SFLA-based Line Balancing and Task Optimization in Master-Slave Controlled Hanger Transportation Systems for Garment Production

Duc Hoang Nguyen
Faculty of Electrical and Electronics Engineering
HCMC University of Technology
Vietnam National University Ho Chi Minh City

ABSTRACT
Effective task allocation and workload balancing are critical challenges in garment production, particularly in systems that involve hanger transportation for moving garments between workstations. This paper presents a novel method for optimizing assembly line balancing by integrating the Shuffled Frog Leaping Algorithm (SFLA) with a task grouping strategy based on skill level, machine type, and precedence constraints, within a Master-Slave control framework for hanger transportation systems (HTS). The Master controller leverages SFLA to globally optimize task assignments aiming to minimize the number of stations, balance workloads, and reduce idle time, while ensuring task precedence and grouping requirements are met. Meanwhile, Slave controllers execute tasks locally and provide real-time feedback, facilitating dynamic adjustments based on system conditions. Simulations based on real-world garment assembly data demonstrate that the SFLA-based method significantly improves task allocation efficiency, reduces the number of stations, and enhances throughput compared to traditional approaches. This method increases the flexibility and adaptability of garment production lines, offering a robust solution for complex, dynamic manufacturing environments.

General Terms
Algorithms

Keywords
Optimization, SFLA, Line Balancing, Master-Slave Control, Hanger Transportation System, Garment Production.

1. INTRODUCTION

The garment production industry faces numerous challenges, particularly when managing complex and dynamic assembly lines that process different garment types and styles. Hanger Transportation Systems (HTS) are widely used to automate the movement of garments between workstations, such as cutting, sewing, and finishing. However, optimizing task assignments, balancing workloads across workstations, and minimizing bottlenecks in such systems remain significant challenges. Inefficient task assignments can lead to unbalanced lines, increased cycle times, idle workstations, and delays in production. To address these issues, effective line balancing and task optimization techniques are essential. Traditional methods for scheduling and balancing often struggle in dynamic environments, where factors like garment flow, worker availability, and machine performance fluctuate rapidly. Metaheuristic optimization algorithms such as Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) have shown promise in solving such complex problems due to their ability to explore large solution spaces and avoid local optima. Studies by Zhang et al. [1] and Wong et al. [2] have demonstrated the effectiveness of these algorithms in improving workload balancing and throughput in garment production systems. Additionally, hybrid metaheuristics, such as those developed by Hamta et al. [3] and Li et al. [4], address flexible operation times and sequence-dependent setup times, critical factors for efficient garment assembly lines. Moreover, Boysen et al. [5] have emphasized that modern assembly lines require optimization models that account for fluctuating production demands and complex precedence constraints.

The Shuffled Frog Leaping Algorithm (SFLA) has emerged as a powerful metaheuristic for optimization problems that involve both local and global search components. Initially proposed by Eusuff and Lansey [6], SFLA has been successfully applied to various engineering optimization problems, including water distribution networks, project scheduling, and machine layout optimization. SFLA's ability to combine local search (within memeplexes) and global exploration (via shuffling) makes it well-suited for complex, multi-objective optimization problems such as line balancing and task assignment.

Despite the effectiveness of SFLA in other domains, its application to HTS and garment production remains relatively unexplored. This paper proposes a novel approach to line balancing and task optimization in HTS by integrating the SFLA within a Master-Slave control framework. In this system, the Master controller uses SFLA to globally optimize task assignments while Slave controllers execute these tasks and provide real-time feedback for dynamic adjustments. The primary goal is to achieve balanced workloads across workstations, reduce cycle times, and improve overall system throughput. The proposed approach is evaluated through simulations of a garment assembly line. The results demonstrate that the integration of SFLA with the Master-Slave control system leads to significant improvements in line balancing and task optimization, ultimately enhancing the flexibility and efficiency of garment production processes. This work contributes to the growing body of research on applying metaheuristic algorithms to industrial automation and offers practical insights for optimizing HTS in real-world garment factories

The remaining paper is organized as follows: Section 2 details the methodology used, including the application of the SFLA for task allocation and line balancing. The Master-Slave control mechanism is explained. Section 3 presents the simulation results for a T-shirt production line using Technomatix Plant Simulation. Section 4 concludes the paper, highlighting the

feasibility of applying SFLA combined with a Master-Slave control system to optimize garment production lines.

