KLE Society's
KLE Technological University, Hubballi.

A Mini Project Report on

Parallel Assembly Line Balancing With Human Robot Collaboration

Submitted in partial fulfillment of the requirement for the degree of

Master of Technology
In
Computer Science and Engineering

Submitted by:
TaranpreetSingh H Rababi
[01FE23MCS009]

Under the guidance of
Dr. Jayalaxmi G N

Under the Co-guidance of
Dr. Sachin Karadgi

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

KLE TECHNOLOGICAL UNIVERSITY, HUBBALLI – 580 031

2023-24

KLE Technological University
Creating Value, Leveraging Knowledge

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that Mini project on "**Parallel Assembly Line Balancing With Human Robot Collaboration"** is a bonafide work carried out by the student team comprising of **TARANPREETSINGH H RABABI (01FE23MCS009)** for partial fulfillment of completion of Second-semester M.Tech in Computer Science and Engineering during the academic year 2023-24. The training report has been approved as it satisfies the academic requirement concerning the project work prescribed for the above-said course.

Guide	Co-Guide
Dr. Jayalaxmi G N	Dr. Sachin Karadgi
P. G. Coordinator	Head, SoCSE
Dr. Vishwanath. P. Baligar	Dr. Vijayalaxmi M

External Viva -Voice:

Name of the Examiners	Signature with date
1.	
2.	

ACKNOWLEDGEMENT

This is to express my heartfelt appreciation and gratitude to everyone who helped make this endeavor possible. I am deeply grateful to Dr. Ashok Shettar, Pro-Chancellor, Dr. P G Tewari, Vice Counselor and Dr. Basavaraj Anami, Registrar, KLE Technological University, for their advice, support, and all the resources necessary to finish this project.

I would also want to use this occasion to offer my heartfelt gratitude and appreciation to Dr. Vijayalakshmi M, the head of the School of Computer Science and Engineering.

I am grateful to our P.G. Coordinator, Dr. Vishwanath P. Baligar and and co-guide, Dr. Sachin Karadgi for their enthusiasm, advice, and useful recommendations that helped us complete this project successfully.

I am grateful to our guide, Dr. Jayalaxmi G N, for her enormous source of inspiration and valuable assistance in demanding efforts in the correct path.

I would also want to thank K.L.E Technological University for providing me with a virtual learning environment and enough input to complete my academics.

TaranpreetSingh H Rababi

[01FE23MCS009]

Abstract

Human-robot-collaboration (HRC) is emerging as a extensive production mode in the era of clever production. This research investigates the utility of HRC in parallel assembly lines and explores its capability for enhancing productivity, aid utilization, and versatility. The look at addresses the combination of human-robotic-collaboration into the parallel assembly line balancing hassle (PALBP), which we talk over with as PALBP-HRC. The collaboration mode lets in both parallel and collaborative challenge executions of human people and robots. A genetic algorithm (GA) is designed to reduce cycle time, optimize human and robot venture assignments with constraints on zoning constraints. Experimental results show that the proposed GA correctly solves the PALBP-HRC with the aid of accomplishing discounts in cycle times, improvements in undertaking venture, and flexibility. Human-robotic collaboration absolutely integrated into parallel assembly traces significantly complements both efficiency and versatility.

Contents

Abstract iv
Table of contents v
List of Figures vi

1 INTRODUCTION **1**
1.1 Assembly Line Balancing Problem (ALBP) 1
1.2 Parallel Assembly Line Balancing Problem (PALBP) 3

2 LITERATURE REVIEW **6**

3 REQUIREMENT SPECIFICATION **8**
3.1 Motivation 8
3.2 Problem Statement 8
3.3 Objectives 8
3.4 Functional Requirements 9
3.5 Non-Functional Requirements 9
3.6 Hardware Requirements 10
3.7 Software Requirements 10

4 METHODOLOGY **12**
4.1 General Flow Diagram 12
4.2 High-level design 15

5 Implementation **18**
5.1 Dataset Description 18
5.2 Code Implementation 18

6 RESULTS AND DISCUSSION **24**
6.1 Cycle Times 24
6.2 Zoning Constraint 26
6.3 Assignment of Tasks to Stations 27

7 CONCLUSION AND FUTURE WORK **32**
References 33
Appendix 35

List of Figures

1.1.1 Car Manufacturing Assembly Line 2
1.1.2 TV Manufacturing Assembly line 2
1.2.1 Parallel Assembly Line with Human-Robot 3
1.2.2 Parallel Assembly Line with Human-Robot 4
1.2.3 Parallel Assembly Line with Human-Robot 4

4.1.1 General Flow Diagram 12
4.2.1 High-Level Design 15

6.1.1 Cycle Time Bar Graph 25
6.1.2 Cycle Time Line Graph 25
6.2.1 Zoning Constraint Percentage Bar Graph 27
6.2.2 Zoning Constraint Percentage Line Graph 27
6.3.1 Task Assignment of Jackson Data 29
6.3.2 Task Assignment of jaeschke Data 29
6.3.3 Task Assignment of MANSOOR Data 30
6.3.4 Task Assignment of mertens Data 30
6.3.5 Task Assignment of MITCHELL Data 31
6.3.6 Task Assignment of BOWMAN Data 31

Chapter 1

INTRODUCTION

The development of more efficient and effective assembly line balancing solutions has been led by the advancement of industrial automation. The increasing complexity of manufacturing systems requires strong methodologies to allocate tasks across multiple stations while considering several constraints. The study focuses on implementing a genetic approach to maximize task assignments, reduce cycle time, and ensure feasibility of solutions in a parallel assembly line environment.

1.1 Assembly Line Balancing Problem (ALBP)

The assembly Line Balancing problem (ALBP) is a crucial optimization venture in manufacturing and production structures. It entails distributing obligations amongst workstations along an meeting line to obtain specific objectives, which includes minimizing cycle time, reducing the range of workstations, or maximizing performance, whilst adhering to various constraints like undertaking priority, zoning, and workload stability. green meeting line balancing is crucial for enhancing productiveness, lowering expenses, and making sure smooth operations in industries starting from automotive to electronics. The trouble is inherently combinatorial and regularly solved the usage of optimization strategies, including heuristic, meta-heuristic, or exact techniques, to address its complexity and ranging industrial necessities. meeting line balancing allows in optimizing labor, assets, and device usage by way of cautiously distributing duties based totally on time requirements and priority. Balancing properly can cause fee financial savings, quicker manufacturing, and higher throughput. however, choppy project intervals, aid obstacles, and variability in performance complicate the balancing process. Fig 1.1.1 shows the automobile production meeting line and Fig 1.1.2 indicates the tv manufacturing assembly line.

Figure 1.1.1: Car Manufacturing Assembly Line

Figure 1.1.2: TV Manufacturing Assembly line

1.2 Parallel Assembly Line Balancing Problem (PALBP)

Human-robot collaboration (HRC), as an rising manufacturing mode, has garnered significant interest inside the context of the growing clever manufacturing. Parallel meeting strains, they perform with numerous traces jogging concurrently. (for instance in car manufacturing wherein a car may have frame assembly, painting and engine installation simultaneously.) those parallel lines can be balanced the use of multi-line stations in which obligations for adjacent traces are completed together. This situation is known as the parallel assem- bly line balancing problem (PALBP). the combination of collaborative robots into parallel assembly traces can in addition increase the performance of the assembly device called PALBP- HRC. Parallel line balancing challenges contain allocating obligations correctly throughout parallel lines in such a manner that no line ends up turning into a bottleneck and every stays in stability regarding paintings distribution and cycle time. This entails handling the priority con- straints of the mission and balancing the workload such that no unmarried line acts as a bottleneck, at the same time as all strains run at top performance. This process seeks to ensure a minimized downtime, decreased put off and a smooth glide of productions in all traces. Parallel line balancing may be very sizeable in which there is a huge demand for production and only a single line can't make to the goal manufacturing hence necessitates multiple strains to cater to the demand. This balances the useful resource usage by using providing higher throughput and lets in most flexi- bility in manufacturing. Fig 1.2.1 shows the parallel assembly traces with two traces and human and robot, and Fig 1.2.2 shows the human and robotic working together in an assembly line. Fig 1.2.2 suggests the meeting of automobile door in human robotic collaboration environment.

