive AI applications. Nevertheless, frequent context switching may introduce overhead, which can reduce efficiency in workloads involving prolonged computations. Priority Scheduling provides flexibility in prioritizing critical tasks, making it suitable for time-sensitive AI applications. However, it may lead to starvation of lower-priority processes, especially in heavily loaded systems.

Round Robin, designed to ensure fairness in multi-user environments, distributes CPU time evenly among tasks. However, its performance is strongly influenced by the size of the time quantum. Smaller quantum sizes may result in excessive context switching, increasing waiting times and reducing overall efficiency in resource-intensive AI workloads.

Overall, the analysis indicates that no single scheduling algorithm universally outperforms the others, as each possesses specific strengths and limitations depending on the nature of the workload. However, among the five algorithms analyzed, SRTF consistently demonstrated superior performance in dynamic environments that require adaptability and low latency. In contrast, algorithms such as SJF or Priority Scheduling may be more appropriate for predictable or structured AI tasks.

These findings highlight the importance of selecting scheduling algorithms according to the specific requirements of AI workloads. Future work may explore hybrid scheduling strategies or adaptive scheduling mechanisms capable of dynamically balancing overhead, fairness, and responsiveness to further optimize AI system performance.

REFERENCES

[1] T. O. Omotehinwa, "Examining the developments in scheduling algorithms research: A bibliometric approach," *Heliyon*, vol. 8, no. 5, 2022. https://doi.org/10.1016/j.heliyon.2022.e09510

[2] T. A. Alghamdi, S. M. Ali, F. H. Almuhsin, R. F. Alsahrani, and E. E. El-Sharawy, "A review on the CPU scheduling algorithms: Comparative study," *International Journal of Computer Science and Network Security*, vol. 21, no. 1, 2021. https://doi.org/10.22937/IJCSNS.2021.21.1.4

[3] N. Kulkarni, S. A. Mahajan, and A. Patil, "A study of available process scheduling algorithms and their improved substitutes," *International Research Journal of Engineering and Technology*, vol. 7, no. 8, 2020.

[4] S. Mehta and H. Mehta, "Detailed analysis and simulation of various process scheduling algorithms," *International Journal of Algorithms Designs and Analysis*, vol. 6, no. 2, 2021.

[5] K. Hao, "The computing power needed to train AI is now rising seven times faster than ever before," *MIT Technology Review*, Aug. 23, 2024. https://www.technologyreview.com/2019/11/11/132004/the-computing-power-needed-to-train-ai-is-now-rising-seven-times-faster-than-ever-before/

[6] Y. Xu *et al.*, "Artificial intelligence: A powerful paradigm for scientific research," *The Innovation*, vol. 2, no. 4, 100179, 2021. https://doi.org/10.1016/j.xinn.2021.100179

[7] Cloudian, "6 types of AI workloads, challenges & critical best practices," Jun. 26, 2024. https://cloudian.com/guides/data-lake/6-types-of-ai-workloads-challenges-and-critical-best-practices/

[8] M. Pacheco, "AI workloads: Data, compute, and storage needs explained," TierPoint, Jul. 11, 2024. https://www.tierpoint.com/blog/ai-workloads/

[9] K. Vohra, "AI workload management in data centres: What you need to know," Hyperstack, Jun. 25, 2024. https://www.hyperstack.cloud/blog/case-study/ai-workload-management-in-data-centres

[10] J. Sekar, "Optimizing cloud infrastructure for AI workloads: Challenges and solutions," *International Journal of All Research Education & Scientific Methods*, vol. 12, no. 8, pp. 296–307, 2024.

[11] GeeksforGeeks, "Hardware requirements for artificial intelligence," Aug. 13, 2024. https://www.geeksforgeeks.org/hardware-requirements-for-artificial-intelligence/

[12] A. Mohammadjafari and P. Khajouie, "Optimizing task scheduling in heterogeneous computing environments," *arXiv*, May 13, 2024. https://arxiv.org/abs/2405.08187

[13] D. Nemirovsky, N. Markovic, M. Nemirovsky, and T. Arkose, "A machine learning approach for performance prediction and scheduling on heterogeneous CPUs," in *Proc. 29th Int. Symp. Computer Architecture and High Performance Computing*, 2017. https://doi.org/10.1109/SBAC-PAD.2017.23

[14] Tableau, "Artificial intelligence (AI) algorithms: A complete overview." https://www.tableau.com/data-insights/ai/algorithms

[15] M. Y. Shakor, "Scheduling and synchronization algorithms in operating system: A survey," *Journal of Studies in Science and Engineering*, vol. 1, no. 2, pp. 1–16, 2021. https://doi.org/10.53898/josse2021121

[16] M. González-Rodríguez, E. González-Rufino, L. Otero-Cerdeira, and F. J. Rodríguez-Martínez, "Study and evaluation of CPU scheduling algorithms," *Heliyon*, vol. 10, no. 9, 2024. https://doi.org/10.1016/j.heliyon.2024.e29959

[17] H. K. Omar, K. H. Jihad, and S. F. Hussein, "Comparative analysis of the essential CPU scheduling algorithms," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2742–2750, 2021. https://doi.org/10.11591/eei.v10i5.2812

[18] N. Goel and R. B. Garg, "A comparative study of CPU scheduling algorithms," *International Journal of Graphics & Image Processing*, vol. 2, no. 4, 2013.

[19] S. Pemasinghe and S. Rajapaksha, "Comparison of CPU Scheduling Algorithms: FCFS, SJF, SRTF, Round Robin, Priority Based, and Multilevel Queuing," in *Proc. IEEE Region 10 Humanitarian Technology Conference*, 2022.

[20] G. Salton, "Introducing Run:ai's CPU Scheduling," *Run:ai*, Jul. 19, 2024. https://www.run.ai/blog/introducing-run-ais-cpu-scheduling-improved-productivity-and-utilization-for-cpu-only-clusters

[21] "Different types of non-preemptive CPU scheduling algorithms." https://www.turing.com/kb/different-types-of-non-preemptive-cpu-scheduling-algorithms

[22] M. J. Rene and D. Kagaris, "Equitable Shortest Job First: A preemptive scheduling algorithm for soft real-time systems," Department of Electrical and Computer Engineering, 2014.

[23] N. Brooks, "Shortest job first (SJF): Preemptive, Non-Preemptive example," *Guru99*, Aug. 12, 2024. https://www.guru99.com/shortest-job-first-sjf-scheduling.html

[24] S. N. Chandra and V. Karthik, "Analysis of priority scheduling algorithm on the basis of FCFS & SJF," *International Journal of Engineering Research in Computer Science and Engineering*, 2017.

[25] T. Baware, S. Chauhan, S. Naik, S. Patil, R. Thakur, and K. Vayadande, "A survey paper on CPU process scheduling," *River Publishers*.

[26] W. M. Lim, "What is quantitative research? An overview and guidelines," *Australasian Marketing Journal*, 2024. https://doi.org/10.1177/14413582241264622

[27] S. M. Miri and Z. D. Shahrokh, "A short introduction to comparative research," 2019.

[28] A. Peter, "CPU scheduling: arrival, burst, completion, turnaround, waiting, and response time," *Baeldung*, Sep. 4, 2024. https://www.baeldung.com/cs/cpu-scheduling

[29] GeeksforGeeks, "Difference between arrival time and burst time in CPU scheduling," Sep. 19, 2024. https://www.geeksforgeeks.org/difference-between-arrival-time-and-burst-time-in-cpu-scheduling/

[30] M. Algabri *et al.*, "Performance assessment of CPU scheduling algorithms," *Journal of Computer Science*, vol. 20, no. 9, 2024. https://doi.org/10.3844/jcssp.2024.972.985

[31] M. Leonelli, "What is simulation," Apr. 19, 2021. https://bookdown.org/manuele$_leonelli/SimBook/what-is-simulation.html$

[32] C. Yin and A. McKay, "Introduction to modeling and simulation techniques," 2018.

[33] N. Kulkarni, S. A. Mahajan, and A. Patil, "A study of available process scheduling algorithms and their improved substitutes," *International Research Journal of Engineering and Technology*, 2020.

[34] GeeksforGeeks, "Difference between turn around time (TAT) and waiting time (WT) in CPU scheduling," Sep. 30, 2024. https://www.geeksforgeeks.org/difference-between-turn-around-time-tat-and-waiting-time-wt-in-cpu-scheduling/

[35] GeeksforGeeks, "Data partitioning techniques in system design," Nov. 1, 2024. https://www.geeksforgeeks.org/data-partitioning-techniques/

International Journal on Optimization and Applications
IJOA. Vol. 06, Issue No. 01, Year 2026, www.ijoa-journal.com

Implementing DevSecOps: A Systematic Literature Review on Integrating Security into DevOps

Rjanna Miki M. Balaybay, Jenny Ann T. Guyong, Justine Mae B. Macario
College of Information Technology and Computer Science
University of the Cordilleras
Baguio City, Philippines
rmb2105@students.uc-bcf.edu.ph, jtg6745@students.uc-bcf.edu.ph, jbm1737@students.uc-bcf.edu.ph

***Abstract*—The migration from DevOps to DevSecOps aims to address the growing needs for security in software development, but the integration of security itself into the workflow introduces new challenges that organizations often struggle to overcome. This study aimed to explore DevSecOps to gain a comprehensive understanding on the impact of its implementation as a security-integrated approach to software development. Through qualitative research design and Systematic Literature Review (SLR), existing studies related to the objectives were gathered and analyzed. In hand with this, using Thematic Analysis, the data was examined and patterns were extracted related to common barriers to successful DevSecOps adoption, its advantages, and strategies to mitigate the identified challenges. The findings revealed twelve recurring barriers across key aspects of software development, which are centered around people, processes, and technology. On the other hand, eight advantages were outlined, highlighting the ability of DevSecOps to reduce vulnerabilities and risks and improve the robustness of the system. Finally, five general strategies were found, emphasizing the role of automation in every strategy to improve security integration. These results highlight that the implementation of DevSecOps brings both advantages and disadvantages that must be considered. Therefore, its impact is dependent on how organizations adopt, implement, and utilize strategies to ensure a seamless and successful transition to DevSecOps.**

***Index Terms*—Agile software development, software development methods, software development process management, systems security, security and privacy.**

I. Introduction

Nowadays, modernization has pushed organizations to adopt advanced technologies in order to stay competitive and meet increasing demands for efficiency and innovation. This shift has transformed how businesses operate, with a growing reliance on digital tools and automated processes to streamline workflows and deliver products and services effectively. As software becomes an essential component of business operations, organizations must adopt development methodologies that enable them to build, deploy, and maintain systems efficiently. One such solution is Agile methodologies, which promote a flexible and iterative approach to software development, allowing faster software releases [1]. While Agile methodologies enhance development speed and adaptability, they fail to fully address the challenges of managing development and operations [2].

This limitation led to the emergence of DevOps, a critical software development technique that bridges the gap between development (Dev) and operations (Ops) teams to enhance collaboration, automate processes, and accelerate software delivery. Research demonstrates that DevOps adoption significantly improves productivity, reduces time-to-market, and enables organizations to respond rapidly to changing market needs [3]. For this reason, DevOps has become a cornerstone of modern agile software development [4]. However, a major drawback of DevOps is its tendency to prioritize speed over security until the later stages of the development lifecycle. Studies have identified misconfigurations, insecure code, and exposure to cyber threats as common risks in DevOps environments, often resulting from delayed security integration [4].

To address this issue, DevSecOps was introduced as a paradigm that integrates security practices into every phase of the DevOps lifecycle [5]. By embedding security into planning, development, testing, and deployment processes, DevSecOps aims to overcome the security limitations of DevOps while maintaining agility and efficiency. Nevertheless, integrating security practices into DevOps introduces certain trade-offs that may significantly alter development workflows. Unlike DevOps, which primarily emphasizes speed and delivery, DevSecOps incorporates tasks such as security assessments and compliance verification that may slow down development if not implemented effectively. This shift in workflow

may impact several key aspects of software development, including OPC (organization, people, culture), process capabilities, technology, and business strategy [6], [7].

Although DevSecOps provides a potential solution to the limitations of DevOps, it also introduces complexities that organizations may find difficult to manage. Many organizations struggle to successfully implement security practices while maintaining efficiency during the transition [7], [8]. By identifying these complexities, organizations can develop informed strategies that balance security with agility and ensure the development of secure and robust software systems.

For these reasons, this research aims to conduct a *Systematic Literature Review (SLR)* on DevSecOps to obtain a comprehensive understanding of the impact of implementing DevSecOps as a security-integrated approach to software development. Furthermore, the study seeks to identify both the advantages and disadvantages of DevSecOps in order to provide insights into more effective implementation strategies.

Specifically, the study addresses the following research questions:

1) What are the common barriers that prevent organizations from achieving successful DevSecOps implementation?
2) What are the key advantages that DevSecOps offers to organizations?
3) What strategies can be employed to address the challenges of successfully implementing DevSecOps?

II. Methodology

This study employed a qualitative research design using a *Systematic Literature Review (SLR)* as the primary research method. This systematic approach allowed for an extensive examination of existing literature to obtain valuable insights on DevSecOps and develop a comprehensive understanding of the topic in relation to the research questions.

The primary databases used to identify relevant literature included ACM Digital Library, IEEE Xplore, Springer, and ScienceDirect. These databases were selected due to their extensive collections of studies related to DevOps and DevSecOps.

To ensure the relevance and quality of the selected studies, specific inclusion criteria were established. The reviewed papers had to focus on DevSecOps or security practices within DevOps environments. Additionally, publications were limited to journal articles or conference proceedings published between 2019 and 2025. To ensure credibility, studies from predatory journals were excluded, and only publications written in English and available in full text were included in the review.

The main keywords used for the literature search included "DevSecOps" and "Secure DevOps" across all objectives. Additional keywords such as "challenges" OR "barriers," "advantages" OR "benefits," and "strategies" OR "solutions" were applied specifically to address the first, second, and third research objectives respectively.

After applying the filtering process, a total of 68 studies were selected for analysis. The researchers followed a structured process to organize and review the collected data thoroughly. Using the *Thematic Analysis Method*, the selected studies were systematically examined by identifying and labeling relevant features aligned with the research objectives. These features were then coded, analyzed for potential thematic patterns, and grouped into broader themes.

The application of thematic analysis provided a structured and systematic approach for examining the literature, enabling an in-depth exploration of the research domain. This process allowed the study to effectively evaluate the impact of integrating security into software development practices and address the research questions.

III. Findings and Discussions

A. Common Barriers to DevSecOps Implementation

To identify the common barriers that prevent organizations from successfully implementing DevSecOps, the researchers conducted a literature search using the keywords "DevSecOps," "Secure DevOps," "challenges," and "barriers" across several academic databases. The initial search yielded 875 results. After screening the studies for relevance to the research objective and applying the defined inclusion criteria, the number of selected papers was reduced to 27.

To ensure the reliability of the findings, only themes that appeared across multiple studies were considered. This approach ensured that the identified challenges represent widely recognized barriers in DevSecOps adoption.

The recurring themes identified from the analyzed studies are summarized in Table I.

TABLE I
THEMES OF THE COMMON BARRIERS TO DEVSECOPS IMPLEMENTATION

Themes	**Description**	**Citations**
Balancing Speed and Security	The extensive security measures add up to additional time that causes delays in deployment cycles.	[7] [9] [10] [11] [12] [13]
Resistance to Change	The migration to DevSecOps faces resistance because of the cultural shift required, where security becomes a responsibility shared by development and operations teams. Developers and operators who are accustomed to traditional methods may become reluctant to adopt the changes.	[8] [10] [14] [15] [16] [17] [19] [20] [21] [22]
Knowledge and Skills Gap	Many engineers are outdated or lack the necessary expertise to integrate security measures effectively into the development pipeline.	[7] [8] [10] [15] [16] [18] [21] [22] [23] [24]
Security Tools	The complexity of security tools combined with insufficient training of the team often results in tool selection and integration challenges.	[7] [10] [15] [16]
Complexity of Automation	The automation of security processes introduces an additional layer of complexity to workflows, especially when integrating advanced technologies such as AI/ML models that require specialized expertise.	[10] [21] [25]
Integration to Complex Infrastructure	Integrating security into complex infrastructures may create technical integration challenges and increase financial costs that demotivate organizations.	[7] [15] [26]
Team Collaboration	Developers and security teams often have conflicting priorities along with unclear communication, which can lead to instability in security implementations.	[7] [8] [14] [15] [17] [27] [28] [29]
Unrestricted Collaborations	Excessive collaboration can sometimes increase security risks because more people have access to sensitive information.	[30] [31]
Compliance Issues	Ensuring security measures while adhering to compliance requirements and regulations makes integration of security practices difficult, particularly when policies are inconsistent or poorly defined.	[8] [15]
Resource Constraints	Migration to DevSecOps requires significant financial and human resources, along with system modifications that may cause financial strain.	[7] [17] [22] [26]
More Complex Development Process	Multiple security measures introduced during development increase process complexity.	[13] [23]
Lack of Planning	Many organizations implement DevSecOps without a well-defined strategy and standardized methodology, which increases the risk of disruptions and inefficiencies in development.	[32] [33]

The findings reveal several themes that hinder successful DevSecOps adoption. Resistance to Change emerges as a primary barrier as integrating security into the workflow requires a significant cultural shift that teams accustomed to traditional methods tend to resist [8], [10], [14], [17]. This problem is further reinforced by the steep learning curve teams struggle to overcome, as found in the theme Knowledge and Skills Gap. They are found to lack expertise in security practices [15], secure coding standards [8], or the operation of advanced security tools [7]. This makes Balancing Security and Speed a major challenge, especially when additional security measures must be taken into account. As emphasized by Desai and Nisha [2021], DevOps emphasizes agility and delivery, but the integration of security into these practices introduces delays. The excessive security scans and additional automated processes to ensure security compromise deployment speed [12], [13]. Along with the integration of Sec into DevOps comes the Team Collaboration issues, often resulting from conflicting priorities of the security and DevOps teams [17], [29]. While the promotion of collaboration is indeed crucial for team management, Unrestricted Collaboration may rather expose sensitive information and increase security risks when teams collaborate too closely [30], [31].

Moreover, with an overwhelming number of options and a lack of expertise to make informed choices, organizations often struggle with the selection and integration of Security Tools. As addressed by several authors [7], [15], [16], selecting and configuring the right tools can get complex, especially for teams that lack training. Similarly, the Complexity of Automation can become technically challenging, especially in environments handling large volumes of data [10]. For organizations operating in cloud-native or hybrid infrastructures, Integration into Complex Infrastructure introduces both technical and financial challenges. This contributes to Resource Constraints, which are particularly evident for smaller organizations with limited budgets and personnel [11], [17].

Organizations also face Compliance Issues, as considering the adherence to security standards and regulatory requirements, while in the process of a cultural shift, can restrict the integration of security [15]. Additionally, the multiple security measures needed for the integration increase the effort required for security validation, leading to a More Complex Development Process [13], [23]. Finally, many implementations fail due to Lack of Planning, where pipelines that are built rapidly without proper engineering can lead to disruptions and inefficiencies that

may happen during the development process [32], [33]. These findings reveal that the barriers to the successful transition to DevSecOps are not isolated but are deeply intertwined, which affects one another if not resolved. Ultimately, these persistent barriers identified suggest that while DevSecOps provides a strong framework for security integration, it requires organizations to consider strategic plans to overcome these barriers and ensure success in implementation.

B. Advantages of adopting DevSecOps

The table below (Table 2) provides an overview of the positive impacts associated with adopting DevSecOps. To address the study's second objective, which focuses on evaluating these benefits, only 27 papers from 662 results were analyzed. These papers were identified using keywords such as "Secure DevOps," "DevSecOps," "advantages," and "benefits." Table II outlines the key themes derived from the reviewed literature, along with their descriptions.

Despite the challenges associated with its adoption, research indicates that the implementation of DevSecOps also offers several advantages by integrating security throughout the development process. One of the most cited benefits is how it leads to More Robust Systems. By embedding security early and into every stage of development, organizations can detect vulnerabilities sooner [18], [49], reduce risks [11], [22], and build stronger infrastructures [24], [26]. Not only does it reduce the risks of breaches, but it also introduces automated self-healing mechanisms [24] that address any detected issues without manual intervention. The reduction of delays as a result of automation leads to Operational Efficiency. As noted by Qasim and Bilal [2024], the automation of security practices minimizes any repetitive tasks, which results in faster detection of security issues, reduced rework, and improved productivity [12], [41]. Consequently, DevSecOps supports Faster Software Delivery, with the reduction of deployment delays reaching 40% [12].

Another important advantage is Enhanced Collaboration due to having a shared responsibility. The breaking down of traditional silos enforces collaborative culture, which speeds up feedback cycles and enhances deployment speed [36], [39], [48]. Moreover, by detecting issues earlier, DevSecOps helps in Cost Efficiency. Since issues are detected sooner, fixing them during development is considerably more cost-effective than resolving them after. The automation also helps in continuously fixing security problems during the development process [43]. Beyond technical efficiency, the agile method also supports Compliance Efficiency due to security controls in the pipeline that help accelerate adherence to regulatory requirements [18], [22], [49]. Automating compliance checks and embedding them directly into the development pipeline not only simplifies how organizations meet security regulations and industry standards but also helps them avoid penalties, reduce risks, and maintain trust with customers without slowing down development. Subsequently, DevSecOps helps companies to strengthen their Reputation Management by demonstrating security commitment to stakeholders and creating an edge over competitors through faster and more secure releases. As noted by Mao et al. [2020], a strong emphasis on security earns organizations customer trust and satisfaction, which is an essential element in a competitive market. Combining evidence from these studies, it can be said that adopting DevSecOps transforms traditional development by aligning speed, security, resources, collaboration, and compliance, which all provide value for both technical and business improvement.

