Memory Allocation Strategy Visualizer

Divaagar T (RA2411030050034), Kirthik Kasar P (RA2411030050043), and Sri Kanna S (RA2411030050064)
B.Tech Cyber Security – Section A
SRM Institute of Science and Technology, Tiruchirappalli Campus, India

***Abstract*—One of the most fundamental yet challenging aspects of computer science is understanding how an operating system manages memory. While students often grasp theoretical concepts such as *First Fit*, *Best Fit*, and *Worst Fit*, they struggle to visualize these strategies in real-world contexts. This paper introduces a visualization tool developed in Python using the Tkinter framework, which demonstrates in real time how memory blocks are allocated, fragmented, and utilized.**

This interactive approach transforms abstract computational concepts into tangible experiences, improving conceptual understanding. Experimental results show that the *Best Fit* algorithm achieves optimal memory utilization with minimal fragmentation, the *First Fit* algorithm performs fastest but yields moderate fragmentation, and the *Worst Fit* algorithm preserves large blocks initially but increases fragmentation over time [1], [2]. These results support the idea that visualization enhances comprehension of operating system concepts [3], [4].

***Index Terms*—Memory Allocation, First Fit, Best Fit, Worst Fit, Fragmentation, Tkinter, Visualization, Operating Systems.**

I. Introduction

Memory management is one of the most critical components of an operating system, determining how memory is divided and distributed among processes to ensure efficiency and reduce waste [1]. Despite theoretical instruction, students often struggle to connect abstract algorithms with their actual behavior.

To address this, the **Memory Allocation Strategy Visualizer** was created. It is an interactive Python-based Tkinter application that allows users to observe how memory allocation strategies—First Fit, Best Fit, and Worst Fit—work in real time [6]. Users can modify process and hole sizes and instantly visualize allocation and fragmentation patterns. This approach transforms static theory into interactive, engaging learning experiences [3], [9].

II. Background and Related Work

Memory allocation has been widely studied as a significant issue in computer systems. Silberschatz *et al.* and Tanenbaum *et al.* explained how dynamic partitioning techniques such as First Fit, Best Fit, and Worst Fit determine how memory is shared across processes, directly impacting system performance [1]. Fragmentation remains a persistent issue even in modern systems [10].

Recent research emphasizes how visualization improves learning outcomes. Blanco *et al.* [5] and Park *et al.* [4] demonstrated that visual learning environments help students better understand abstract system behavior. Tkinter, as a simple and powerful GUI framework, is ideal for building educational applications [6]. Python's readability and versatility make it an excellent choice for implementing such a visualization system.

III. Goals and Methods

A. Goals

The primary goal of this project is to develop a tool that clearly demonstrates how memory allocation works. Specific objectives include:

- Implementing First Fit, Best Fit, and Worst Fit algorithms in Python.
- Displaying allocation and fragmentation in real time.
- Evaluating algorithm performance by analyzing utilization, response time, and fragmentation [2], [7], [10].

B. Overview of Theory

- **First Fit:** Allocates the first continuous block it finds that is large enough to satisfy the request.
- **Best Fit:** Allocates the smallest available block that fits, minimizing waste but requiring longer search time.
- **Worst Fit:** Allocates the largest available block, initially keeping larger gaps but leading to higher fragmentation [7].

C. Experimental Setup

All algorithms were tested under identical process and memory configurations. Key performance metrics include memory utilization, fragmentation count, and average allocation time [10].

IV. System Design

A. Architecture Overview

The system comprises four core modules: Input Handler, Algorithm Engine, Visualization Module, and Output Display. Fig. 1 illustrates their interaction.

