Solving Single-Machine, Parallel-Machines and Flowshop Problems by using Dispatching Rules, Local Search and Threshold Accepting Methods

IE 439
FALL 2024-2025

Hilal Aslan

Görkem Görgeç

Mehmet Mert Kılıçarslan

TABLE OF CONTENTS

INTRODUCTION

Machine scheduling is a vital area of combinatorial optimization with applications in manufacturing, logistics, and various service industries. This project focuses on three core machine environments: Single Machine, Parallel Machines, and Flow Shop, each representing distinct scheduling complexities and challenges.

The Single Machine environment is the simplest and most fundamental setting, involving a single resource where jobs are processed sequentially. This model serves as a foundational framework for understanding more complex systems. The primary objective in this environment is often to optimize criteria such as total completion time, tardiness, or lateness. Despite its simplicity, the single-machine problem has significant theoretical and practical importance in scenarios with limited resources, such as small-scale production settings. The Parallel Machines environment extends the single-machine model by introducing multiple identical machines, adding complexity to scheduling decisions. In this setup, the challenge lies in determining both the assignment of jobs to machines and the sequence of jobs on each machine. The additional flexibility provided by parallel machines must be carefully managed to balance workloads and minimize overall completion time or delays. This environment is highly applicable in industries with multiple production lines or parallel processing units. The Flow Shop environment, in contrast, represents a multi-stage production system where jobs must follow a predefined sequence of operations across multiple machines. This model is commonly seen in industries like automotive assembly or chemical processing, where each job undergoes a series of sequential processes. The complexity in flow shop scheduling arises from the need to synchronize operations and minimize downtime between stages while optimizing objectives such as makespan or lateness[1].

These three machine scheduling environments highlight the breadth of challenges and solutions in optimizing resource utilization and job sequencing. Each environment offers unique insights into scheduling dynamics, providing a comprehensive foundation for addressing practical scheduling problems in diverse operational contexts.

Objective Criterias

Total completion time, total tardiness and total number of late jobs each have their own weighted objective criteria, which are crucial for prioritizing and assigning importance to jobs within a scheduling system. Each of these objective functions aligns with specific operational goals and reflects distinct priorities and constraints inherent to the system. The selection of the appropriate objective function depends on the desired outcomes and the contextual requirements of the operational environment. The criteria associated with each objective function under these constraints are outlined as follows.

Makespan (C_{max}):

The total time required to complete all jobs, from the start of the first job to the end of the last job. Ensures the shortest possible schedule length, often used in flow shop or parallel machine environments.

Total Completion Time ($\sum C_j$):

The sum of the completion times of all jobs. Reduces overall waiting time for all jobs, often improving system efficiency.

Total Tardiness ($\sum T_j$):

The sum of the tardiness of all jobs, where tardiness is the delay beyond a job's due date. Focuses on reducing late deliveries or penalties for overdue jobs.

Total number of Late Jobs ($\sum U_j$):

The total number of jobs completed after their due dates. Important for systems where the number of late deliveries matters more than their exact tardiness.

Maximum Lateness (L_{max}):

The maximum lateness among all jobs. Ensures no job is excessively late.

Application of the Algorithm

1. Single Machine Problem

In this study, it is aimed to solve scheduling problems by utilizing three different machine environments(α), incorporating a variety of job characteristics(β), and defining optimality criteria(γ) . For this purpose, a graphical user interface (GUI) has been designed. Upon running the code, the user is presented with an interface where an Excel input is expected to be uploaded, as shown in Figure 1. Once the input is successfully uploaded, a confirmation message is displayed to inform the user of the successful completion of this step.

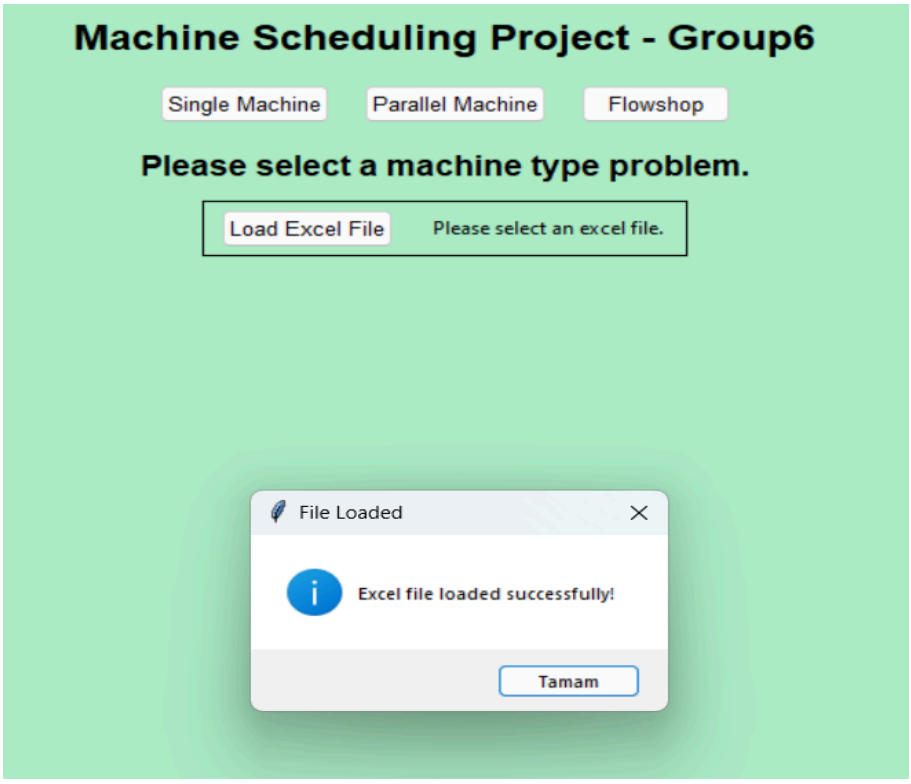


Figure 1. GUI for Scheduling Problem with File Upload Notification

In the context of the single machine scheduling problem, the graphical user interface (GUI) provides users with the ability to select from five dispatching rules commonly used for job scheduling, as shown in Figure 2.1 These rules help determine the sequence in which jobs are processed on the single machine, based on specific criteria designed to optimize performance or meet operational goals. The rules and their respective functionality are detailed as follows.