C. Strategies for Implementing DevSecOps

In identifying the strategies that can be employed to address the challenges of implementing DevSecOps, the researchers conducted a literature search using the keywords "DevSecOps," "Secure DevOps," "strategies," and "solutions" to look for relevant literature. From the initial set of 789 results retrieved from academic databases, 55 research papers were relevant and were subsequently analyzed in relation to the third objective. The recurring themes under the strategies are summarized in Table III.

TABLE II
THEMES OF THE ADVANTAGES OF ADOPTING DEVSECOPS

Themes	**Description**	**Citations**
More Robust Systems	Embedding security practices throughout the development cycle enables organizations to detect vulnerabilities earlier and mitigate risks effectively.	[11] [18] [22] [24] [26] [34] [35] [40] [42] [43] [44] [46] [48] [50]
Reputation Management	This protects organizational credibility by preventing breaches and adherence to regulatory standards, which not only avoids financial penalties but also builds the trust of stakeholders.	[10] [17] [21] [36]
Operational Efficiency	The adoption optimizes workflow through automation, which eliminates manual bottlenecks, reduces human error, and allows faster delivery cycles.	[12] [16] [38] [41] [42] [45]
Faster Software Delivery	Integrating security checks creates a balance between speed and security, ensuring rapid releases without compromising safety.	[12] [16] [22] [43]
Enhanced Collaboration	Through shared tools, cross-training, and cultural alignment, different teams foster a unified approach to security.	[36] [39] [48] [50]
Cost Efficiency	Expenses are reduced through proactive security measures and automation.	[18] [37] [41] [43] [47] [50]
Compliance Efficiency	Adhering to regulatory requirements is simplified by automation and real-time monitoring.	[18] [22] [49]
Competitive Advantage	This enables companies and organizations to deliver secure, high-quality software faster than traditional approaches, which drives customer trust and market differentiation.	[17] [21]

TABLE III
THEMES OF THE STRATEGIES FOR IMPLEMENTING DEVSECOPS

Themes	**Description**	**Citations**
Automation and Integration	Automation and integration enable early vulnerability detection, consistent security measure implementation, and streamlined processes. These include the integration of proper automation into pipelines to ensure continuous security testing and seamless workflows.	[5] [18] [19] [20] [22] [25] [30] [32] [34] [35] [37] [38] [46] [51] [53] [56] [57] [60]
Governance and Compliance	This theme focuses on strategies that emphasize compliance and rules as essential for aligning security practices with industry standards while reducing manual effort.	[9] [31] [34] [39]
Collaboration and Communication	Collaboration and Communication points to breaking down silos between development, operations, and security teams critical for effective DevSecOps adoption.	[14] [27] [29] [31] [33] [42] [45] [48] [50] [53] [59] [62]
Tool Selection and Optimization	This refers to selecting the right tools that are vital for streamlining security processes while maintaining agility. Strategies focus on leveraging open-source tools, ensuring compatibility with existing workflows, and adopting advanced technologies like AI/ML for real-time threat detection.	[5] [7] [19] [20] [21] [24] [26] [28] [43] [44] [47] [52] [56] [57] [58] [59] [60] [67]
Risk Management	This refers to integrating threat modeling and continuous risk assessment from the start (shift-left approach) and throughout the development lifecycle. Strategies include automating these processes to identify vulnerabilities early and adapting to evolving threats effectively.	[21] [30] [40] [47] [49] [51] [54] [55] [60] [69] [71]

To overcome challenges in the integration of security within DevSecOps, organizations can adopt several strategic solutions. As one of the most cited strategies in literature, the findings reveal that Automation and Integration is central to DevSecOps as it ensures security measures across the development process. The integration of real-time scanning tools and services helps maintain security beyond deployment, helping organizations shift from reactive to more proactive approaches. As emphasized across numerous studies [18, 20, 38, 66], automation not only enhances efficiency but also reduces human error, lowers financial costs, and speeds up regulatory compliance, which ultimately makes it an essential aspect of a successful implementation of DevSecOps. Under this theme are practices such as Security as Code (SaC) [37] and Infrastructure as Code (IaC) [63] that treat security configurations and infrastructure as programmable objects, making them easier to audit and maintain. Moreover, with the integration of automation within the pipelines comes continuous Governance and Compliance. The role of AI can be seen in the automation of compliance checks to verify adherence to standards while maintaining speed. Desai and Nisha [2021] emphasize codifying security policies to ensure that the environment is regulated and that suitable security controls are in place. These compliance standards should be regularly updated to align with the latest security frameworks. Collaboration and Commu-

nication among the teams are another critical strategy to the successful execution of DevSecOps practices. Li and Zalialetdzinau [2022] point out that in DevSecOps, security is no longer isolated to a single department but is now embedded across all teams. Essentially, a successful collaboration entails behavioral change [14], cross-training [54], and constant communication between teams [62]. Training programs and appointing "Security Champions" or security specialists are also highly suggested to embed security practices in the team [45, 54]. A crucial addition is the feedback loop, which places channels for continuous feedback that improve coordination and address security challenges at the same time [33]. When teams are properly trained and collaborate well, they are better equipped for selecting the appropriate tools. Under the theme of Tool Selection and Optimization, studies recommend that the tools should be automated [44, 47, 52], up-to-date [5], and suitable to the infrastructure [7, 26] to be effectively utilized in rapid development cycles. Using the proper tools should be taken into consideration to streamline the processes while maintaining agility. Many studies also highlight the integration of Artificial Intelligence (AI) into tools to make it more efficient, though it still has to overcome ethical and adoption challenges as noted by Pakalapti et al. [2023]. Fundamental to every other solution is Risk Management, which involves integrating security considerations throughout the development lifecycle. A widely cited strategy under this theme is the shift-left approach, where security is incorporated early in the development [30, 49, 54]. Complementing this is continuous risk assessment, which automatically and regularly updates security measures to evolving threats [40, 55, 60]. Techniques like threat modeling further strengthen risk management by identifying potential vulnerabilities during the planning phase to ensure a robust foundation [51, 54, 55, 69]. Given these points, AI is essentially a core strategy in DevSecOps to enable efficient and more secure processes. These findings highlight that the implementation of DevSecOps requires a holistic transformation as its benefits can only be realized when supported by cultural, procedural, and governance changes.

IV. CONCLUSION

The study aimed to conduct a literature review to gain a comprehensive understanding of the impact of DevSecOps as a security-integrated paradigm to software development. Based on the findings, it can be concluded that the effect of the adoption is neither inherently positive nor negative. Rather, its success largely depends on how it is adopted and sustained within an organization. Its effectiveness is deeply influenced by the practices employed, as proper implementation strategies are needed to leverage the positive effects of DevSecOps on software development. This study was able to examine the common barriers, advantages, and strategies of adopting DevSecOps, focusing on its ability to provide security throughout the DevOps lifecycle while making sure operations are still running efficiently. The findings revealed twelve barriers that are deeply interconnected rather than isolated. These related issues disrupt collaboration, slow down deployment and delivery, and increase the risk of insecure products. With barriers identified, such as Team Collaboration, Resistance to Change, and Security Tools, the results imply that DevSecOps is not just simply a methodical process, but rather a more complicated approach that entails holistic changes in people, processes, and technology. Despite the multifaceted challenges, the study highlights eight advantages organizations experience during the adoption. With a more secure posture, the integration of early and continuous security enables faster detection of vulnerabilities, fewer issues to mitigate post-release, and overall, more robust systems. These results demonstrate that when integrated properly, it creates synergy between speed, security, collaboration, and business value. For a more effective implementation, the study identifies five general themes for strategies that can be applied to overcome the identified barriers. Results proved that the integration of automation is highly suggested in enhancing processes and reducing vulnerabilities. Moreover, cooperation and collaboration between the teams were critical factors in successfully adopting the methodology. When done right, DevSecOps not only mitigates risks but also improves compliance and lessens resource constraints. Ultimately, the impact of DevSecOps lies not in the methodology alone, but in how it is applied and how the organization utilizes available strategies to enable secure and agile development even in fast-paced environments.

V. RECOMMENDATION

The following suggestions from the methodological and objective perspective can be considered to improve the paper. From a methodological standpoint, expanding the database searches beyond the resources used can contribute to a more comprehensive literature review. Moreover, using gray literature or industry white papers, code documentation, and other technical reports can provide more insights into information overlooked in academic writing. Additionally, real-world validation by actual DevSecOps professionals through a mixed-methods approach can enhance the depth of the study. There should also be a focus on the long-term impact of implementing DevSecOps to review its efficacy. The

objectives standpoint can focus on the following topics: sector specificity and emerging technologies. The current study can be enhanced by focusing on specific industries and tailoring the DevSecOps methodology for each field. Furthermore, given the role of automation in DevSecOps as found in the results, concentrating on emerging technology, such as cloud-native architectures, quantum computing, and AI/ML in DevSecOps practices, can be valuable for future topics. To better facilitate the implementation of the DevSecOps methodology and to open opportunities for future research, the previously discussed recommendations are provided.

a) ACKNOWLEDGMENTS: We thank Mr. Efraim Jededdia Z. Pangan and Artificial Intelligence Chatbots for their support in completing this paper. With the lectures, material, and feedback provided by Mr. Pangan, we have successfully navigated the complexities of the research and refined the arguments presented within the paper. Multiple AI chatbots, namely ChatGPT, Gemini, DeepSeek, Perplexity, and Claude, have been used throughout this process. Leveraging their expansive data pools and file upload capabilities, their assistance and aid proved crucial for completing this work. Specifically, these chatbots facilitated a deeper understanding of the DevSecOps methodology and the Systematic Literature Review procedure, ultimately contributing to the rigor and comprehensiveness of our findings.

REFERENCES

[1] M. Doshi and P. Virparia, "Quality, speed, and collaboration in Agile vs. traditional models," *J. Electrical Systems*, vol. 20, no. 11, 2024.

[2] S. M. Masud, M. Masnun, A. Sultana, A. Sultana, F. Ahmed, and N. Begum, "DevOps enabled Agile: Combining Agile and DevOps methodologies for software development," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 11, Jan. 2022. http://dx.doi.org/10.14569/ijacsa.2022.0131131

[3] A. Wiedemann, M. Wiesche, and H. Krcmar, "DevOps in practice: A systematic literature review," in *Proceedings of the International Conference on Software Engineering (ICSE)*, 2020. https://doi.org/10.1002/spe.3096

[4] T. Leppänen, A. Honkaranta, and A. Costin, "Trends for the DevOps Security: A Systematic Literature Review," in *Proceedings of the 12th International Symposium on Business Modeling and Software Design (BMSD 2022)*, ACM, New York, NY, USA, 2022.

[5] C. Feio, N. Santos, N. Escravana, and B. Pacheco, "An empirical study of DevSecOps focused on continuous security testing," in *2024 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, Vienna, Austria, Jul. 8–12, 2024, pp. 610–617. http://dx.doi.org/10.1109/eurospw61312.2024.00074

[6] X. Zhao, T. Clear, and R. Lal, "Identifying the primary dimensions of DevSecOps: A multi-vocal literature review," *Journal of Systems and Software*, vol. 214, p. 112063, Aug. 2024. http://dx.doi.org/10.1016/j.jss.2024.112063

[7] R. N. Rajapakse, M. Zahedi, M. A. Babar, and H. Shen, "Challenges and solutions when adopting DevSecOps: A systematic review," *Information and Software Technology*, vol. 141, no. 6, p. 106700, Jan. 2022. http://dx.doi.org/10.1016/j.infsof.2021.106700

[8] M. A. Akbar, K. Smolander, S. Mahmood, and A. Alsanad, "Toward successful DevSecOps in software development organizations: A decision-making framework," *Information and Software Technology*, vol. 147, p. 106894, Jul. 2022. http://dx.doi.org/10.1016/j.infsof.2022.106894

[9] R. Desai and N. T. N., "Best practices for ensuring security in DevOps: A case study approach," *Journal of Physics Conference Series*, no. 4, Jul. 2021. http://dx.doi.org/10.1088/1742-6596/1964/4/042045

[10] A. Mustyala, "DevSecOps: Integrating security into the DevOps lifecycle," *ISAR Journal of Multidisciplinary Research and Studies*, vol. 1, no. 5, Nov. 2023.

[11] P. S. S. Patchamatla, "Security in DevOps: A DevSecOps Approach to Mitigating Software Vulnerabilities," *Recent Innovations in Wireless Network Security*, vol. 7, no. 2, pp. 14–16, Feb. 2025. https://doi.org/10.5281/zenodo.14921426

[12] V. Veeramachaneni, "A Systematic Review of DevSecOps: Bridging Security and Agile Development for Resilient Software Systems," *NeuroQuantology*, vol. 21, no. 7, pp. 1251–1255, Dec. 2023.

[13] A. Caniglia, V. Dentamaro, S. Galantucci, and D. Impedovo, "FOBICS: Assessing Project Security Level through a metrics framework that evaluates DevSecOps performance," *Information and Software Technology*, vol. 178, p. 107605, Feb. 2025. http://dx.doi.org/10.1016/j.infsof.2024.107605

[14] M. Sánchez-Gordón and R. Colomo-Palacios, "Security as Culture: A Systematic Literature Review of DevSecOps," in *Proceedings of the IEEE/ACM 42nd International Conference on Software Engineering Workshops (ICSEW'20)*, ACM, New York, NY, USA, 2020, pp. 266–269. https://doi.org/10.1145/3387940.3392233

[15] S. Afaneh, M. R. Al-Mousa, H. S. Al-hamid, B. S. AL-Awasa, M. Alia, H. Almimi, and A. A. Alkhatib, "Security Challenges Review in Agile and DevOps practices," in *2023 International Conference on Information Technology (ICIT)*, Aug. 2023. http://dx.doi.org/10.1109/icit58056.2023.10226018

[16] N. Qasim and M. Bilal, "DevSecOps: Integrating Security into IT Development and Operations," *Revista de Inteligencia Artificial en Medicina*, vol. 15, no. 1, Oct. 2024.

[17] X. Zhou, R. Mao, H. Zhang, Q. Dai, H. Huang, H. Shen, J. Li, and G. Rong, "Revisit security in the era of DevOps: An evidence-based inquiry into DevSecOps Industry," *IET Software*, vol. 17, no. 4, pp. 435–454, Jul. 2023. http://dx.doi.org/10.1049/sfw2.12132

[18] G. Bollieddula, "Challenges and Solutions in the Implementation of DevOps Tools & Security (DevSecOps): A Systematic Review," *Culminating Projects in Information Assurance*, no. 127, 2022.

[19] K. Zalialetdzinau, "Secure Change Management Process: On the Effectiveness of DevSecOps," *Journal of Computer Science and Information Technology*, vol. 10, no. 4, pp. 37–51, Dec. 2022. http://dx.doi.org/10.13189/csit.2022.100401

[20] Z. Ahmed and S. C. Francis, "Integrating security with DevSecOps: Techniques and challenges," in *2019 International Conference on Digitization (ICD)*, Sharjah, United Arab Emirates, Nov. 18–19, 2019, pp. 178–182. http://dx.doi.org/10.1109/icd47981.2019.9105789

[21] X. Ramaj, M. Sánchez-Gordón, V. Gkioulos, S. Chockalingam, and R. Colomo-Palacios, "Holding on to compliance while adopting DevSecOps: An SLR," *Electronics*, vol. 11, no. 22, p. 3707, Nov. 2022. https://doi.org/10.3390/electronics11223707

[22] R. Ticu-Jianu, "Continuous Resilience: DevSecOps Strategies for Cloud and Quantum Platforms," *Informatica Economică*, vol. 28, no. 4, pp. 63–73, Dec. 2024.

[23] V. Pendyala, "Evolution of integration, build, test, and Release Engineering into DevOps and to DevSecOps," in *Tools and Techniques for Software Development in Large Organizations: Emerging Research and Opportunities*, Jan. 2020, pp. 1–20. http://dx.doi.org/10.4018/978-1-7998-1863-2.ch001

[24] J. Alonso, R. Piliszek, and M. Cankar, "Embracing IaC through the DevSecOps philosophy: Concepts, challenges, and a reference framework," *IEEE Software*, vol. 40, no. 1, pp. 56–62, Jan. 2023. http://dx.doi.org/10.1109/ms.2022.3212194

[25] N. Pakalapati, S. Venkatasubbu, and S. M. Sistla, "The convergence of AI/ML and devsecops: Revolutionizing software development," *Journal of Knowledge Learning and Science Technology ISS*, vol. 2, no. 2, pp. 189–212, Aug. 2023. http://dx.doi.org/10.60087/jklst.vol2.n2.p212

[26] S. Nagasundari, P. Manja, P. Mathur, and P. B. Honnavalli, "Extensive review of threat models for DevSecOps," *IEEE Access*, vol. 13, pp. 45252–45271, Mar. 2025. http://dx.doi.org/10.1109/access.2025.3547932

[27] D. Ashenden and G. Ollis, "Putting the SEC in DevSecOps: Using social practice theory to improve secure software development," *New Security Paradigms Workshop 2020*, pp. 34–44, Oct. 2020. http://dx.doi.org/10.1145/3442167.3442178

[28] R. N. Rajapakse, M. Zahedi, and M. A. Babar, "Collaborative Application Security Testing for DevSecOps: An empirical analysis of challenges, best practices and tool support," *arXiv.org*, Nov. 22, 2022. https://doi.org/10.48550/arXiv.2211.06953

[29] M. A. Akbar and A. A. AlSanad, "Empirical investigation of key enablers for secure DevOps practices," *IEEE Access*, vol. 13, pp. 43698–43715, 2025. http://dx.doi.org/10.1109/access.2025.3549183

[30] D. Anjaria and M. Kulkarni, "Effective DevSecOps Implementation: A Systematic Literature Review," *Cardiometry*, vol. 11, no. 4, pp. 4931–4945, Nov. 2022. https://doi.org/10.18137/cardiometry.2022.24.410417

[31] S. Rafi, W. Yu, and M. A. Akbar, "Towards a Hypothetical Framework to Secure DevOps Adoption: Grounded Theory Approach," in *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering (EASE '20)*, ACM, New York, NY, USA, 2020, pp. 457–462. https://doi.org/10.1145/3383219.3383285

[32] T. Scanlon and J. Morales, "Revelations from an Agile and DevSecOps Transformation in a Large Organization: An Experiential Case Study," in *Proceedings of the International Conference on Software and System Processes and International Conference on Global Software Engineering (ICSSP '22)*, ACM, New York, NY, USA, 2022, pp. 77–81. https://doi.org/10.1145/3529320.3529329

[33] J. Díaz, J. E. Pérez, M. A. Lopez-Peña, G. A. Mena, and A. Yagüe, "Self-service cybersecurity monitoring as enabler for DevSecOps," *IEEE Access*, vol. 7, pp. 100283–100295, Jul. 2019. https://doi.org/10.1109/ACCESS.2019.2930000

[34] L. Prates and R. Pereira, "DevSecOps practices and tools," *International Journal of Information Security*, vol. 24, no. 11, Nov. 2024. https://doi.org/10.1007/s10207-024-00914-z

[35] A. K. Sandu, "DevSecOps: Integrating Security into the DevOps Lifecycle for Enhanced Resilience," *Technology and Management Review*, vol. 6, no. 1, pp. 1–19, Feb. 2021.