2. PROPOSED METHODOLOGY

2.1 Overview of the Shuffled Frog Leaping Algorithm (SFLA)

The Shuffled Frog Leaping Algorithm (SFLA) is a population-based metaheuristic algorithm inspired by the natural behavior of frogs searching for food in their environment. It combines the benefits of both memetic and particle swarm optimization strategies by dividing the population into subgroups, known as memeplexes, where local search is conducted. After a certain number of local iterations, the frogs from different memeplexes are shuffled globally to exchange information and avoid local optima, thus improving global exploration. The SFLA operates with two major processes:

- **Local Search in Memeplexes**: Frogs within each memeplex perform local optimization by exchanging information to improve their current solutions based on their relative performance.
- **Global Shuffling**: After several rounds of local search, frogs are shuffled globally to prevent stagnation and diversify the search for better solutions.

In SFLA, each solution (frog) is represented as a set of decision variables that are optimized iteratively. The fitness function drives the optimization process, aiming to minimize or maximize a target objective depending on the problem. The algorithm has proven to be effective for combinatorial optimization problems due to its ability to balance exploration and exploitation [7-8].

2.2 Application to the Assembly Line Balancing Problem (ALBP)

The proposed methodology applies the SFLA to optimize task allocation and workload balancing in an Assembly Line Balancing Problem (ALBP), specifically in the garment production industry. The objective is to minimize the number of stations required while respecting cycle time, task precedence constraints, and the need to group tasks based on skill level and machine requirements. Each workstation (station) is assigned to handle a single task, with one or more machines depending on the task's processing time. Tasks with long processing times are divided across multiple machines when they exceed the cycle time.

The ALBP is modeled as a combinatorial optimization problem where the fitness function evaluates each frog (solution) based on the following criteria:

- **Minimizing the number of stations**: Solutions using fewer stations are prioritized.
- **Minimizing idle time**: The total idle time at each station is minimized to increase efficiency.
- **Task grouping compliance**: Penalties are applied if tasks requiring the same skill level or machine type are split between stations.
- **Precedence constraint violations**: Solutions violating the task precedence constraints are penalized to ensure proper task sequencing.

2.3 Task Grouping and Machine Allocation Strategy

A key innovation in the proposed approach is the use of task grouping based on shared characteristics such as skill level and machine type. Tasks are divided into groups to ensure that tasks requiring similar resources are assigned to the same workstation, minimizing machine changeovers and improving the ease of resource management.

For each task, the following parameters are considered:

- **Processing time:** The amount of time required to complete the task.
- **Skill level:** The worker expertise required for the task.
- **Machine type:** The specific machine or equipment needed to execute the task.

Tasks with long processing times are split across multiple machines at the same workstation to ensure that no task exceeds the defined cycle time. The SFLA ensures that tasks within the same group (i.e., those requiring the same skill level and machine type) are prioritized for allocation to the same workstation. This strategy reduces overhead and ensures smoother transitions between tasks.

2.4 Fitness Function Definition

The fitness function is crucial to guiding the SFLA optimization process. For each frog (solution), the fitness function evaluates how well the solution satisfies the objectives of the problem. The fitness function is defined as follows:

$$f(x) = \omega_1 NS + \omega_2 IT + \omega_3 PGV \quad (1)$$

where:

- NS = Number of Stations: The total number of stations used in the solution.
- IT = Idle Time: The total idle time across all stations.
- PGV = Penalty for Group Violations: A penalty applied if tasks that should be grouped together (based on skill level or machine type) are split between different stations.
- ω_1, ω_2, and ω_4 are positive weighting factors

By minimizing the fitness function, the algorithm ensures that the solution uses the fewest possible stations, minimizes idle time, and respects both task groupings and precedence constraints.