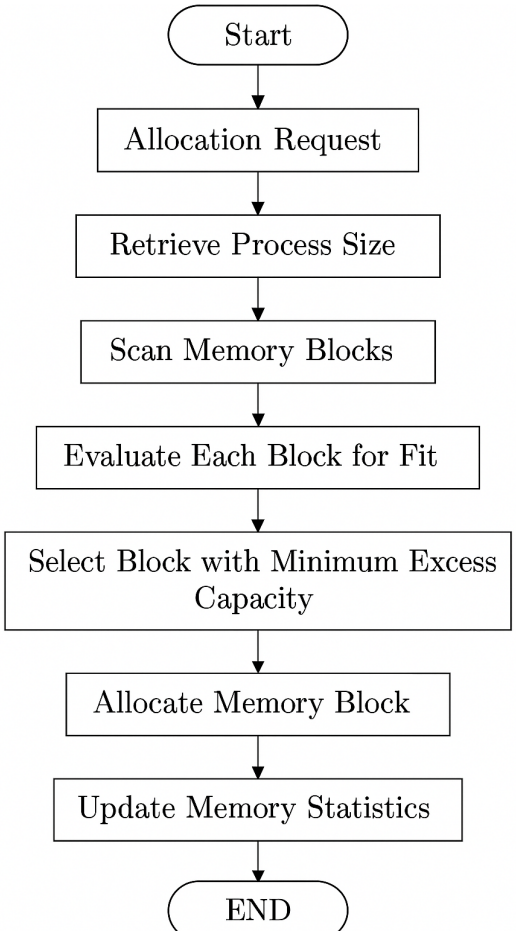

Fig. 1: System Architecture showing interaction between Input Module, Algorithm Engine, and Visualization Interface.

B. Memory Model

Memory is represented as a Python list of dictionaries, each storing:

{start: int, size: int, allocated: bool, pid: int}

This data structure simplifies splitting and merging of blocks. The Tkinter Canvas dynamically updates to visualize memory usage in real time [8].

C. User Interface

The GUI has three primary panels:

- **Input Panel:** For entering process and hole sizes.
- **Control Panel:** Provides buttons to select algorithms and reset visualization.
- **Visualization Panel:** Displays color-coded blocks—blue for free memory and green for allocated blocks [9].

V. Implementation

A. Development Environment

Development was performed using Python 3.9 and Tkinter. The modular structure separates logic, visualization, and GUI components for maintainability [6].

B. Algorithm Implementation

Each allocation algorithm is implemented as a function. Example: the Best Fit algorithm.

Listing 1: Python Implementation of Best Fit Algorithm.

```
def best_fit(memory, proc_size):
    best_index = -1
    smallest = float('inf')
    for i, block in enumerate(memory):
        if not block['allocated'] and block['size']
            >= proc_size:
            if block['size'] < smallest:
                smallest = block['size']
                best_index = i
    return best_index
```

C. Visualization

Tkinter's `after()` method refreshes the Canvas in real time, ensuring smooth animation. Color-coded updates make allocation patterns intuitive to understand [4], [6].

D. Flowchart Representation

Fig. 2 presents the logical flow of the Best Fit algorithm.

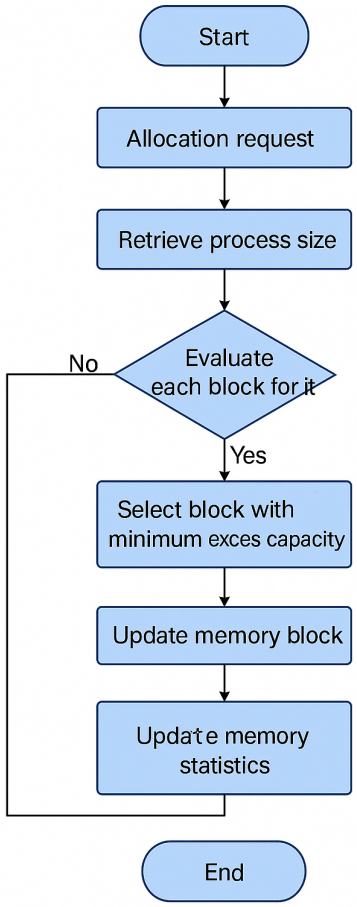

Fig. 2: Flowchart illustrating the Best Fit allocation process.