[36] O. O. Abiona, O. J. Oladapo, O. T. Modupe, O. C. Oyeniran, A. O. Adewusi, and A. M. Komolafe, "The emergence and importance of DevSecOps: Integrating and reviewing security practices within the DevOps pipeline," *World Journal of Advanced Engineering Technology and Sciences*, vol. 11, no. 2, pp. 127–133, Mar. 2024. http://dx.doi.org/10.30574/wjaets.2024.11.2.0093

[37] S. Chittala, "Securing DevOps pipelines: Automating security in DevSecOps frameworks," *Journal of Recent Trends in Computer Science and Engineering*, vol. 12, no. 5, pp. 31–44, Nov. 2024. http://dx.doi.org/10.70589/jrtcse.2024.5.5

[38] K. Mittal, M. Sharma, M. Gupta, and K. Sheoran, "DevSecOps: A boon to the IT industry," *SSRN Electronic Journal*, Apr. 2021. http://dx.doi.org/10.2139/ssrn.3834132

[39] K. Boisrond, P. M. Tardif, and F. Jaafar, "Ensuring the integrity, confidentiality, and availability of IOT data in industry 5.0: A systematic mapping study," *IEEE Access*, vol. 12, pp. 107017–107045, Jul. 2024. http://dx.doi.org/10.1109/access.2024.3434618

[40] L. Prates, J. Faustino, M. Silva, and R. Pereira, "DevSecOps metrics," in *Lecture Notes in Business Information Processing*, Aug. 2019, pp. 77–90. http://dx.doi.org/10.1007/978-3-030-29608-7$_7$

[41] T. Chen and H. Suo, "Design and Practice of Security Architecture via DevSecOps Technology," in *2022 IEEE 13th International Conference on Software Engineering and Service Science (ICSESS)*, Beijing, China, Oct. 21–23, 2022, pp. 310–313. https://doi.org/10.1109/ICSESS54813.2022.9930212

[42] Dapshima, B. A., and S. K. Ahmad, "Evaluation and assessment of software security risks and vulnerabilities within the realm of secure DevOps," *International Journal for Multidisciplinary Research*, vol. 6, no. 4, Jul. 2024. https://doi.org/10.36948/ijfmr.2024.v06i04.25026

[43] G. Sanders, T. Morrow, N. Richmond, and C. Woody, "Integrating Zero Trust and DevSecOps," *Software Engineering Institute*, Feb. 2022. https://doi.org/10.1184/R1/19193099.v1

[44] E. Sermpezis, D. Karapiperis, and C. Tjortjis, "Integration of security in the DevOps methodology," in *2024 15th International Conference on Information, Intelligence, Systems & Applications (IISA)*, Chania Crete, Greece, Jul. 17–19, 2024, pp. 1–6. http://dx.doi.org/10.1109/iisa62523.2024.10786669

[45] M. Chen, B. Liang, and X. Lu, "The practice and application of a novel DevSecOps platform on Security," in *2024 5th International Seminar on Artificial Intelligence, Networking and Information Technology (AINIT)*, Nanjing, China, Mar. 29–31, 2024, pp. 558–562. http://dx.doi.org/10.1109/ainit61980.2024.10581700

[46] A. Bahaa, A. Abdelaziz, A. Sayed, L. Elfangary, and H. Fahmy, "Monitoring real time security attacks for IOT systems using DevSecOps: A systematic literature review," *Information*, vol. 12, no. 4, p. 154, Apr. 2021. http://dx.doi.org/10.3390/info12040154

[47] M. Bedoya, S. Palacios, D. Díaz-López, E. Laverde, and P. Nespoli, "Enhancing DevSecOps practice with large language models and security chaos engineering," *International Journal of Information Security*, vol. 23, no. 6, pp. 3765–3788, Oct. 2024. https://doi.org/10.1007/s10207-024-00909-w

[48] M. Zaydi and N. Bouchaib, "DevSecOps practices for an agile and secure IT service management," *Journal of Management Information and Decision Sciences*, vol. 23, no. 2, pp. 134–149, Jun. 2020.

[49] K. K. Voruganti, "Implementing security by design practice with DevSecOps Shift Left Approach," *Journal of Technological Innovations*, vol. 2, no. 1, Feb. 2021.

[50] T. Li and K. Zalialetdzinau, "Attemps of scientific reflection on the role of e-learning of the future in the area of digital transformation: Nw opportunities and experiences with DevSecOps," *Futurity Research Publishing*, vol. 2, no. 4, pp. 52–63, Dec. 2022. http://dx.doi.org/10.57125/fed.2022.25.12.06

[51] J. de Kock and J. Ophoff, "Critical success factors for integrating security into DevOps environment," in *Proceedings of the 15th Dewald Roode Workshop on Information Research*. https://ifip.byu.edu/00000188-e1b8-d3db-afbf-e3bd83ff0000/drw-2023-paper-17

[52] M. Fu, J. Pasuksmit, and C. Tantithamthavorn, "AI for DevSecOps: A landscape and future opportunities," *ACM Transactions on Software Engineering and Methodology*, Jan. 2025. http://dx.doi.org/10.1145/3712190

[53] S. Tatineni, "Compliance and audit challenges in DevOps: A security perspective," *International Research Journal of Modernization in Engineering Technology and Science*, vol. 5, no. 10, Oct. 2023. http://dx.doi.org/10.56726/irjmets45309

[54] R. Mao, H. Zhang, Q. Dai, H. Huang, G. Rong, and H. Shen, "Preliminary findings about DevSecOps from Grey Literature," in *2020 IEEE 20th Conference on Software Quality, Reliability and Security (QRS)*, Macau, China, Dec. 11–14, 2020, pp. 450–457.

[55] R. M. Czekster, "Continuous risk assessment in secure DevOps," *arXiv.org*, Sep. 2024. https://doi.org/10.48550/arXiv.2409.03405

[56] B. S. Akula, "Vulnerability management in DevSecOps," *IRJMETS*, vol. 6, no. 4, Apr. 2024.

[57] S. Yautsiukhin, G. Dupont, A. Ginis, G. Iadarola, S. Fagnano, F. Martineli, C. Ponsard, A. Legay, and P. Massonet, "Product incremental security risk assessment using DevSecOps practices," in *Lecture Notes in Computer Science*, Feb. 2023, pp. 666–685. http://dx.doi.org/10.1007/978-3-031-23641-7$_3$2

[58] N. A. Bernardino, B. Sequeira, E. Piza, F. Henriques, F. Neves, and C. I. Reis, "Enhancing DevSecOps: Three custom tools for continuous security," in *2024 IEEE 11th International Conference on Cyber Security and Cloud Computing (CSCloud)*, Shanghai, China, Jun. 28–30, 2024, pp. 53–58. https://doi.org/10.1109/CSCloud62866.2024.00017

[59] N. O. Omoike, "DevSecOps in AWS: Embedding security into the heart of DevOps practices," *International Journal of Science and Research Archive*, vol. 13, no. 2, pp. 1309–1313, Nov. 2024. https://doi.org/10.30574/ijsra.2024.13.2.2306

[60] X. Ramaj, "A DevSecOps-enabled framework for risk management of critical infrastructures," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings (ICSE '22)*, ACM, New York, NY, USA, 2022, pp. 242–244. https://doi.org/10.1145/3510454.3517053

[61] S. T. Makani and S. Jangampeta, "DevOps security tools: Evaluating effectiveness in detecting and fixing security holes," *International Journal of DevOps*, vol. 1, no. 2, Jul. 2021. https://iaeme.com/Home/issue/IJDO?Volume=1Issue=2

[62] A. Parashar, A. Dwivedi, A. Kumar, and A. A. Khan, "DevSecOps: A case study on a sample implementation of devsecops," *International Journal of Engineering Applied Sciences and Technology*, vol. 5, no. 2, pp. 156–158, Jun. 2020. http://dx.doi.org/10.33564/ijeast.2020.V05i02.022

[63] H. Yasar and S. E. Teplov, "DevSecOps In Embedded Systems: An Empirical Study Of Past Literature," in *Proceedings of the 17th International Conference on Availability, Reliability and Security (ARES '22)*, ACM, New York, NY, USA, Art. 155, pp. 1–6, 2022. https://doi.org/10.1145/3538969.3544451

[64] H. Haverinen, T. Päivärinta, J. Vänskä, and H. Joutsijoki, "Information-centric adoption and use of standard compliant DevSecOps for operational technology: From experience to design principles," in *Lecture Notes in Business Information Processing*, Feb. 2024, pp. 400–415. http://dx.doi.org/10.1007/978-3-031-53227-6$_2$8

[65] N. Tomas, J. Li, and H. Huang, "An Empirical Study on Culture, Automation, Measurement, and Sharing of DevSecOps," in *2019 International Conference on Cyber Security and Protection of Digital Services (Cyber Security)*, Oxford, UK, Jun. 3–4, 2019, pp. 1–8. https://doi.org/10.1109/CyberSecPODS.2019.8884935

[66] B. Gajbhiye, A. Aggarwal, and S. Jain, "Automated Security Testing in DevOps environments using AI and ML," *International Journal for Research Publication and Seminar*, vol. 15, no. 2, pp. 259–271, Jun. 2024. http://dx.doi.org/10.36676/jrps.v15.i2.1472

[67] M. Cankar, N. Petrovic, J. P. Costa, A. Cernivec, J. Antic, T. Martincic, and D. Stepec, "Security in DevSecOps: Applying Tools and Machine Learning to Verification and Monitoring Steps," in *Companion of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23 Companion)*, ACM, New York, NY, USA, 2023, pp. 201–205. https://doi.org/10.1145/3578245.3584943

[68] A. Sadovykh and V. Ivanov, "Enhancing DevSecOps with continuous security requirements analysis and testing," *Computer Research and Modeling*, vol. 16, no. 7, pp. 1687–1702, Nov. 2024. https://doi.org/10.20537/2076-7633-2024-16-7-1687-1702

[69] J. A. Morales, T. P. Scanlon, A. Volkmann, J. Yankel, and H. Yasar, "Security impacts of sub-optimal DevSecOps implementations in a highly regulated environment," in *Proceedings of the 15th International Conference on Availability, Reliability and Security (ARES '20)*, ACM, New York, NY, USA, Art. 63, pp. 1–8, 2020. https://doi.org/10.1145/3407023.3409186

[70] M. A. Akbar, S. Rafi, S. Hyrynsalmi, and A. A. Khan, "Towards People Maturity for Secure Development and Operations: A vision," in *Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering (EASE '24)*, ACM, New York, NY, USA, 2024, pp. 528–533. https://doi.org/10.1145/3661167.3661238

[71] T. Okubo and H. Kaiya, "Efficient secure DevOps using process mining and Attack Defense Trees," *Procedia Computer Science*, vol. 207, pp. 446–455, Oct. 2022. https://doi.org/10.1016/j.procs.2022.09.079

International Journal on Optimization and Applications
IJOA. Vol. 06, Issue No. 01, Year 2026, www.ijoa-journal.com

Development of an IDS/IPS based on machine learning technique to detect suspicious behavior in a network

Salma Doukkar
ENSA - Ibn tofail University
Salma.Doukkar@uit.ac.ma

Sophia ALAMI KAMOURI
ENSA - Ibn tofail University
s.alami@um5r.ac.ma

***Abstract*—As the cybersecurity landscape evolves, the development of advanced and efficient malware detection systems has become increasingly crucial. This research addresses the persistent challenge of identifying malware and safeguarding digital assets against cyber threats. Traditional detection methods often struggle to adapt to the dynamic nature of malicious activities [1], highlighting the need for innovative approaches that enhance detection accuracy while considering current cybersecurity challenges.**

This study focuses specifically on detecting malware through network traffic analysis. We propose a machine-learning-based framework for malware detection, emphasizing the effectiveness of two prominent algorithms: Random Forest (RF) and Support Vector Machine (SVM).

Our comparative performance analysis reveals that the model utilizing the Random Forest algorithm exhibits superior results. Notably, both classifiers achieve a True Positive Rate (TPR) exceeding 97%, while maintaining a False Positive Rate (FPR) of less than 4%. This indicates a robust capability in accurately identifying malware without generating excessive false alarms.

The model developed in this research has significant potential to assist organizations and cybersecurity professionals in anticipating and mitigating malware threats. By integrating this model into daily network operations, it can facilitate proactive decision-making and enhance the overall security posture of organizations. Additionally, our findings contribute to the broader field of cybersecurity by offering insights into effective machine-learning strategies for malware detection.

***Index Terms*—Machine learning, Malware Detection, Intrusion Detection, Malware Analysis, Random Forest, SVM**

I. Introduction

In today's digital age, many electronic devices face significant risks from malware. The term "malware" refers to malicious software specifically designed to damage or disrupt a target system. Malware can infiltrate networks, infect computers and other smart devices, steal sensitive data, damage vital infrastructure, and more [2].

These programmes include malware such as ransomware, rootkits, worms, spyware, bots, and viruses. According to [3], IT services claims that in only one year, one billion emails were exposed, impacting one in five internet users, and resulting in data breaches that cost organisations, on average, $4.35 million in 2022. The first half of 2022, there were about 236.1 million ransomware assaults worldwide.

In 2021, the accounts of one in two internet users in America were compromised and we can also give the example of the National Social Security Fund (CNSS), on April 8, 2025 in Morocco, which was the victim of a major cyberattack that resulted in the leak of sensitive data. This attack targeted the CNSS's information systems, resulting in a data leak concerning millions of citizens.

Malware attacks are becoming more complicated over time, despite improvements in detection, proper family class classification, and continual evolution, malware continues to be a serious threat to the internet [4]. Malware has also increased the risk of sophisticated attacks, such as multi-stage attacks [5]–[8] and Distributed Denial of Service (DDoS) attacks [9], [10], which have been a serious threat in recent years.

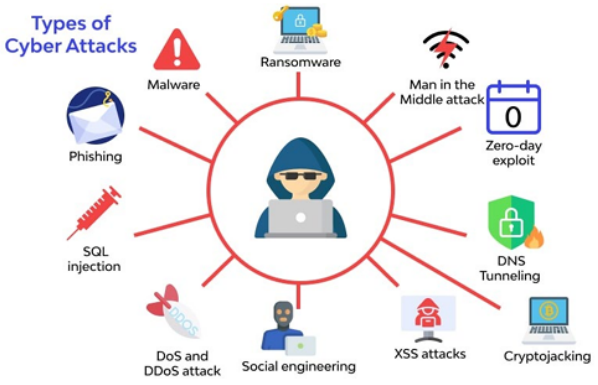

Fig. 1. Types of Cyber Attacks

To reduce the risk of cyberattacks, Intrusion Detection Systems (IDSs) are employed to monitor network traffic. The capability to detect viruses on a computer enables the development of malware prevention solutions, often incorporating unique signatures to identify infections. Malware can manifest in various forms, such as ransomware designed to extort money and spyware intended for surveillance.

Despite the advancement of several Machine Learning (ML) techniques for detecting anomalies in network traffic, human expertise remains crucial in creating many tools and methods. Specifically, handcrafted features play a vital role in traditional ML-based malware analysis methods. These features reflect what cybersecurity professionals consider the most important characteristics of malware. However, the feature engineering process can be labor-intensive, and the features created are often specific to particular tasks and subject to individual judgment.

Traditional ML methods have proven effective for malware detection, but they heavily rely on the knowledge and experience of security professionals to define the features characterizing malware. Research has shown a need for representations of malware that are less dependent on human expertise, addressing a significant yet unresolved challenge in the field.

This paper proposes a machine-learning-based approach for malware detection, with particular attention to the Random Forest (RF), Support Vector Machine (SVM).

The remainder of this paper is organised as follows. Section 2 presents the related to malware detection, section 3 describes the proposed methodology, section 4 shows the evaluation results, Section 5 presents a performance analysis of the utilised ML algorithms and Section 6 concludes the paper.

II. Related Work

With the rapid evolution of technology, malware threats and cybersecurity challenges are becoming increasingly difficult to counter. Malware, commonly referred to as a computer virus, is malicious software designed to infiltrate systems, damage files, and compromise user data. Some forms of malware, such as Trojans, disguise themselves as legitimate applications in order to gain unauthorized access and potentially take full control of a victim's computer [11].

The continuous emergence of malware variants and vulnerabilities in cloud environments raises the critical question of how assets can be effectively protected. To address this, various malware recognition techniques, particularly those leveraging machine learning, have been explored as promising solutions [12].

Machine learning offers significant advantages for classification tasks such as malware detection. By learning behavioral patterns and malicious activities, machine learning models can help mitigate the damage caused by malware. However, as highlighted in [13], detecting sophisticated and unpredictable malware remains challenging. Therefore, models must be trained and tested on diverse datasets containing a wide range of malware in order to improve detection accuracy, especially in high-risk scenarios involving destructive attacks. This demonstrates the need for rigorous and comprehensive evaluation.

Beyond malware detection, machine learning also provides useful applications in other domains, with several studies showcasing its versatility.

A. *Different Categories of Malware*

Different categories of malware are commonly discussed in the literature. For example, viruses are pieces of code that replicate until they corrupt files or system structures [14]. Worms spread automatically across networks to infect multiple machines. Trojans, disguised as harmless programs, trick users into downloading them before revealing their malicious intent.

Runtime malware may steal or encrypt data, often demanding ransom for recovery. Such attacks can be especially harmful to organizations managing sensitive customer information, as they can lead to data breaches, financial loss, and reputational damage.

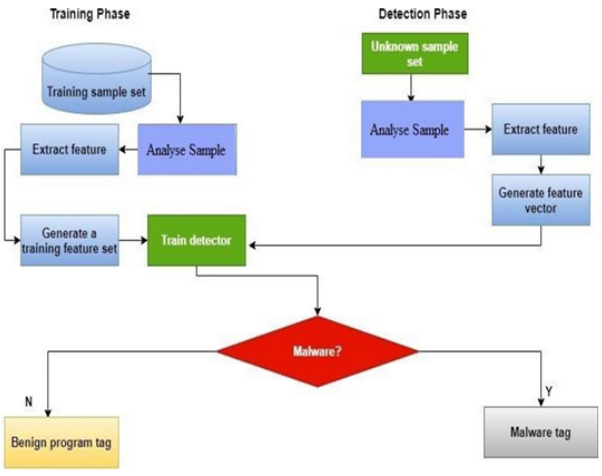

Fig. 2. Proposed ML malware detection method

Because malware is constantly evolving, rule-based and signaturebased detection methods are often insufficient. Signature-based detection identifies threats using

byte patterns stored in a database. While accurate for known threats, this approach fails against novel or polymorphic malware and requires frequent human updates.

Signature-less methods attempt to classify programs based on predefined suspicious behaviors (e.g., unusual assembly calls, unauthorized file permission changes, or abnormal file modifications). A threshold is then applied to determine whether the activity should be flagged as malicious. However, these approaches also require careful tuning and large datasets for effective training.

Anomaly detection techniques, capable of identifying new or unknown threats, offer another line of defense but suffer from high false-positive rates. Malware analysis is generally divided into two categories: static analysis, which inspects code without execution (allowing style recognition and code flow profiling), and dynamic analysis, which observes behavior during execution. Both approaches are widely used in research and industry.

Recent studies confirm that malware continues to grow in scale and sophistication, exploiting vulnerabilities in modern operating systems and even in browsers with built-in VPN services. These weaknesses are often leveraged by attackers, sometimes targeting inexperienced users, to launch severe cyberattacks that compromise system and network security.

III. Proposed Methodology

The proposed approach consists of several key stages:

A. Stage 1 – Environment Setup

The first step involves setting up the development environment.

B. Stage 2 – Log Generation with Anomalies

Synthetic log data containing both normal and anomalous events is generated. A custom Python script (`logs_generator.py`) is used to create a structured dataset, stored in CSV format, which serves as the foundation for the training and evaluation phases.

C. Stage 3 – Data Preprocessing

The generated logs are preprocessed to ensure quality and usability. This includes cleaning, normalization, feature extraction, and transformation into numerical vectors suitable for machine learning models.

D. Stage 4 – Model Training

Two supervised machine learning algorithms are employed: Random Forest (RF) and Support Vector Machine (SVM). Random Forest, an ensemble-based method, is chosen for its robustness and ability to handle high-dimensional data, while SVM is selected for its effectiveness in classifying anomalies in complex feature spaces. Both models are trained to distinguish between normal and abnormal log patterns.

E. Stage 5 – API Development

To enable real-time usability, an API is created to serve the trained models. This API allows external systems to send logs and receive classification results (normal or anomalous).

F. Stage 6 – Log Simulation

A simulation process is implemented to mimic the continuous flow of logs. The logs are sent through the API to test the model's ability to process and classify data streams in near real time.

G. Stage 7 – Deployment and Monitoring

Finally, the solution is deployed in a test environment with monitoring mechanisms in place to track model performance and adapt to changes in data patterns.

H. Evaluation Metrics

The performance of the models is assessed using standard metrics, including accuracy, precision, recall, and F1-score, to measure the detection quality of both algorithms.

IV. System Design and Workflow

A. Environment Setup

The development environment was prepared using Python and its scientific ecosystem. Key libraries such as NumPy and Pandas were used for data manipulation and numerical analysis, while Redis was integrated as an in-memory database for caching and session management. Celery handled asynchronous and background tasks, and Uvicorn served as the lightweight server for deploying Python applications. Additionally, other essential libraries, such as Matplotlib for data visualization and Scikit-learn for machine learning, were also utilized. This setup ensured scalability, efficiency, and smooth execution throughout the project.

B. Logs Generation

To create a realistic dataset for training and testing the anomaly detection models, a Python script was developed to generate synthetic logs containing both normal and anomalous events. Using the Faker library, the script produces random IP addresses, HTTP methods, status codes, API endpoints, and response times within a defined time range.

The process involves three main steps:

- **Timestamp creation:** Generating ordered timestamps across a specified period.

- **Anomaly interval definition:** Introducing intervals of abnormal behavior, such as client errors, server errors, and timeouts, by randomly selecting IP addresses and error types.
- **Dataset assembly:** Writing all generated events into a structured CSV file, including normal traffic and injected anomalies.

This automated generation ensures a diverse and controlled dataset that closely mimics real-world conditions, making it suitable for model training and evaluation.

Fig. 3. Logs generations

TABLE I
DESCRIPTION OF DATASET FEATURES

Column	Description
method	The HTTP request method used (e.g. GET, POST).
end_point	The targeted resource or URL (e.g. /login, /data).
response_time	The server's response time in milliseconds.
hour_of_day	The hour when the request was made (0–23).
day_of_week	The day of the week (0 = Monday, 6 = Sunday).
month	The month of the year (1–12).
is_weekend	Indicates whether the request occurred on a weekend (1 = yes, 0 = no).
index	The original log entry index in the dataset.
error_count	Number of errors (4xx or 5xx) for the IP address.
unique_error_types	Number of distinct error types encountered.
max_response_time	Average response time for the IP address.

C. Data Preprocessing

To prepare the log data for analysis and machine learning, a custom preprocessing pipeline was implemented. The process includes several key steps:

1) a. Feature Parsing and Transformation:

- Converted status_code to string format and extracted temporal features from the timestamp (hour, day of week, month).
- Added a binary indicator is_weekend to distinguish weekend traffic.
- Grouped logs by user_ip to calculate statistics such as error_count, unique_error_types, avg_response_time, and max_response_time.
- Created a categorical feature response_time_category (fast, normal, slow) based on response time.
- Added an is_potential_anomalous flag to mark potential anomalies (e.g., high latency or error codes).