2.4.1 Idle Time

Idle Time refers to the amount of unused time at each workstation (station) during a production cycle. Since each workstation must complete its assigned tasks within the defined Cycle Time (C_T), any remaining time after completing the task(s) at a station is considered idle time.

$$IT = \sum_{i=1}^{n} (C_T - PT_i) \quad (2)$$

where:

- C_T is the cycle time,
- PT_i is the sum of the processing times for all tasks assigned to station i,
- n is the number of stations.

2.4.2 Penalty for Group Violations

Group Violations occur when tasks that should be grouped together (e.g., tasks requiring the same skill level or machine type) are split across different stations. Grouping tasks together reduces setup time and improves workflow efficiency, so violations incur a penalty.

Approach to Calculate Group Violations:

- ✓ **Define Groups**: Group tasks based on common characteristics (e.g., machine type, skill level).
- ✓ **Identify Violations**: For each task group, check whether tasks in the group are assigned to the same station. If they are split across multiple stations, it counts as a violation.
- ✓ **Apply Penalty**: For each group violation, a predefined penalty is applied.

$$PGV = \sum_{j=1}^{m} (PV * VG_j) \quad (3)$$

where:

- m is the number of task groups,
- VG_j is the number of violations in group j,
- PV is a constant penalty applied for each group violation.

2.5 Master-Slave Control System

2.5.1 Overview of Master-Slave Control

The Master-Slave Control System or Decentralized Control System is a hierarchical structure widely used in manufacturing, where the Master controller is responsible for global coordination and optimization, while decentralized Slave controllers manage the execution of specific tasks at each workstation. This structure allows for global decision-making while ensuring localized control, improving system adaptability and scalability [9-11].

The Master controller is designed to optimize the allocation of tasks, manage resources, and respond to changes in real-time, while Slave controllers execute tasks at workstations and provide feedback, enabling dynamic adjustments to the production process.

2.5.2 Master Controller Functions

The Master controller employs optimization techniques to assign tasks to workstations, ensuring efficient operation and resource utilization. Some of its primary functions include:

- **Global Task Assignment**: The Master controller uses optimization algorithms such as the Shuffled Frog Leaping Algorithm (SFLA) to allocate tasks in a way that balances workloads, minimizes idle time, and ensures tasks are completed in the correct order.
- **Hanger Routing Optimization**: It also optimizes the routing of hangers between workstations, reducing travel time and ensuring smooth transitions.
- **Minimizing Cycle Time**: By optimizing task assignments and reducing delays, the Master controller minimizes the overall cycle time, improving system throughput.
- **Handling Precedence Constraints**: The controller respects task precedence, ensuring that tasks are completed in the correct sequence, preventing bottlenecks.

2.5.3 Slave Controller Functions

Each Slave controller is responsible for executing tasks assigned by the Master controller and reporting real-time updates. The main functions include:

- **Local Task Execution**: The Slave controllers execute tasks using assigned resources, ensuring efficient operation.
- **Real-Time Feedback**: Slave controllers monitor task progress and provide updates to the Master controller. This feedback includes task status, machine performance, and any potential disruptions.
- **Dynamic Adjustments**: Based on the feedback, the Master controller can make adjustments to task assignments, resource allocation, or hanger routing to adapt to real-time changes.

2.5.4 Integration of Master-Slave Control with SFLA

The integration of SFLA within the Master-Slave system enhances task optimization and resource allocation in production systems. The Master controller uses SFLA to globally optimize task assignments, while the Slave controllers ensure local task execution. This enables the system to maintain flexibility and responsiveness in dynamic manufacturing environments.

- **Global Optimization**: The Master controller leverages SFLA to reduce the number of workstations and optimize workload distribution across the system.
- **Local Execution and Feedback**: Slave controllers provide real-time feedback, allowing dynamic adjustments based on current conditions.

2.6 Overview of Tecnomatix

Tecnomatix is a suite of digital manufacturing solutions offered by Siemens. It focuses on improving production efficiency through digital simulations and real-time optimization of manufacturing processes. Tecnomatix enables manufacturers to simulate, optimize, and validate their manufacturing systems and workflows virtually, reducing errors, improving time to market, and optimizing resources. It is commonly used for tasks such as production planning, process simulation, robotics, and ergonomic analysis [12].