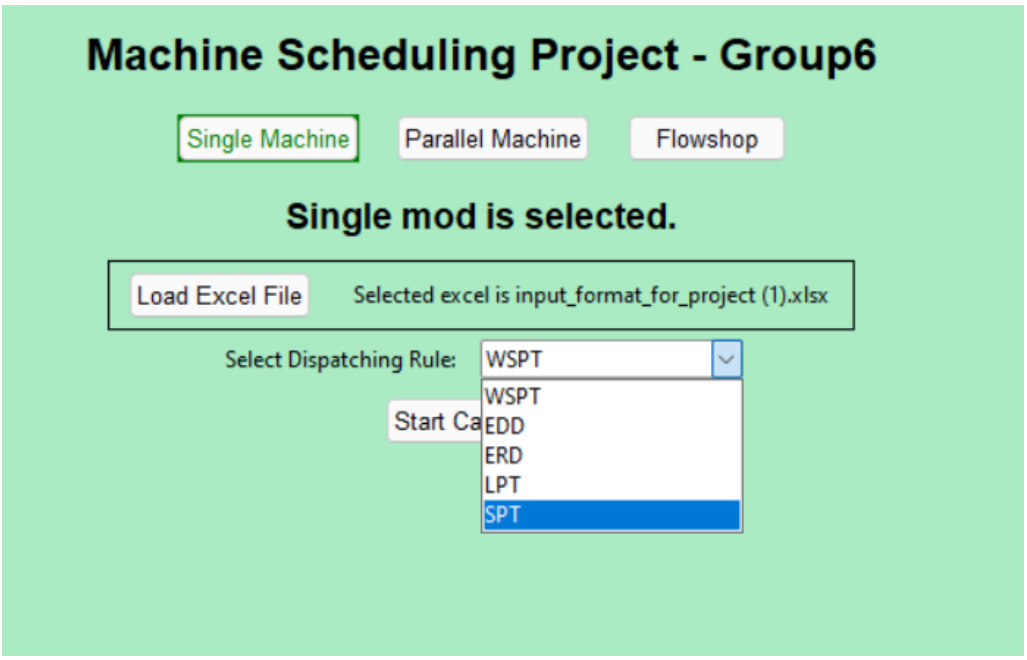
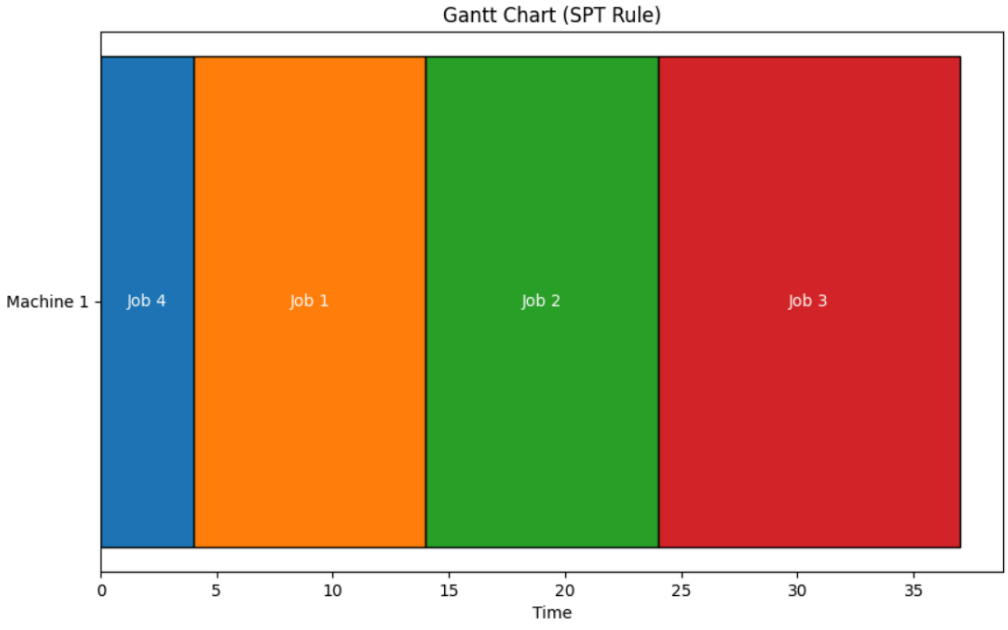


Figure 2.1 GUI for Single Machine Scheduling Problem with Dispatching Rule Selection

Shortest Processing Time (SPT):

The results obtained for the single machine environment using the Shortest Processing Time (SPT) rule, as shown in Figure 2.2, demonstrate the scheduling outcomes aimed at minimizing makespan. The Gantt chart illustrates the sequence of jobs processed on the machine, where jobs are arranged in ascending order of their processing times. Key performance metrics are provided, including a makespan of 37 units, a total completion time of 79 units, and a total tardiness of 68 units. Additional measures, such as total weighted completion, lateness, and weighted tardiness, further emphasize the effectiveness of the selected dispatching rule in achieving the desired scheduling objectives. The sequence generated by the SPT rule, shown as "4 1 2 3," reflects the optimal order based on the criterion used.

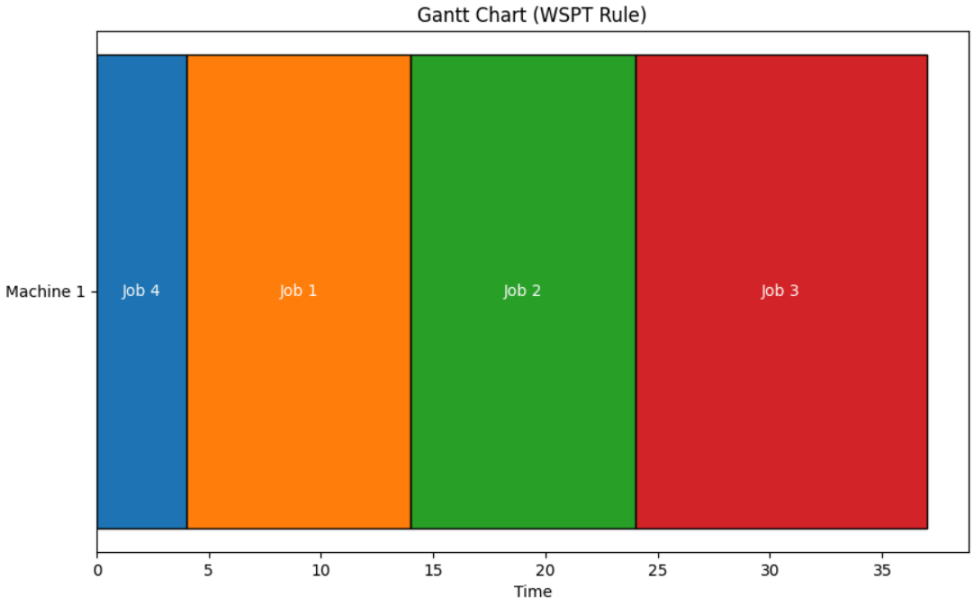


Objectives	Values
Makespan	37
Total completion time	79
Total tardiness	68
Total weighted completion	569
Total lateness (Uj)	3
Total weighted lateness (WjUj)	27
Total weighted tardiness (WjTj)	440
SPT Sequence	4 1 2 3

Figure 2.2 Minimizing Makespan for Single Machine with SPT Rule

Weighted Shortest Processing Time (WSPT):

In the Single Machine environment, the Weighted Shortest Processing Time (WSPT) rule prioritizes jobs based on the ratio W_j / p_j, where W_j is the job's weight and p_j is its processing time. Jobs are scheduled in decreasing order of this ratio, ensuring that more important jobs with shorter processing times are prioritized. As shown in Figure 2.3, this method effectively balances job importance and efficiency, minimizing weighted tardiness and lateness.

