Jurnal Teknik Industri, Vol. 28, No. 2, December 2026
ISSN 1411-2485 print / ISSN 2087-7439 online
DOI: https://doi.org/10.9744/jti.28.2.103-116

Optimization of Multi-Manned Assembly Line Balancing with Workload Smoothing using Simulated Annealing Based Hyper-Heuristic

Ferry Mario Sugiarta, Suharjito*
Industrial Engineering Department, BINUS Graduate Program – Master of Industrial Engineering,
Bina Nusantara University, Jakarta, 11480, Indonesia
Email: suharjito@binus.edu
*Corresponding author

Abstract: In most cases, an assembly line with multi-manned configuration will increase efficiency in productivity. However, the complexity of multi-manned assembly can sometimes lead to bottlenecks rather than improvements in productivity. The smoothness index is a crucial performance indicator in the assembly line balancing problem, as uneven smoothness can create bottlenecks and disparities among workers; unfortunately, it is often neglected. This study developed a model that represents the multi-manned assembly line balancing problem that considers workload smoothing. Although an exact method may give the most optimal answer, due to the NP hard nature of the problem, the exact method becomes infeasible as it will take time to solve. Although in some cases, metaheuristic or hyper-heuristic methods may not guarantee the best solution quality, a near optimal solution can be given in a more reasonable time. Data from real-world cases of assembly lines and benchmark datasets were used to test the model and examine the algorithm's performance. It was found that the hyper-heuristic method based on simulated annealing could find an optimal solution for the number of workers and stations, although it required a slightly longer computation time compared to pure metaheuristic methods.

Keywords: Multi-manned ALBP, workload smoothing, simulated annealing based hyper-heuristic.

Introduction

The assembly line plays an important role in flow-line production systems. A well-designed assembly line enables a company to produce standardized outputs in large quantities. In an assembly line, an unbalanced distribution of workload can lead to imbalances in the production flow, as the entire line pace depends on the station with the longest time [1]. This imbalance typically appears in the form of bottlenecks in the production process. One of the most common impacts of bottlenecks is waiting at the subsequent station. Many severe bottlenecks can negatively affect a company's level of efficiency. To prevent or reduce the impact of bottlenecks, manufacturers can adopt appropriate assembly-line configurations. Generally, assembly line configurations can be categorized into three main types: simple, two-sided, and multi-manned assembly lines [2]. In a simple assembly-line configuration, each workstation is assigned to a single worker. A two-sided assembly line allows two workers to operate simultaneously at each workstation from opposite sides of the line. A multi-manned assembly line is another type of assembly-line configuration in which one or more workers are assigned to a single workstation depending on the station workload. In most cases, multi-manned assembly lines may have advantages over simple assembly lines or two-sided assembly lines. These advantages may include shorter assembly lines, smaller work-in-progress (WIP), and lower material handling costs. However, not all instances can apply multi-manned assembly lines. The size of the product that is being assembled needs to be considered so that operators do not block one another [3]. Another thing is that designing a multi-manned assembly line is more complex than when designing or solving simple assembly-line balancing problems (ALBP). The complexity is due to the unavoidable idle time between operations in the workstations [4]. Several studies have addressed the multi-manned ALBP.

Several studies have applied optimization techniques to multi-manned assembly line balancing problems (MMALBP). To find the best solution, researchers can use exact methods to solve these optimization problems. However, the hard nature of MMALBP, where the amount of computation increases exponentially as the number of instances grows, resulting in a larger search space, has led most researchers to use heuristic or metaheuristic methods for optimizing such problems [2]. Some of the heuristic methods include largest