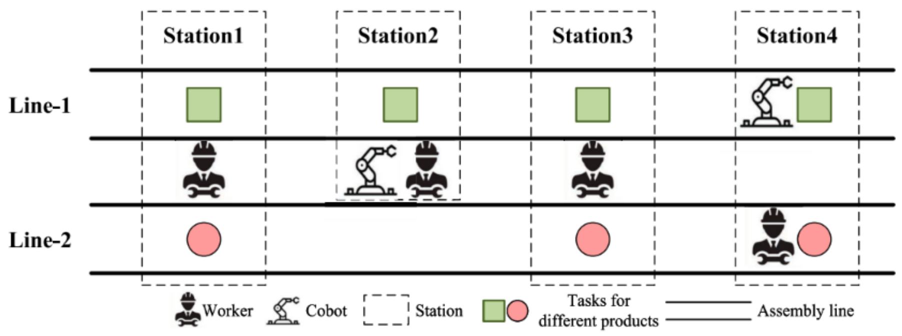


Figure 1.2.1: Parallel Assembly Line with Human-Robot

Figure 1.2.2: Parallel Assembly Line with Human-Robot

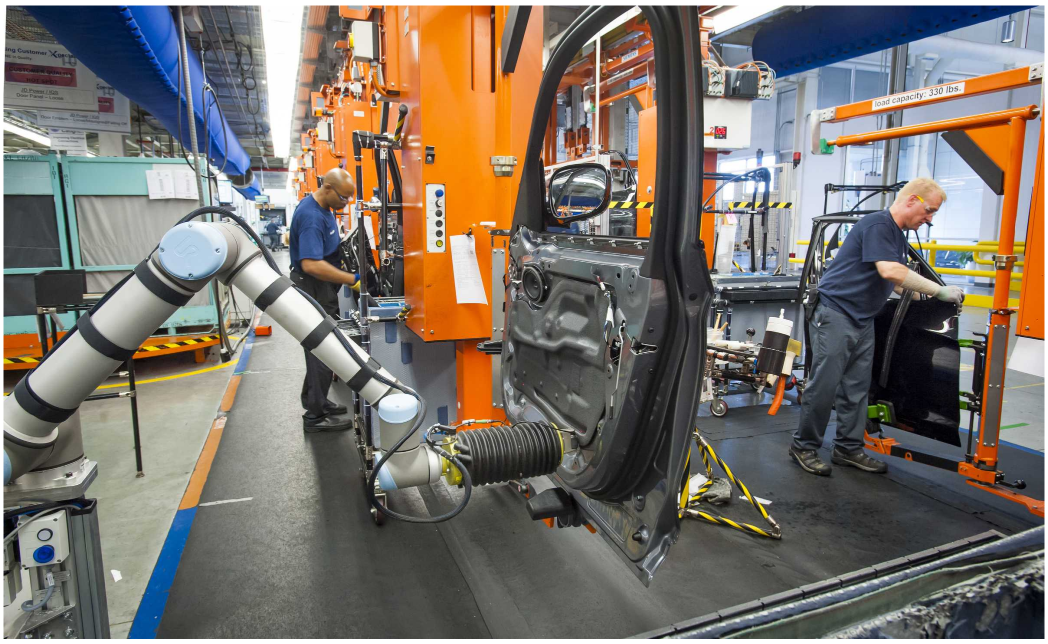


Figure 1.2.3: Parallel Assembly Line with Human-Robot

The organization of report is as follows, the assignment ”Parallel assembly Line Balancing With Human-robotic Collaboration” goals at exploring how we can maximize the efficiency of the meeting line by incorporating a genetic algorithm at the same time as employing human people and robots working together. Literature evaluate is chapter 2, which takes stock of the prevailing work on meeting line balancing and human-robot collaboration. hardware, software program, and purposeful necessities of the system are designated in chapter three: Requirement Specification. The techniques that were evolved to optimize undertaking assignments and collaboration among human companions and robotic marketers are defined in bankruptcy four: methodology. bankruptcy 5: version architecture is intended to look at the layout of AI models toward improving system performance. bankruptcy 6: Implementation describes the actual-international scenarios of coding and model training for effective device development. bankruptcy 7: outcomes and dialogue includes the evaluation of the system overall performance and highlights strengths and weaknesses of it. subsequently, bankruptcy 8: end and future work takes into consideration the observations made and proposes some avenues for future studies so one can up-scale the gadget and adeptly observe it in a broader business context.

Chapter 2

LITERATURE REVIEW

The hassle of assembly line balancing has been drastically studied with several optimization strategies proposed to cope with the complex troubles in this vicinity. those variety from traditional strategies through blended-integer programming and heuristic algorithms for arriving at an most suitable or near-highest quality solutions challenge to predefined constraints. This, these days, meta-heuristic techniques like genetic algorithms, simulated annealing, and particle swarm optimization have come within the highlight due to the ability of solving multi-objective issues. on this segment, an in depth evaluation of to be had works in assembly line balancing is supplied with emphasis on their methodologies, obstacles, and applicability to parallel and bendy manufacturing structures.

1. **Title:** Balancing parallel assembly lines with human-robot collaboration: problem definition, mathematical model and tabu search approach.
   Authors: Zhaofang Mao, Jiaxin Zhang, Yiting Suna, Kan Fang and Dian Huang.
   Published On: INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH April 2024.
   DOI: https://doi.org/10.1080/00207543.2024.2356627
   About: in this look at, they proposed lines balancing trouble with human-robotic collaboration (PALBP-HRC-II). There are multiple line (especially or 3 lines) positioned parallel to every other.The design of parallel meeting lines with collaborative robots combines the power of parallel strains with the efficiency of human-robotic collaboration The objective of the PALBP-HRC is to balance adjacent traces collectively with minimising the cycle time.

2. **Title:** Balancing and scheduling human-robot collaborated assembly lines with layout and objective consideration
   Authors: Amir Nourmohammadi, Masood Fathi, Amos H.C. Ng.
   Published On: November 2023 Elsevier
   DOI: https://doi.org/10.1016/j.cie.2023.109775
   About: The assembly line consists of stations with embedded sensing and interacting software program and hardware that carry out the tasks one by one, in parallel, or collaboratively by way of multiple humans and robots with exclusive competencies and talents.This study considers three one-of-a-kind objective types to be optimized relying on the character of the problem, as, type-1: Minimizes the NS given a CT. kind-2: Minimizes the CT given an NS. kind-rw: Minimizes the full fees of stations, operators, and robotic strength consumption given a CT.

3. **Title:** Balancing assembly lines with industrial and collaborative robots: Current trends and future research directions
   Authors: Masood Fathi, Arash Sepehri, Morteza Ghobakhloo, Mohammad Iranmanesh, Ming-Lang Tseng.
   Published On: May 2024 Elsevier.
   DOI: https://doi.org/10.1016/j.cie.2024.110254
   About: ambitions to clarify the key variations between hassle sorts, this segment outlines the mathematical fashions for basic SALBP, RALBP, and ALBP-HRC. these fashions serve as references for information the number one inputs, selection variables, and constraints of each hassle type. This take a look at addresses two common objectives for every simple problem kind: minimizing the variety of workstations (kind-I) and minimizing the cycle time (type-II). In total, six fashions are presented: SALBP-I, SALBP-II, RALBP-I, RALBP-II, ALBP-HRC-I, and ALBP-HRC-II.

4. **Title:** Balancing of assembly lines with collaborative robots.
   Authors: Christian Weckenborg, Karsten Kieckha fer, Christoph Muller, Martin Grunewald and Thomas S.
   Published On: Spengler 6 August 2019 Springer.
   DOIhttps://doi.org/10.1007/s40685-019-0101-y.
   About: The look at characterized the hassle by way of the opportunity that human and robots can concurrently execute responsibilities on the equal workpiece both in parallel or in collaboration. The proposed version comes to a decision on both the project of collaborative robots to stations and the distribution of workload to people and robotic partners. the main goal of the observe is to limit the cycle time and allocate robots to stations.

5. **Title:** Balancing U-type assembly lines with human–robot collaboration
   Authors: Zhaofang Mao, Jiaxin Zhang, Kan Fang, Dian Huang and Yiting Sun.
   Published On: July 2023 Elsevier
   DOI: DOI:10.1016/j.cor.2023.106359 **About:**U-kind meeting line balancing hassle with human-robot collaboration (UALBP-HRC), wherein the collaborative robotic (cobot) can either parallelly method unique duties or collaboratively process the identical mission with employees. the principle contributions of this have a look at are as follows. (i) a new UALBP-HRC from the angle of parallel-based collaborative way is studied for the primary strive. (ii) a new blended integer programming model is formulated for UALBP-HRC to limit the cycle time. (iii) Enhancement techniques, which includes tight decrease and higher bounds and preliminary answer, are proposed to improve the MIP.

Chapter 3

REQUIREMENT SPECIFICATION

Machine Requirement a portion of the hardware and software program requirements are designated in the specification, which additionally covers the parts required for the system's effective operation. Requirement for software specs the specification, which covers the components required for the system's effective operation, specifies some of the hardware and software requirements.

3.1 Motivation

Minimizing cycle time in a parallel assembly line with human-robot collaboration offers substantial benefits including enhanced productivity, cost reduction, and improved quality. To understand the domain and applications in parallel assembly line with collaboration of robots, which can increase the efficiency and productivity. Existing literature highlights that traditional assembly line setups often face challenges such as bottlenecks, inefficiencies, and limitations in flexibility, which can hinder production output.

3.2 Problem Statement

Implement genetic algorithm for assigning tasks to human, robot and human-robot (operator, cobot or operator-cobot). Assume two parallel lines with each line have a input and an output and multiple stations. Minimize the cycle time for completing the tasks by considering the constraints, Precedence Relationship between tasks, Zoning Constraints and Operator Specific Tasks.

3.3 Objectives

1. Explore different models and frameworks for task allocation in hybrid work environments where humans, robots, and cobots work together.
2. Review different methods for balancing workloads between stations to optimize cycle time.
3. Develop mixed-integer linear programming (MILP) genetic algorithm (metaheuristic) for task assignment

4. Reduce the cycle time for completing the assigned tasks of the two parallel lines with multiple stations.

3.4 Functional Requirements

- **Time Minimization:** The set of rules must prioritize solutions that reduce the cycle time for each meeting line, reaching the main objective effectively.
- **Precedence Relationship Constraint Handling:** The algorithm must ensure that tasks follow a predefined sequence (precedence relationships) without violating task order dependencies.
- **Zoning Constraint Enforcement:** It should assign tasks based on specific zones, ensuring that some tasks are only performed in designated areas to avoid safety risks and maximize efficiency.
- **Operator-Specific Task Assignment:** The set of rules must allow for unique duties that can only be achieved via human operators or robots, accounting for the skill and capability variations.
- **Genetic Operations (Selection, Crossover, Mutation):** Implement selection, crossover, and mutation operations to evolve solutions over generations. These operations should be designed to respect the assembly line constraints.
- **Fitness Evaluation Parameters:** outline a health feature that evaluates each answer based on cycle time and penalizes constraint violations. This feature have to drive the optimization procedure.
- **Human-Robot Collaboration Optimization:** It should optimize the distribution of tasks between humans and robots based on capabilities, balancing workload and enhancing collaboration efficiency.

3.5 Non-Functional Requirements

- **Performance and Scalability:** The algorithm should perform efficiently, even as the number of tasks and constraints grows, and should scale well for larger assembly lines with more parallel tasks and resources.
- **Reliability and Robustness:** Ensure the algorithm consistently produces feasible and optimized solutions without crashing, even if the problem complexity increases.

- **Adaptability:** The system should be adaptable to changes in constraints or objectives, such as adding new precedence relationships or adjusting zoning rules without major reconfiguration.
- **Usability:** The solution should be user-friendly, with an interface (or script structure) that allows engineers to easily input constraints, task data, and adjust parameters.
- **Maintainability:** Code and system design should be modular and well-documented, allowing for future adjustments or enhancements without extensive rewrites.
- **Compliance with Safety Standards:** Ensure that task allocations adhere to safety regulations for human-robot collaboration in an industrial setting, preventing unsafe task assignments or zoning violations.
- **Flexibility in Genetic Algorithm Parameters:** The set of rules need to permit tuning of genetic parameters (mutation price, crossover rate, population size) for testing and optimizing exceptional setups.