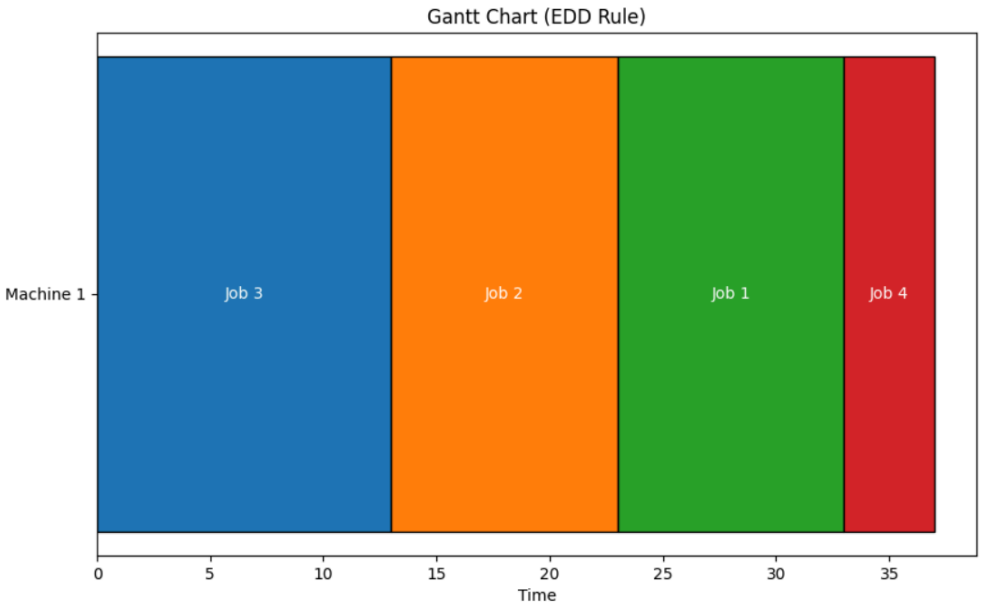


Objectives	Values
Makespan	37
Total completion time	79
Total tardiness	68
Total weighted completion	569
Total lateness (Uj)	3
Total weighted lateness (WjUj)	27
Total weighted tardiness (WjTj)	440
WSPT Sequence	4 1 2 3

Figure 2.3 Minimizing Makespan for Single Machine with WSPT Rule

Earliest Due Date (EDD):

In the Single Machine environment, the Earliest Due Date (EDD) rule schedules jobs in increasing order of their due dates, prioritizing jobs with earlier deadlines. This approach aims to minimize lateness and tardiness by ensuring jobs with tight deadlines are completed first. As shown in Figure 2.4, the EDD rule organizes job priorities effectively, reducing delays and improving on-time performance in the scheduling process.

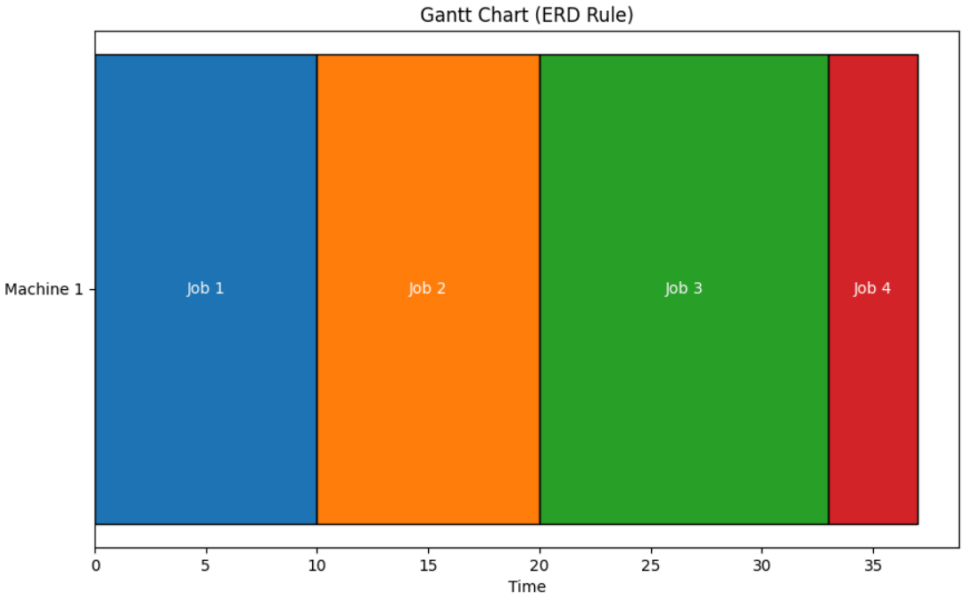


Objectives	Values
Makespan	37
Total completion time	106
Total tardiness	87
Total weighted completion	1195
Total lateness (Uj)	4
Total weighted lateness (WjUj)	39
Total weighted tardiness (WjTj)	970
EDD Sequence	3 2 1 4

Figure 2.4 Minimizing Makespan for Single Machine with EDD Rule

Earliest Release Date (ERD):

In the Single Machine environment, the Earliest Release Date (ERD) rule schedules jobs in increasing order of their release dates. This ensures that jobs which become available earlier are prioritized and processed first.This approach minimizes idle machine time and ensures an efficient allocation of resources. As shown in Figure 2.5, the ERD rule effectively organizes the job sequence, prioritizing availability to improve overall scheduling performance.