Tecnomatix Plant Simulation is widely applied in optimizing production lines by modeling, simulating, and analyzing workflows. It helps manufacturers understand system behavior, identify bottlenecks, and evaluate "what-if" scenarios before physical implementation. Tecnomatix offers tools like Discrete Event Simulation (DES), allowing users to predict the performance of their production systems, adjust task scheduling, and optimize resource allocation [13-14].

In this paper, Tecnomatix Plant Simulation is utilized to model and optimize the garment production line by simulating task assignments, resource allocation, and hanger transportation across workstations. The software allows for detailed analysis of different configurations and scenarios, ensuring that cycle time constraints are met and minimizing the number of workstations. Tecnomatix provides a virtual environment to test task groupings, identify bottlenecks, and improve overall efficiency in the production process, making it ideal for solving the Assembly Line Balancing Problem (ALBP) in garment manufacturing.

3. SIMULATION RESULTS

3.1 Problem Formulation

The Assembly Line Balancing Problem (ALBP) in this paper focuses on optimizing task allocation and workload balancing for a T-shirt production system in the company TC. The system consists of 16 tasks, each with defined processing times, machine requirements, skill levels and precedence constraints as Table 1 and Fig. 1. The goal is to minimize the number of stations while ensuring all tasks are completed within a specified Cycle Time (C_T).

Table 1. Task List and Processing Time

#	*Task name*	*Machine type*	*Worker skill*	*Processing time(sec)*
1	Collar sewing	Single-needle	C	6.99
2	Stitching collar edge	Single-needle	C	11.80
3	Shoulder overlock	4-thread overhanging	B	14.14
4	Shoulder topstitch	Single-needle	B	11.88
5	Collar overlock	4-thread overhanging	B	30.18
6	Attaching back neck	Single-needle	B	25.41
7	Front neckline topstitch	Single-needle	B	24.72
8	Back neckline + label	Single-needle	B	16.23
9	Overlock sleeve	4-thread overhanging	B	29.46
10	Armhole finishing	Kansai	B	22.25
11	Side seam overlock	4-thread overhanging	B	39.27
12	Side seam topstitch	Single-needle	B	27.89
13	Sewing bottom rib	Single-needle	C	7.48
14	Bottom hem overlock	4-thread overhanging	B	28.07
15	Sewing cuff	Single-needle	B	6.13
16	Cuff overlock	4-thread overhanging	B	17.44

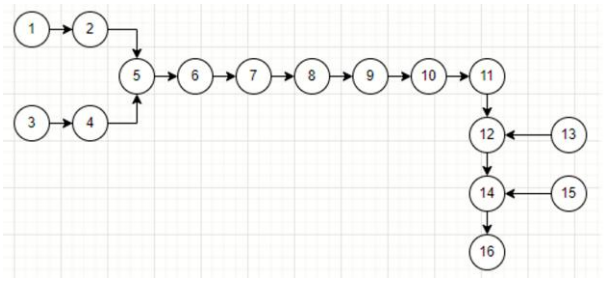

Fig. 1: Precedence constraints

3.2 Simulation parameters

- **Weighting factors of the fitness function**

The fitness function in this problem evaluates the efficiency of task assignments by focusing on minimizing the number of workstations while considering idle time and penalties for group violations. In the simulation, weighting factors are assigned as follows: $\omega_1 = 0.6$, $\omega_2 = 0.3$, and $\omega_3 = 0.1$, where ω_1 prioritizes minimizing the number of workstations, ω_2 targets reducing idle time, and ω_3 represents the penalty for group violations. Task precedence is treated as a constraint, ensuring all task dependencies are maintained.

- **SFLA parameters**

The parameter settings for the SFLA applied in this problem are provided in Table 2, outlining key algorithmic parameters such as population size, number of memeplexes, maximum iterations, and other essential values guiding the optimization process. These parameters are carefully selected to strike an optimal balance between exploration and exploitation, enhancing the algorithm's ability to effectively explore the solution space and optimize task allocation in the Assembly Line Balancing Problem.