Submitted: April 28, 2026; Revised: June 23, 2026; Accepted: July 13, 2026

Table 1. Related research

Author	Problem			Objective				Solving method			
	Simple	Multi	Other	Worker	Station	Cost	Workload Smoothing	Exact	Heuristic	Metaheuristic	Hyper-heuristic
Roshani and Giglio [14]		O				O				O	
Sahin and Kellegoz [15]		O		O	O					O	
Zhang *et al.* [16]		O		O	O					O	
Pilati *et al.* [7]		O		O	O		O			O	
Hazir *et al.* [10]	O						O		O		
Dinler and Tural [17]	O						O	O			
Ozbakir and Secme [12]	O		Parallel Line Balancing			O					O
Muller and Bonilha [18]			Dynamic Optimization			O					O
Muklason *et al.* [13]			Transit Routing Problem			O					O
Current research		O		O			O				O

candidate rule, Kilbridge and Wester, ranked positional weight method [5], relax-and-fix procedure [6], and many more. Several metaheuristic algorithms that are usually used to solve ALBP are Simulated Annealing (SA) [7], Ant Colony Optimization [8], and Particle Swarm Optimization [9].

Workload smoothing in assembly lines aims to assign workload to workstations evenly, help in planning workforce, and to optimize the resources usage [10]. When designing or balancing an assembly line, workload smoothing or smoothing index need to be considered because of reasons such as to increase production output, establish a sense of equity among workers, reduce machine downtime, and reduce the risk of ergonomic issues [11]. Workload smoothing itself has been extensively explored in the context of simple assembly lines. In a multi-manned setting, workload smoothness is arguably even more critical; a fair task distribution balances idle time among stations and mitigates the bottleneck effect.

Despite this, research addressing MMALBP with the secondary objective of minimizing the smoothness index between workers remains underexplored. The problem domain of balancing multi-manned configurations has been prominently explored by Pilati *et al.* [7]. Pilati et al. successfully optimized metrics such as line length, station counts, and station-level smoothing, and solved it using mixed-integer linear programming and standard simulated annealing. However, their model only analyzed station-level smoothing and did not delve further to analyze worker-level stations. Özbakır and Seçme [12], on their research, introduced Simulated-Annealing Based Hyper-Heuristic framework for assembly line balancing problem (ALBP). Due to how hyper-heuristic algorithms operate at a higher level by managing low-level heuristics, on several studies that studied NP-hard optimization problems, they can outperform regular metaheuristics and able to give near-optimal solutions in reasonable computational times [13]. However, the assembly line balancing specific framework by Özbakır and Seçme was only tested on parallel ALBP driven by a cost-based objective function. It has yet to be tested on different types of assembly line configurations.

Table 1. summarizes the related studies that have been conducted. While previous studies have investigated MMALBP and the hyper-heuristic framework independently, a limited search has explored the application of the hyper-heuristic framework to MMALBP with worker-level smoothing. We take the multi-manned problem topology which was discussed by Pilati *et al.* [7] and solve it using a highly adaptive variant of the SA-HH framework proposed by Özbakır and Seçme [12]. Using data from both real-world commercial assembly lines and benchmark datasets, this study compares the performance of SA-HH against pure metaheuristics to optimize the number of workstations, number of workers, and workload fairness, offering practical insights for manufacturers.

The remainder of this paper outlines the research methodology and the data analysis procedures employed in this study. Following this section, we will present and discuss the results obtained from the research. The paper concludes by summarizing the main findings and providing recommendations for future research as well as practical applications of the results.

Methods

This research was conducted in four stages. First, a mathematical model that represents the multi-manned assembly line problem was developed. Second, benchmark and real-world case-study datasets were collected and prepared for processing. Third, the proposed Simulated Annealing Hyper-Heuristic and two other metaheuristics, namely ACO and SA, were implemented to solve the mathematical model. Finally, the results from the three algorithms were compared using several performance measurements, such as the number of workers, smoothness index, and computational time.