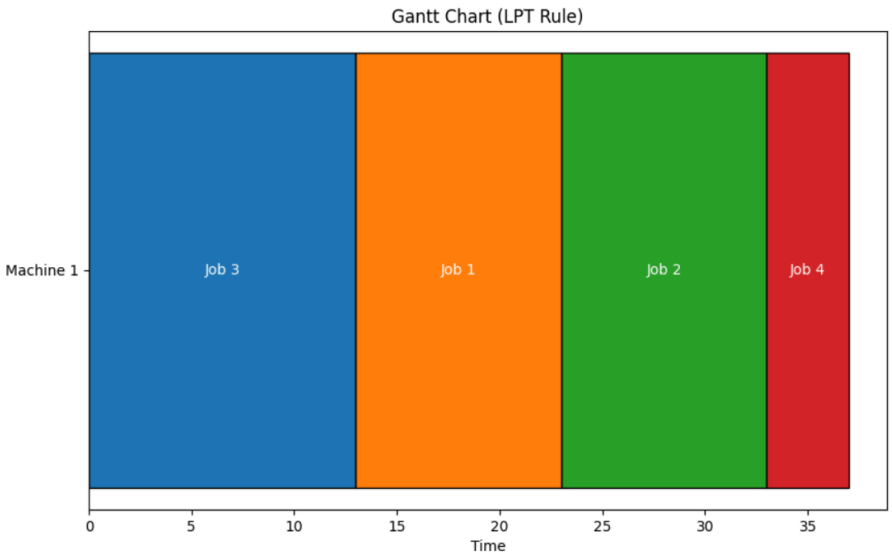


Objectives	Values
Makespan	37
Total completion time	100
Total tardiness	81
Total weighted completion	857
Total lateness (Uj)	4
Total weighted lateness (WjUj)	39
Total weighted tardiness (WjTj)	632
ERD Sequence	1 2 3 4

Figure 2.5 Minimizing Makespan for Single Machine with ERD Rule

Longest Processing Time (LPT):

In the Single Machine environment, the Longest Processing Time (LPT) rule schedules jobs in decreasing order of their processing times. This method prioritizes longer jobs first, which helps balance resource utilization by ensuring that larger tasks are addressed early in the schedule. The approach reduces the risk of leaving lengthy jobs unprocessed near the end of the scheduling process, thereby improving overall efficiency. As shown in Figure 2.6, the LPT rule organizes jobs effectively, demonstrating its usefulness in scenarios where workload balancing is a key objective.



Objectives	Values
Makespan	37
Total completion time	106
Total tardiness	87
Total weighted completion	1175
Total lateness (Uj)	4
Total weighted lateness (WjUj)	39
Total weighted tardiness (WjTj)	950
LPT Sequence	3 1 2 4

Figure 2.6 Minimizing Makespan for Single Machine with LPT Rule

2. Parallel Machine Problem

In the algorithm designed for the identical parallel machines problem, two primary methods are utilized for obtaining solutions. The problem can be addressed either through the application of dispatching rules or by employing optimization tools, specifically Local Search and Threshold Accepting methods. The code is structured to allow the selection of the desired number of identical parallel machines, providing flexibility in configuring the scheduling environment.

In the GUI for the parallel machines environment, as shown in Figure 3, users configure and solve scheduling problems step by step. The process begins by loading an Excel input file, which is then acknowledged by the system. Users specify the number of identical parallel machines and select a dispatching rule and a solving method, such as Local Search or Threshold Accepting, from the provided menus. Once these parameters are set, the "Start Calculation" button is used to execute the scheduling process, generating optimized job assignments and performance metrics based on the selected configurations.

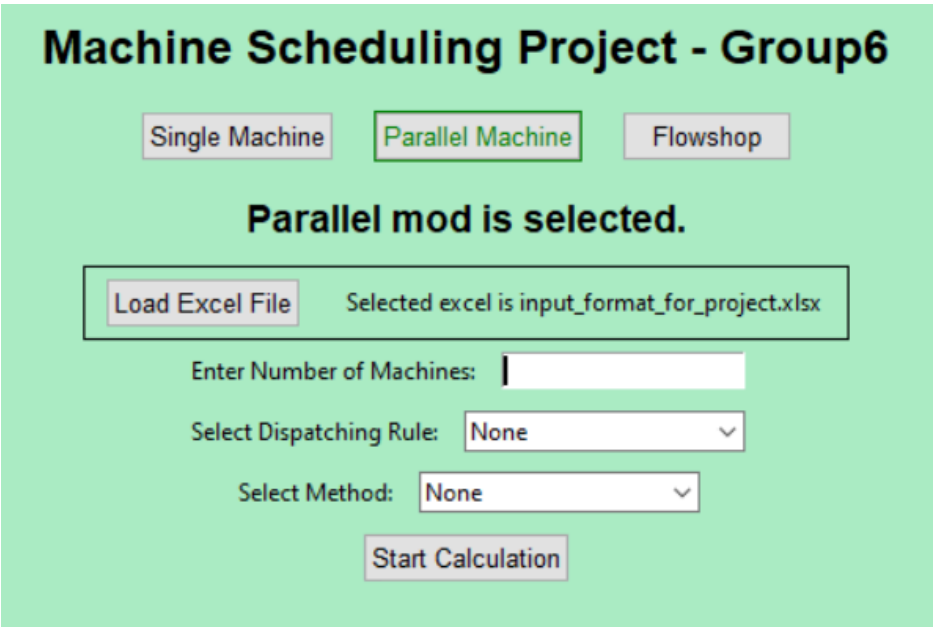


Figure 3. GUI for Parallel Machines Environment Configuration and Calculation

a. SPT Rule for Parallel Machines:

In this approach, jobs are assigned to machines in increasing order of their processing times. Smaller jobs are prioritized and allocated first, ensuring that they are completed efficiently and in the shortest possible time, as shown in Figure 3.1.