3.6 Hardware Requirements

The hardware requirements for this project are minimal due to the use of Google Colab, which provides cloud-based GPUs. The specific hardware setup includes:

- **GPU:** Access to **NVIDIA Tesla K80**, **T4**, or **P100** provided by Google Colab for accelerated model training.
- **Intel** core i3 processor
- **RAM:** Up to **12GB** of RAM available on Google Colab.
- **Storage:** Cloud storage provided by Google Colab, integrated with **Google Drive** for dataset storage and access.

3.7 Software Requirements

- **Platform: Google Colab** was used as the primary development and execution environment, providing cloud-based GPUs for model training and testing.
- **Programming Language: Python 3.x** was used for implementing the models and handling data.
- **Libraries:**
  - **ZipFile:** For managing and extracting compressed datasets.

- **NumPy:** For numerical computations and handling arrays.
- **OS:** Used for directory and file management.
- **random:** Used for generating random numbers

Chapter 4

METHODOLOGY

The project used a genetic algorithm to maximize scheduling in a lineless mobile assembly system. Modeling the problem was part of the methodology. To account for uncertainties, design an adaptive genetic algorithm for scheduling and simulation. It is possible to address dynamic job arrivals. The approach is focused on flexibility, less delay, and all-round efficiency.

4.1 General Flow Diagram

A genetic set of rules is genuinely a search and optimization approach, stimulated through the standards of natural choice and evolutionary biology. In different phrases, it mimics the technique of evolution, evolving solutions to complicated optimization problems over successive generations through mechanisms like selection, crossover, and mutation.

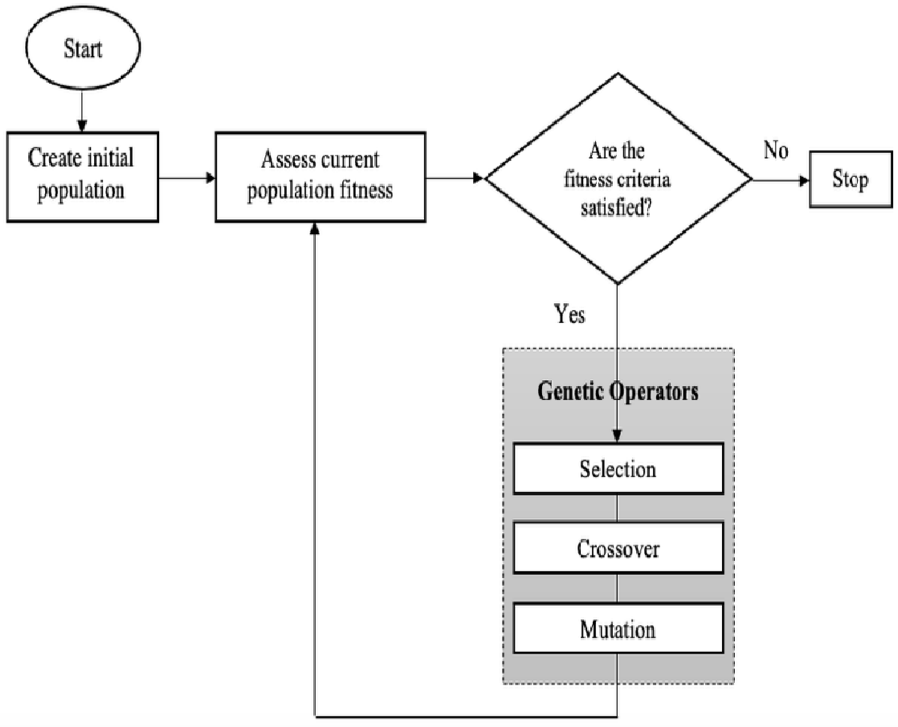

Start
Create initial population
Assess current population fitness
Are the fitness criteria satisfied?
No
Stop
Yes
Genetic Operators
Selection
Crossover
Mutation

Figure 4.1.1: General Flow Diagram

Fig 4.1.1 shows the flowchart describing the general architecture of a Genetic Algorithm (GA) typically involves the following steps:

1. **Start**:

- Thus begins the GA. Prior to launching into the algorithm, however, it is contingent upon having a thorough definition of the problem at hand.
- Define the objective function of the problem, such as minimize cost, maximize efficiency, or any other relevant objective function.
- Generate representations for the solutions. Commonly, solutions are represented as chromosomes typified by strings of numbers or binary values.
- Delimit and characterize the bounds such as population size, mutation rate, crossover probability, and the number of generations.

2. **Create Initial Population**:
   - The population is a collective bunch of potential solutions (individuals/chromosomes).
   - A random generation or heuristic basis is produced for the first solution set.
   - Each one of them-namely, potential solution to the optimization issue.
3. **Assess Current Population Fitness**:
   - An assessment of the health characteristic is the most crucial a part of the GA.
   - The fitness function provides a measure for quantifying how ”fit” or adept a solution is with respect to a particular problem.
   - It measures fitness for every individual in an entire population. This helps the algorithm converge towards better solutions over the generations.
4. **Are The Fitness Criteria Satisfied**:
   - A particular solution has attained the prescribed fitness, whether in terms of cost minimization or profit maximization.
   - It attains the pre-specified maximum number of generations characterized by the algorithm.
   - There is no significant improvement over several generations, and the fitness starts converging.
5. **Genetic Operators**: These operators are the mimickers of the biological evolution process. They further help the algorithm in both exploring and exploiting the searched solution space.
   (a) Selection
      - Selection determines which individuals will be the ones to pass on their genes to the next generation.

- People have higher chances of selection as parents based on qualifying criteria.
- The ways of selection most commonly used are:
  - Roulette Wheel selection: chance of selection is proportional to health.
  - Tournament Selection: Randomly selects a bunch of individuals before choosing the best individual among all the chosen ones.
  - Rank Selection: Ranks individuals based on their fitness and are selected according to their ranks.

(b) **Crossover**: This operator creates an offspring by combining the genetic material (chromosomes) of two parent individuals.

- Purpose:
  - Exchanging and combining good characteristics from parents.
  - Search space exploration and diversity generation in population.
- Common crossover methods:
  - Single-point crossover: where it separates the chromosomes of the parents with a unique point and swaps the segments.
  - Two-point crossover: Two splitting points are used for swapping.
  - Uniform crossover: Each gene is selected randomly by either of the two parents.

(c) **Mutation**:

- Mutations are randomly created changes that occur in the chromosomes of an individual.
- Purpose: To keep genetic diversity afloat and preventing premature convergence of the population at a local optimum.
- It is usually utilized with less probability so as to avoid destroying good solutions.
- For example: Binary mutation: Flipping bits (e.g., $1 \rightarrow 0$ or $0 \rightarrow 1$).
  Real-valued mutation: Add small random value on gene.
  Swap mutation (for permutations): Swap two positions in the sequence.

6. **Loop Back**:

- Upon applying genetic operators, an algorithm gives rise to a new population.
- This newer population is reassessed, and this is where the loop again comes back to step 3.
- Using this approach, the algorithm is expected to provide better solutions in each successive generation (higher fitness).

7. **Stop**:
    - The algorithm ends when Solution meets the stopping criteria to be defined as satisfactory.
    - The upper limit on iterations (or generations) is reached.
    - The best found solution (individual) is output as the result.

4.2 High-level design

Fig 4.2.1 shows the high flow of a meta-heuristic algorithm for the solution of optimization problems.

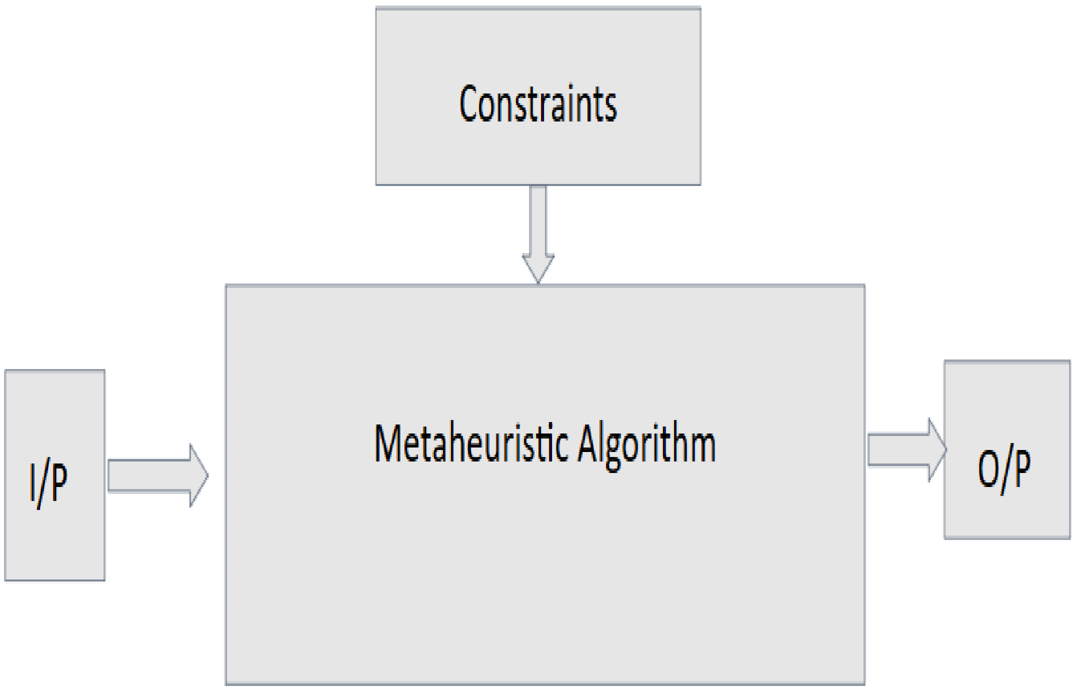


Figure 4.2.1: High-Level Design

1. Input:
    - At this step, problem data is given to the meta-heuristics algorithm.
    - It usually contains:
        - Define the problem (e.g., a scheduling task, a traveling salesman problem, or a resource allocation problem).
        - Initial solution(s) or search space parameters.
        - Algorithm-specific parameters such as population size (if population-based), iteration limits, or step size.