Table 2. The SFLA parameter settings

G	n	m	$iter$	D
50	100	10	20	∞

- **Termination Criteria**

The termination criteria for the optimization process in this study are based on two key factors: stability of the fitness value and maximum number of iterations. First, an upper limit on iterations is set to prevent the process from running indefinitely and to conserve computational resources. The second criterion focuses on fitness stability – if the fitness value remains constant for a predetermined number of iterations, the optimization process is terminated. This implies that further improvement is unlikely. These criteria ensure efficient resource use while achieving high-quality solutions.

3.3 Task Assignment Results

Based on the production plan for one shift of the company, the SFLA algorithm returns the result of task distribution across groups, as shown in Table 3. The SFLA successfully optimized the distribution of tasks into groups, resulting in the allocation of tasks to a total of 20 workstations and 28 machines. This grouping and assignment ensure that tasks are completed efficiently within the shift's constraints.

- **Group 1** handles **Task 3** using 1 workstation and 1 machine, with a processing time of 14.14 seconds.
- **Group 2** handles **Task 4** using 1 workstation and 1 machine, with a processing time of 11.88 seconds.
- **Group 3** combines **Tasks 1 and 2**, requiring 1 workstation and 2 machines, with a combined processing time of 18.79 seconds.
- **Group 4** handles **Task 5**, which is one of the more time-intensive tasks, using 2 workstations and 2 machines for a total of 30.18 seconds.
- **Group 5** combines **Tasks 6 and 7**, requiring 3 workstations and 3 machines, with a processing time of 50.13 seconds.
- **Group 6** is dedicated to **Task 8**, using 1 workstation and 1 machine, with a processing time of 16.23 seconds.
- **Group 7** handles **Tasks 9 and 10**, utilizing 3 workstations and 6 machines to process in 51.71 seconds.

- **Group 8** is assigned to **Tasks 11 and 12**, the most time-consuming tasks in the process, requiring 4 workstations and 8 machines, with a processing time of 67.16 seconds.
- **Group 9** manages **Tasks 13 to 15**, using 1 workstation and 1 machine, processing in 13.61 seconds.
- **Group 10** combines **Tasks 14 to 16**, requiring 3 workstations and 3 machines for a total processing time of 45.51 seconds.

Table 3. Task group

Group	*Task*	*# workstations*	*# machines*	*Processing time(sec)*
1	1-2	1	2	18.79
2	3	1	1	14.14
3	4	1	1	11.88
4	5	2	2	30.18
5	6-7	3	3	50.13
6	8	1	1	16.23
7	9-10	3	6	51.71
8	11-12	4	8	67.16
9	13-15	1	1	13.61
10	14-16	3	3	45.51
Sum		20	28	319.34

The results in Table 3 illustrate the output of the SFLA optimization, where tasks are grouped based on their processing times, worker skills, and dependencies, ensuring that the production line operates smoothly throughout the shift. This optimized grouping minimizes idle time and increases the overall efficiency of the production process, aligning with the company's operational goals for the shift.

Furthermore, the flexibility of the SFLA ensures that the task assignments can be adjusted in real-time to respond to any unexpected changes in production conditions, such as machine downtimes or sudden changes in task priority. This adaptability helps maintain the continuity of the production line and contributes to higher throughput and reduced cycle times.

3.4 Simulation Results of Master-Slave Control

Based on the task grouping results as outlined above, a simulation model was developed using Technomatix Plant Simulation, as shown in Fig. 2. In which, the workstations correspond to the groups as shown in Table 4. The Master-Slave control mechanism was applied to this model to optimize task distribution and the flow of garments through the production line.

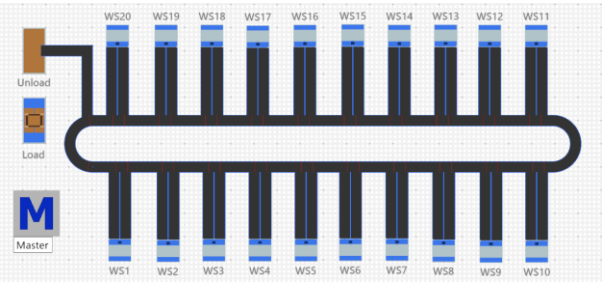

Fig. 2: Simulation model

Table 4. Workstations

Group	*# workstations*	*Workstations*
1	1	WS1
2	1	WS3
3	1	WS4
4	2	WS5, WS6
5	3	WS7, WS8, WS9
6	1	WS10
7	3	WS11, WS12, WS13
8	4	WS14, WS15, WS16, WS17
9	1	WS2
10	3	WS18, WS19, WS20

- **Simulation Results Before Balancing**

Statistics on the working time and waiting time of the 20 workstations are provided as shown in Fig. 3. Table 5 contains detailed data as well as the number of products entering and exiting each workstation.