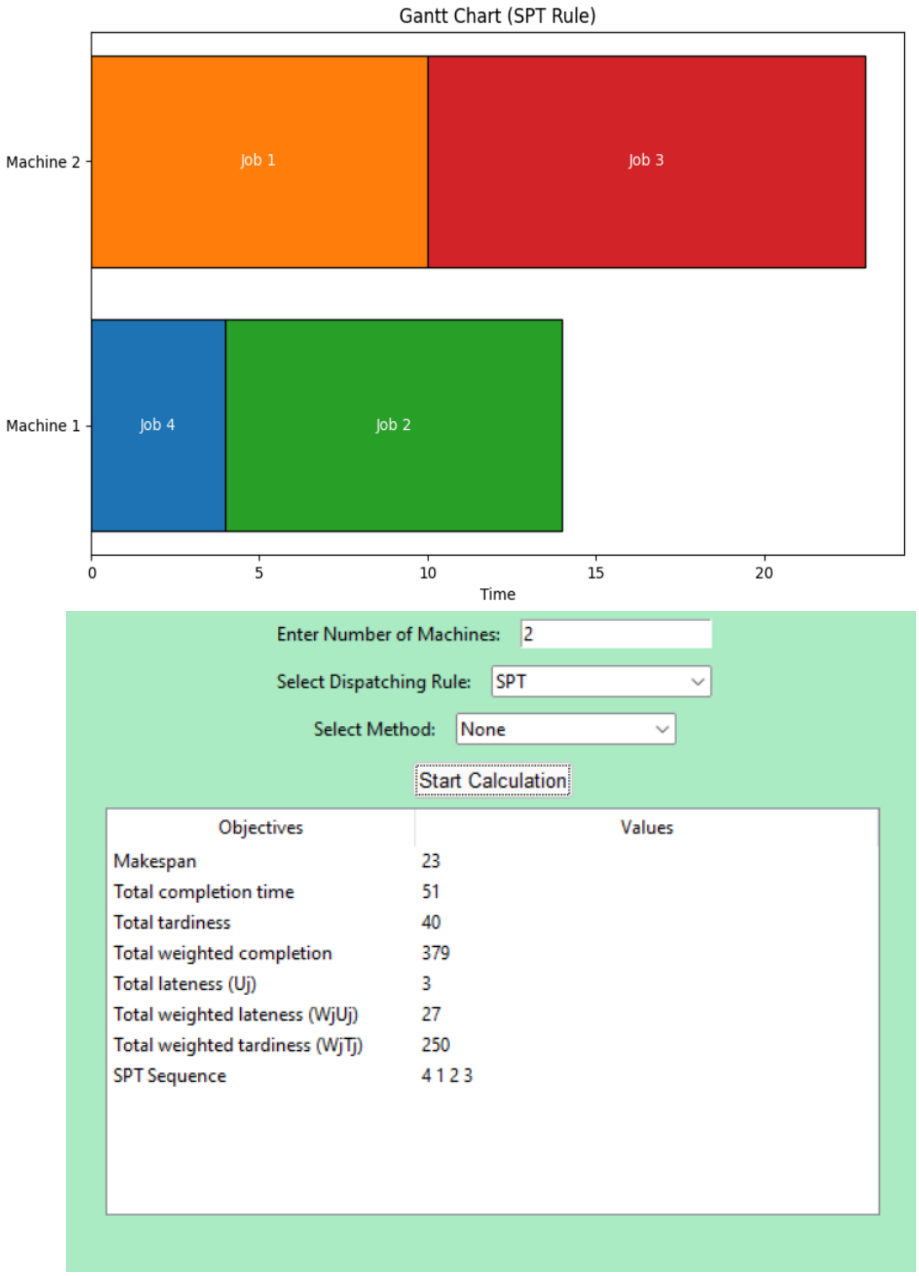


Figure 3.1 Results of Minimizing Makespan for Parallel Machines with SPT Rule

b.LPT Rule for Parallel Machines:

This method involves assigning jobs to machines in decreasing order of their processing times. Larger jobs are prioritized to ensure they are distributed across the machines early in the scheduling process, balancing workload effectively, as shown in Figure 3.2.

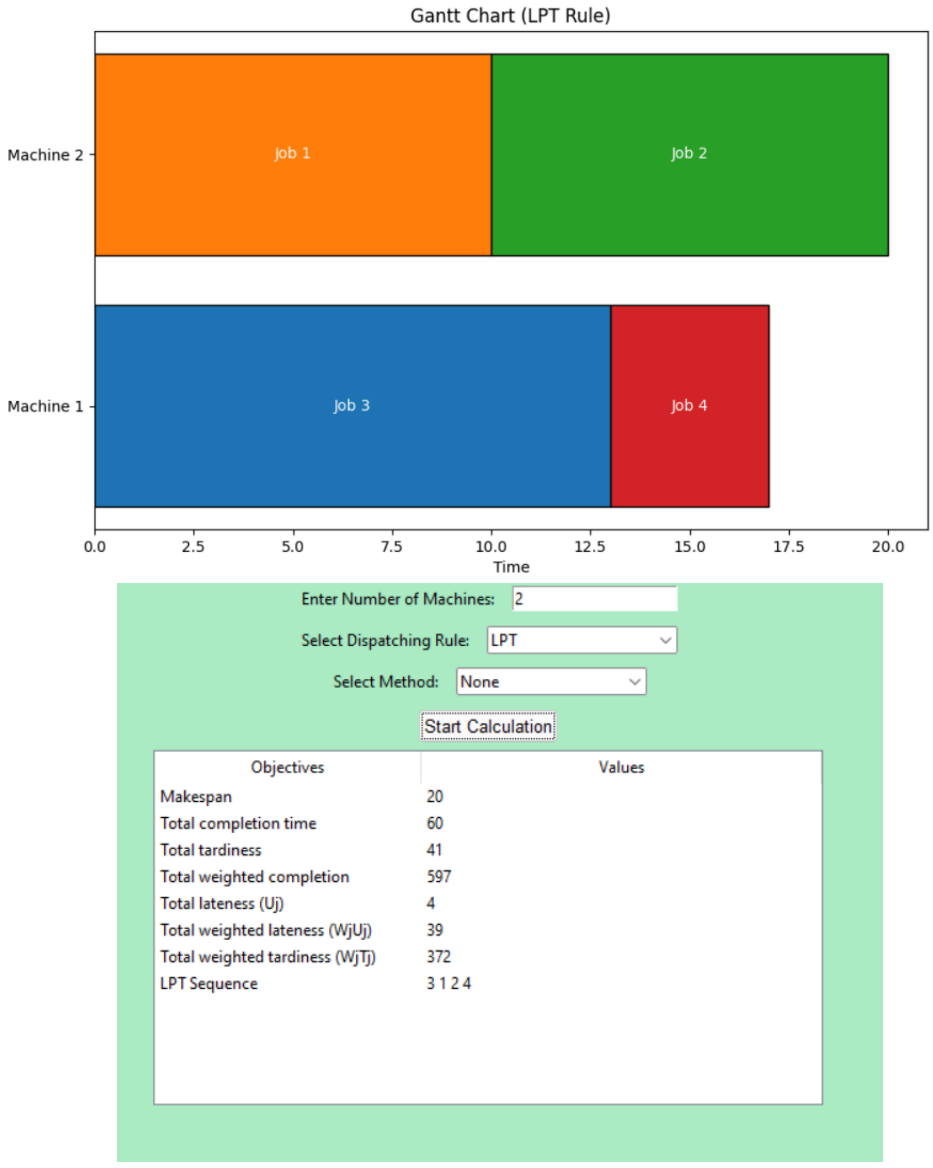


Figure 3.2 Results of Minimizing Makespan for Parallel Machines with LPT Rule

c.Wrap-around Rule for Parallel Machines:

Jobs are distributed sequentially across the machines in a cyclic manner. The first job is assigned to the first machine, the second job to the next machine, and this pattern is repeated until all jobs are allocated. This method ensures an even distribution of jobs across all machines as shown in Figure 3.3.