2. Constraints:

   - Constraints are the rules or conditions that are satisfied for a solution to hold a validity.
   - they are defining the feasibility of solutions in the search space. Constraints, for instance, inside the case of scheduling
     - problems: ”No two events can overlap”.
     - In route optimization: ”A vehicle cannot exceed its maximum capacity”.
     - Task assignments: ”A task must respect precedence relationships”.
   - These constraints guide the meta-heuristic algorithm in avoiding invalid solutions while looking for an optimal one.

3. Meta-Heuristic Algorithm:

   - A core portion in this diagram is the optimization which in fact derives from the meta-heuristic algorithm.
   - Meta-heuristic deals with high-level strategies that enable solving highly complicated optimization problems by intelligently exploring the solution space.
   - It balances two critical sides:
     - Exploring: Searching diverse regions of the solution space.
     - Exploiting: Refinement of the current best solutions.
   - Examples of Meta-heuristic Algorithms:
     - Genetic Algorithm (GA): which derives its line from the natural selection and evolution.
     - Simulated Annealing (SA): which is similar to the cooling process of metals.
     - Particle Swarm Optimization (PSO): which derives its line from the social behavior of birds or fish.
     - Ant-Colony-Optimizations (ACO): which is an imitation of how ants find the shortest paths.
     - Tabu Search: uses memory to avoid revisiting previous solutions.
     - Greedy Randomized Adaptive Search Procedures (GRASP): combines completely greedy algorithms with randomization.

4. Output:

   - This is the final output as produced by the optimization process.
   - The outputs are between:

- The best solution found using the meta-heuristic algorithm within the confines of constraints.
- Some key performance indicators such as minimized cost, maximized profit, optimized schedule.

Chapter 5

Implementation

Implementation of the proposed solution includes the layout of a genetic algorithm that includes precedence, zoning, and cycle time constraints. It starts with an initial population of capability answers. via choice, crossover, and mutation operators, those capacity solutions are iteratively advanced. The health characteristic computes a measure of the quantity to which each answer meets these constraints and is consequently effective in trendy. it's going to then assign obligations to human and robotic sources in line with their capabilities.

5.1 Dataset Description

facts Set of Gökçen et al. (2006): This set comprises several assessments on PALBP as referred inside the article of Göokçen et al.(2006). each sort of take a look at problem includes 4 components, o1, o2, z1, and z2. right here o1 and z1 contain the priority relations of the version produced on first parallel meeting lines and the venture time of the version respectively; o2 cease and z2 imply the precedence relations of the (different) model produced in the 2nd parallel meeting line and the undertaking time of the model respectively.

5.2 Code Implementation

1. **Fitness Function**

Listing 5.1: Fitness Function

```python
def fitness_with_robot(individual, processing_times,
    robot_density):
    station_loads = np.zeros(number_of_stations)
    for task in range(len(individual)):
        station = individual[task]
        station_loads[station] += processing_times[task]

    # Apply the cycle time rules based on station type
    adjusted_station_loads =
        np.zeros_like(station_loads)
    for i, load in enumerate(station_loads):
        if robot_density[i] == 0:  # Worker-only
            adjusted_station_loads[i] = load
        else:  # Mixed (Worker and Robot)
            adjusted_station_loads[i] = 0.7 * load

    return max(adjusted_station_loads)  # Minimize the maximum cycle time
```

2. **Initial Population**

Listing 5.2: Initialize Population Function

```python
def initialize_population_with_precedence(pop_size, initial_assignment, precedence_constraints_line1, precedence_constraints_line2):
    population = []
    for _ in range(pop_size):
        while True:
            individual = initial_assignment.copy()
            np.random.shuffle(individual[:tasks_nt1])  # Shuffle Line 1
            np.random.shuffle(individual[tasks_nt1:])  # Shuffle Line 2
            if (respects_precedence(individual, precedence_constraints_line1, 0, tasks_nt1) and
                    respects_precedence(individual, precedence_constraints_line2, tasks_nt1, total_tasks)):
                population.append(individual)
                break
    return np.array(population)
```

3. **Selection Function**

Listing 5.3: Selection Function

```python
def tournament_selection(population, fitness_scores, tournament_size=3):
    selected = []
    for _ in range(len(population)):
        competitors = random.sample(list(enumerate(fitness_scores)), tournament_size)
        winner = min(competitors, key=lambda x: x[1])[0]
        selected.append(population[winner])
    return np.array(selected)
```

4. **Crossover Function**

Listing 5.4: Crossover Function

```
    def single_point_crossover_with_precedence(parent1,
       parent2, precedence_constraints_line1,
       precedence_constraints_line2):
    point = random.randint(1, len(parent1) - 1)
    offspring1 = np.concatenate((parent1[:point],
       parent2[point:]))
    offspring2 = np.concatenate((parent2[:point],
       parent1[point:]))

    if (not respects_precedence(offspring1,
       precedence_constraints_line1, 0, tasks_nt1) or
            not respects_precedence(offspring1,
               precedence_constraints_line2, tasks_nt1,
               total_tasks)):
        offspring1 = parent1.copy()
    if (not respects_precedence(offspring2,
       precedence_constraints_line1, 0, tasks_nt1) or
            not respects_precedence(offspring2,
               precedence_constraints_line2, tasks_nt1,
               total_tasks)):
        offspring2 = parent2.copy()

    return offspring1, offspring2
```

5. **Mutation Function**

Listing 5.5: Mutation Function

```
def swap_mutation_with_precedence(individual,
   precedence_constraints_line1,
   precedence_constraints_line2):
    for _ in range(10):  # Try multiple swaps to find a
       valid one
        mutated = individual.copy()
        idx1, idx2 = random.sample(range(len(mutated)),
           2)
        mutated[idx1], mutated[idx2] = mutated[idx2],
           mutated[idx1]
```

```python
        if (respects_precedence(mutated, precedence_constraints_line1, 0, tasks_nt1) and
                respects_precedence(mutated, precedence_constraints_line2, tasks_nt1, total_tasks)):
            return mutated
    return individual  # Return original if no valid mutation found
```

6. **Zoning Constraint Function**

Listing 5.6: Zoning Constraints Function

```python
def apply_zoning_constraints(population, precedence_constraints_line1, precedence_constraints_line2):
    def zoning_check(task_a, task_b):
        for constraint in precedence_constraints_line1 + precedence_constraints_line2:
            if constraint == (task_a, task_b) or constraint == (task_b, task_a):
                return True  # Positive zoning
        return False  # Negative zoning

    for individual in population:
        for i in range(len(individual) - 1):
            for j in range(i + 1, len(individual)):
                if individual[i] // tasks_nt1 == individual[j] // tasks_nt1:  # Same line check
                    if not zoning_check(individual[i], individual[j]):
                        # Repair the solution by swapping
                        individual[i], individual[j] = individual[j], individual[i]
    return population
```

7. **Survivors Function**

Listing 5.7: Survivors Function

```python
def elitism_selection(population, fitness_scores,
   elite_size=2):
    elites = np.argsort(fitness_scores)[:elite_size]
    return population[elites]
```

8. **Genetic Algorithm Function**

Listing 5.8: Genetic Algorithm Function

```python
def
   genetic_algorithm_with_precedence_and_zoning(pop_size,
   generations, precedence_constraints_line1,
   precedence_constraints_line2):
    initial_assignment = assign_tasks_by_precedence()
    population =
       initialize_population_with_precedence(pop_size,
       initial_assignment,
       precedence_constraints_line1,
       precedence_constraints_line2)

    for generation in range(generations):
        fitness_scores =
           np.array([fitness_with_robot(individual,
           processing_times, robot_density) for
           individual in population])

        # Selection using tournament
        selected = tournament_selection(population,
           fitness_scores)

        # Generate next generation with
           precedence-respecting crossover
        next_generation = []
        for i in range(0, len(selected), 2):
            parent1 = selected[i]
            parent2 = selected[i + 1] if i + 1 <
               len(selected) else selected[0]
            offspring1, offspring2 =
               single_point_crossover_with_precedence(
                parent1, parent2,
                    precedence_constraints_line1,
                    precedence_constraints_line2)
            next_generation.append(offspring1)
            next_generation.append(offspring2)

        # Mutation with precedence-respecting mutation
        for individual in next_generation:
            if random.random() < 0.2:
                individual[:] =
                    swap_mutation_with_precedence(
                     individual,
                        precedence_constraints_line1,
                        precedence_constraints_line2)

        # Apply zoning constraints
        next_generation =
            apply_zoning_constraints(next_generation,
            precedence_constraints_line1,
            precedence_constraints_line2)

        # Apply elitism
        elites = elitism_selection(population,
            fitness_scores)
        next_generation = np.array(next_generation)
        next_generation[:len(elites)] = elites

        # Update population
        population = next_generation

    # Get the best solution
    best_index =
        np.argmin([fitness_with_robot(individual,
        processing_times, robot_density) for individual
        in population])
    return population[best_index]
```

Chapter 6

RESULTS AND DISCUSSION

The consequences of the proposed genetic algorithm are analyzed to check effectiveness in optimizing parallel meeting line balancing. The overall performance of all of the check instances is in comparison concerning cycle time, undertaking allocation efficiency, and constraint pleasure. This evaluation is presented through graph, chart, and Gantt diagrams for illustrating mission assignments and overall gadget performance. This phase also gives a essential dialogue at the found results, with a focus on strengths and limitations of the algorithm.

6.1 Cycle Times

Such a thing is often witnessed in graphs. Different bars describe different cycles for different folders, where each bar gives the time taken by the task processed through that folder. The x-axis refers to folder names, and the y-axis describes the respective cycle times in numerical units. Fig 6.1.1 and Fig 6.1.2 shows the cycle times of all dataset.
Observation in Detail:

- **Longest Cycle Time:** This folder, MANSOOR, contains the longest vertical bar in the graph, it reflects the highest cycle time, which is over 90 units, indicating that the amount of time taken to work on tasks or operations related to this folder is substantially longer than that of other folders.
- **Shortest Cycle Time:** This is the folder of mertens, with the shortest height of the bars by which it is indicated as this is the minimum cycle time of all, approximately 10 units, indicating that the tasks associated with this folder can be done much faster than any other folder's tasks.
- **Intermediate Cycle Times:**
  - The average points for MITCHELL and BOWMAN are in between, falling somewhere between 40 and 60 units. These two folders seem to be balanced, neither being the fastest at task completion nor being the slowest.
  - JACKSON and jaeschke have pretty much the same cycle times, being in the range of 20 to 30 units, which shows that the processing duration is between low and average.