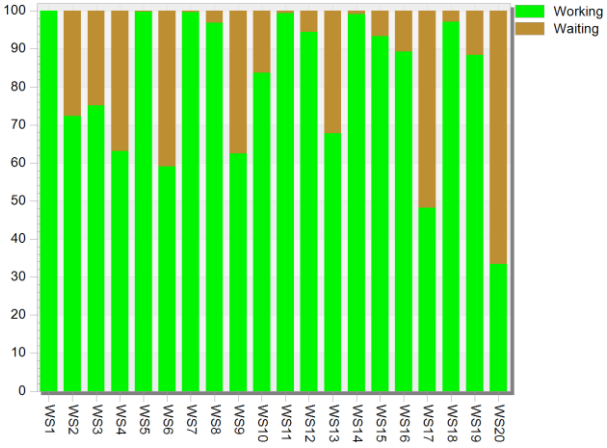

Fig. 3: Statistical results before balancing

Table 5. Statistical results before balancing

WS#	*Working*	*Waiting*	*NumIN*	*NumOut*
1	99.99%	0.01%	1725	1724
2	72.38%	27.62%	1723	1723
3	75.15%	24.85%	1722	1722
4	63.10%	36.90%	1721	1721
5	99.75%	0.25%	1071	1070
6	59.06%	40.94%	634	634
7	99.64%	0.36%	644	643
8	96.85%	3.15%	626	625
9	62.50%	37.50%	404	403
10	83.70%	16.30%	1671	1671
11	99.39%	0.61%	623	622
12	94.44%	5.56%	592	591
13	67.83%	32.17%	425	425
14	99.21%	0.79%	479	478
15	93.30%	6.70%	451	450
16	89.30%	10.70%	431	430
17	48.28%	51.72%	233	232
18	97.10%	2.90%	692	691
19	88.45%	11.55%	630	629
20	33.46%	66.54%	239	238

Based on the aforementioned results, the following observations can be drawn: The production quantity at each workstation within the same group exhibits significant disparities, predominantly concentrated at the first workstation of the group in the production line. This imbalance leads to extended waiting times for subsequent workstations and an inequitable distribution of workload across the stations, ultimately resulting in a suboptimal utilization of workstation productivity.

- **Simulation Results After Balancing**

Statistical results after balancing are provided as shown in Fig. 4. Table 6.

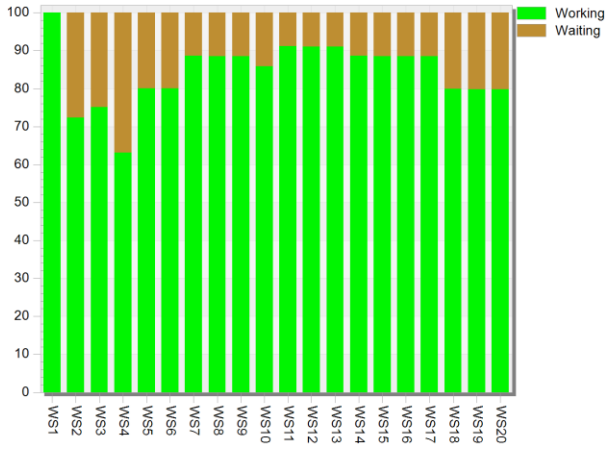

Fig. 4: Statistical results after balancing

Table 6. Statistical results after balancing

WS#	*Working*	*Waiting*	*NumIN*	*NumOut*
1	99.99%	0.01%	1725	1724
2	72.38%	27.62%	1723	1723
3	75.15%	24.85%	1722	1722
4	63.10%	36.90%	1721	1721
5	80.11%	19.89%	861	860
6	80.07%	19.93%	860	859
7	88.63%	11.37%	573	572
8	88.56%	11.44%	573	572
9	88.50%	11.50%	572	572
10	85.91%	14.09%	1716	1715
11	91.18%	8.82%	572	571
12	91.13%	8.87%	571	570
13	91.07%	8.93%	571	570
14	88.67%	11.33%	428	427
15	88.60%	11.40%	428	427
16	88.54%	11.46%	428	427
17	88.50%	11.50%	427	426
18	79.92%	20.08%	569	569
19	79.87%	20.13%	569	568
20	79.81%	20.19%	569	568