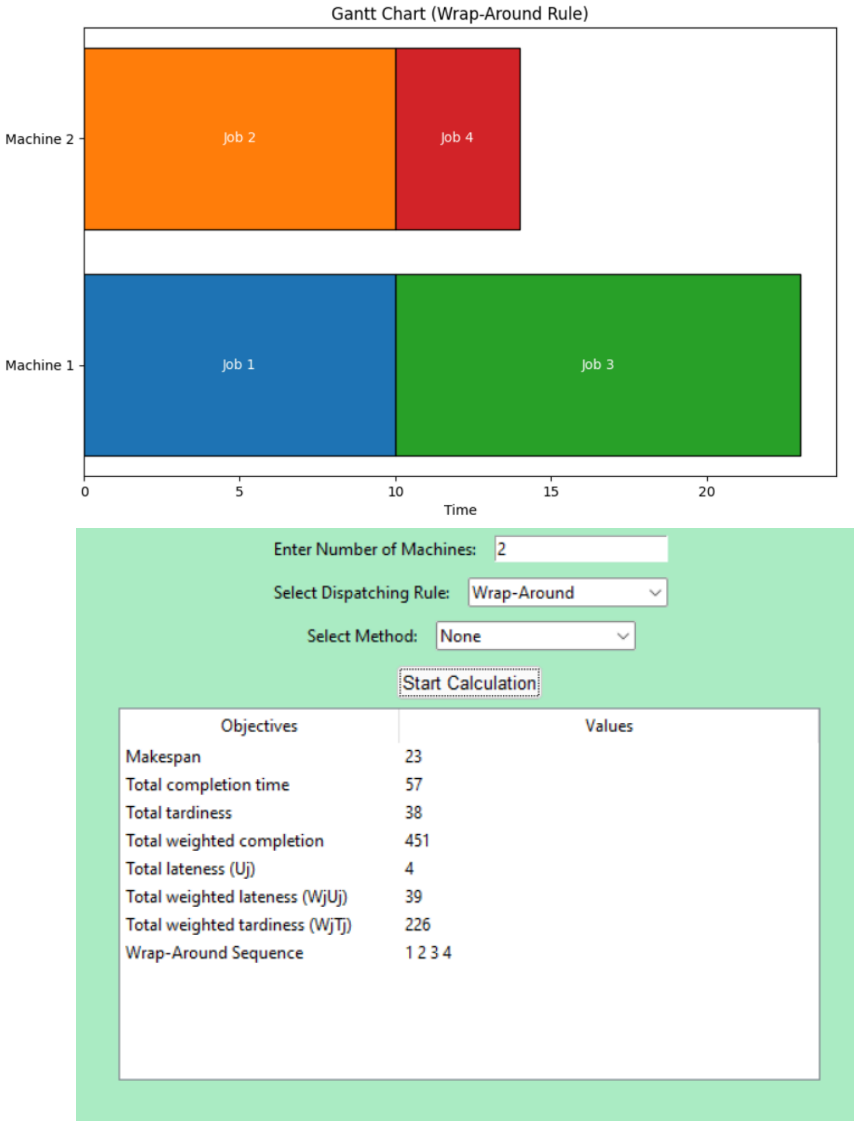


Figure 3.3 Results of Minimizing Makespan for Parallel Machines with Wrap-Around Rule

2.1. Solving the Parallel Machines Problem with the Local Search Method

In this section, three initial solutions are generated using the SPT, LPT, and Wrap-Around dispatching rules, as shown in Figure 4. These initial solutions are displayed in the GUI, providing users with an overview of the starting points for the optimization process.

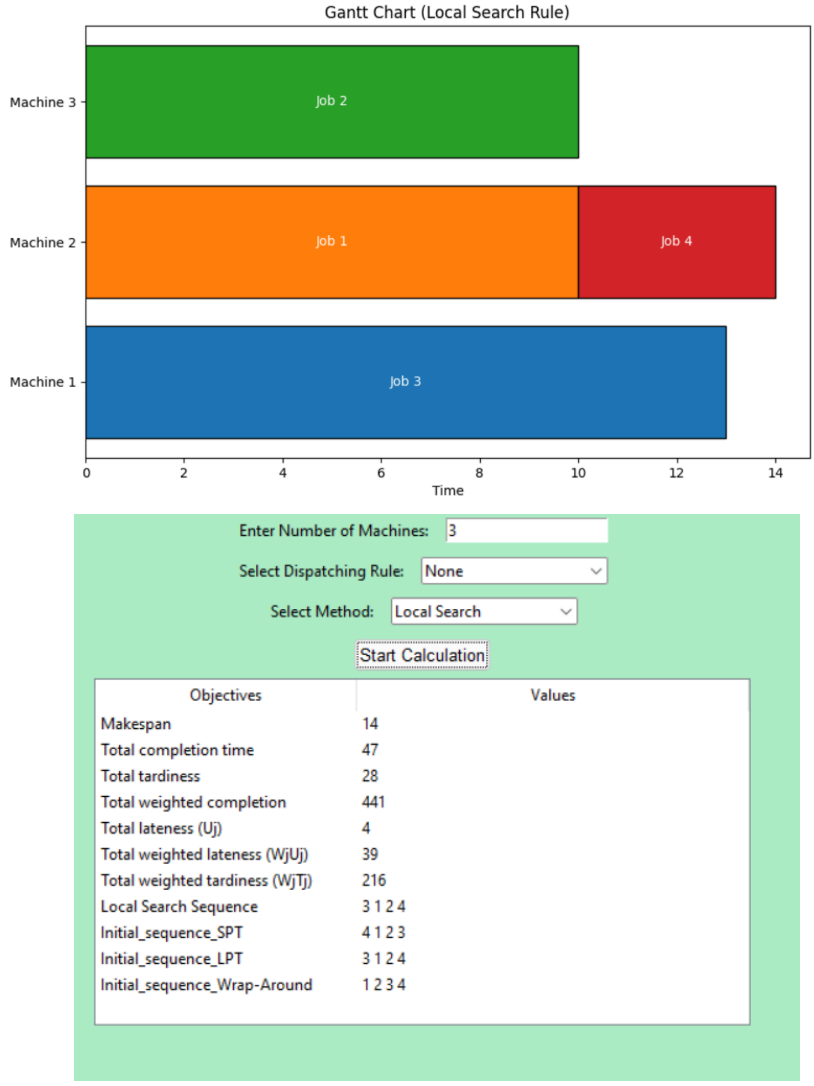


Figure 4. Optimized Gantt Chart and Results for Parallel Machines with Local Search Method

Once the initial solutions are generated, the Local Search method is applied. The method utilizes a random swap technique to explore the neighborhood structure of each initial solution. This process involves performing 500 iterations for each initial solution to identify improved sequences within the neighborhood. The random swaps enable the algorithm to refine the initial

sequences by searching for configurations that optimize the scheduling objectives.The best sequence obtained from the Local Search process is selected based on the evaluation criteria. The resulting Gantt chart, along with the updated job sequence and performance metrics, is displayed in the GUI.