2) b. Feature Engineering for Machine Learning:

- Selected numeric features (e.g., response times, error counts) to scale using StandardScaler.
- Selected categorical features (e.g., method, endpoint, weekend flag) to encode using OneHotEncoder.

3) c. Preprocessing Pipeline:

- Combined numeric and categorical transformations using ColumnTransformer.
- Ensured unknown categories during inference are handled gracefully.

4) d. Dataset Splitting:

- Implemented a function to split the dataset into training and testing subsets according to a configurable ratio.

This modular design ensures the dataset is properly cleaned, transformed, and ready for further anomaly detection and machine learning tasks.

D. Model Training

To detect abnormal behaviors in application logs, we implemented a machine learning-based anomaly detection module. Two unsupervised models were used:

1) Isolation Forest – an ensemble algorithm that isolates anomalies by randomly partitioning the data.
2) One-Class SVM (Support Vector Machine) – a boundary-based algorithm that identifies data points that deviate significantly from the normal profile.

The workflow is as follows:

- **Data preparation:** The preprocessed and engineered features obtained from the log dataset are split into training and testing sets.
- **Model training:** Both models are trained on the training data using the IsolationForest and OneClassSVM implementations from Scikit-learn.
- **Model persistence:** Trained models are saved to disk using joblib for later reuse without retraining.
- **Prediction and scoring:** For any new dataset, the system can predict whether each entry is normal or anomalous and compute anomaly scores for deeper analysis.

- **Evaluation:** The distribution of anomaly scores is visualized using histograms, and the ratio of detected outliers is calculated for each model.

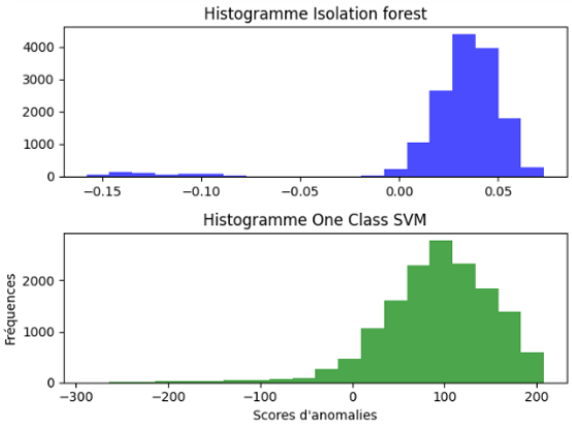

Fig. 4. Model Training

E. Results Interpretation

1) Top Histogram: Isolation Forest:

- **Description:** The histogram for the Isolation Forest algorithm is shown at the top.
- **X-axis:** Represents anomaly scores ranging from -0.15 to 0.05.
- **Y-axis:** Indicates the frequency of occurrences for each score.
- **Interpretation:** The majority of the data points are clustered near the 0.05 score, suggesting that the Isolation Forest algorithm has identified most of the data as normal. Scores close to zero indicate typical behavior, while lower scores may indicate detected anomalies.

2) Bottom Histogram: One-Class SVM:

- **Description:** The histogram for the One-Class SVM algorithm is displayed at the bottom.
- **X-axis:** Represents anomaly scores, with a wider range of values.
- **Y-axis:** Indicates the frequency of occurrences for each score.
- **Interpretation:** The One-Class SVM shows a different distribution of anomaly scores. The scores appear to be more spread out, indicating a wider variation in detected anomalies. This suggests that the One-Class SVM may have identified more outliers compared to the Isolation Forest, with some scores reflecting significant deviations from normal behavior.

F. API Development

In the API creation phase, I installed FastAPI and developed the `main.py` file to establish a robust framework for log anomaly detection. The FastAPI application is designed to handle various HTTP requests, providing endpoints for adding log entries, checking model statuses, and retrieving prediction results.

Utilizing Pydantic, I defined a LogEntry model to ensure data validation when logs are submitted. Redis was integrated as an in-memory database for efficient caching and session management, while Celery was employed to manage asynchronous tasks, enabling batch processing of log data.

Key functionalities include the ability to trigger anomaly detection when the log buffer exceeds a specified threshold, retrieve prediction results by their IDs, and monitor the status of processing tasks.

Fig. 5. Instalation of redis

Fig. 6. API Development

G. Log Simulation

A simulation process has been implemented to mimic the continuous flow of logs. The log simulator code reads entries from a CSV file containing log data and sends

each log to the previously designed API. This approach tests the model's ability to process and classify data streams in near real time.

Using the LogSimulator class, batches of logs are sent to the API, allowing for the evaluation of the anomaly detection system's effectiveness in a dynamic environment.

H. Deployment and Monitoring

Finally, the solution was deployed in a test environment with monitoring mechanisms in place to track model performance and adapt to changes in data patterns. The application generated status logs, indicating that the log processing and prediction tasks are functioning correctly.

Continuous monitoring ensures that the model remains effective and can be adjusted based on new trends detected in the data.

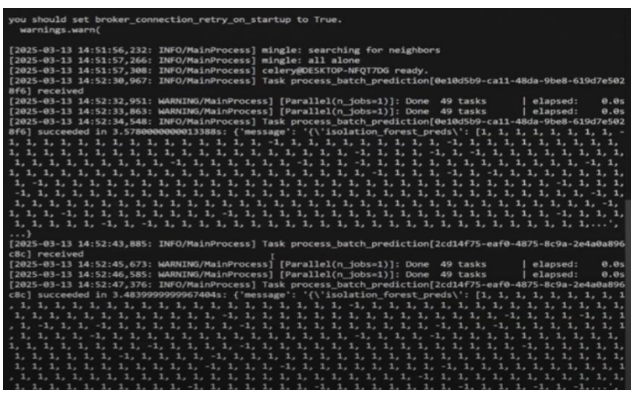

Fig. 7. Test environment

V. Conclusion

In conclusion, the development of a machine learning-based Intrusion Detection System (IDS) for malware detection represents a significant advancement in the ongoing battle against cyber threats. By leveraging powerful algorithms such as Random Forest and Support Vector Machine, this research not only enhances detection accuracy but also addresses the dynamic and evolving nature of malware.

The findings demonstrate that robust detection capabilities can be achieved, with high True Positive Rates and low False Positive Rates, ultimately enabling organizations to proactively safeguard their digital assets. As cyber threats continue to grow in complexity, integrating such intelligent detection systems into daily operations becomes imperative.

Future research should focus on refining these models and exploring innovative techniques that reduce reliance on human expertise in feature engineering, paving the way for more autonomous and adaptive cybersecurity solutions. This approach not only strengthens the resilience of networks but also contributes to the broader landscape of cybersecurity, fostering a more secure digital environment for all users.

References

[1] U.V. Nikam and V.M. Deshmukh, "Performance evaluation of machine learning classifiers in malware detection," in *Proceedings of the 2022 IEEE International Conference on Distributed Computing and Electrical Circuits and Electronics (ICDCECE)*, Ballari, India, 23–24 April 2022, pp. 1–5.

[2] K.V. Uma and E. Sharon Blessie, "Survey on Android malware detection and protection using data mining algorithms," in *2018 2nd International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, IEEE, 2018, pp. 209–212.

[3] AAG IT Services, "The latest ransomware statistics," Available: https://aag-it.com/the-latest-ransomware-statistics/, Feb. 7, 2024.

[4] S. S. M. Rahman, F. B. Rafiq, T. R. Toma, S. Hossain, and K. Beepan Badruzzaman, "Performance assessment of multiple machine learning classifiers for detecting phishing URLs," in *Proceedings of 3rd ICDECT-2K19*, Springer, 2020, pp. 285–296.

[5] F. J. Aparicio-Navarro *et al.*, "Addressing multi-stage attacks using expert knowledge and contextual information," in *2019 IFIP/IEEE Symposium on Integrated Network and Service Management Workshops (IM)*, IEEE, 2019.

[6] F. J. Aparicio-Navarro *et al.*, "Multi-stage attack detection using contextual information," in *MILCOM 2018 - IEEE Military Communications Conference*, IEEE, 2018, pp. 1–9.

[7] M. F. Khan *et al.*, "Multi-stage attack detection: Emerging challenges for wireless networks," in *2022 International Conference on Smart Applications, Communications and Networking (SmartNets)*, IEEE, 2022.

[8] M. F. Khan *et al.*, "Latent Dirichlet allocation for the detection of multi-stage attacks," in *2023 International Conference on Information Technology*, IEEE, 2023.

[9] D. M. El Asaad *et al.*, "Anomaly detection using dynamic time warping," in *2019 IEEE International Conference on Computational Science and Engineering (CSE)*, IEEE, 2019, pp. 193–198.

[10] D. M. El Asaad *et al.*, "Denial of service detection using dynamic time warping," *International Journal of Network Management*, vol. 31, no. 6, 2021.

[11] M. Hammad *et al.*, "A service-oriented approach for sensing in the Internet of Things," *IEEE Sensors Journal*, vol. 14, no. 11, 2020, pp. 15753–15761.

[12] M. Hammoud and R. Newman, "Information extraction from sensor networks using the Watershed transform algorithm," *Information Fusion*, vol. 22, 2015, pp. 309–319.

[13] S. Kumar *et al.*, "MCFT-CNN: Malware classification with fine-tune convolution neural networks using traditional and transfer learning in Internet of Things," *Future Generation Computer Systems*, vol. 125, 2021, pp. 334–351.

[14] S. M. Lima *et al.*, "Artificial intelligence-based antivirus in order to detect malware preventively," *Progress in Artificial Intelligence*, vol. 10, no. 1, 2021.

International Journal on Optimization and Applications
IJOA. Vol. 06, Issue No. 01, Year 2026, www.ijoa-journal.com

Using the Linear Programming Model to Improve Institutional Finance Decisions: An Applied Study on the Sherifian Office of Phosphates OCP (Morocco, 2023-2027)

Fatimaezzahra Amanzoui*, Rokaya Saoudi*, Wissal Maaroufi*, Fatimezzahra Dandane*, Souhaila Jorafi*, Asmaa Faris*, Elhachloufi†
* Faculty of Legal, Economic, and Social Sciences – Ain Sbaa,
University Hassan II Casablanca, Casablanca, Morocco
Master's student in Participatory Finance Engineering and Artificial Intelligence
fatimaezzahra.amanzoui-etu@etu.univh2c.ma, rokaya.saoudi-etu@etu.univh2c.ma, wissal.maaroufi1-etu@etu.univh2c.ma,
fatimezzahra.dandane-etu@etu.univh2c.ma, souhaila.jorafi-etu@etu.univh2c.ma, asmaa-faris-etu@etu.univh2c.ma
† Department of Statistics and Applied Mathematics for Economics and Management,
University Hassan II Casablanca, Casablanca, Morocco
elhachloufi@yahoo.fr

***Abstract*—Financing decision-making is one of the main challenges facing organizations in a complex and multi-alternative economic environment. In this context, this research ought to improve funding decisions by employing the linear programming model, in particular the transport model, supported by the simulation of linear weighting, in the distribution of financing resources within OCP to four strategic projects. The research aims to provide a quantitative model that supports decision-making in the absence of accurate data on demand, with a focus on achieving a balance Between the strategic priorities and financial constraints associated with the various sources of financing, represented in international bonds, strategic partnerships and international loans. The methodology relied on building a mathematical model that represents the funding relationships between projects and sources, and used linear weight simulation to determine the relative importance of each project, then the data was processed and analyzed using Python programming tools. The results showed that the proposed model contributed to providing an effective and objective funding distribution that reflects the relative importance of projects without exceeding the imposed restrictions, and also allowed flexibility in testing multiple scenarios and possible data updates. The added value of this model is highlighted in its ability to support decisions Finance in environments of uncertainty and a multitude of alternatives, providing a quantitative tool applicable at the level of strategic institutions. Thus, the research concludes that the integration of linear programming with linear weighting simulation constitutes a practical framework for improving the efficiency of resource allocation and enhancing the quality of financial decision in contemporary institutions. .**

***Index Terms*—linear programming, transfer problems, linear weighting simulation of weight distribution, institutional finance.**

I. Introduction

In light of the increasing economic challenges and complexities of the modern financial environment, improving institutional financing decisions has become critical to ensure the sustainability of institutions and enhance their competitiveness. Mathematical models have emerged as effective tools to achieve this goal, offering an accurate quantitative framework for analyzing financing alternatives and optimizing the allocation of financial resources. This study aims to explore how mathematical programming models, especially linear programming and transport models, can be used to improve financing decisions for organizations, focusing on practical applications in diverse contexts. OCP Group has launched the Green Investment Program for 2023-2027, which is part of its strategy to achieve carbon neutrality by 2040. This ambitious program aims to invest approximately 136578 million MAD in sustainable projects, including the development of renewable energies, water desalination, the production of green hydrogen and green ammonia, and the strengthening of industrial infrastructure associated with fertilizer production. This program is an advanced model for integrating environmental and economic objectives, where Adopting accurate quantitative analysis tools

is required to effectively channel funding towards changing and complex strategic priorities. This study seeks to provide an advanced mathematical model based on linear programming and transport models, supported by linear weighting simulation, to improve the distribution of funding among strategic projects based on relative importance, and in accordance with available funding constraints, ensuring the highest levels of efficiency and transparency in resource management. The importance of this study lies in its theoretical and practical contribution. In theory, it offers a new framework for integrating mathematical models into financial decision-making, broadening the scope of the existing literature. In practice, the proposed model provides a viable tool for organizations to improve the allocation of financial resources, reduce costs, and increase efficiency. The problem of this research is the challenges faced by institutions when making decisions related to the distribution of financing resources between multiple projects in light of funding constraints and disparities in strategic priorities, with the absence of accurate data on the size of the funding demand for each project. This challenge is increasing in large institutions that rely on different sources of funding, which imposes the need for a quantitative model that helps in scientific and flexible decision-making. Based on this, the problem is that: How can the transport model, supported by linear weighting simulations, be employed to improve the efficiency of funding distribution in a multi-project and resource-constrained environment? This research seeks to achieve a set of theoretical and applied objectives, most notably building a mathematical model using linear programming ”transport model” to distribute funding in a systematic manner between multiple projects with the employment of linear weighting simulation to estimate the relative weights of projects in the absence of accurate data on demand and provide a quantitative tool that supports financing decision-making within institutions, as well as applying the model to the case of the OCP process. To test its validity and interpret its results while proposing practical recommendations that contribute to improving the efficiency of the allocation of financial resources. This study is divided into several main sections: it begins with a theoretical framework that identifies basic concepts such as institutional finance and mathematical models in finance, followed by a presentation of research gaps in the previous literature. The proposed mathematical model and its application methodology are then presented, followed by analysis and discussion of the results. Finally, the study concludes with recommendations for future research and a summary of the most prominent methodology: The research is based on a quantitative analytical approach, based on modeling the decision to allocate funding using linear programming, in particular the transfer model, which allows the distribution of resources between multiple units under quantitative constraints. This model was supported by linear weighting simulation that enables the conversion of qualitative estimates (materiality) into mathematical processable numerical weights. The model was applied to real-world OCP data, and Python and Excel Solver tools were used.To perform calculations, simulations, and quantitative analysis of the results.

II. Study of the Literature

A. Previous Research

A study entitled: “The use of linear programming and goal programming models in the effectiveness of choosing the optimal production mix is a comparative study on engineering industry companies in the public and private sectors in Syria”, is one of the studies that aims to apply the models of linear programming and goal programming in the companies under study in order to choose the optimal production mix, and then compare the results of the application of the two models and the actual reality in public and private companies in the field of engineering industries, and conduct a sensitivity analysis of the results of solving these two models to reach alternatives and multiple options for the decision maker that help them choose the optimal production mix. The results can then be applied to companies that are similar in nature of work to the companies under study. In this study, (Al-Sheikh Hassan, 2024) relied on the descriptive approach in the theoretical framework of the research. As for the practical study, the researcher conducted a survey study by designing a survey list and distributing it to workers in the senior and middle management in the engineering industry companies under study, and then analyzing the answers using appropriate statistical methods and programs. The researcher also relied on the practical side by applying linear programming models and goal programming in the companies under study through building models, selecting them, analyzing their results, and conducting sensitivity analysis of the results of the models used. The study found that the use of linear programming and goal programming models in the companies under study helps to choose the optimal production mix effectively.

In the same vein, a study by (Aarab, 2020) entitled: “Improving the Performance of Economic Enterprises Using Numerical Linear Programming: A Case Study of the Metal Packing Corporation IMP Algeria”, aimed to show the importance of applying the numerical linear programming model on the metal packet institution, as well as understanding the productive activity of the

institution and the resources involved in the production process. The study then built and analyzed the numerical linear programming model of the institution. The study relied on the descriptive approach when presenting the theoretical side by collecting scientific material from secondary sources, while the practical side was represented by applying the numerical linear programming model in the metal packages institution and solving the model using the LINDO program. The study found that the proposed production program enabled the institution to achieve profits that exceeded its actual earnings, highlighting the importance of linear programming in enhancing institutional performance and supporting effective decision-making, whether to maximize profits or reduce costs.

In this context, a study titled "The Use of Linear Programming Method in Planning the Optimal Production of Al-Mamoun Factory for the Year 2017 – Iraq" by (Faeq & Hassan, 2024) aimed to develop a scientifically grounded production plan to minimize resource waste and improve productivity. This approach allowed the utilization of surplus resources to increase production levels and meet the needs of citizens. The researchers adopted a descriptive method for the theoretical framework and used the IN CANE program to define the constraints of the linear programming model and analyze the data. The study concluded that linear programming resulted in an optimal production plan that involved fewer products than the actual plan but achieved a higher profit margin. Moreover, the study found a surplus in the number of workers at the Al-Mamoun factory.

On another level, the study by (Tayebnasab, Mohebali, Farhad, & Hamid Reza Maleki, 2021) titled "Linear programming model to reduce patient payments and increase hospital income at the same time in Iran" explored the application of a bi-level linear programming model in a specialized hospital setting. The goal was to maximize hospital income while simultaneously reducing financial burdens on patients. The researchers used a descriptive method for the theoretical part and applied a two-level linear programming model in the practical phase. Secondary data was collected from the Specialized Hospital in Iran in order to solve the model using statistical methods for operations research. The application of the linear programming model showed its effective role in the performance of the hospital, as the results indicated a decrease in the costs paid by the patient and an increase in hospital revenues.

B. Theoretical Concepts

Building the conceptual framework is an essential step in any scientific research because of its role in clarifying the basic concepts on which the analysis is based. In this context, the most important concepts related to the research topic will be presented and interpreted, with reference to approved scientific references.

In this context, institutional finance is defined as a type of financing provided by large financial institutions (such as banks, investment funds, and insurance companies) to productive or service institutions. This type of financing aims to support economic growth by financing major projects and providing the necessary liquidity for the continuity of economic activity. Institutional finance is a vital component of modern financial markets, contributing to more efficient resource allocation and supporting the stability of the financial system [15]. It also constitutes the backbone of the global financial system, with institutional financial institutions providing essential services to corporations, governments, and other large organizations. They also play a pivotal role in facilitating economic growth, capital formation, and maintaining financial stability [17].

On the other hand, financial decision-making is the essence of the administrative process within the institution, as it is related to choosing the most appropriate financing or investment alternatives in order to achieve the strategic objectives of the organization. These decisions include project financing, resource allocation, profit distribution, and cost and risk management. Financial decision-making is based on quantitative and qualitative bases that include financial analysis, predictive models, and risk assessment [6]. In this regard, finance is about decision-making, which requires weighing costs and benefits. When the benefits are bigger than the costs, the decision is a good one; when smaller, it is a bad one [13].

Moreover, mathematical models in finance refer to formulas and equations that are used to represent and analyze financial phenomena with the aim of understanding the behavior of markets and making decisions based on quantitative data. There are many of these models, including statistical models, linear programming, Markov series, probabilistic models, and simulation models. These tools provide the possibility of evaluating alternatives and predicting the outcome of decisions, thereby improving the quality of financial decision-making [26].

In the same vein, risk analysis is the process of identifying and assessing risks that may affect the outcome of a financial decision. These risks include market risk, credit risk, liquidity risk, and operational risk. Risk analysis improves the organization's resilience and ability to adapt to unexpected fluctuations through the use of tools such as Value at Risk (VaR) models and scenario simulation models [16].

Finally, optimization is a branch of applied mathematics used to find the best possible solutions under a set of constraints. In finance, optimization techniques are used to determine the optimal combination of funding sources or to maximize return with the least risk. Linear and nonlinear programming are among the most widely used optimization techniques in this field [10].

C. Identifying Research Gaps in Previous Studies

A review of previous studies that have been relied upon shows that they have made significant contributions to employing mathematical programming models to improve institutional performance, especially in the productive and operational aspects. However, a critical analysis of these studies reveals a set of scientific and methodological gaps that justify the need for a new study in this area.

With regard to limiting itself to the productive field without financial consideration, previous studies, such as the studies of Fidaa Sheikh Hassan, Aarab, Yassin Faeq and Marwa Hassan, have focused on the application of linear programming and goal programming to improve production performance or to choose the optimal production mix. The study of Tayebnasab *et al.* examined a two-tier model with the aim of improving hospital income and reducing costs for patients. Despite the importance of these efforts, these studies did not address strategic financial decisions as an area of application of these models. In particular, there was no attempt to improve financing structures, financial resource planning, or the trade-off between sources of financing and investment.