From the above results, we can make the following observations:

- The distribution of products to the workstations is performed uniformly, reflecting balance and optimization in the allocation of workload to each station. This results in the optimal utilization of all workstation resources, ensuring that no station is either underutilized or overburdened.
- Examining the workload statistics of the workstations reveals a more equitable and balanced distribution of tasks among them. This indicates that workstations within the same group, having similar configurations, will receive tasks uniformly.

- **Simulation of Various Incidents**

To further highlight the performance of the balancing algorithm, we will simulate two error cases as follows: Error Case 1 involves the critical workstation number 10 (where only one workstation performs the task in the line), and Error Case 2 involves workstation 7 in group 5 experiencing a malfunction. Both incidents result in a 30-minute halt after 4 hours of operation, followed by a return to normal operation until the end of the shift. The statistical results regarding the distribution of the workstations are illustrated in Fig. 5 and 6, respectively

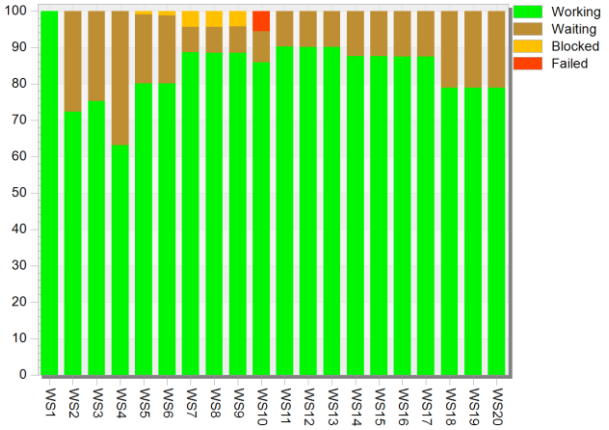

Fig. 5: Statistical results of Error Case 1

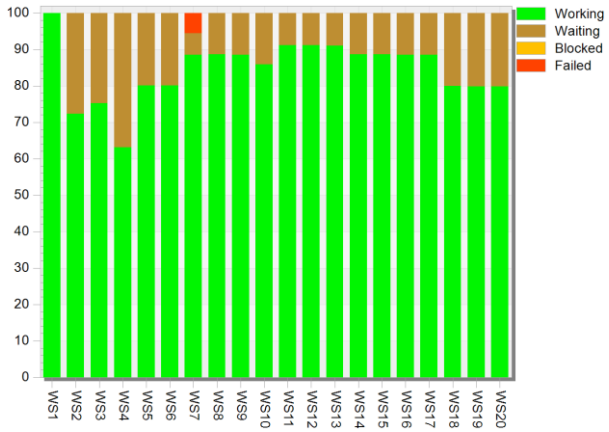

Fig. 6: Statistical results of Error Case 2

From the statistical charts of the workstation distribution, we can observe that the load remains balanced across the workstations. This indicates that when a malfunction occurs and is fully resolved, the dynamic load-balancing algorithm still demonstrates its effectiveness.

4. CONCLUSION

This study demonstrates the effectiveness of integrating the Shuffled Frog Leaping Algorithm (SFLA) within a Master-Slave control system for optimizing task assignment and routing in a Hanger Transportation System (HTS) for garment production. The simulation results highlight that the primary benefit of this approach was in achieving a more balanced workload distribution across workstations. This balance led to

smoother production flow and improved overall system performance.

The Master-Slave control architecture allowed for dynamic adjustments based on real-time feedback from the Slave Controllers, which enabled the system to adapt to changing conditions and prevent bottlenecks from stalling production. This flexibility proved to be a key strength of the system, allowing it to maintain high efficiency even when faced with unexpected changes in workload or task availability.