Proposed Model

First, the problem in this research will take two functions, number of workers and workload smoothing, as its objective functions. In this problem, the number of workers is solved first, and then the secondary objective is solved. Eq. (2) in this study covers the objective of solving this problem. The equation can be separated into two segments, the first one ($\sum_s w_s$) corresponds to the minimization number of workers objective. The second part of the equations corresponds to minimizing the smoothness index.

$$Min\ Z = \sum_s w_s + \varepsilon \sqrt{\sum_{k \in K} (W_{max} - W_k)^2} \quad (1)$$

ε in this model is sufficiently small number act as weighting factor, it is to make sure that the second part remain as secondary objective and does not have much influence on the primary objective. In this paper, we will define the ε as $1x10^{-5}$.
Subject to:

$$\sum_{s \in S} x_{is} = 1 \quad \forall i \in N \quad (2)$$
$$\sum_{s \in S} s \times x_{is} \leq \sum_{s \in S} s \times x_{js} \quad \forall (i,j) \in P \quad (3)$$
$$\sum_{i \in N} a_i x_{is} \leq CT \times w_s \quad \forall s \in S \quad (4)$$
$$w_s \geq m_i \times x_{is} \quad \forall i \in N, \forall s \in S \quad (5)$$
$$x_{is} \leq y_s \quad \forall s \in S \quad (6)$$
$$y_s \leq w_s \leq M \times y_s \quad \forall s \in S \quad (7)$$
$$W_{max} \geq W_k \quad (8)$$

Decision variables:

$$x_{is} \in \{0,1\} \quad (9)$$
$$z_{ik} \in \{0,1\} \quad (10)$$
$$y_s \in \{0,1\} \quad (11)$$
$$w_s \in \{0,1,\ldots,M\} \quad (12)$$

Where:
i, j : index of tasks
s : index of workstations
k : index of workers
N : set of all tasks
S : set of all candidate workstations
K : set of all workers
P : set of precedence relations
w_s : number of workers assigned to stations-s
t_i : process time for task-i
CT : cycle time
M : maximum number of workers per station
m_i : required number of manpower to perform task-i
W_{max} : maximum worker workload across all station ($W_{max} = max(W_k)$)
W_k : total workload of worker-k ($W_k = \sum_{i \in N} t_i z_{ik}$)
a_i : workload total of task-i ($a_i = m_i \times t_i$)
ε : balancing weight parameter

$y_s \in \{0,1\}$	1 if station-s is used; 0 if station-s not used
$x_{is} \in \{0,1\}$	1 if task-i is assigned to station-s; 0 if task-i is not assigned to station-s
$z_{ik} \in \{0,1\}$	1 if worker-k participates in task-i; 0 if worker-k did not participate in task-i

Equation (2) ensures that each task is assigned to only one workstation. Equation (3) guarantees that task precedence and the order of operations are maintained. Equation (4) ensures that the workload assigned to each station does not exceed its available capacity. Equation (5) verifies that the number of workers at each station meets the required number necessary to perform specific tasks. Equation (6) confirms that tasks are assigned only to active stations. Finally, Equation (7) ensures that the number of workers falls within the established lower and upper limits.

Dataset

This study uses data obtained from real-world cases. Assembly data from the commercial HVAC assembly line will be utilized. As previously explained, multi-manned configurations can only be applied to assembly lines producing sufficiently large products. Therefore, a case study of large-capacity air conditioning unit assembly, where unit sizes can reach approximately 3 m x 3 m x 2 m, is suitable for this kind of configuration. Data will be collected through direct observation in the field and from existing documents. The data to be collected in this study include the process time data for each task, the required manpower to perform said task, and the sequence of the assembly process. The data presented in this study are publicly available in Sugiarta [19].