- **Variation Across Folders:** The graph here shows variation in cycle times considerably from folder to folder, meaning they have a different demand in processing or complexity in the type of tasks tackled. The difference between the maximum and minimum cycle time is huge, with MANSOOR being more than 9 times slower than mertens.
- **Distribution Pattern:** Most folders (4 out of 6) will have cycle times less than 50 units, except for MITCHELL and BOWMAN who approach or go above that threshold. Hence, most tasks or data sets are quick to process, while only a few names take much longer.

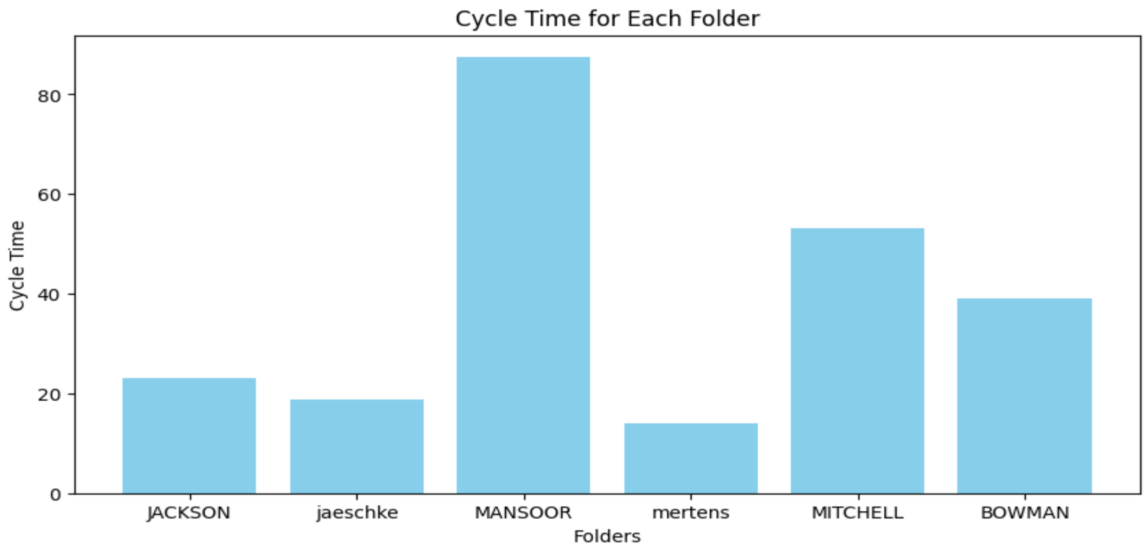


Figure 6.1.1: Cycle Time Bar Graph

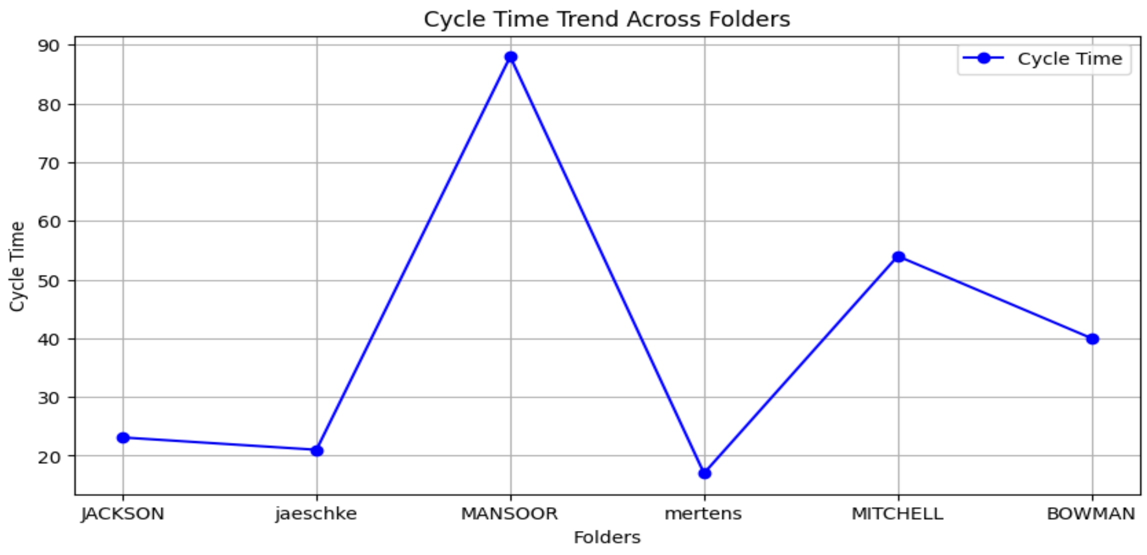


Figure 6.1.2: Cycle Time Line Graph

6.2 Zoning Constraint

The zoning percentages amounts per folder have been considered in this graph. The X-axis refers to the name of the folder, while the Y-axis shows the zoning constraint percentages of 0% - 100%. Each folder has its bar presented in two- the bar showing a positive zoning percentage sharing green color and another bar for negative zoning percentage in red color. Fig 6.2.1 and Fig 6.2.2 shows the zoning constraint percentages.
In Detailed Observations:

- **Positive Zoning (Green Bars):**
  - Positive zoning percentages have been low throughout the folders, being below 30% for most of the criteria.
  - Folders jaeschke, mertens, and BOWMAN are slightly ahead in zoning positive percentages with respect to JACKSON, MITCHELL, and MANSOOR. However even these are too deficient to show much inclination towards positive zoning.
- **Negative Zoning (Red Bars):**
  - Negative zoning tops all folders with values greater than 60% and at some point may even reach 100% (like in the case of MANSOOR).
  - The place that holds the highest at negative-zoning is MANSOOR, who has the 100% at representing all the negative-zoning-qualified violations in this folder.
  - Then there is JACKSON together with other folders like jaeschke, mertens, MITCHELL, BOWMAN ranking high in negative zoning with their values ranging approximately between 70
- **Comparative Results of the Above:**
  - A clear distinction is visible between green and red bars of all folders, which shows imbalance in it. All positive percentages seem to pale against negative percentages.
  - Thus, it indicates that negative zoning infringements have been general to all folders, whereas positive zoning constraints with very little compliance are met.
- **Folder-Specific Analysis:**
  - It cannot be considered that this folder has some positive zoning since this folder has an entirely negative zone; that is 0% positive with a red bar taking the complete height at 100%.

- The rest, jaeschke and BOWMAN, have around 20-30% higher positive zoning compared to the other folders.

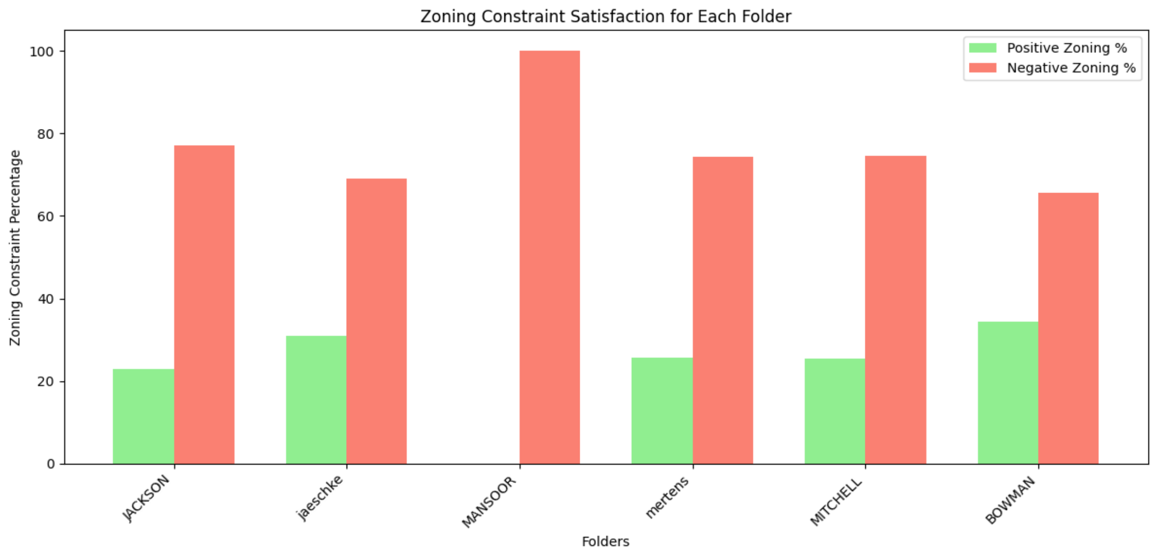


Figure 6.2.1: Zoning Constraint Percentage Bar Graph

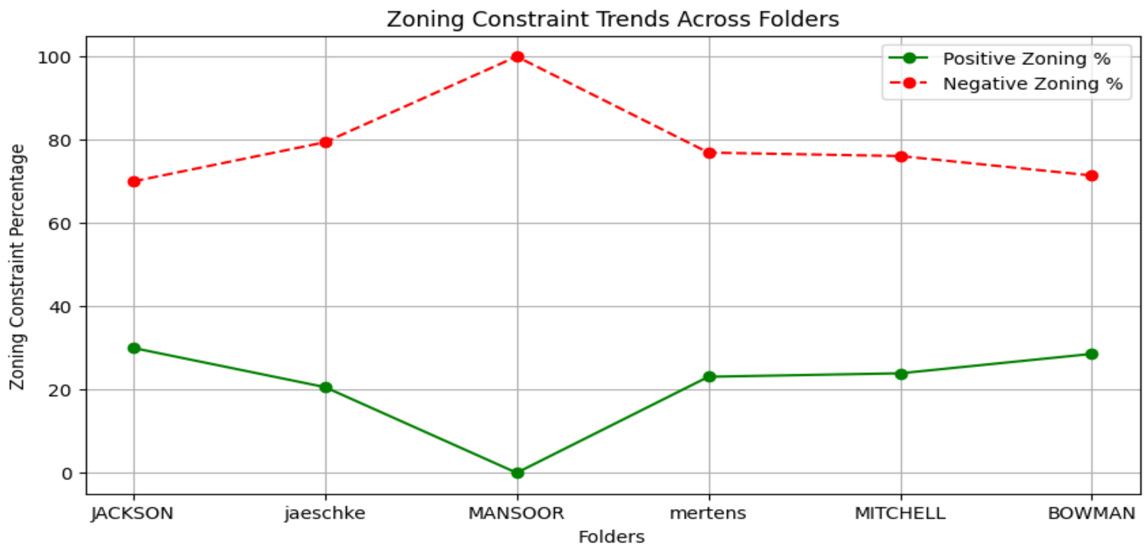


Figure 6.2.2: Zoning Constraint Percentage Line Graph

6.3 Assignment of Tasks to Stations

The created Gantt charts for each folder as a part of the assignment, which would give the visualization of the distribution of tasks at various stations. The Gantt charts would show how jobs are being assigned to different stations, with initiation timings and duration. These helped in judging the efficiency and timeline possibility of the assembly line operation.