Moving forward, future research can explore the scalability of this approach to larger production lines and more complex garment assembly processes. Additionally, refining the fitness function to focus more heavily on cycle time or other specific objectives may further enhance the system's effectiveness in different manufacturing contexts. Overall, this study shows that the combination of SFLA and a Master-Slave control system offers a flexible and powerful solution for optimizing complex production systems like HTS in garment manufacturing.

5. ACKNOWLEDGMENT

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNU-HCM for supporting this study.

6. REFERENCES

[1] J. Zhang, Y. Zhang, and P. Wang. 2018. Application of intelligent hanging production system in garment industry. Advanced Manufacturing and Automation VII, Singapore: Springer.

[2] W.K. Wong, P.Y. Mok, and S.Y.S. Leung. 2013. Optimizing apparel production systems using genetic algorithms. Optimizing Decision Making in the Apparel Supply Chain Using Artificial Intelligence, Springer.

[3] Hamta, N., Fatemi Ghomi, S., Jolai, F., and Akbarpour Shirazi, M. 2012. A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. International Journal of Production Economics, 141(1), 99-111.

[4] Z. Li, I. Kucukkoc, and Q. Tang. 2017. New MILP model and station-oriented ant colony optimization algorithm for balancing U-type assembly lines. Comput. Ind. Eng., vol. 112, pp. 107–121.

[5] Boysen, N., Schulze, P., and Scholl, A. 2022. Assembly line balancing: What happened in the last fifteen years? European Journal of Operational Research, 301(3), 797-814.

[6] Eusuff, Muzaffar, Kevin Lansey, and Fayzul Pasha. 2006. Shuffled Frog-Leaping Algorithm: A Memetic Meta-Heuristic for Discrete Optimization. Engineering Optimization 38 (2): 129–54.

[7] Nguyen, DH., Ngo, MD. 2016. Comparing Convergence of PSO and SFLA Optimization Algorithms in Tuning Parameters of Fuzzy Logic Controller. In: Duy, V., Dao, T., Zelinka, I., Choi, HS., Chadli, M. (eds) AETA 2015: Recent Advances in Electrical Engineering and Related Sciences. Lecture Notes in Electrical Engineering, vol 371. Springer, Cham.

[8] Duc Hoang Nguyen. 2023. Modified DE-based Fuzzy PD Controller for UP6 Manipulator. International Journal of Computer Applications. 185, 38, 36-40.

[9] Tkáčik, Milan, Ján Jadlovský, Slávka Jadlovská, Anna Jadlovská, and Tomáš Tkáčik. 2024. Modeling and Analysis of Distributed Control Systems: Proposal of a Methodology. Processes 12, no. 1: 5.

[10] Y. Qamsane, A. Tajer and A. Philippot. 2014. Synthesis and implementation of distributed control for a flexible manufacturing system. Second World Conference on Complex Systems (WCCS), Agadir, Morocco, pp. 323-329

[11] Shahgholian, G. 2021. A brief review on microgrids: Operation, applications, modeling, and control. International Transactions on Electrical Energy Systems, 31(6), e12885.

[12] Tecnomatix Plant Simulation. 2024. Siemens. Accessed: Oct 17, 2024. [Online]. Available: https://plm.sw.siemens.com/en-US/tecnomatix/products/plant-simulation-software/

[13] More, Y.Y., Buktar, R.B. 2024. Investigating smart manufacturing process implementation in the Indian manufacturing industries using tecnomatix and response surface methodology. *Int J Interact Des Manuf.*

[14] Sobrino, D.R.D., Kotian, M., Škuba, M., Cazañas, R.D. 2024. A Complex Take into the Autonomous Supply Process to Workplaces in Manufacturing Companies: A Particular Case Study Supported by the Use of Tecnomatix Plant Simulation. In: Tamás, P., Bányai, T., Telek, P., Cservenák, Á. (eds) Advances in Digital Logistics, Logistics and Sustainability . CECOL 2024. Lecture Notes in Logistics. Springer, Cham.