Alongside the dataset obtained from the real-world case study, benchmark datasets from Boysen *et al.* [20] will also be included. These benchmark datasets will be used to evaluate the performance of both algorithms across various problem sizes, ranging from small-scale ($1 \leq N \leq 30$) to medium scale ($31 \leq N \leq 70$) and large-scale ($N > 70$) challenges. The benchmark datasets to be assessed include the Bowman ($N = 8$), Mitchell ($N = 21$), Heskiaoff ($N = 28$), Sawyer ($N = 30$), Kilbridge ($N = 45$), Warnecke ($N = 57$), Tonge ($N = 70$), and Arc ($N = 83$), Lutz ($N = 89$), Barthold ($N = 148$) and Scholl ($N = 297$) problems. However, since the dataset only provides task processing times and precedence relationships, additional parameters are necessary to execute the proposed model. Several adjustments, including cycle time, minimum number of workers for each task, and maximum number of workers per station are introduced under specific assumptions. For cycle time, it was set to match the longest processing time for each problem to ensure that assignments are feasible. The minimum number of workers is set to one, as the original data set was developed specifically for simple assembly problems. It is important to note that this assumption does not eliminate the multi-worker aspect of the system, as multiple workers can still be assigned to the same stations. Additionally, the maximum number of workers per station has been determined based on the size of the problem. Given that the context of the dataset is unknown, it is assumed that larger problems indicate more complex assembly lines, which can accommodate more workers per station. Consequently, we have set the maximum number of workers as follows: For small-scale problems, two workers are assigned to each station; for medium-scale problems, four workers; and for large-scale problems, six workers.

Solving Algorithm

Dey defines Ant Colony Optimization (ACO) as a metaheuristic optimization method capable of solving combinatorial optimization problems within an acceptable computational time [21]. Fattahi *et al.* [3] previously researched the multi-manned ALBP problem using the ACO method. In their study, they evaluated 18 different combinations of ACO parameters to determine their effectiveness in solving the problem. Table 2 shows the optimal parameter combination identified for the multi-manned ALBP.

Simulated Annealing (SA) is an optimization method based on a concept from physics known as material annealing, in which a material is heated and then gradually cooled [22]. The SA method begins at an initial temperature, allowing the algorithm to explore the solution space. To avoid being trapped in local optima, the algorithm may accept worse solutions with a certain acceptance probability, which is determined by the temperature that decreases over time [23]. After completing one iteration, the temperature is reduced according to the cooling ratio. This process is repeated until the temperature reaches a final temperature. Pilati *et al.* [7] conducted a study to optimize the multi-manned ALBP problem using the SA method.

Table 2. Previous research best parameter setting

ACO [3]		SA [7]	
Parameter	Value	Parameter	Value
Pheromone influence (α)	0.25	Cooling ratio (Λ)	0.9
Heuristic influence (β)	1.00	Epoch Length (K)	50
Evaporation rate (ρ)	0.35	Initial Temperature (T_i)	21°
Pheromone deposit constant (q)	1.00	Final Temperature (T_f)	1.5°
Number of ants	20.00	Probability of acceptance (XP)	15%
Number of iterations	80.00	Cycle time fraction of SI^{min} (S)	3%

They identified a set of parameters, as shown in Table 2, that yielded optimal results for the tested problems. This study adopts both the ACO and SA methods with the same parameter settings as one of the benchmark approaches for comparison.

Simulated Annealing based Hyper-Heuristic (SA-HH) is search strategy that manages and selects sets of low-level heuristic (LLH) on each iteration, or decoding, during the optimization process. Unlike regular metaheuristics, which explore directly within the solution space, a hyper-heuristic approach operates at a higher level by managing and selecting the LLH. The set of LLH is then responsible for constructing and exploring candidate solution during the search process. Thus, rather than searching directly within solution space, the hyper-heuristic searches within the space of heuristic combinations.

In the proposed SA-HH, Simulated Annealing acts as the high-level controller that determines the acceptance and rejection of the sequence and combination of LLH used during the search process. Thus, the exploration of the solution space is performed by the LLH, while the Simulated Annealing mechanism steers the search towards the combinations of LLH that may give better results. The proposed method employs ten LLHs to control task-selection decisions during solution construction. These heuristics consist of Longest Processing Time (LLH1), Shortest Processing Time (LLH2), Maximum Number of Predecessor Tasks (LLH3), Minimum Number of Predecessor Tasks (LLH4), Maximum Number of Successor Tasks (LLH5), Minimum Number of Successor Tasks (LLH6) and Maximum Total Processing Time of Predecessor Task (LLH7) Minimum Total Processing Time of Predecessor Task (LLH8), Maximum Total Processing Time of Successor Tasks (LLH9), and Minimum Total Processing Time of Successor Tasks (LLH10). These ten LLHs are chosen based on their simple structures, adaptability for all types of assembly line, ease of implementation, and lower computation times [12].