Accordingly, a clear gap emerges in the absence of a link between optimization models and financial decision-making within organizations, leaving a scientific gap in the literature. In terms of the limited types of models used, although these studies succeeded in applying tools such as linear or numerical programming or even simple binary models, they nevertheless remained within the traditional framework of mathematical programming.

Thus, these studies could not exploit the broad potential of advanced models such as multi-objective programming, which reflects the reality of organizations where multiple goals coexist (profit, risk, liquidity). Similarly, dynamic programming, which addresses time-varying decisions, and probabilistic programming, which takes into account uncertainty in the financial environment, were not explored. Hence, an additional gap emerges: the limited methodological diversity and the continued reliance on classical approaches despite the increasing complexity of financial environments.

With regard to the absence of overlap with the concepts of quantitative finance, previous studies do not indicate any real use of quantitative financial concepts and instruments, despite their increasing importance in supporting financial decision-making. Examples include risk management using optimization techniques, asset allocation models, variance analysis, options pricing, and composite financial instruments.

Accordingly, previous studies have remained isolated from recent literature on quantitative finance despite the direct relationship between these two fields. Thus, a third gap can be identified: the weak integration between mathematical programming and quantitative financial approaches, which reduces the effectiveness of the proposed models in real-world applications.

D. Justification of the Contribution of the Present Study

In light of the previous gaps, the current study makes a distinct scientific and academic contribution through several aspects. While the efforts of previous studies focused mainly on solving operational problems, this study aims to employ optimization models in developing financial decision-making within institutions. Specifically, the study proposes a model that helps in selecting the best financing structure, distributing financial resources efficiently, and comparing investment alternatives based on quantitative indicators.

Thus, optimization models are transformed from production-only tools into instruments that support strategic financial planning. On the other hand, the study aims to employ advanced mathematical models by going beyond traditional approaches and integrating linear programming into the proposed framework in order to better simulate complex financial realities.

This methodological shift contributes to providing a more flexible and realistic model, enabling decision-makers to evaluate financial alternatives more accurately within a rigorous mathematical framework. Consequently, the proposed approach enhances the applied value of the study and increases its usability within institutional decision-making environments.

III. Methodology

A. Description of the Methods Used

1) Linear Programming: Linear Programming is a mathematical tool used to solve optimization problems, and it has been widespread since the development of the simplex algorithm in 1947, which made it applied in various fields such as banking, education, transportation, forestry, and petroleum (Wayne L. Winston & Jeffrey B. Goldberg, 2004). Linear programming is primarily concerned with optimizing a linear objective function while respecting linear constraints including equality or inequality imposed on decision variables. It forms a basic

foundation for many optimization techniques and is used in practice in areas such as production and transportation planning (Michel Goemans, 2015).

In addition, linear programming is defined as a mathematical method of allocating scarce resources in order to achieve a specific goal, which can be expressed in linear constraints. Thus, it contributes to making optimal decisions related to the allocation of human and material resources to achieve the maximum return or the lowest cost within a set of constraints (Students, 2016).

$$\min z = \sum_{i=1}^{m} \sum_{j=1}^{n} C_{ij} X_{ij} \tag{III.1}$$

$$X_{ij} \geq 0 \tag{III.2}$$

In this research, the Transportation Model was applied as one of the applications of linear programming to represent the relationship between available sources of financing (such as loans, bonds, and partnerships) and the needs of the four targeted projects. This model aims to distribute funding in a way that balances supply (from sources) and demand (from projects), while taking into account the maximum limits of each funding source.

We chose linear programming because the nature of the problem (the problem of optimizing financing in OCP) is suitable for linear modeling. This approach provides a strict quantitative model for distributing funding from multiple sources to different projects in order to reduce costs while respecting the financial constraints of each source. In addition, this model is characterized by scalability and flexibility, allowing the addition of new sources of funding or the modification of variables and constraints.

2) Linear Weighting Simulations for Weight Distribution: Simulation is a set of processes that imitate real-world systems or processes over a given period, whether these systems are physical or computational. Simulation involves studying the system and observing the impact related to the operational characteristics of the system in the real world (Faez Hassan, 2020). To achieve this, assumptions must be made about how these systems operate. These assumptions usually lead to mathematical and logical equations that together form the system model (Al-Qutli, 2018).

When it becomes difficult to solve problems using analytical or numerical methods, simulation represents an important alternative approach. In many situations, it may be the only viable method for solving complex problems. Simulation methods rely on resampling techniques and the generation of random numbers and variables with specific characteristics (Bari, 2002).

3) Weight Function: The weight function is a mathematical tool used when performing operations such as addition, integration, or averaging in order to give certain elements greater importance or influence than others within the same set. The application of a weight function produces either a weighted sum or a weighted average.

Weighting functions are widely used in statistics and data analysis and are closely related to the concept of measurement. They can be applied in both discrete and continuous environments and are used in constructing mathematical frameworks known as "weighted calculus" (Jane, Michael, & Robert, 1980).

In this study, the Linear Weighting Method was adopted within a deterministic simulation framework in order to distribute financial resources among several projects in a quantitative and systematic manner. This method relies on assigning relative weights to each project, reflecting its priority or strategic importance within the framework of the approved investment plan.

Algorithm 1 Funding Distribution Algorithm Using Linear Weighting Simulation

Import required libraries: numpy, pandas, matplotlib, seaborn, Arabic_reshaper, bidi.algorithm Define Arabic reshaping function: reshape_ar(text) = get_display(Arabic_reshaper.reshape(text)) Define list of projects and reshape project names using reshape_ar Define dictionary of funding sources with corresponding total amounts Reshape funding source names using reshape_ar Define weight vector $weights$ such that $\sum weights = 1$ Define function distribute_exact(total, weights) to: Compute raw allocation: $raw = weights \times total$ Apply floor: $base = \lfloor raw \rfloor$ Compute $remainder = total - \sum(base)$ Identify indices with largest fractional parts Increment top remainder indices in $base$ Return final integer allocation each funding source s in sources Compute allocation using distribute_exact($amount_s$, weights) Store result as a pandas Series indexed by project names Combine all allocations into a pandas DataFrame Reshape DataFrame column names and index labels using reshape_ar Print allocation table and display sum totals for each funding source Plot stacked bar chart of the allocations using matplotlib Apply reshaped Arabic labels to title, axis labels, and legend Display the final plot

B. Justification of the Methods Used

We have chosen the linear weighting method in this study due to its simplicity, clarity, and effectiveness in

dealing with decisions that require rational distribution of resources among multiple alternatives. This method is one of the applications of the SAW (Simple Additive Weighting) technique, where the decision-maker assigns an importance weight for each criterion. These weights are later used as coefficients multiplied by the numerical evaluations of each criterion, and the results are then aggregated to obtain the total score for each alternative. This process enables the construction of a clear and effective quantitative model for decision support [**?**].

As for digital simulations, they were adopted because they provide high flexibility in testing the distribution of funding under different scenarios without the need for accurate data on demand, which is consistent with the nature of the problem that depends on estimates and strategic directions rather than final quantitative data. Simulation is also considered an implementation of the model over time, bringing the model to life and showing how a particular object or phenomenon will behave. It is useful for testing, analysis, and training by representing realistic systems or concepts through a model [**?**].

Moreover, the model used is able to generate accurate and logical results while maintaining the sum of the original funding without decimals, thanks to the adopted algorithm that relies on approximation of values. This enhances the applicability of the results in practice. Overall, this choice combines quantitative accuracy with applied flexibility and provides an appropriate tool to support financing decision-making in strategic contexts.

1) Solver: Solver is an advanced analytical tool used to make data-driven decisions by building mathematical models based on techniques such as mathematical optimization, decision-making rules, Monte Carlo simulation, risk analysis, data mining, and forecasting. This tool is often integrated into the Excel environment through a range of products such as Analytic Solver, enabling analysts and managers to build complex decision models without the need for prior programming expertise.

These models are used to analyze uncertainty and allocate limited resources such as money, equipment, and human resources, often resulting in improved performance and significant financial savings. Solver has become a leading academic choice in business schools around the world, where it is taught in hundreds of universities and used in dozens of specialized scientific books [**?**].

We have chosen Excel Solver because it is easy to use and includes a visual interface within Excel without the need for programming. It helps in solving complex problems that may include hundreds of variables and constraints. In addition, it provides rapid analysis capabilities, allowing users to modify parameters and immediately observe the impact on results. Therefore, it is particularly suitable for research studies and scientific reports.

Algorithm 2 Algorithm for Distributing Funding Sources to Green Investment Program Projects (2023–2027)

1: Define Arabic project names and funding data for each source
2: Reshape Arabic text for correct display using `Arabic_reshaper` and `python-bidi`
3: Create a DataFrame *df* with projects as index and funding sources as columns
4: **for** each text element in the DataFrame columns and index **do**
5: Apply Arabic reshaping and the bidi algorithm
6: **end for**
7: Generate a stacked bar chart from the DataFrame using `matplotlib`
8: Add title, axis labels, legend, and grid to the chart
9: Display the chart using `plt.show()`

2) Python: Python is a programming language that is currently used very widely. It was created by Guido van Rossum in the late 1980s. Python is a powerful language whose code can be easily and simply read, which makes it easier for programmers to quickly develop applications. In addition, Python can be executed on some of the most popular operating systems such as Windows and Mac OS [22].

Python is also considered a highly readable and versatile language, and its name is inspired by a British comedy group called Monty Python. Its syntax is straightforward and provides immediate error reports, making it an excellent choice for beginners and newcomers to programming [21].

Going back to its origin, Python is a high-level programming language created by Guido van Rossum in 1986 while working at the Centrum Wiskunde & Informatica research center. Since then, the language has continued to develop with the addition of many features, becoming one of the most popular programming languages ever [9].

Moreover, Python has been advancing at an exceptional rate compared to all other programming languages, and it has been the most widely used language since 2017 until now. It is also ranked as the second easiest language in the world after Ruby, while surpassing it in terms of performance, popularity, number of libraries, and technical support [2].

Finally, Python is a portable, dynamic, free, and extensible programming language. It allows (but does not require) the use of modular and object-oriented program-

ming (OOP) approaches. Python was developed in 1989 by Guido van Rossum along with a large number of volunteers and contributors. It is a portable language, not only across various Unix systems but also on operating systems such as Mac, BeOS, NextStep, MS-DOS, and different versions of Windows [19].

Python was selected for this research due to its flexible, precise, and open-source programming environment, which allows the implementation of complex mathematical algorithms and the graphical representation of results, even in Arabic. This supports financial decision-making in the absence of accurate data. It is also scalable and easily updatable, which aligns with the requirements of the applied research on OCP.

C. Data Used

In this study, three sources of financing were relied upon, represented by financing through bonds, loans, as well as partnerships with companies, as shown in the following tables.

The first table aims to provide a comprehensive overview of the loans granted to OCP projects, specifying the financing banks, loan amounts, and the targeted projects. The table highlights the diversity of banking funding sources and their contributions to various environmental and industrial sectors.

Table 1 show that the largest loans were directed towards water desalination and green hydrogen projects, while renewable energy and climate technology projects received relatively lower funding. This reflects the company's preference for projects with direct strategic impact and higher banking financing costs. The second table shows the different bond issuances aimed at financing OCP projects, with details on maturity dates, amounts, and interest rates. This demonstrates the importance of bonds as a financing tool to secure long-term resources

Table 2 indicates that the company relied on multiple maturity issuances to distribute financial risks and achieve a balance between interest obligations and the total financing amount. The additional releases in February 2025 also reflect the company's flexibility in meeting changing financial needs. As for partnerships, two partnerships have been signed, the first with ENGER and the second with IFC Bank, but so far, the value of the partnerships has not been declared either for OCP or other companies.

IV. Results and Discussion

A. Presentation of Results

Within the framework of this study, and after applying the transportation model for the distribution of OCP project financing, which has a total value of 136578 million MAD, the analysis focused on four main projects: industrial infrastructure and fertilizers, water desalination, green hydrogen and ammonia, as well as renewable energies and climate technology.

The financing structure is based on three main sources of funding: loans, bonds, and partnerships. The obtained results reveal the pattern of financial resource distribution allocated to each project. This discussion aims to analyze how these resources are distributed and to evaluate the efficiency of this allocation in meeting project demand and directing funding according to project priorities. From the table 3, it is evident that international partnerships were allocated evenly across all projects, while loans and bonds were directed to specific projects. This reflects different financing strategies for each source, with partnerships representing a long-term strategic approach, whereas loans and bonds tend to cover specific needs.

The fourth table shows the final distribution results after using Excel Solver to optimize funding allocation, achieving the minimum total cost while meeting all project requirements.

TABLE I
LOANS PROVIDED FOR A COMPANY PROJECT OCP

Banks	Loan Value (Million Dirhams)	Target Project
IFC	1088.26	Renewable energies and climate technology
KFW	2162.4	Green hydrogen, ammonia and desalination
EBRD	2103.44	Desalination
AFDB	1511.4	Desalination
CACF	181.368	Desalination
CIF	201.52	Renewable energies and climate technology
Total	**7249 M MAD**	

Source: Produced by the author using data from the institution

TABLE II
BONDS ISSUED FOR A COMPANY PROJECT OCP

	History Due	
Publications	**2 May 2034**	**2 May 2054**
May 2024 release	12312.5M MAD at 7.5% interest	7387.5M MAD at 6.75% interest rate
Additional release in February 2025	2537.25M MAD	746.25M MAD

Source: Produced by the author using data from the institution

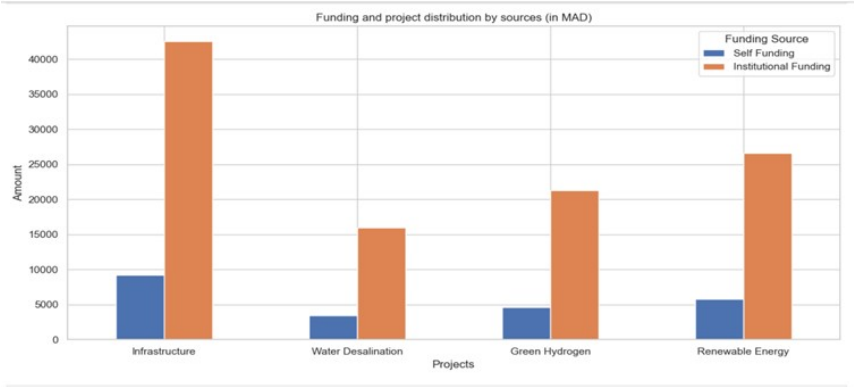

Fig. 1. Chart of Funding Allocation Using Linear Weighting Simulation for Weight Distribution (Python Output Produced by the Author)

TABLE III
DISTRIBUTED SPREADSHEET

	Industrial Infrastructure & Fertilizers	Desalination	Green Hydrogen and Ammonia	Energies and Climate Tech
Financing & Loans	0	5418	541	1290
International Bonds	9197	3449	4599	5748
International Partnerships	42534	15951	21267	26584
Total (Ask)	51731	24818	26407	33622

Source: Produced by the author using data from the institution

The table 4 indicates that bank loans were directed entirely to the industrial infrastructure and fertilizers project, while bonds focused on other main projects. International partnerships covered all four projects evenly, reflecting efficient allocation of financial resources according to project priorities and the goals of the Green Investment Program for the period 2023–2027.

$$\begin{aligned}\text{Min } Z = {} & 26584 \times 33622 + 15951 \times 24818 + 21267 \times 26407 \\ & + 0 \times 7249 + 9197 \times 22993 + 42534 \times 21489 \end{aligned} \tag{IV.1}$$

$$Z = 2\,976\,756\,582 \tag{IV.2}$$

TABLE IV
DISTRIBUTION RESULTS AFTER USING THE EXCEL SOLVER PROGRAM

	Industrial Infrastructure & Fertilizers	Desalination	Green Hydrogen and Ammonia	Energies and Climate Techn
Financing & Loans	7249	0	0	0
International Bonds	22993	0	0	0
International Partnerships	21489	24818	26407	33622
Total (Ask)	51731	24818	26407	33622

Source: Produced by the author using data from the institution

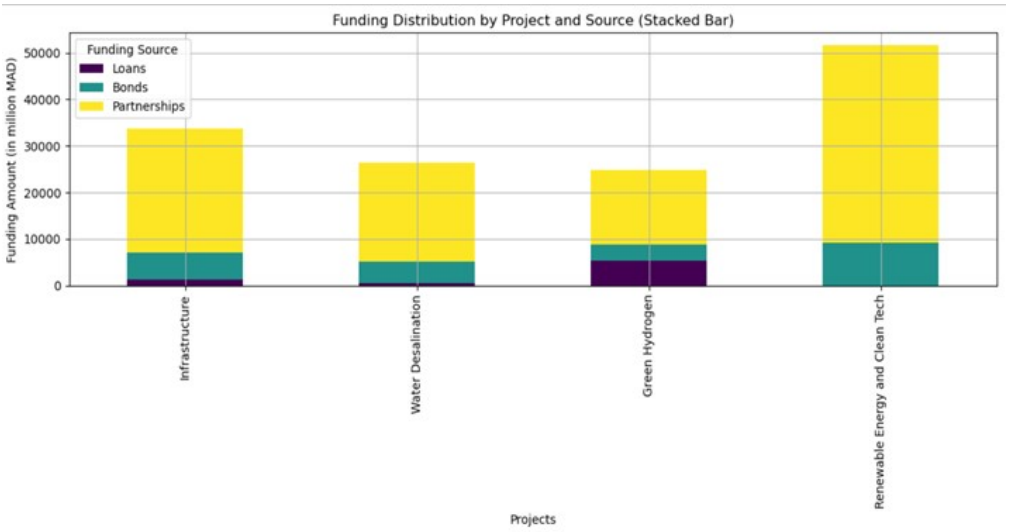

Fig. 2. Chart of Funding Sources Allocation to the Green Investment Program Projects (2023-2027) (Python Output Produced by the Author)

Through the results obtained from the above table, we note that the total offer value is equal to the total value of the demand which is 136578 million dirhams, which will contribute to solving the table directly without adding an imaginary column or row. Which is added with the aim of balancing the total values of demand and supply as for the distribution of funding sources, we note that international companies were directed to the four projects with similar values for a total of 106335.97 Million dirhams. This shows that it is the lowest cost of the three sources. International bonds were also directed to the same project only at a value of 22,993 million dirhams, which means that they were likely resorted to after exhausting the maximum allocated amount of strategic partnerships. While the international loans were fully directed to the industrial infrastructure and fertilizer project, they only used partial financing of the latter in the amount of 7249 million MAD, which indicates that it is relatively more expensive and therefore will only be used when needed. To cover the remaining demand and from here it appears to us that it has Greater emphasis was placed on the infrastructure project, which received funding from the three sources worth 51,731 million MAD. This reflects its great importance in the Green Investment Program for the period between 2023 and 2027 as for projects. that received a value of 0 in some funding sources This means that they have not received any funding from the available funding sources due to the fact that if they take funding from sources with a value of zero, it will become a cost Funding is high. Hence, we note that loan financing reflects the desire of the institution to use high-cost and interest financing in projects that achieve a quick return, of course, after directing it mainly to the industrial infrastructure project. As for bond financing, it represents a suitable medium term financing tool, which makes it suitable for projects that require stable financing without the need for rapid recovery as is the case with loans. Finally, partnership financing reflects the strategic dimension of these projects within the framework of sustainable development and energy transition, which also confirms the interest of international partners in supporting projects with environmental dimensions and positive effects on environmental resources. As for the achieved result MIN Z =2976756582 , it represents the lowest possible cost to achieve the financing targets within the framework of improving the financing deci-

sions of the Green Investment Program for the period 2023-2027.

B. Comparison of Results with Previous Studies

In light of the results of the current study, which relied on a linear programming model supported by linear weighting simulation, an optimal distribution of financing resources was reached among four OCP strategic projects, based on two main sources of financing: international bonds and partnerships. The Solver tool in Excel was used to solve the model, allowing great flexibility in adjusting criteria and funding constraints and simulating multiple scenarios. This approach made it possible to account for the relative importance of each project, especially in the absence of accurate demand data. The results showed the efficiency of this approach in improving resource allocation and balancing financing constraints with strategic objectives, while remaining reusable and adaptable to market changes.

Compared to previous studies, the methods and methodologies used are diverse, but the common denominator is the pursuit of improved resource allocation and support for managerial and productive decisions. For example, a study relied on linear programming models and goal programming to choose the optimal production mix in engineering industry companies in Syria and focused on sensitivity analysis to offer alternatives to decision makers. The study employed numerical linear programming using the LINDO program [1], [4] to present a model that achieves profits exceeding the actual results observed in an Algerian institution, highlighting the importance of mathematical models in improving economic performance.

On the other hand, another study aimed to plan optimal production in an Iraqi factory using the IN CANE program [7], where the model demonstrated the ability to allocate resources efficiently and reduce waste, resulting in increased profitability. At the service level, Tayebnasab *et al.* [23] proposed an advanced two-level linear programming model to reduce patient payments while simultaneously maximizing hospital income in a complex healthcare environment requiring a balance between social and financial objectives.

In terms of the methods used, our study is characterized by the adoption of linear weighting simulation as a decision-support mechanism. Unlike some previous studies that relied on goal programming or binary models, this study focused on a unified linear programming framework using the Solver tool to generate the optimal solution based on estimated weights. This model is simpler and more flexible, particularly in environments that lack accurate data or require easy-to-use decision-support tools.