A clear and visual representation of the task assignment process is an important tool to measure the feasibility and efficiency in the direction and allocation of tasks throughout the assembly line. Fig 6.3.1, Fig 6.3.2, Fig 6.3.3, Fig 6.3.4, Fig 6.3.5, Fig 6.3.6 shows the overall assignments of tasks to stations respectively.

Important observations:

- **X-axis (Time):** That is the time axis, during which the tasks will be performed.
- **Y-axis (Stations):** The stations are listed on the Y-axis, and each one corresponds to different task assignment within the folder.
- **Bars:** Each bar in the Gantt chart describes a task that is assigned to an individual station, where:
- The left facet of the bar corresponds to the time the project had began. The duration of the bar indicates the duration of the mission at that station.
- The Gantt charts are colour-coded by station, every with a one-of-a-kind colour. This permits easy identity of obligations assigned to the identical station. The duties are marked with their assignment number (as an example, ”project 1,” ”task 2”), and the legend could distinguish the different duties.

Folder-wise Analysis and Insights:

- **Task Distribution Across Stations:** The gantt charts in the folders show how tasks are distributed across various stations. For example, Folder A might get an even share of task distribution whereas Folder B will have some of the stations overloaded at certain times.
- **Task Scheduling and Duration:** The start and duration times of each task under each folder give a feeling of the well the tasks could be executed. A task could be at a station with a long duration, which could mean a bottleneck or improvement area.
- **Task Conflicts and Overlaps:** Gantt charts would show potential overlaps in work at a station. For instance, if two or more tasks are to start together at the same station, then it indicates a possible conflict.
- **Optimization Opportunities:** Gantt charts can show where tasks can be rescheduled or where the work can be redistributed to improve the overall efficiencies of the assembly line. For instance, two tasks, which are so close to each other at one station, can be moved up or down from the scheduled time to avoid idle time or congestion.