Figure 1 shows the pseudocode of the SA-HH algorithm used in this experiment. The LLH is encoded in a rule vector corresponding to one of the ten LLH. Each position in the rule vector represents a decision point during the constructive decoding process. The LLH is then selected through the controlled Adaptive Heuristic Selection (AHS) mechanism. A weight matrix is maintained in which each row corresponds to a decision position and

SA-HH Pseudocode:

```
Initialization
Construct AHS
Construct LLH vector
Generate initial solution
While Non-improved iteration < Max non-improved iteration:
   a. Create a new LLH vector by making small changes (Swap).
   b. Decode the vector into a new solution (adjacent solution).
   c. Evaluate the new solution.
   If the new solution is Better:
      - Accept the adjacent solution.
      - Positive update of heuristic rating.
      - Update global best.
   Else, check the Random Acceptance Ratio:
      - If accepted, then:
         * Accept the adjacent solution.
         * Positive update of heuristic rating.
      - If rejected, then:
         * Reject the adjacent solution.
         * Negative update of heuristic rating.
   d. Update AHS rating, temperature, and iteration number.
Display Best Solution.
Finish
```

Figure 1. SA-HH Pseudocode

each column corresponds to an LLH. Initially, all heuristics are assigned equal weights. During the search process, heuristics are selected probabilistically according to their weights. Whenever a generated solution improves the incumbent best solution, the weights associated with the selected heuristics are rewarded with positive points. A solution that is worse but accepted will also be rewarded with lower positive points. While solution that is worse and rejected will receive a point deduction. This adaptive mechanism allows the algorithm to gradually choose heuristics that demonstrate better performance during the search and encourage exploration of the search space, thus helping to avoid becoming trapped in local optima.

In SA-HH, the neighborhood structure operates on the rule vector rather than directly on task assignments. A new neighboring solution candidate is generated by making small changes to the rule vector. The rule vector position is selected randomly and is mutated by replacing the current heuristic with another heuristic chosen from the AHS matrix. The mutation size in this experiment is temperature-dependent, allowing larger modifications during early stages and smaller modifications when the temperature is sufficiently low. The resulting rule vector is used during the decoding process to select which task from the feasible task list to be assigned to that position.

The decoding subprocess transforms a rule vector into a complete assembly-line solution through a COMSOAL-like methodology. First, a new workstation is opened, and all feasible tasks whose precedence constraints have been satisfied are listed. From the feasible task list, the task is ranked according to the LLH in the rule vector. Candidate tasks are evaluated to check whether they violate the station or manning capacity. The highest-ranked feasible task is assigned to the current workstation, after which the workstation workload, worker requirements, and feasible task list are updated. This process is repeated until no additional task can be assigned to the current workstation, after which a new workstation is opened, and the procedure continues. The decoding is completed once all tasks have been assigned, which gives a complete assembly line configuration. The complete configuration is then evaluated using the objective function. The pseudocode for the decoding process is shown in Figure 2.

Meanwhile, the acceptance strategy follows the Simulated Annealing (SA) framework. At each iteration, a new candidate solution is generated and compared with the current solution. Δ represent the difference between the objective values of the candidate solution and the current solution. If the candidate solution has a better objective value, it is always accepted and becomes the new current solution. If the candidate solution is worse than the current solution, it may still be accepted with a probability given by $(P = \exp(-\Delta/T))$, where T is the current temperature. During the search, the temperature gradually reduced following its cooling ratio. We set the lower bound for the temperature, so that the temperature will never go down below that value, which may result in algorithm behaving greedily and trapped in local optimum. As the temperature decreases, the algorithm becomes more selective and increasingly focuses on improving solution quality. The search process will stop once the search didn't give any improvement best solution after 5,000 iterations.