Therefore, the present study can be considered both complementary and distinctive. It provides a simplified and effective model that combines widely available analytical tools (Excel/Solver) with a linear weighting approach, making it practically applicable within financial environments characterized by uncertainty and data scarcity. The results also reinforce the applied value of linear models in improving funding allocation, consistent with previous studies that demonstrated the feasibility of these models in different production and service contexts.

C. Discussion of Practical Implications and Limitations

The transportation model is an analytical tool that enables the distribution of funds to be organized in a deliberate manner, especially when multiple projects and diverse funding sources are involved. By applying this model to the situation of OCP, several practical advantages can be derived, as well as a number of challenges that may affect the accuracy and feasibility of the results. In practice, this model provides an effective support tool that assists decision-makers within the company, as it offers a quantitative framework based on linear programming and transportation models. This framework helps determine the optimal allocation of funding between different projects, based on the available financing sources and the needs of each project.

In addition to improving resource utilization, the model helps reduce waste and maximize the use of available financial resources, particularly when there is disparity in financing capacities among different sources (such as costly loans versus free or semi-free partnerships). Moreover, the model is flexible and easily updatable, as the data allocated to each project can be modified when new information becomes available. The model also provides an analytical perspective on the distribution of financing according to its source (bonds, loans, and partnerships), which contributes to understanding the level of dependence on each financing instrument and evaluating its potential risks or suitability. Furthermore, the same model can be reused for future projects simply by modifying the initial data, making it a long-term planning tool for financial decision-making.

However, despite the practical advantages offered by the model, several limitations must be considered. First, the model does not incorporate a temporal dimension, meaning that it does not provide funding distribution across time phases (short, medium, or long term), while some projects may require financing over multiple years.

Second, the model ignores detailed economic feasibility. It does not evaluate the financial return or economic

viability of each project, relying instead on estimated weights, which may affect its accuracy in complex financing situations.

Third, the model ignores the interaction between funding sources, as each financing source was analyzed independently without examining potential overlaps or complementarities between them, such as covering specific costs through a mix of financing sources.

Fourth, the absence of sensitivity and risk analysis is another limitation. The model does not test how the funding allocation would change if the weights or the total available budget were modified. Such analysis is essential in real-world environments where financial conditions and data can change rapidly.

V. Conclusion

This study found significant results that enhance the effectiveness of using linear programming models and linear weighting simulation techniques in improving financing decisions within OCP. The financial transportation model demonstrated the ability to distribute resources optimally among four strategic projects, taking into account different sources of financing, especially international bonds and international partnerships. This contributed to achieving the lowest possible cost of financing within realistic financial constraints.

In the same vein, the use of simulation techniques, such as the Linear Weighted Scoring Model, enabled the integration of multiple criteria including the cost of funding and the degree of risk, resulting in more balanced and objective financing decisions. The results showed that diversifying funding sources provides greater financial and strategic flexibility compared to relying on only one source of funding.

Moreover, the study showed that the model is flexible and adaptable to changing financial data, making it a reusable tool in long-term investment planning. Overall, the results confirm that quantitative models such as linear programming and simulation represent practical and effective tools that support decision-makers in addressing financing complexities and help identify optimal alternatives according to the company's priorities and objectives.

However, it is necessary to note some limitations that may affect the accuracy or comprehensiveness of the results. The model does not take into account the time dimension of funding distribution, ignores detailed economic analysis of each project, and addresses funding sources independently without considering potential overlap or complementarity between them. In addition, no sensitivity analysis or testing of the impact of changing data such as budget or weights was conducted, which is necessary to ensure that the model adapts to changing economic conditions.

Based on the above, this study provides a set of research recommendations to improve financing decision-making and enhance its effectiveness in applied business environments. First, future research should study the impact of global market fluctuations on optimal financing models by exploring how interest rates and international bond markets influence financing decisions in large industrial companies such as OCP. Second, a comparison between linear programming results and artificial intelligence models could be conducted to analyze the effectiveness of techniques such as neural networks or genetic algorithms in improving funding decisions.

Third, the model could be expanded to include risk analysis by integrating uncertainty through stochastic programming or Monte Carlo simulation to evaluate funding decisions in unstable environments. Fourth, the development of multi-criteria decision-making (MCDM) models such as AHP or the Linear Weighted Scoring Model alongside linear programming could help incorporate environmental and social considerations. Fifth, future work could include a comparative study between different Moroccan companies to analyze the efficiency of various financing sources. Sixth, long-term dynamic analysis of financing decisions could be performed by developing a dynamic programming model that examines the impact of financing on a company's financial performance over several years. Finally, future studies may explore the integration of Islamic finance instruments such as Musharaka and Mudaraba into traditional mathematical models and assess their impact on profitability and Shariah compliance.

In practical applications, the Linear Weighted Scoring Model and its simulation techniques represent effective quantitative tools that play a pivotal role in supporting financial decision-making, especially in contexts requiring the evaluation of multiple criteria. This methodology is widely used in several practical applications, particularly in investment budget planning, where it contributes to the allocation of financial resources among projects according to well-defined strategic priorities.

It also provides opportunities for investment portfolio management by allocating assets based on criteria such as expected return, risk, and liquidity, ensuring balanced portfolios aligned with investors' objectives. In the field of banking and corporate finance, the model is valuable for analyzing financing alternatives and selecting the most appropriate source, whether loans, bonds, or partnerships, based on criteria such as financing cost, repayment period, and risk level.

Furthermore, it is useful for assessing the relative

feasibility of investment projects, analyzing costs and benefits, and directing resources toward high value-added activities. The model also helps prioritize investment spending under financial constraints and supports financial performance analysis using multiple indicators.

Finally, linear weighting simulation provides a framework for testing multiple scenarios and evaluating the flexibility of financing decisions under changing market conditions. This enhances organizational adaptability and supports long-term financial planning. Thanks to these characteristics, the model represents a flexible and comprehensive framework for making financial decisions based on quantitative and objective analysis.

REFERENCES

[1] Z. Aarab, "Improving the Performance of Economic Institutions Using Numerical Linear Programming: A Case Study of the Metal Planting Institution," *Journal of Studies in Economics, Commerce and Finance*, 2020.
[2] A. Al-Hosiny, *Al-Kafi in Python*. 2024.
[3] R. Al-Qutli, *Simulation and Modeling*. Syrian Virtual University, 2018.
[4] F. Al-Sheikh Hassan, "Using Linear Programming Model and Importance Degree in Selecting the Optimal Investment Projects: A Comparative Study on Petrochemical Industry Companies," *Al-Bahth University Journal*, vol. 46, no. 2, Series of Economic and Tourism Sciences, 2024.
[5] A. Bari, *Modeling and Simulation*. King Saud University, 2002.
[6] E. Brigham and M. Ehrhardt, *Financial Management: Theory and Practice*, 16th ed. Cengage Learning, 2020.
[7] Y. Faeq and M. Hassan, "Using the Linear Programming Method in Optimal Production Planning for the Yamoon Factory for the Year 2017 in Iraq," *Al-Baath University Journal*, vol. 46, no. 2, Series of Economic and Tourism Sciences, 2024.
[8] A. Faez Hassan, *Statistics and Operations Research*, 2020.
[9] Y. Faraj, *Basics of the Python Programming Language*, 2009.
[10] F. S. Hillier and G. J. Lieberman, *Introduction to Operations Research*, 5th ed., 2005.
[11] H. O. Alanazi, A. H. Abdullah, and M. Larbani, "Dynamic Weighted Sum Multi-Criteria Decision Making: Mathematical," *International Journal of Mathematics and Statistics Invention*, 2013.
[12] G. Jane, G. Michael, and K. Robert, "The First Systems of Weighted Differential and Integral Calculus," 1980.
[13] R. Michael, *Financial Decision-Making*. The Wharton School, University of Pennsylvania, 2020.
[14] M. Goemans, *Linear Programming*, 2015.
[15] F. Mishkin and S. Eakins, *Financial Markets and Institutions*. Pearson Education, 2018.
[16] J. Philippe, *Value at Risk: The New Benchmark for Managing Financial Risk*, 2007.
[17] E. Samer, "Unlocking Opportunities: Understanding Institutional Banking," *Journal of Financial Markets*, Department of Psychiatry, American University of Beirut, Lebanon, 2023.
[18] L. Students, *Mathematics Lecture*, Faculty of Economics, Commerce and Management Sciences, University of Batna, 2016.
[19] G. Swinnen, *Learn Programming with Python*. The Arab Linux Community, 2013.
[20] F. Systems, *Analytic Solver Documentation*, 2009.
[21] L. Tagliaferri, *Programming in Python Language*. Hasoub Academy, 2020.
[22] H. Taher, *Learn Python for Beginners*.
[23] S. F. Tayebnasab, R. Mohebali, H. Farhad, and H. R. Maleki, "Introducing a Bi-Level Linear Programming Model to Reduce Patient Payment and Increase Hospital Income Simultaneously," *Hospital Practices and Research*, 2021.
[24] D. A. University, *Systems Engineering Fundamentals*. Defense Acquisition University Press, Fort Belvoir, VA, USA, 2001.
[25] W. L. Winston and J. B. Goldberg, *Operations Research*, 2004.
[26] P. Wilmott, *Paul Wilmott on Quantitative Finance*, 2nd ed., 3 vols. John Wiley & Sons, 2006.

International Journal on Optimization and Applications
IJOA. Vol. 06, Issue No. 01, Year 2026, www.ijoa-journal.com

Management Practices Among Small Moroccan Investors: An Analysis Between Economic Rationality and Behavioral Constraints

Karima LAMRANI
Department of Management Sciences of Organizations
Ibn Tofail University – National School of Business and Management (ENCG)
Kenitra, Morocco
lamranikarima1995@gmail.com

***Abstract*—This paper examines the financial management practices of small Moroccan investors by investigating the extent to which their decisions are driven by economic rationality or influenced by behavioral and contextual constraints. The central research question is as follows: to what extent do the financial choices of small Moroccan investors reflect economic rationality, and to what extent are they shaped by psychological biases and local specificities?**

To address this issue, a mixed-methods approach combining a structured questionnaire and qualitative insights was conducted with a sample of 250 individual investors. The findings reveal that, although decision-making exhibits elements of rationality, it is significantly influenced by behavioral biases such as loss aversion, overconfidence, social herding, and limited portfolio diversification.

Furthermore, the Moroccan context—characterized by restricted access to reliable financial information, low levels of financial literacy, and the strong influence of social and familial networks—intensifies these hybrid decision-making patterns. The study also highlights a marked preference for tangible assets, particularly real estate and gold.

These results emphasize the relevance of behavioral finance in explaining the actual practices of small investors and underline the necessity of enhancing financial education and improving access to reliable information. Finally, this research opens avenues for future studies aimed at expanding the sample, incorporating real market data, and assessing the impact of financial education programs on investment rationality and diversification.

***Index Terms*—Small investors, Morocco, Economic rationality, Behavioral finance, Psychological biases, Financial practices**

I. Introduction

In recent years, the Moroccan financial landscape has witnessed the emergence of a growing number of small investors, primarily originating from middle- and lower-income groups. These actors engage in various forms of investment, ranging from traditional savings to stock market participation, real estate investments, and even entrepreneurial initiatives. Their role is becoming increasingly significant, both in terms of their contribution to economic development and their involvement in promoting financial inclusion.

Classical economic theory assumes that investors make rational decisions aimed at maximizing returns while minimizing risks. However, empirical observations suggest that the financial choices of small investors often deviate from this rational framework. Indeed, psychological factors such as loss aversion, overconfidence, and herding behavior strongly influence their decision-making processes. These effects are further amplified by contextual constraints specific to the Moroccan environment, including limited access to reliable financial information and low levels of financial literacy.

Accordingly, the central research question of this study is as follows: to what extent are the financial practices of small Moroccan investors driven by economic rationality, and to what extent are they affected by behavioral and contextual constraints?

This research aims to examine both the rational foundations underlying investors' financial decisions and the behavioral factors that may limit such rationality. It also seeks to identify the local specificities that shape these practices. The study contributes to the theoretical development of behavioral finance within the Moroccan context while providing practical insights for financial institutions and financial education programs.

To better understand the financial behavior of small Moroccan investors, it is essential to position this study within the broader body of existing knowledge. Classical finance literature emphasizes economic rationality as a fundamental principle, assuming that investors make optimal decisions based on their objectives and risk preferences. However, behavioral finance research has demonstrated that real-world decisions are often influenced by cognitive biases, emotions, and contextual

constraints.

Accordingly, this paper is structured into three main sections. The first section presents the literature review and conceptual framework, focusing on economic rationality, behavioral constraints, and the specific characteristics of small Moroccan investors. The second section describes the research methodology, including data collection and analysis techniques. Finally, the third section presents the results and discussion, analyzing observed financial practices and comparing them with existing theoretical frameworks.

II. Literature Review and Conceptual Framework

To understand the financial practices of small Moroccan investors, it is essential to situate this study within the existing theoretical framework. Classical finance literature emphasizes economic rationality, considering that financial decisions are primarily driven by the pursuit of optimal returns and risk minimization. However, a growing body of research in behavioral finance has demonstrated that actual investor behavior often deviates from this theoretical model. These deviations are explained by cognitive biases, emotional influences, and contextual constraints that shape decision-making processes.

This section provides a synthesis of the relevant theoretical and empirical contributions. It first examines the principle of economic rationality and its limitations, then explores the main behavioral biases likely to affect small investors. Finally, it highlights the specific characteristics of the Moroccan context, particularly in terms of financial literacy, investment preferences, and social influences, in order to better contextualize the observed practices.

A. Economic Rationality in Financial Management

Classical economic theory is based on the assumption that investors are rational agents who systematically seek to maximize their utility while minimizing their exposure to risk. This perspective relies on the expected utility theory and assumes that individuals have access to complete information, are capable of processing it without bias, and consistently choose the optimal alternative. This representation, embodied in the concept of the *homo economicus*, constitutes the foundation of traditional financial models and portfolio theories developed by Markowitz [**?**] as well as the Efficient Market Hypothesis proposed by Fama [**?**].

Within this framework, financial decisions are interpreted as rational trade-offs, where investors compare expected returns with anticipated risks in order to allocate their resources efficiently. This approach is central to neoclassical finance and underlies standard financial models such as the Capital Asset Pricing Model (CAPM) and modern portfolio theory.

However, such pure rationality is rarely observed in practice, particularly among small investors. These individuals often lack the technical resources and access to comprehensive information required to objectively evaluate their investment choices. Moreover, their decisions are influenced by time constraints, limited financial resources, and contextual factors. In the Moroccan context, for instance, unequal access to financial information, the complexity of financial products, and the prevalence of informal investment practices further hinder the strict application of economic rationality. As a result, a significant gap exists between theoretical models and the actual behavior of small investors.

B. Behavioral and Psychological Constraints

Behavioral finance has emerged as a fundamental research field precisely to explain the gap between theoretical rationality and observed investor behavior, particularly since the seminal work of Kahneman and Tversky [**?**]. It highlights the influence of cognitive biases, emotions, and social environments on financial decision-making. Unlike traditional finance, which assumes optimal and fully rational choices, behavioral finance acknowledges that individuals rely on heuristics—mental shortcuts that facilitate decision-making but often lead to systematic errors.

Among the most extensively studied behavioral biases, loss aversion occupies a central position. Investors tend to exhibit a stronger preference for avoiding losses than for achieving equivalent gains. This tendency often leads them to retain underperforming assets rather than liquidating them promptly, in the hope of a future recovery. In the Moroccan context, this bias is further reinforced by limited financial literacy and the cautious attitude of small investors toward markets perceived as complex and uncertain.

Overconfidence represents another prevalent bias. Many investors overestimate their ability to predict market movements or identify profitable opportunities. This overestimation can result in excessive trading activity or a lack of diversification, thereby increasing exposure to risk. In Morocco, this phenomenon is particularly observable among novice investors or those relying on unverified information obtained from social networks or informal sources.

The confirmation bias also plays a significant role in shaping investment decisions. Individuals tend to favor information that confirms their prior beliefs while disregarding contradictory evidence. Such behavior can limit portfolio diversification and increase vulnerability

to misjudgments. In the Moroccan context, where access to reliable financial information may be constrained, this bias is amplified by dependence on non-professional or biased information sources.

Finally, social herding—commonly referred to as herd behavior—constitutes a major driver of financial decisions. Investors often imitate the perceived actions of others, especially in environments characterized by low institutional trust and a strong reliance on social networks for information. In Morocco, the influence of family advice and social circles further strengthens this tendency, potentially leading to collective and sometimes irrational investment movements.

These psychological biases are not merely anomalies; they actively shape financial market dynamics and help explain phenomena such as excessive volatility, speculative bubbles, and collective panic. Therefore, their analysis is essential for understanding the behavior of small Moroccan investors.

III. Empirical Research Approach

To rigorously examine the financial behavior of small Moroccan investors, it is necessary to formulate research hypotheses that allow the confrontation of the theoretical framework with the collected empirical data. These hypotheses reflect the expected relationships between socio-demographic variables, financial knowledge, information sources, and investment behaviors. Their statistical testing constitutes a crucial step in assessing the respective influence of economic rationality and behavioral constraints on the decisions under study.

A. Research Hypotheses

H1: Financial literacy positively influences the rationality of investment decisions.
H0: Financial literacy has no significant impact on the rationality of investment decisions.
H1: Financial literacy has a positive and significant impact on the rationality of investment decisions.

H2: Behavioral biases significantly affect the investment choices of small investors.
H0: Behavioral biases have no significant impact on investment choices.
H1: Behavioral biases have a significant impact on investment choices.

H3: Information sources influence perceived reliability and shape financial behaviors.
H0: Information sources have no significant effect on financial behaviors.
H1: Information sources have a significant effect on financial behaviors.

H4: Socio-demographic characteristics (age, income, occupation) influence investment practices.
H0: Socio-demographic characteristics have no significant effect on investment practices.
H1: Socio-demographic characteristics have a significant effect on investment practices.

B. Type of Research

This study adopts a mixed-methods approach combining quantitative and qualitative techniques in order to provide a comprehensive and nuanced understanding of the financial practices of small Moroccan investors. The quantitative approach constitutes the core of the research, as it enables the identification of general trends, the measurement of behavioral frequencies, and the analysis of statistical relationships between the variables under study. It is based on the administration of a structured questionnaire to a representative sample, facilitating the quantification and comparison of results.

In addition, a qualitative dimension has been incorporated through open-ended questions and, in some cases, exploratory interviews, in order to capture respondents' perceptions, motivations, and subjective reasoning. This dual methodological design aims not only to identify measurable correlations but also to better understand the complexity of financial behavior within its specific socio-economic and cultural context in Morocco.

C. Population and Sample

The target population of this study consists of small Moroccan investors, defined as individuals making personal investments of relatively modest amounts, generally below 50,000 Moroccan dirhams, in various financial or non-financial assets. This threshold was selected to distinguish this category from individuals with higher financial capacity and institutional investors.

Participant selection was based on several criteria: being at least 18 years old, having a minimum of six months of investment experience, representing geographical diversity including major metropolitan areas (such as Casablanca and Rabat) as well as medium-sized cities (such as Meknès, Nador, and Agadir), and providing informed consent to participate in the survey.

The final sample consists of 250 respondents, a size considered sufficient to ensure representativeness and to support robust statistical analyses, including factor analysis and multiple regression, with an acceptable level of confidence.

D. Data Collection Instrument

The primary data collection instrument used in this study is a structured questionnaire, developed based on

insights from behavioral finance literature as well as empirical studies conducted in the Moroccan context. The questionnaire was designed to comprehensively cover the key dimensions related to financial practices and behavioral biases among small investors. It consists of five complementary sections.

The first section gathers general information about respondents, including age, gender, education level, employment status, and income, in order to establish a detailed socio-demographic profile. The second section focuses on investment experience, collecting data on duration, invested amounts, and preferred asset types. The third section examines exposure to financial information, including the sources consulted, the role of social media, and perceptions of information reliability. The fourth section addresses investment behaviors and decision-making, analyzing saving habits, reactions to gains and losses, and diversification strategies. Finally, the fifth section evaluates the level of financial knowledge and literacy by assessing the understanding of key concepts and including self-assessment measures.

Additionally, several open-ended questions were incorporated to allow respondents to elaborate on the motivations and reasoning underlying their investment decisions, thereby enriching the quantitative analysis with a more nuanced qualitative dimension.

E. Data Analysis Methods

The data analysis relies on a set of complementary techniques aimed at providing a descriptive, explanatory, and interpretative understanding of the financial behaviors under study. First, descriptive statistics (frequencies, means, and standard deviations) were used to characterize the socio-demographic profile of respondents and their investment practices.

Second, an exploratory factor analysis (EFA) was conducted to identify the underlying dimensions of financial behaviors and psychological biases, allowing variables to be grouped into coherent and interpretable factors. Furthermore, correlation tests and multiple regression analyses were employed to examine the relationships between key variables, particularly socio-demographic characteristics, financial literacy, and investment behaviors.

Finally, a qualitative content analysis was performed on the open-ended responses, enabling the identification of emerging themes specific to the Moroccan context and enriching the interpretation of quantitative findings.

F. Reliability and Validity

To ensure the scientific rigor of the study, particular attention was given to the reliability and validity of the measurement instruments. Initially, the questionnaire was pre-tested on a pilot sample of twenty participants, which made it possible to assess the clarity of the questions, identify potential ambiguities, and implement necessary adjustments to improve internal consistency.