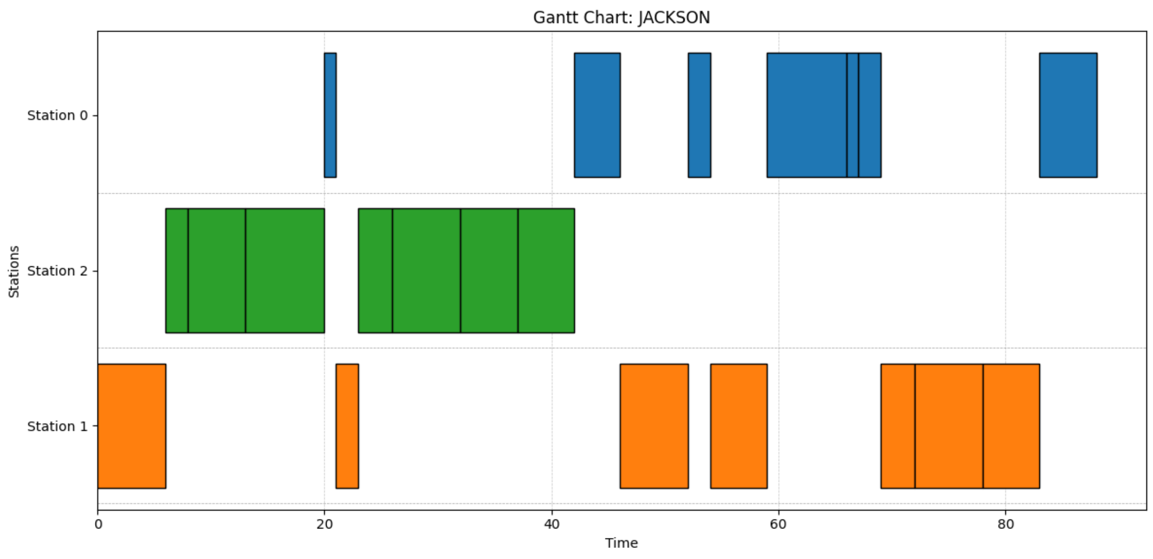


Figure 6.3.1: Task Assignment of Jackson Data

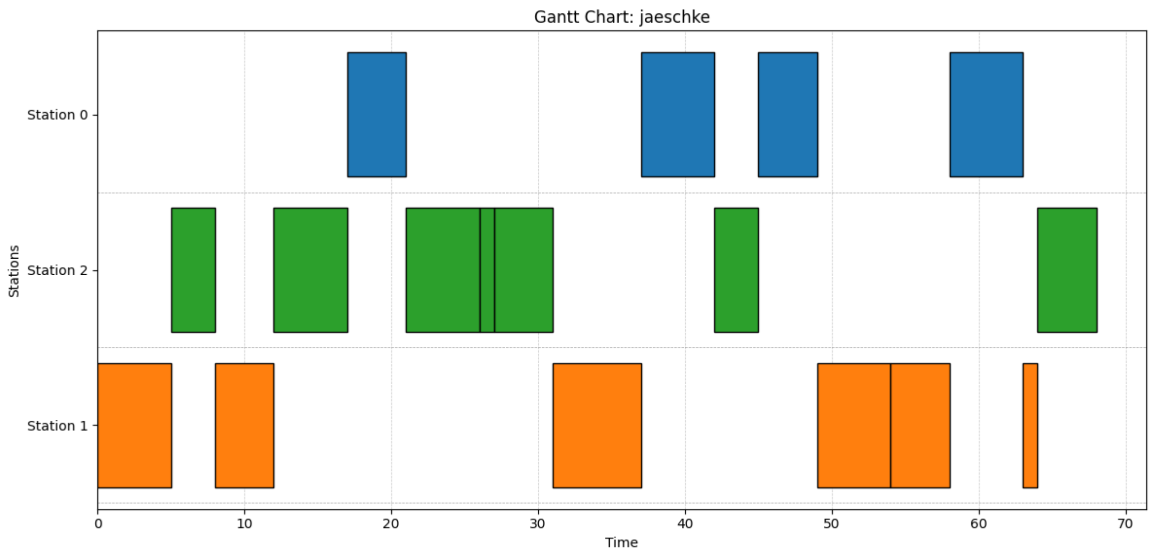


Figure 6.3.2: Task Assignment of jaeschke Data

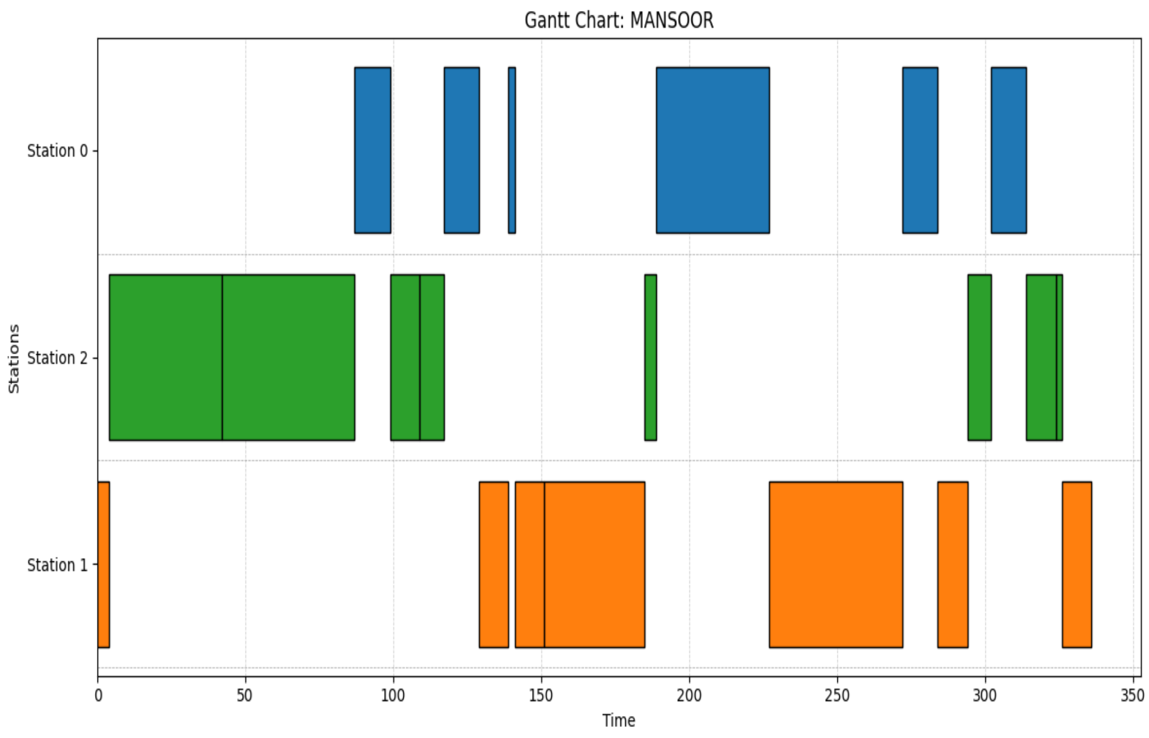


Figure 6.3.3: Task Assignment of MANSOOR Data

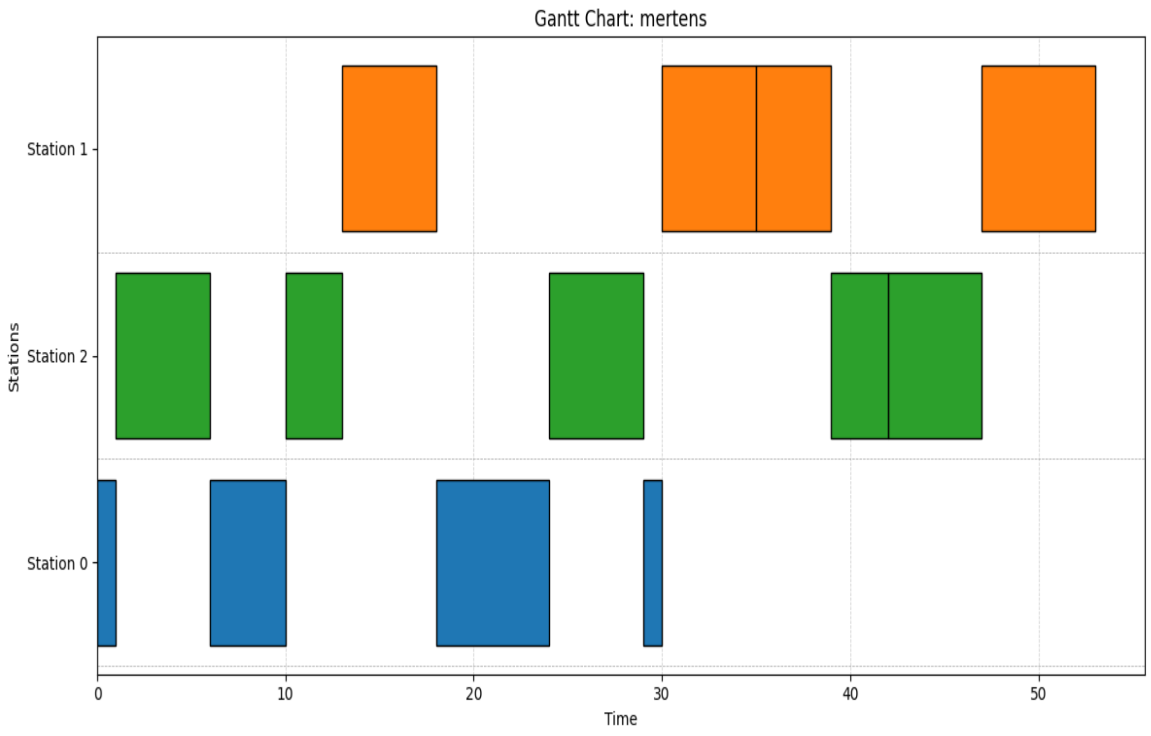


Figure 6.3.4: Task Assignment of mertens Data

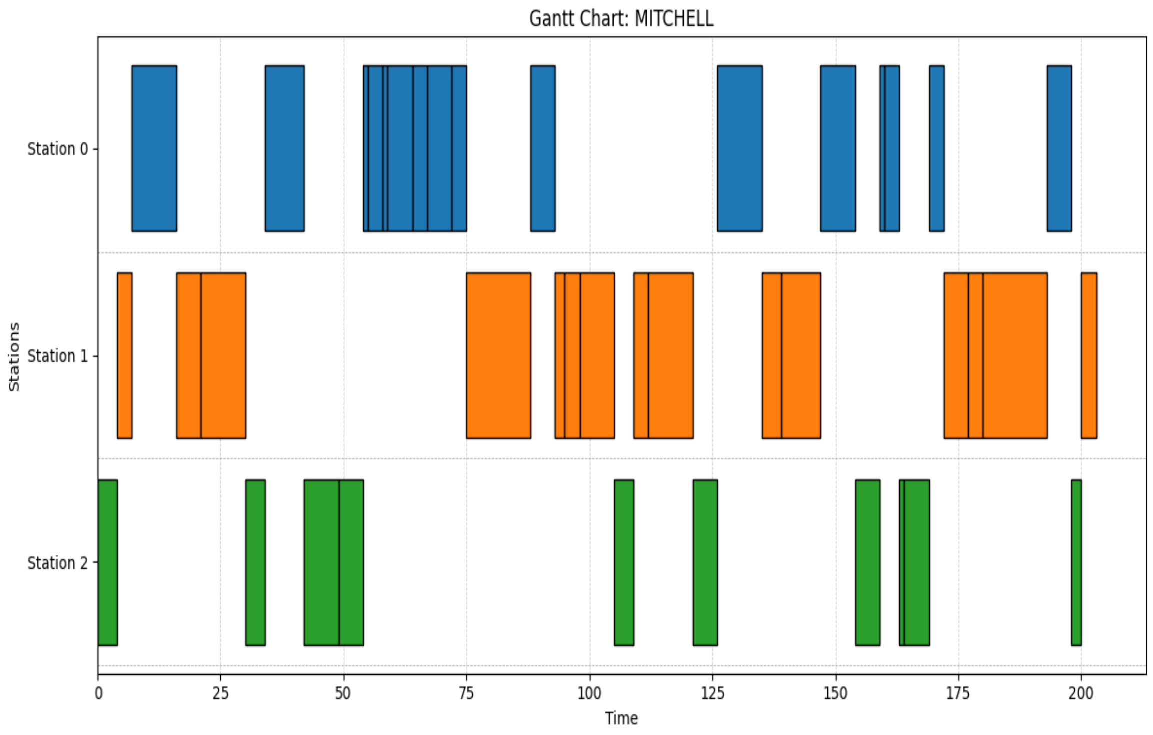


Figure 6.3.5: Task Assignment of MITCHELL Data

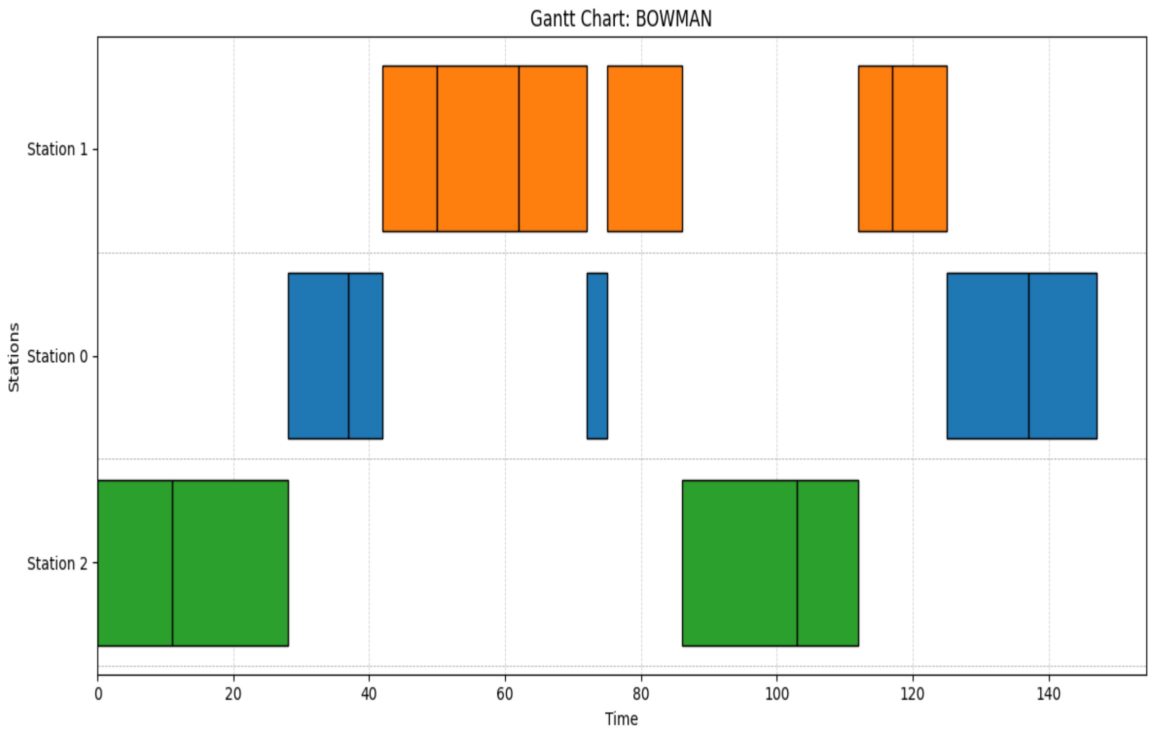


Figure 6.3.6: Task Assignment of BOWMAN Data

Chapter 7

CONCLUSION AND FUTURE WORK

The trouble lies within the allocation of responsibilities to stations, while the sub issues include the choice of processing alternatives for every of the responsibilities. A key factor is the zoning constraint, which lets in the coupling of two duties to be done on the identical station if they fall inside a pre-defined sector. this would allow the parallel execution of tasks such that it adds performance to the meeting line. Zoning constraints determine the challenge organization, i.e., it validates that a mission falls in a chosen zone so that the duties may be done concurrently, therefore nullifying useless time and increasing throughput. On a extraordinary notice, extending the idea of parallel lines with a cooperative format to distinct configurations, together with U-kind traces and -sided traces, will create a great opportunity for optimization. The mission was in finding out how exclusive layouts should have an effect on task assignment and whether or not or no longer line configurations could improve the overall line performance. extra static is the hassle concerning whether or not parallelism of the collaborative framework amongst exceptional varieties of lines can yield efficiency upgrades. this could include the interaction between multiple line types underneath one shared assignment allocation strategy. ultimately, as we strive to end up greater attentive to marketplace demands, the addition of combined fashions based totally on the above techniques remains a promising avenue. Merging mixed version production techniques with zoning constraints and cooperative line layouts would offer the agility now required to reply unexpectedly to changing marketplace conditions with out sacrificing performance across different project assignments.