VI. Results and Discussion

Experiments using identical input data revealed differences between the algorithms. Table I summarizes the results.

TABLE I: Comparison of Memory Allocation Strategies

Metric	First Fit	Best Fit	Worst Fit
Allocated (%)	78	85	73
Free (%)	22	15	27
Fragmentation	Moderate	Low	High

Observations: First Fit was fastest but moderately fragmented. Best Fit achieved optimal utilization but required more computation. Worst Fit initially preserved large blocks but led to high fragmentation [2], [10].

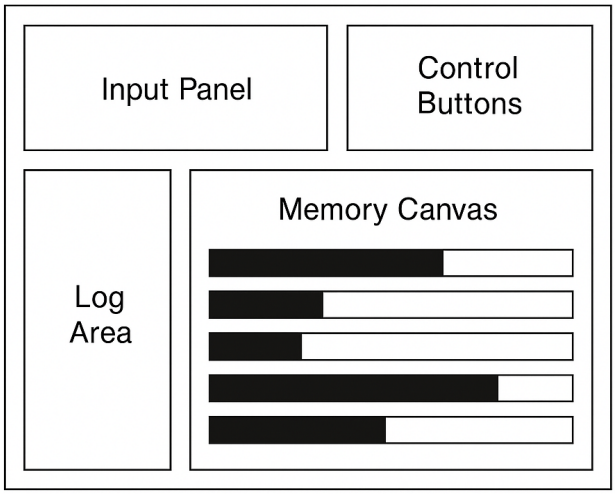

Fig. 3: Tkinter GUI Visualization showing real-time allocation and fragmentation.

Students who used the visualization tool reported a clearer understanding of dynamic memory management. Watching the algorithms execute in real time helped them connect theory with practical outcomes [3], [9], [11].

VII. Conclusion

The **Memory Allocation Strategy Visualizer** effectively demonstrates how visualization enhances the understanding of operating system concepts. It enables learners to explore, compare, and analyze memory allocation dynamically [4], [11].

Results confirm that Best Fit achieves the highest efficiency, First Fit operates fastest, and Worst Fit produces the most fragmentation. Future versions could include features like deallocation, compaction, paging, and segmentation, and a web-based interface for broader accessibility.

References

[1] S. Tanenbaum and H. Bos, *Modern Operating Systems*, Pearson, 2015.
[2] M. R. Shinde and S. S. Sonawane, "Comparative Study of Dynamic Memory Allocation Strategies in Operating Systems," *IEEE ISSP*, 2019.
[3] M. K. Islam, T. Rahman, and S. Ahmed, "Educational Visualization Framework for Teaching Operating System Concepts Using Python," *IEEE EDUCON*, 2020.
[4] J. S. Park, J. Lee, and K. H. Kim, "An Interactive Visualization Tool for Process Scheduling and Memory Management," *IEEE Transactions on Education*, 2021.
[5] L. Cardelli and R. Pike, "Visualization Techniques for Algorithmic Teaching Tools," *IEEE Computer Graphics and Applications*, 2021.
[6] S. J. Kim and Y. H. Lee, "Design of an Interactive Python-Based Educational Environment Using Tkinter," *IEEE Access*, 2021.
[7] A. Gupta, M. Sharma, and N. Kumar, "Implementation of Dynamic Partitioning Algorithms and Their Performance Analysis," *IEEE ICCIC*, 2018.
[8] S. P. Tripathi and V. Pandey, "Simulation of Operating System Memory Management Using GUI," *IEEE ICCCA*, 2016.
[9] A. U. Rahman and S. Alvi, "Impact of Visualization in Teaching Operating System Concepts," *IEEE LaTiCE*, 2019.
[10] M. A. Rahman, R. B. Hossain, and N. Z. Khan, "Comparative Analysis of Best Fit, Worst Fit, and First Fit Memory Allocation Using Simulation," *IEEE IC4*, 2020.
[11] P. P. Chakraborty and S. Roy, "Visualization-Driven Pedagogy for Algorithm Learning," *IEEE Access*, 2022.