2.2. Solving the Parallel Machines Problem with the Meta-Heuristic Method

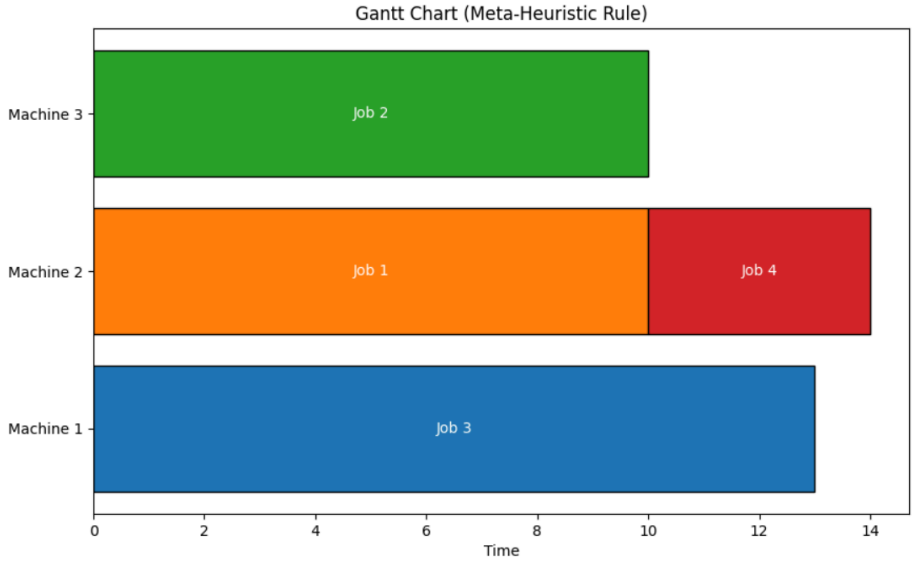
In this section, the parallel machines problem is addressed using the Meta-Heuristic method, which builds upon the Local Search process by incorporating the Threshold Accepting technique. This approach provides additional flexibility in exploring solutions that may initially appear suboptimal but have the potential to lead to better overall results.

The Meta-Heuristic process begins by generating initial solutions using SPT, LPT, and Wrap-Around dispatching rules, as displayed in the GUI. These initial solutions are then refined through Local Search, where random swaps are performed to explore the neighborhood structure. Following this step, the Threshold Accepting method is applied to further optimize the solution.

Threshold Accepting Process in the Algorithm:

1. **Initial Threshold Setting:** The process starts with a defined threshold value (e.g., 5), which allows for the acceptance of worse solutions if the deterioration in performance is below this threshold.
2. **Iterative Improvement:** A new sequence is generated by performing a random swap of job assignments. The objective value for the new sequence is calculated, and the sequence is accepted if:
    - It provides a better objective value than the current sequence.
    - It is worse but within the allowable threshold.
3. **Threshold Reduction:** After a set number of iterations, the threshold is gradually reduced (e.g., multiplied by 0.95 for a 5% reduction) to encourage convergence toward a refined solution. This step is optional but helps the algorithm focus on more promising solutions as the process progresses.
4. **Stopping Criterion:** The algorithm terminates after a fixed number of iterations (e.g., 500), ensuring a balance between computational efficiency and solution quality.

The final output of the Meta-Heuristic method includes the optimized sequence, the Gantt chart representing the schedule, and the associated performance metrics, as shown in the GUI and in Figure 5. This structured approach ensures a robust exploration of the solution space while gradually refining the schedule to achieve the best possible result for the parallel machines problem.



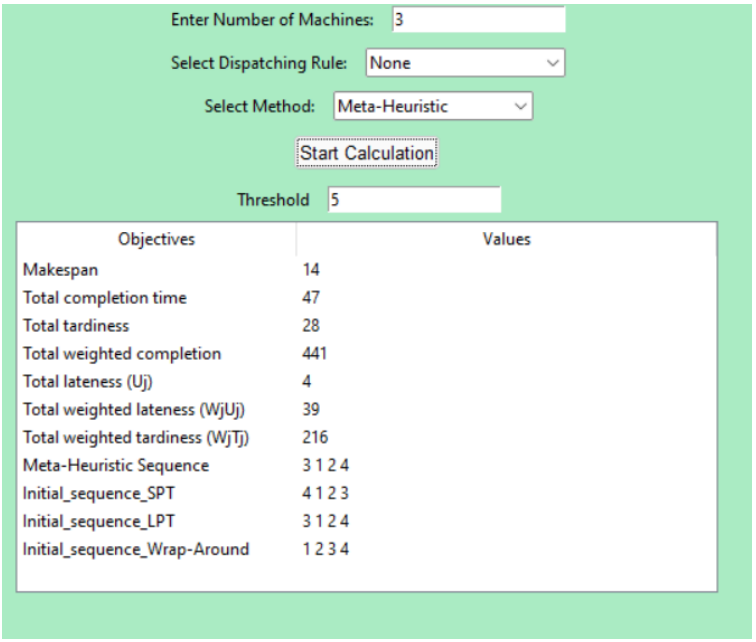
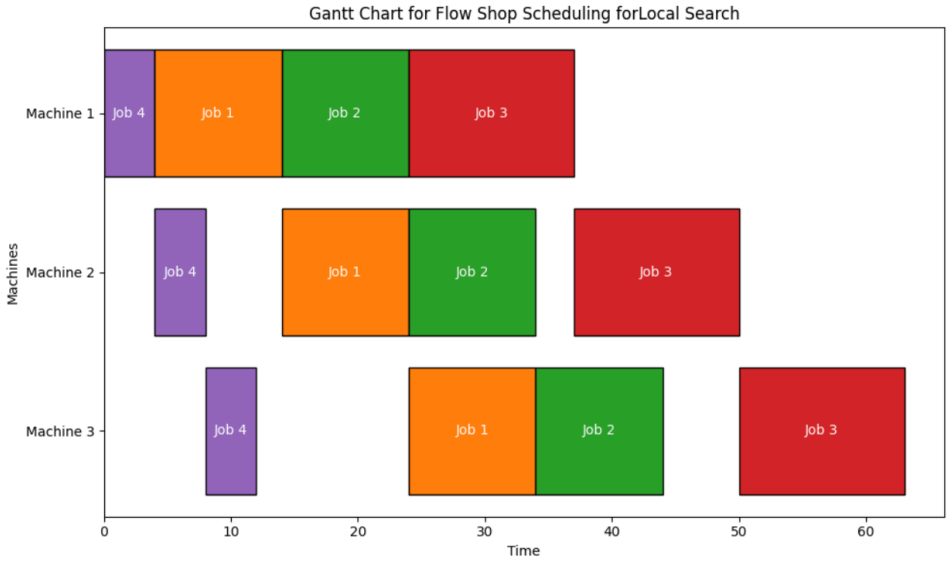


Figure 5. Results of the Parallel Machines Problem Solved with the Meta-Heuristic Method

3. Solving the Flow Shop with Local Search and Threshold Accepting

In the flow shop scheduling problem, jobs are required to follow the same sequence on all machines. This ensures that each job proceeds through the machines in a predefined order. To optimize the scheduling process, the Threshold Accepting method is applied, enhancing the search for better solutions while allowing some flexibility in accepting suboptimal intermediate results.