References

[1] Z. Mao, J. Zhang, Y. Sun, K. Fang, and D. Huang, “Balancing parallel assembly lines with human-robot collaboration: problem definition, mathematical model and tabu search approach,” *INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 2024. [Online]. Available: https://doi.org/10.1080/00207543.2024.2356627

[2] A. Nourmohammadi, M. Fathi, and A. H. Ng, “Balancing and scheduling human-robot collaborated assembly lines with layout and objective consideration,” *Elsevier*, 2023. [Online]. Available: https://doi.org/10.1016/j.cie.2023.109775

[3] M. Fathi, A. Sepehri, M. Ghobakhloo, M. Iranmanesh, and M.-L. Tseng, “Balancing assembly lines with industrial and collaborative robots: Current trends and future research directions,” *Elsevier*, 2024. [Online]. Available: https://doi.org/10.1016/j.cie.2024.110254

[4] C. Weckenborg, K. Kieckhäfer, C. Müller, M. Grunewald, and T. S. Spengler, “Balancing of assembly lines with collaborative robots,” *Springer*, 2019. [Online]. Available: https://doi.org/10.1007/s40685-019-0101-y

[5] Z. Mao, J. Zhang, K. Fang, D. Huang, and Y. Sun, “Balancing u-type assembly lines with human-robot collaboration,” *Elsevier*, 2023. [Online]. Available: https://doi.org/10.1016/j.co

[6] H. Aguilar, A. García-Villoria, and R. Pastor, “A survey of the parallel assembly lines balancing problem,” *Computers & Operations Research*, vol. 124, p. 105061, 2020. [Online]. Available: https://doi.org/10.1016/j.cor.2020.105061

[7] F. Araújo, F. B. Costa, M. Alysson, and C. Miralles, “Balancing parallel assembly lines with disabled workers,” *European Journal of Industrial Engineering*, vol. 9, no. 3, pp. 344–365, 2015. [Online]. Available: https://doi.org/10.1504/EJIE.2015.069343

[8] O. Battaïa and A. Dolgui, “A taxonomy of line balancing problems and their solution approaches,” *International Journal of Production Economics*, vol. 142, no. 2, pp. 259–277, 2013. [Online]. Available: https://doi.org/10.1016/j.ijpe.2012.10.020

[9] N. Boysen, P. Schulze, and A. Scholl, “Assembly line balancing: What happened in the last fifteen years?” *European Journal of Operational Research*, vol. 301, no. 3, pp. 797–814, 2022. [Online]. Available: https://doi.org/10.1016/j.ejor.2021.11.043

[10] H. Çerçioğlu, U. Ozcan, H. Gökçen, and B. Toklu, "A simulated annealing approach for parallel assembly line balancing problem," *Journal of the Faculty of Engineering and Architecture of Gazi University*, vol. 24, no. 2, 2009.

[11] F. Chen, K. Sekiyama, F. Cannella, and T. Fukuda, "Optimal subtask allocation for human and robot collaboration within hybrid assembly system," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 4, pp. 1065–1075, 2013. [Online]. Available: https://doi.org/10.1109/TASE.2013.2274099

[12] P. Chutima, "Assembly line balancing with cobots: An extensive review and critiques," *International Journal of Industrial Engineering Computations*, vol. 14, no. 4, pp. 785–804, 2023. [Online]. Available: https://doi.org/10.5267/j.ijiec.2023.7.001

[13] P. Chutima and P. Jirachai, "Parallel u-shaped assembly line balancing with adaptive moea/d hybridized with bbo," *Journal of Industrial and Production Engineering*, vol. 37, no. 2-3, pp. 97–119, 2020. [Online]. Available: https://doi.org/10.1080/21681015.2020.1735544

[14] Z. A. cil, Z. Li, S. Mete, and E. Özceylan, "Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration," *Applied Soft Computing*, vol. 93, p. 106394, 2020. [Online]. Available: https://doi.org/10.1016/j.asoc.2020.106394

[15] M. Dalle Mura and G. Dini, "Designing assembly lines with humans and collaborative robots: A genetic approach," *CIRP Annals*, vol. 68, no. 1, pp. 1–4, 2019. [Online]. Available: https://doi.org/10.1016/j.cirp.2019.04.006

[16] E. Dar-El, "Malb–a heuristic technique for balancing large single-model assembly lines," *AIIE Transactions*, vol. 5, no. 4, pp. 343–356, 1973. [Online]. Available: https://doi.org/10.1080/05695557308974922

[17] Y. Delice, E. K. Aydoğan, U. Özcan, and M. S. İlkay, "Balancing two-sided u-type assembly lines using modified particle swarm optimization algorithm," *4OR*, vol. 15, no. 1, pp. 37–66, 2017. [Online]. Available: https://doi.org/10.1007/s10288-016-0320-4

[18] G. R. Esmaeilian, S. Sulaiman, N. Ismail, M. Hamedi, and M. M. H. M. Ahmad, "A tabu search approach for mixed-model parallel assembly line balancing problem (type ii)," *International Journal of Industrial and Systems Engineering*, vol. 8, no. 4, pp. 407–431, 2011. [Online]. Available: https://doi.org/10.1504/IJISE.2011.041803

[19] F. Glover, "Tabu search–part i," *ORSA Journal on Computing*, vol. 1, no. 3, pp. 190–206, 1989. [Online]. Available: https://doi.org/10.1287/ijoc.1.3.190

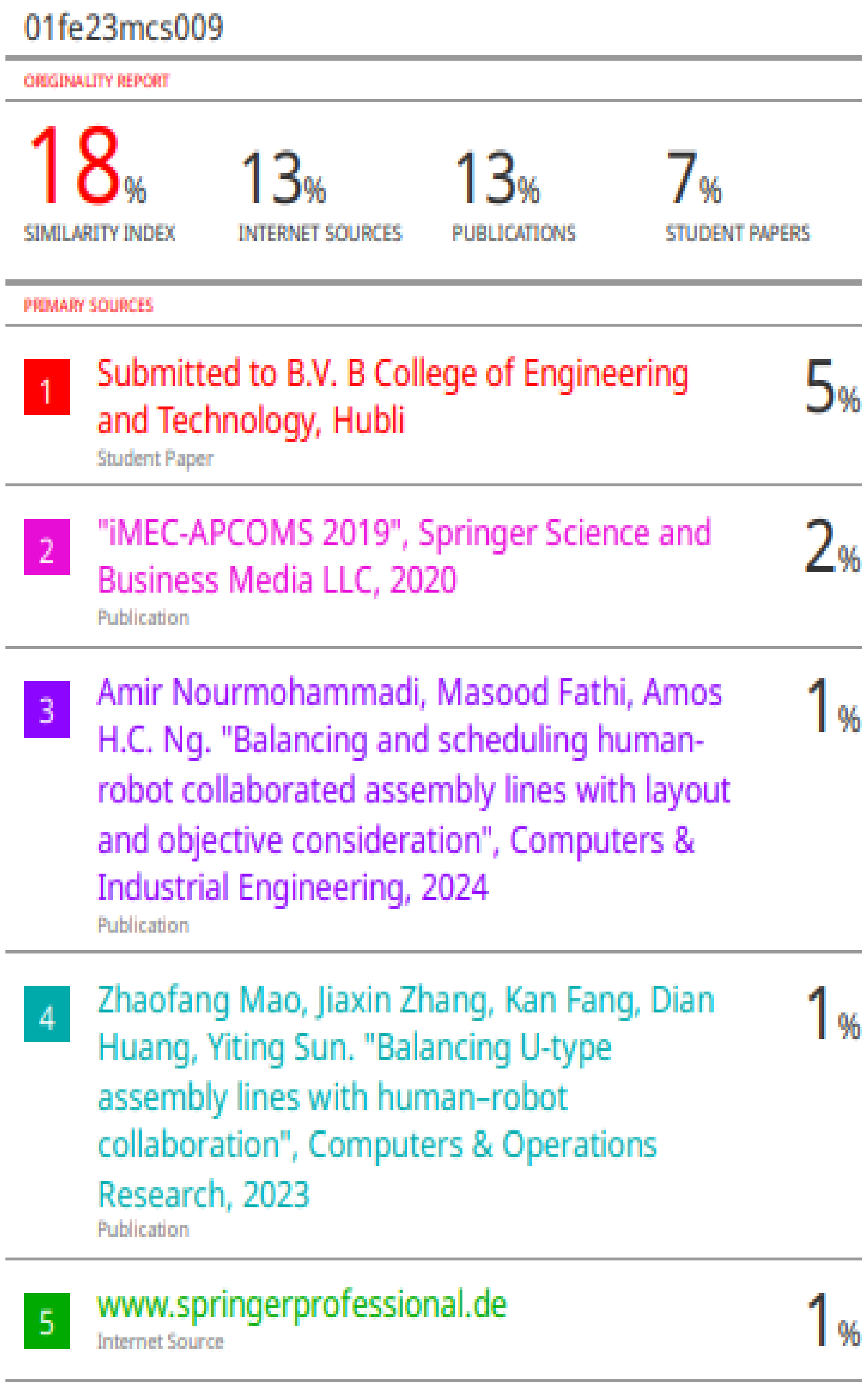
Appendix

Abbreviations

- **ALBP**: Assembly-Line-Balancing-Problem
- **PALBP**: Parallel-Assembly-Line-Balancing-Problem
- **HRC**: Human-Robot-Collaboration
- **GA**: Genetic-Algorithm
- **CT**: Cycle-Time
- **NS**: Number-of-Stations
- **MILP**: Mixed-Integer-Linear-Programming
- **U-ALBP**:U-shaped-Assembly-Line-Balancing-Problem

Experiment Environment

- Development Platform: Google Colab (Primary development and execution environment)
- Language: Python 3.x
- Main Libraries and Packages:
  - NumPy: array manipulation, mathematical computations.
  - Matplotlib: bar chart generation, Gantt chart generation, line graph.
  - Random: generation of random numbers to be used to initialize or mutate.
  - OS/ZipFile: File management, Data Set
- Hardware Configuration
  - GPU: Cloud GPU available with Google Colab. T4 from NVIDIA
  - RAM: 12 GB
  - Processor: Equivalent to i3 and above
  - Storage: Cloud Storage available along with Google Drive.
- Operating System: Linux-based (Google Colab environment) provide this in latex



01fe23mcs009

ORIGINALITY REPORT

18%	13%	13%	7%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to B.V. B College of Engineering and Technology, Hubli Student Paper	5%
2	"iMEC-APCOMS 2019", Springer Science and Business Media LLC, 2020 Publication	2%
3	Amir Nourmohammadi, Masood Fathi, Amos H.C. Ng. "Balancing and scheduling human-robot collaborated assembly lines with layout and objective consideration", Computers & Industrial Engineering, 2024 Publication	1%
4	Zhaofang Mao, Jiaxin Zhang, Kan Fang, Dian Huang, Yiting Sun. "Balancing U-type assembly lines with human–robot collaboration", Computers & Operations Research, 2023 Publication	1%
5	www.springerprofessional.de Internet Source	1%