The validity of the research design was further strengthened by grounding the questionnaire in well-established theoretical concepts from behavioral finance, particularly the foundational work of Kahneman and Tversky [**?**], while also incorporating recent empirical studies conducted in the Moroccan context (Hmimnat, 2024; Massiki et al., 2023). This combined approach, integrating theoretical grounding and empirical validation, enhances the relevance and robustness of the results.

IV. Results and Discussion

This section presents the main findings of the study, derived from the questionnaire administered to small Moroccan investors. The objective is to highlight observed financial behaviors, cognitive biases, and contextual influences that shape investment decisions. The data are first analyzed descriptively to establish the socio-economic profile and investment practices of participants. Subsequently, relationships between variables are examined using appropriate statistical tests in order to validate the formulated hypotheses.

The discussion goes beyond a mere presentation of results by situating the observed behaviors within the frameworks of classical and behavioral finance theories, while taking into account the cultural, economic, and institutional specificities of the Moroccan context. The analysis aims to explain the gap between expected economic rationality and actual practices, and to identify the psychological and social factors influencing the financial decisions of small investors.

A. General Information (Questions 1–5)

The socio-demographic characteristics of participants provide an essential framework for understanding their investment practices. They help identify the most represented profiles in terms of age, gender, education level, employment status, and monthly income.

TABLE I
SOCIO-DEMOGRAPHIC PROFILE OF RESPONDENTS

Variable	Categories	Frequency (n=250)	Percentage (%)
5*Age	Under 25 years old	40	16%
	25–34 years old	90	36%
	35–44 years old	60	24%
	45–54 years old	40	16%
	55 years and above	20	8%
2*Gender	Male	138	55%
	Female	112	45%
4*Education Level	Primary or below	20	8%
	Secondary	65	26%
	University (Bachelor/Master)	145	58%
	Doctorate	20	8%
4*Monthly Income	$<$ 3,000 DH	50	20%
	3,001–6,000 DH	110	44%
	6,001–10,000 DH	70	28%
	$>$ 10,000 DH	20	8%

Source: SPSS Software

The sample is relatively young, with a majority (60%) of respondents aged between 25 and 44 years, and slightly male-dominated (55%). The population is predominantly composed of university graduates (58%), which may facilitate the understanding of basic financial concepts.

In terms of income distribution, respondents are mainly concentrated in the middle-income category, with 44% earning between 3,001 and 6,000 DH, confirming that the study effectively targets small-scale investors. Furthermore, the diversity in professional status allows for capturing a wide range of investment behaviors, from risk-averse salaried individuals to more proactive entrepreneurial profiles.

B. B. Investment Experience and Practices (Questions 6–9)

The investment experience and behavior of participants are crucial for understanding their financial habits and their exposure to risk. This section examines the duration of investment activity, the average amount invested, the preferred asset classes, and the frequency of investment decisions.

The majority of small investors (55%) report having between one and three years of investment experience, indicating relatively recent but progressively increasing participation in financial markets. The amounts invested remain modest, with nearly half of the respondents (48%) allocating less than 10,000 DH, which further confirms their classification as small-scale investors.

[0]*Multiple responses were allowed; therefore, percentages may exceed 100%.

TABLE II
INVESTMENT EXPERIENCE AND PRACTICES

Question	Categories	Frequency (n=250)	Percentage (%)
4*Investment Duration	$<$ 1 year	40	16%
	1–3 years	138	55%
	3–5 years	50	20%
	$>$ 5 years	22	9%
3*Average Amount Invested	$<$ 10,000 DH	120	48%
	10,001–30,000 DH	90	36%
	30,001–50,000 DH	40	16%
5*Preferred Asset Types*	Real Estate	105	42%
	Gold / Precious Metals	88	35%
	Savings Products	75	30%
	Stocks / Bonds	35	14%
	Entrepreneurship Projects	28	11%
3*Investment Frequency	Rarely (1–2 times/year)	95	38%
	Occasionally (3–5 times/year)	110	44%
	Regularly ($>$ 6 times/year)	45	18%

Source: SPSS Software

Regarding asset preferences, investment choices are predominantly oriented toward tangible assets. Real estate (42%) and gold (35%) emerge as the most favored options, whereas more abstract financial instruments, such as stocks and bonds, remain marginal (14%). This pattern highlights a strong degree of risk aversion and a clear preference for security and capital preservation.

In terms of investment frequency, 44% of respondents make investment decisions occasionally (three to five times per year), reflecting an active yet cautious engagement. Such behavior suggests that investment decisions are often driven by perceived market opportunities rather than continuous strategic planning.

C. C. Access to and Exposure to Financial Information (Questions 10–11)

Access to financial information and the perceived reliability of its sources play a critical role in shaping the investment decisions of small investors. This section examines the primary information channels used by respondents, as well as the level of trust they place in these sources.

The results indicate that social media constitutes the primary source of financial information for 55% of respondents, followed by advice from friends and family (48%). Traditional media remains a relevant source (38%), while professional channels, such as financial

[0]*Multiple responses were allowed; therefore, percentages may exceed 100%.

TABLE III
ACCESS TO AND PERCEPTION OF FINANCIAL INFORMATION

Question	Categories	Frequency (n=250)	Percentage (%)
5*Main Information Sources	Traditional media (TV, press, radio)	95	38%
	Social media	138	55%
	Friends / family advice	120	48%
	Financial advisors / banks	65	26%
	Specialized websites / platforms	50	20%
4*Perceived Reliability	Very reliable	40	16%
	Fairly reliable	110	44%
	Slightly reliable	75	30%
	Not reliable at all	25	10%

Source: SPSS Software

advisors and specialized platforms, are less frequently consulted.

This distribution highlights a strong reliance on informal and easily accessible information channels, which may expose investors to cognitive biases and information asymmetry. The limited use of professional sources suggests potential gaps in financial literacy or accessibility to expert advice.

Regarding perceived reliability, the majority of respondents consider financial information to be either fairly reliable (44%) or only slightly reliable (30%), while a smaller proportion expresses high confidence (16%). This indicates a moderate level of trust in available information, reflecting a certain degree of skepticism or uncertainty among small investors.

Overall, these findings suggest that investment decisions are shaped by a combination of informal influence and partial trust in information sources, which may contribute to suboptimal decision-making and reinforce behavioral biases such as herding and overconfidence. Regarding the perception of reliability, only 16% of respondents consider their information sources to be "very reliable," while 44% rate them as "fairly reliable." This indicates that although small investors actively seek financial information, a substantial proportion remains uncertain about its quality and credibility.

Such uncertainty may significantly influence the rationality of their investment decisions and increase their susceptibility to behavioral biases. In particular, reliance on partially trusted information sources can amplify cognitive distortions such as overconfidence, herding behavior, and heuristic-driven decision-making.

D. D. Financial Behavior and Psychological Attitudes (Questions 12–15)

The financial behavior of small investors reflects the significant influence of psychological biases and emotional factors on their decision-making processes. This section analyzes investors' reactions to losses, their level of portfolio diversification, the key factors driving their investment choices, and their self-perception regarding the rationality of their decisions.

TABLE IV
FINANCIAL BEHAVIORS AND PSYCHOLOGICAL ATTITUDES

Question	Categories	Frequency (n=250)	Percentage (%)
3*Reaction to Financial Loss	Hold assets expecting recovery	170	68%
	Sell quickly	60	24%
	Seek advice before acting	20	8%
4*Portfolio Diversification	Yes, always	55	22%
	Sometimes	95	38%
	Rarely	65	26%
	Never	35	14%
4*Decision Influencing Factors	Personal analysis	120	48%
	Behavior of other investors	110	44%
	Advice from experts or relatives	80	32%
	Intuition or emotions	95	38%
4*Perception of Decisions	Very rational	40	16%
	Rather rational	125	50%
	Rather emotional	70	28%
	Very emotional	15	6%

Source: SPSS Software

TABLE V
FINANCIAL KNOWLEDGE AND EDUCATION

Question	Categories	Frequency (n=250)	Percentage (%)
2*Participation in financial training / investment management	Yes	80	32%
	No	170	68%
2*Knowledge of risk diversification concept	Yes	95	38%
	No	155	62%

Source: SPSS Software

The observed behaviors in response to financial losses indicate a strong degree of risk aversion, as 68% of respondents prefer to hold their assets in anticipation of a potential recovery rather than selling immediately. This tendency reflects the influence of loss aversion, a central concept in behavioral finance.

Portfolio diversification appears to be only partially implemented. While 22% of respondents consistently diversify their investments, a majority (38%) do so only occasionally, suggesting a moderate and inconsistent approach to risk management.

Investment decisions are influenced by multiple factors. Although personal analysis remains the primary driver for 48% of respondents, peer influence and intuition also play a significant role, accounting for 44% and 38%, respectively. This highlights the coexistence of rational evaluation and behavioral influences in the decision-making process.

Regarding the perception of their own decisions, 50% of respondents consider their choices to be "rather rational." However, nearly one-third (34%) perceive them as either "rather" or "very emotional," illustrating the ongoing interaction between bounded rationality and psychological constraints.

E. E. Financial Knowledge and Education (Questions 16–17)

Financial knowledge and access to education are key determinants of the rationality of investment decisions. This section examines participation in financial training programs and the level of understanding of key concepts, such as risk diversification. The results indicate that the majority of small investors have not received any formal financial training (68%), which limits their ability to objectively assess investment risks and opportunities.

Furthermore, only 38% of respondents are familiar with the concept of risk diversification, a fundamental technique for reducing exposure to potential losses. This finding confirms that low financial literacy constitutes a key factor explaining the presence of irrational decision-making and the behavioral biases observed in previous sections.

These findings suggest that targeted and accessible financial education programs could enhance economic rationality and improve investment management among small investors in Morocco.

F. F. Hypothesis Testing

To test the first hypothesis, which postulates a relationship between education level and the rationality of financial decisions, a Chi-square test was conducted. The results ($\chi^2(9) = 19.83$, $p = 0.019$) indicate a statistically significant relationship at the 5% significance level. Thus, higher levels of education are associated with a greater likelihood of perceiving financial decisions as rational. The null hypothesis is therefore rejected in favor of the alternative hypothesis.

Regarding the second hypothesis, the analysis aimed to examine whether loss aversion influences portfolio diversification. The Chi-square test results ($\chi^2(2) = 17.08$, $p < 0.001$) reveal a highly significant relationship. Individuals who prefer to hold declining assets rather than sell them exhibit a strong tendency toward limited diversification. This finding leads to the rejection of the null hypothesis and supports the conclusion that loss aversion negatively affects diversification.

The third hypothesis focused on the role of social media in the emergence of herding behavior. The Chi-square test results ($\chi^2(1) = 16.71$, $p < 0.001$) show a highly significant association. Small investors who rely

primarily on social media for information are more likely to follow the behavior of other investors, confirming the presence of herding effects. Therefore, the null hypothesis is rejected in favor of the alternative hypothesis.

Finally, the fourth hypothesis examined whether investment experience improves the rationality of financial decisions. The Chi-square test results ($\chi^2(3) = 16.82$, $p = 0.001$) indicate a significant relationship. Investors with greater experience tend to perceive their decisions as more rational compared to less experienced individuals. Once again, the null hypothesis is rejected and the alternative hypothesis is confirmed.

V. General Synthesis and Discussion of Results

The analysis of the questionnaire responses reveals several important trends in the financial management practices of small investors in Morocco. As shown in :contentReference[oaicite:0]index=0, participants are predominantly young, slightly male-dominated, and largely university-educated, with moderate monthly incomes. This confirms that the sample consists mainly of small-scale investors with limited financial resources.

Their investment experience is generally recent, and the amounts invested remain modest, with nearly half of the respondents allocating less than 10,000 DH. Investment preferences are primarily oriented toward tangible assets, such as real estate and gold, while more abstract financial instruments, including stocks and bonds, are less commonly used. The frequency of investment decisions remains moderate, reflecting a cautious approach and a strong preference for security.

Access to financial information is mainly driven by social media and advice from relatives, whereas professional or specialized sources are less frequently consulted. The perceived reliability of these sources is relatively low, with only 16% of respondents considering them highly reliable. This limited trust may reduce the rationality of investment decisions and increase the impact of behavioral biases.

These biases are clearly reflected in the observed behaviors: loss aversion is dominant, portfolio diversification remains partial, and social influence or intuition plays a significant role in decision-making. Furthermore, the majority of investors have not received formal financial training, and only a minority is familiar with the concept of risk diversification. This lack of financial literacy further increases their vulnerability to irrational decision-making.

Overall, the financial practices of small Moroccan investors appear to be hybrid, combining bounded economic rationality, behavioral constraints, and local contextual factors, such as the influence of social networks, family culture, and traditional investment habits. These findings highlight the importance of strengthening financial education and providing appropriate guidance in order to reduce cognitive and emotional biases, improve diversification, and promote more rational investment decisions.

Financial institutions can play a key role by offering simple and educational financial products, while policymakers should encourage access to reliable information sources and support the development of targeted financial education programs. This study demonstrates that understanding financial behavior requires combining quantitative analysis with qualitative insights, in order to fully capture the interaction between economic rationality, psychological biases, and local context.

A. A. Comparison with Existing Literature

The results obtained largely confirm findings reported in the international literature on small investors. Behavioral biases such as loss aversion, overconfidence, and herding behavior are widely documented in the seminal work of Kahneman and Tversky (1979), as well as in recent studies focusing on individual investors in emerging markets (Massiki et al., 2023; Hmimnat, 2024).

However, some observed behaviors in the Moroccan context, particularly the strong preference for tangible assets and the predominant influence of social networks and family advice, reveal local specificities that are not always present in Western contexts. These differences highlight the importance of considering cultural and contextual constraints when analyzing financial practices.

B. B. Theoretical Implications

The findings of this study provide valuable insights for behavioral finance applied to small investors. They demonstrate that economic rationality remains partial and is strongly influenced by psychological, social, and cultural factors.

The identification of hybrid behaviors, combining limited rational decision-making with cognitive biases, confirms the need to develop financial models that explicitly incorporate behavioral dimensions. These results may also contribute to adapting traditional portfolio theory and risk assessment models to emerging market contexts, where access to information is limited and financial literacy is heterogeneous.

C. C. Study Limitations

Several limitations should be acknowledged. First, although the sample is relatively representative (n = 250), it mainly includes urban investors, which limits the generalizability of the findings to rural populations or less financially connected individuals.

Second, the use of a self-reported questionnaire may introduce response bias, as participants may tend to present their behavior as more rational or socially desirable than it actually is.

Finally, this study focuses exclusively on the Moroccan context, and the results cannot be directly generalized to other countries or financial markets without caution.

D. D. Future Research Directions

This study opens several avenues for further research aimed at deepening the understanding of financial practices among small investors. Future studies could extend the analysis to rural populations, where financial behaviors may differ due to limited access to financial services and information.

Additionally, comparative analyses based on socio-demographic characteristics such as gender, age, and education level could provide more nuanced insights into investment behavior. The use of real market data would also be valuable to validate and complement the self-reported findings of this study.

Moreover, experimental research designs could be employed to assess the impact of targeted financial education programs on improving decision rationality and reducing behavioral biases. Such approaches would allow for a more causal interpretation of the relationship between financial literacy and investment behavior.

VI. Conclusion

This study provides an in-depth analysis of the financial management practices of small investors in Morocco by examining the interplay between economic rationality, behavioral constraints, and contextual factors.

The findings reveal that, although investors aim to optimize their investment decisions, their behavior is significantly influenced by cognitive biases such as loss aversion, overconfidence, and herding behavior. These effects are further reinforced by limited access to reliable information, low levels of financial education, and specific cultural characteristics of the Moroccan context, particularly the strong influence of social networks and close social circles.

The coexistence of partial rationality and emotional decision-making highlights the complexity of financial behavior within this group of investors.

From a theoretical perspective, this study confirms the relevance of behavioral finance in explaining the gap between traditional financial models and actual investment practices. It also emphasizes the need to adapt financial tools and models to local contexts, where information asymmetry and heterogeneous financial literacy levels are prevalent.

From a practical standpoint, the results suggest that strengthening financial education is essential, particularly with regard to portfolio diversification, risk assessment, and financial planning. Improving access to reliable and professional financial information is also crucial in order to reduce dependence on informal sources such as social media and personal networks.

Financial institutions have an important role to play by designing simple, transparent, and secure financial products that promote diversification and reduce risk exposure. Furthermore, raising awareness of behavioral biases and integrating psychological dimensions into financial education programs could significantly enhance decision rationality and improve the management of gains and losses.

Finally, this research highlights the importance of promoting informed and sustainable financial inclusion. Future studies may build on this work by incorporating rural populations, exploring socio-demographic variations, and combining survey-based data with real market data. The ultimate objective is to contribute to a more stable, inclusive, and efficient financial system in Morocco.

References

[1] N. Barberis and R. Thaler, "A Survey of Behavioral Finance," in *Handbook of the Economics of Finance*, G. Constantinides, M. Harris, and R. Stulz, Eds., vol. 1, pp. 1053–1128, Elsevier, Amsterdam, 2003.

[2] E. F. Fama, "Efficient Capital Markets: A Review of Theory and Empirical Work," *Journal of Finance*, vol. 25, no. 2, pp. 383–417, 1970.

[3] H. Hmimnat, "Comportements financiers des investisseurs individuels au Maroc," *Revue Marocaine de Gestion*, vol. 12, no. 1, pp. 45–68, 2024.

[4] D. Kahneman and A. Tversky, "Prospect Theory: An Analysis of Decision under Risk," *Econometrica*, vol. 47, no. 2, pp. 263–291, 1979.

[5] H. Markowitz, "Portfolio Selection," *Journal of Finance*, vol. 7, no. 1, pp. 77–91, 1952.

[6] S. Massiki, H. Berrada, and R. El Amrani, "Éducation financière et décisions d'investissement des particuliers au Maroc," *Revue Africaine de Finance*, vol. 9, no. 2, pp. 101–120, 2023.

[7] Autorité Marocaine du Marché des Capitaux (AMMC), "Rapport sur l'évolution des investisseurs individuels sur la Bourse marocaine," Rabat, Morocco, 2024.

[8] H. Shefrin, *Beyond Greed and Fear: Understanding Behavioral Finance and the Psychology of Investing*, Harvard Business School Press, Boston, 2000.

APPENDIX A
RESEARCH QUESTIONNAIRE

Study Title: *Financial Management Practices of Small Moroccan Investors: Between Economic Rationality and Behavioral Constraints*

Note: This questionnaire is strictly confidential and intended exclusively for scientific research purposes.

A. I. General Information

1. Age:

- ☐ Under 25 years old
- ☐ 25–34 years old
- ☐ 35–44 years old
- ☐ 45–54 years old
- ☐ 55 years and above

2. Gender:

- ☐ Male
- ☐ Female

3. Education Level:

- ☐ Primary or below
- ☐ Secondary
- ☐ University (Bachelor/Master)
- ☐ Doctorate

4. Employment Status:

- ☐ Employee
- ☐ Self-employed / Trader
- ☐ Entrepreneur
- ☐ Student
- ☐ Retired
- ☐ Unemployed

5. Average Monthly Income:

- ☐ Less than 3,000 DH
- ☐ 3,001–6,000 DH
- ☐ 6,001–10,000 DH
- ☐ More than 10,000 DH

B. II. Investment Experience and Practices

6. Investment Experience:

- ☐ Less than 1 year
- ☐ 1–3 years
- ☐ 3–5 years
- ☐ More than 5 years

7. Average Amount Invested:

- ☐ Less than 10,000 DH
- ☐ 10,001–30,000 DH
- ☐ 30,001–50,000 DH

8. Preferred Investment Types (multiple answers possible):

- ☐ Bank savings products
- ☐ Real estate
- ☐ Gold or precious metals
- ☐ Stocks or bonds
- ☐ Entrepreneurship / personal projects
- ☐ Other (please specify): _____

9. Investment Decision Frequency:

- ☐ Rarely (1–2 times/year)
- ☐ Occasionally (3–5 times/year)
- ☐ Regularly (more than 6 times/year)

C. III. Access to and Exposure to Financial Information

10. Main Sources of Information:

- ☐ Traditional media (TV, press, radio)
- ☐ Social media
- ☐ Advice from friends/family
- ☐ Financial advisors / banks
- ☐ Specialized websites / platforms
- ☐ Other (please specify): _____

11. Perceived Reliability of These Sources:

- ☐ Very reliable
- ☐ Fairly reliable
- ☐ Slightly reliable
- ☐ Not reliable at all

D. IV. Financial Behavior and Psychological Attitudes

12. In case of financial loss, you tend to:

- ☐ Hold the asset expecting recovery
- ☐ Sell quickly to limit losses
- ☐ Wait and seek advice before acting

13. Level of Portfolio Diversification:

- ☐ Yes, always
- ☐ Sometimes
- ☐ Rarely
- ☐ Never

14. Factors Influencing Your Decisions:

- ☐ Personal analysis
- ☐ Behavior of other investors (herding effect)
- ☐ Advice from an expert or relative
- ☐ Intuition or emotions

15. Perception of Your Financial Decisions:

- ☐ Very rational
- ☐ Rather rational
- ☐ Rather emotional
- ☐ Very emotional

E. V. Financial Knowledge and Education

16. Participation in financial training / investment management:

- ☐ Yes
- ☐ No

17. Knowledge of risk diversification:

- ☐ Yes
- ☐ No

18. Knowledge of compound interest calculation:

- ☐ Yes
- ☐ No

19. Self-assessment of financial knowledge (1 to 5):

- ☐ 1 Very low
- ☐ 2 Low
- ☐ 3 (Medium)
- ☐ 4 Good
- ☐ 5 Excellent

20. Perception of financial education level in Morocco:

- ☐ Very sufficient
- ☐ Fairly sufficient
- ☐ Insufficient
- ☐ Very insufficient

International Journal on Optimization and Applications
IJOA. Vol. 06, Issue No. 01, Year 2026, www.ijoa-journal.com

Portfolio Optimization under Non-Normal Returns: Why Higher-Order Moments Matter in the Mean–CVaR Model

Adam Lahbous
Mohammed V University in Rabat, Morocco
adam_lahbous@um5.ac.ma

***Abstract*—importance of considering higher order moments in the estimation of the CVaR, when the returns are not normally distributed. The study focuses on a portfolio of 10 stocks listed on the New York Stock Exchange. The results of the portfolio's optimization by the models mean-parametric CVaR and mean-Cornish Fisher CVaR, showed that the risk estimated by Cornish Fisher CVaR, which considers the Skewness and the Kurtosis, is higher than that evaluated by parametric CVaR, which is based only on the mean and the standard deviation in the estimation of the risk.**

***Index Terms*—Portfolio, Optimization, CVaR, Skewness, Kurtosis.**

I. Introduction

Portfolio optimization has always been a concern for researchers in financial mathematics. In this context, Harry Markowitz (1952) introduced the "mean-variance" model. This model uses the variance of portfolio returns as a measure of risk. In effect, the Markowitz model consists of minimizing variance (or standard deviation) for a given level of return or maximizing return for a given level of risk. Advances and research in the field have allowed the use of more mature indicators, such as Value at Risk (VaR) or Conditional Value at Risk (CVaR). VaR can be defined as the worst loss expected from a portfolio, for a certain confidence level (1-), over a time horizon t, assuming an adverse market scenario.