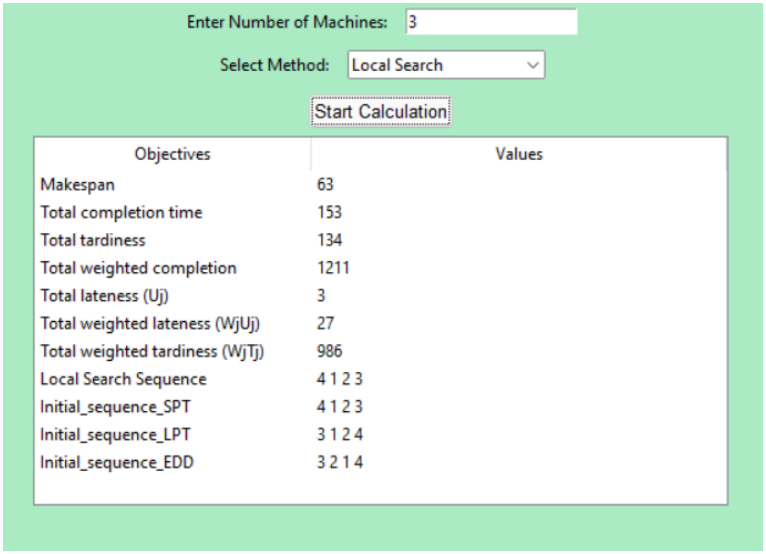
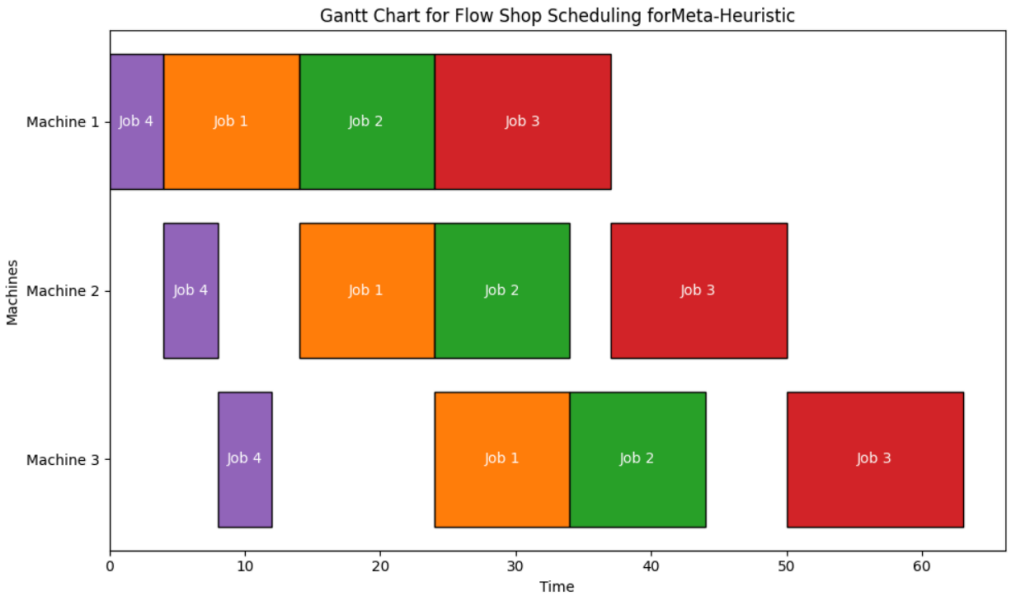


Figure 6. Results of the Flow Shop Problem Solved with the Local Search Method

In the iterative improvement phase, new sequences are created by randomly swapping job assignments. Each new sequence is evaluated, and it is accepted if it results in an improved objective value, such as a reduction in makespan. If the new sequence does not improve the objective value but falls within the acceptable threshold, it is still accepted. This flexibility allows the algorithm to explore solutions that may eventually lead to better results. The threshold value is gradually reduced after a set number of iterations. For instance, it can be decreased by 5% in each step by multiplying it by 0.95. This gradual reduction helps the algorithm focus on more refined solutions as the process progresses.



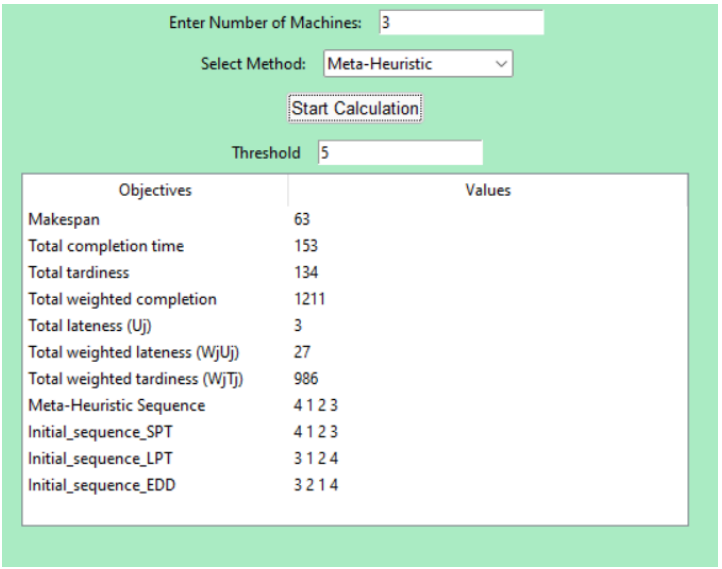


Figure 7. Results of the Flow Shop Problem Solved with the Meta-Heuristic Method

The algorithm concludes after a specified number of iterations, such as 500, ensuring a balance between computational time and solution quality. Figure 6 illustrates the Local Search process for the flow shop environment, while Figure 7 presents the application of the Meta-Heuristic method for the same.

References

[1]M. L. Pinedo, "Scheduling," *SpringerLink*, 2022, doi: https://doi.org/10.1007-978-3-031-05921-6.