Although VaR is a popular and intuitive risk measure, it has two important limitations. First, it does not satisfy the sub-additivity property. Second, it provides no information about the magnitude of losses exceeding the VaR at level . An alternative risk measure is the Conditional Value at Risk (CVaR), which avoids both limitations. CVaR is the expected loss given that losses exceed VaR.

Based on the Markowitz model, the mean-VaR and mean-CVaR models were developed. However, these models are based on the parametric estimation of VaR and CVaR, which assumes that asset returns are normally distributed. This assumption is problematic because financial returns are often not normal. Moreover, the mean-CVaR model assumes a quadratic utility function, in the sense that investors are only interested in the first two moments (mean and variance), while in reality, preferences may extend to higher-order moments such as skewness and kurtosis.

Tobin (1958) and Rubinstein (1973) have already shown that, in choosing a portfolio, higher order moments must be considered, and that the investor's utility function is not quadratic. Arditti (1975) and Krauss and Litzenberger (1976) have shown that the investor generally seeks a strictly positive skewness, and thus the investor's utility function cannot be quadratic. Scott and Horvath (1980) showed that investors have a positive preference for odd central moments and a negative preference for even central moments. Chunhachinda et al. (1997) analyzed in detail the preference of investors for skewness.

Indeed, a negative skewness means that large negative returns occur more frequently than large positive returns. Conversely, if the skewness coefficient is positive, it indicates that large positive returns occur more frequently than large negative returns. On the other hand, kurtosis reflects the presence of extreme events: if kurtosis exceeds 3, the distribution is said to be leptokurtic; otherwise, it is said to be platykurtic.

Empirically, the normality assumption is not satisfied, which has pushed several authors to seek solutions to this problem. Yao et al. (2013) obtained an estimated calculation formula of CVaR using nonparametric loss function density estimation and formulated two nonparametric mean-CVaR portfolio selection models based on two methods of bandwidth selection. Jaman et al. (2014) replaced variance with CVaR in the mean-variance-skewness-kurtosis model and showed that CVaR is a better measure of risk than variance in portfolio optimization. Zhao et al. (2015) proposed a mean-CVaR-skewness portfolio optimization model based on the asymmetric Laplace distribution, which is appropriate to describe the leptokurtosis and asymmetry of financial assets. They

added skewness to the model and proposed a simplified model to solve the multi-objective optimization problem. Zhai et al. (2018) designed a hybrid smart algorithm to solve the mean-risk-skewness model they proposed.

In our study we will focus on the Cornish-Fisher expansion, as a solution to the case of non-normality of returns, which is commonly used by academics and practitioners in portfolio allocation and risk management. It is a simple tool to determine the quantiles of non-normal distributions. It allows us to consider skewness and kurtosis, through a formula in which higher-order moment terms are explicitly included (Amédée-Manesme et al., 2019).

The current challenge is as follows: Does ignoring higher-order moments in mean-CVaR optimization lead to a significant underestimation of portfolio risk, and how important is it to incorporate skewness and kurtosis for more accurate risk assessment?

The following hypotheses are proposed based on the literature review:

- H0: The parametric CVaR underestimates the risk, and the consideration of higher-order moments is important in the estimation of CVaR.
- H1: The parametric CVaR does not underestimate the risk, and the consideration of higher-order moments is not important in the estimation of CVaR.

II. Methodology & Data

The methodology of this study is based on the optimization of a portfolio composed of a set of stocks listed on the New York Stock Exchange, first using the mean-parametric CVaR model, and secondly using the mean-Cornish-Fisher CVaR model, in order to compare the results obtained.

A. Portfolio Stocks Sample

We have chosen to build this portfolio from 10 stocks, listed on the New York Stock Exchange, from different sectors to have a well-diversified portfolio.

TABLE I
Portfolio stocks sample

Companies	Ticker symbol	Sectors
ExxonMobil	NYSE:XOM	Energy
DuPont	NYSE:DD	Materials
Boeing	NYSE:BA	Industrials
Nike	NYSE:NKE	Consumer Discretionary
Coca-Cola	NYSE:KO	Consumer Staples
Pfizer	NYSE:PFE	Health Care
JPMorgan Chase	NYSE:JPM	Financials
IBM	NYSE:IBM	Information Technology
Verizon Communications	NYSE:VZ	Communication Services
Southern Company	NYSE:SO	Utilities

Source: Author's own elaboration

B. Data

The analysis is based on a series of data concerning daily stock prices during the period from 03/01/2012 to 30/12/2022 for each company.

We obtain the return at time t for each stock using the formula:

$$R_t = \ln\left(\frac{P_t}{P_{t-1}}\right) \tag{II.1}$$

Where P_t : the stock's value on the market at t;
And P_{t-1}: the stock's value on the market at $t-1$.

C. CVaR in Portfolio Optimization

The parametric CVaR and Cornish-Fisher CVaR optimization programs will be presented in this section.

1) Parametric CVaR Optimization Program: The parametric method assumes that equity returns follow a normal distribution, and that the $CVaR_\alpha$ of a portfolio P, at a confidence level of $1-\alpha$, can be estimated using the portfolio's mean return μ_p and standard deviation σ_p. According to Huang (2000), the $CVaR_\alpha$ equation can be written as the following density formula:

$$CVaR_\alpha = \mu_p + Z_\alpha \sigma_p \tag{II.2}$$

Where:

$$Z_\alpha = \frac{-e^{-1/2z_\alpha^2}}{\alpha\sqrt{2\pi}} \tag{II.3}$$

z_α : the order α quantile of the standard normal distribution

Thus, the CVaR optimization program for a portfolio is written as follows:

$$\min_X \left| X^T\mu + Z_\alpha \sqrt{X^T M X} \right| \tag{II.4}$$

subject to:

$$X^T\mu = E^* \tag{II.5}$$
$$X^T\mathbf{1} = 1 \tag{II.6}$$
$$X_i \geq 0 \tag{II.7}$$

Where:
μ : expected returns vector of stocks;
X : stock proportions vector in portfolio;
$\mathbf{1} = (1, \ldots, 1)$: unit vector of N dimension;
M : variance-covariance matrix;
E^* : (constant) given expected return of the portfolio.

2) Cornish-Fisher CVaR Optimization Program: When asset returns do not follow a normal distribution, the parametric CVaR would underestimate risk. Thus, adjustments for skewness and kurtosis were implemented in the variance-covariance model of CVaR. This approximation, based on the Taylor series, uses the moments of the distribution that deviate from the normal to calculate its percentiles. According to Maillard (2012), the Cornish-Fisher expansion is used to fit the Z_α of the traditional CVaR, when asset returns do not follow a normal distribution, such that:

$$W_\alpha = \frac{-e^{-1/2z_\alpha^2}}{\alpha\sqrt{2\pi}}\left[1 + Z_\alpha\left(\frac{S}{6}\right) + (1-2Z_\alpha^2)\frac{S^2}{36} + (-1+Z_\alpha^2)\left(\frac{K}{24}\right)\right] \tag{II.8}$$

Hence, the Cornish-Fisher CVaR optimization program for a portfolio is written as follows:

$$\min_X \left| X^T\mu + W_\alpha\sqrt{X^TMX} \right| \tag{II.9}$$

subject to:

$$X^T\mu = E^* \tag{II.10}$$
$$X^T\mathbf{1} = 1 \tag{II.11}$$
$$X_i \geq 0 \tag{II.12}$$

III. Results

Before presenting the results of the portfolio optimization, it is important to verify stocks returns normality.

A. Verification of the normality of returns

The results of the Jarque-bera normality test are presented as follows in Table 2.

TABLE II
NORMALITY TEST OF STOCKS RETURNS

Companies	Skewness	Kurtosis	Test Value	Proba
NYSE:XOM	-0.165409	11.01714	7422.934	0.0000
NYSE:DD	0.103624	10.45335	6409.691	0.0000
NYSE:BA	-0.543452	24.44079	53136.63	0.0000
NYSE:NKE	0.287194	14.03466	14076.38	0.0000
NYSE:KO	-0.858887	13.41912	12856.03	0.0000
NYSE:PFE	0.153746	8.711631	3772.029	0.0000
NYSE:JPM	-0.101892	15.65106	18457.13	0.0000
NYSE:IBM	-0.789558	13.73453	13572.57	0.0000
NYSE:VZ	-0.066413	7.447287	2282.314	0.0000
NYSE:SO	0.203200	28.68250	76064.34	0.0000

According to the Jarque–Bera test, all stock return distributions are non-normal, particularly since their kurtosis values are much higher than the normal benchmark of 3, indicating that they are leptokurtic.

B. Results of the Portfolio Optimization

The study results are summarized in the following tables: Table 3 reports the portfolios optimized using the mean–parametric CVaR model, while Table 4 reports those optimized using the mean–Cornish Fisher CVaR model.

According to Tables 3 and 4 below, the results of the portfolio optimization reveal distinct allocation patterns under the two models.

According to Table 3, all portfolios do not contain NYSE:DD and NYSE:BA after optimization using the mean-parametric CVaR model. In portfolio 1, corresponding to the minimal CVaR (0.018435414), NYSE:KO, NYSE:PFE, and NYSE:VZ have the highest weights, representing more than 70% of the portfolio. On the other hand, portfolio 9, associated with the highest mean return, consists only of NYSE:NKE.

According to Table 4, all portfolios do not contain NYSE:XOM, NYSE:DD, NYSE:BA, NYSE:IBM, and NYSE:SO after optimization using the mean-Cornish Fisher CVaR model. In portfolio 1, corresponding to the minimal CVaR (0.028681269), NYSE:PFE and NYSE:VZ dominate the allocation, accounting for more than 96% of the portfolio. In contrast, portfolio 9, associated with the highest mean return, consists exclusively of NYSE:NKE.

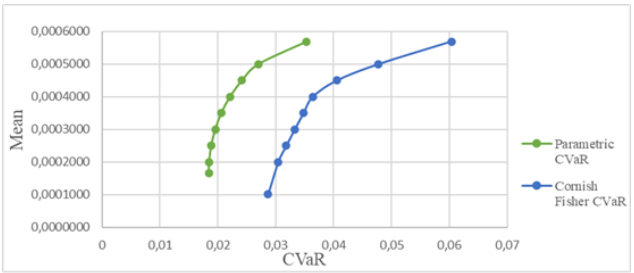

Fig. 1. Efficient Frontiers of Mean-Parametric CVaR and Mean-Cornish Fisher CvaR

Figure 1 shows that mean-Cornish Fisher CVaR efficient frontier is under mean-parametric CVaR efficient frontier.

Tables 3 and 4 and figure 1 prove that for a given mean return level, Cornish Fisher CVaR is higher than parametric CVaR. For example, portfolio 2 which has a mean of 0.0002, parametric CVaR (0.018501662) is lower than Cornish-Fisher CVaR (0.030428659).

TABLE III
PORTFOLIOS OPTIMIZED BY THE MEAN-PARAMETRIC CVAR MODEL

Portfolio	**1 (CVaR min)**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9 (Mean Max)**
Mean	0.0001664	0.0002000	0.0002500	0.0003000	0.0003500	0.0004000	0.0004500	0.0005000	0.0005695
St Dev	0.0090181	0.0090665	0.0092654	0.0096284	0.0101614	0.0109096	0.0119368	0.0133321	0.0174024
Skewness	-0.4391531	-0.4621051	-0.5085350	-0.5400704	-0.5255992	-0.4633764	-0.3705264	-0.2543904	0.2873498
Kurtosis	12.777005	12.741812	13.056199	13.592985	13.428956	11.527408	10.111049	10.051639	11.056799
α	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
z	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.644853627
Z	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.062712808
Parametric CVaR	0.018435414	0.018501662	0.018861885	0.019560688	0.020610012	0.022103441	0.024172179	0.027000226	0.035326583
Stock weight									
NYSE:XOM	0.037405475	0.028979195	0	0	0	0	0	0	0
NYSE:DD	0	0	0	0	0	0	0	0	0
NYSE:BA	0	0	0	0	0	0	0	0	0
NYSE:NKE	0.062258212	0.095200294	0.144226312	0.180334433	0.225421249	0.305232925	0.392574105	0.525654447	1
NYSE:KO	0.250770238	0.267560710	0.276049177	0.265651692	0.260418381	0.184189056	0.067877004	0	0
NYSE:PFE	0.177078497	0.203771064	0.236452721	0.258066357	0.286534889	0.304056398	0.313913709	0.185920366	0
NYSE:JPM	0	0.000176402	0.024101268	0.077302719	0.112799923	0.166421674	0.225635181	0.288425182	0
NYSE:IBM	0.042165190	0.007528701	0	0	0	0	0	0	0
NYSE:VZ	0.307761410	0.273097465	0.197099243	0.105453686	0.009743082	0	0	0	0
NYSE:SO	0.122560977	0.123686169	0.122071281	0.113191114	0.105082476	0.040099959	0	0	0

Source: Author's own elaboration

TABLE IV
PORTFOLIOS OPTIMIZED BY THE MEAN-CORNISH FISHER CVAR MODEL

Portfolio	**1 (CVaR min)**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9 (Mean Max)**
Mean	0.00010	0.00020	0.00025	0.00030	0.00035	0.00040	0.00045	0.00050	0.00057
St Dev	0.009979119	0.010126544	0.010296044	0.010697836	0.011303962	0.012024818	0.012493592	0.013517686	0.017402384
Skewness	-0.01189906	-0.07075662	-0.15731496	-0.20395444	-0.2090435	-0.1951683	-0.22510796	-0.14953065	0.287349798
Kurtosis	5.558918547	6.297584089	6.605835616	6.615974258	6.397023071	6.11158038	7.531875864	9.730223801	11.05679914
α	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05	0.05
z	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.64485363	-1.644853627
Z	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.06271281	-2.062712808
W	-2.8842612	-3.02459169	-3.113734	-3.13733512	-3.10758663	-3.05931636	-3.28126066	-3.56792512	-3.500117889
CVaR-CF	0.028681269	0.030428659	0.031809142	0.033262696	0.03477804	0.036387724	0.040544733	0.047730091	0.060340858
Stock weight									
NYSE:XOM	0	0	0	0	0	0	0	0	0
NYSE:DD	0	0	0	0	0	0	0	0	0
NYSE:BA	0	0	0	0	0	0	0	0	0
NYSE:NKE	0	0.068811108	0.133399583	0.191957865	0.245742614	0.304841538	0.509859218	0.621894201	1
NYSE:KO	0.038783436	0	0	0	0	0	0	0	0
NYSE:PFE	0.291076944	0.497815256	0.537429637	0.587548339	0.645981746	0.695158466	0.490140774	0.236376891	0
NYSE:JPM	0	0	0	0	0	0	0	0.141728905	0
NYSE:IBM	0	0	0	0	0	0	0	0	0
NYSE:VZ	0.670139621	0.433373637	0.329170778	0.220493802	0.108275649	0	0	0	0
NYSE:SO	0	0	0	0	0	0	0	0	0

Source: Author's own elaboration

IV. DISCUSSION

The results of our study are consistent with the findings of Maillard (2012), who showed that parametric CVaR tends to underestimate risk when asset returns do not follow a normal distribution. In contrast, the Cornish–Fisher expansion overcomes this limitation, as it allows the adjustment of Z_α by incorporating skewness and kurtosis.

Our results are also consistent with the contributions of Tobin (1958) and Rubinstein (1973), who demonstrated that higher-order moments should be considered in portfolio selection. Furthermore, they align with the findings of Jaman et al. (2014), Zhao et al. (2015), and Zhai et al. (2018), who proposed optimization models based on CVaR estimation that account for higher-order moments and emphasized their importance.

V. Conclusion

In conclusion, when returns are not normally distributed, it is essential to consider skewness and kurtosis in risk estimation. The Cornish–Fisher expansion provides an effective framework to incorporate these higher-order moments into the estimation of CVaR.

Moreover, relying solely on parametric CVaR, which is based only on the mean and standard deviation, when returns do not follow a Gaussian distribution, may lead investors to underestimate the true level of risk.

References

[1] C. Amédée-Manesme, F. Barthélémy, and D. Maillard, "Computation of the corrected Cornish–Fisher expansion using the response surface methodology: application to VaR and CVaR," *Annals of Operations Research*, vol. 281, no. 1–2, pp. 423–453, 2019.

[2] F. D. Arditti, "Skewness and Investors' Decisions: A Reply," *Journal of Financial and Quantitative Analysis*, vol. 10, no. 1, pp. 173–176, 1975.

[3] P. Chunhachinda, K. Dandapani, S. Hamid, and A. Prakash, "Portfolio selection and skewness: Evidence from international stock markets," *Journal of Banking and Finance*, vol. 21, no. 2, pp. 143–167, 1997.

[4] H. M. Markowitz, "Portfolio selection," *Journal of Finance*, vol. 7, pp. 77–91, 1952.

[5] S. H. Jaaman, W. N. Lam, and Z. Isa, "A new higher moment portfolio optimisation model with conditional value at risk," *International Journal of Operational Research*, 2014.

[6] A. Kraus and R. H. Litzenberger, "Skewness preference and the valuation of risk assets," *Journal of Finance*, vol. 31, no. 4, pp. 1085–1100, 1976.

[7] D. Maillard, "A User's Guide to the Cornish Fisher Expansion," SSRN, 2012.

[8] M. P. Rubinstein, "A Mean-Variance Synthesis of Corporate Financial Theory," *Journal of Finance*, vol. 28, no. 1, pp. 167–181, 1973.

[9] R. A. Scott and P. A. Horvath, "On the Direction of Preference for Moments of Higher Order Than the Variance," *Journal of Finance*, vol. 35, no. 4, pp. 915–919, 1980.

[10] J. Tobin, "Liquidity Preference as Behavior Towards Risk," *The Review of Economic Studies*, vol. 25, no. 2, pp. 65, 1958.

[11] H. Yao, Z. Li, and Y. Lai, "Mean–CVaR portfolio selection: A nonparametric estimation framework," *Computers & Operations Research*, vol. 40, no. 4, pp. 1014–1022, 2013.

[12] J. Zhai, M. Bai, and H. Wu, "Mean-risk-skewness models for portfolio optimization based on uncertain measure," *Optimization*, vol. 67, no. 5, pp. 701–714, 2018.

[13] S. Zhao, Q. Lu, L. Han, Y. Liu, and F. Hu, "A mean-CVaR-skewness portfolio optimization model based on asymmetric Laplace distribution," *Annals of Operations Research*, vol. 226, no. 1, pp. 727–739, 2015.

AUTHORS INDEX

ABOUT THE EDITOR IN CHIEF

Prof. Dr. Hanaa Hachimi, Ph.D in Applied Mathematics & Computer Science and a Ph.D in Mechanics & Systems Reliability, I am Full Professor at National School of Applied Sciences, Ibn Tofail University of Kenitra, Morocco. President of the Moroccan Society of Engineering Sciences and Technology (MSEST). I am the Editor in Chief of "The International Journal on Optimization and Applications" (IJOA). I am Director of the Systems Engineering Laboratory (LGS) and IEEE Senior Member, precisely I am affiliated at the Big Data, Optimization, Service and Security (BOSS) team. I am Scientific Expert-Evaluator at the CNRST. I am Lecture and Keynote Speaker of the courses: Optimization & Operational Research, Graph Theory, Statistics, Probability, Cryptography, Reliability and Scientific Computing. I am Member of the Moroccan Society of Applied Mathematics (SM2A). I'm the General Chair of "The International Conference on Optimization and Applications" (ICOA) & the International Competition of Innovation (Let's Challenge). Lions Club Member and UNESCO UIT-Chair Member.

Previously Associate Professor & ex-Secretary General of Sultan Moulay Slimane University in Beni Mellal.

for more information, visit our website : http://www.hanaahachimi.com/

ISSN : 2737-8314

IJOA.

International Journal on Optimization and Applications

ISSN : 2737-8314