Since all the tested algorithms are stochastic optimization methods, their search and final solutions may vary across different executions due to the use of pseudo-random processes.

```
Initialization
While there are unassigned tasks Do
   Open a new station
   While current station is not full Do
      1. Identify all feasible tasks
      2. Read the next LLH heuristic from the vector
      3. Rank feasible tasks and select the highest-rank
      If selected task fits the station capacity Then
         Assign task to the current station
         If all tasks are assigned Then
            Terminate and return the final solution
      Else If another feasible task fits the station Then
         Retry selection with the alternative task
      Else
         Mark the current station as full
Finish
```

Figure 2. Decoding pseudocode

Table 3. Parameter for SA-HH

Parameter	SA-HH
Initial temperature	Auto calibrated ($\frac{\Delta}{-\ln(P_0)}$)
Target acceptance	95%
Final temperature	$T < 1x10^{-6}$
Probability of acceptance	$\exp(-\Delta/T)$

To ensure reproducibility, each run was executed independently using ten predefined random seeds: 18432, 59103, 72814, 11905, 88172, 44211, 90124, 33781, 77502, and 66219. These seed values were randomly selected as fixed integers for initializing the pseudo-random number generator. To ensure fair comparisons, all three algorithms will be executed with the same set of seeds. In this paper, we limit the number of random seeds for each instance to 10 seeds, as it is considered to provide enough variety while keeping the total computing time practical.

Evaluation

The results from the optimization will then be evaluated and compared between algorithms to see which gives the best results. The evaluation matrix that will be compared in this study includes the total number of workers, total number of working stations, smoothness index between workers, value from multi-manned ALBP with smoothness index objective function, and computation time. The total number of workers is the total number of workers stationed among all open stations (w_s). In the model formulation, the total number of workers is presented as shown in Eq. (13).

$$\sum_s w_s \tag{13}$$

Workload smoothing or smoothness index refers to how even work is distributed among workstations. A lower smoothness index value indicates a more uniform balance between stations. To calculate workload smoothing between workers, we use Eq. (14). Eq. (14). represents the workload variance among workers, where W_k are the total average workload of worker k and W_{max} is the maximum workload across all the stations.

$$\sqrt{\sum(W_{max} - W_k)^2} \tag{14}$$

The objective function value is derived from the value used as the objective of the problem being discussed. The primary objective is to minimize the number of workers. Among the results with the same primary objective value, the smoothness index is used as a tiebreaker to select the best task distribution combination. To calculate the objective function value, Eq. (1) can be used. Lastly, computation time is how long it takes for the algorithm to reach the best solution that it can find

Results and Discussions

In this section, the algorithm results between the current or existing assembly line configuration used by the company are presented. All the algorithms were developed in Python that ran on a 64-bit Windows PC, with an AMD Ryzen 7 9800X3D (8 Core; 4.7 GHz) CPU and 32.0 GB of RAM.

Parameter Calibration

Parameter calibration was conducted beforehand to determine the influence of the SA-HH parameter on the solution quality. Three parameters were selected for evaluation: cooling factor, epoch length, and reward mechanism. The cooling factor controls the rate at which the temperature decreases during the annealing process. Larger values result in slower cooling, allowing the algorithm to maintain exploration for a longer period. Meanwhile, smaller cooling values mean more focus on the exploitation stage as it tries to quickly move from exploration. Four cooling factors were tested: 0.90, 0.95, 0.97, and 0.99. The epoch length determines the number of neighbourhood evaluations performed before the temperature is updated. A larger epoch allows for a more intensive search at each temperature level, where a smaller epoch enables more frequent temperature adjustments. However, as the epoch becomes larger, the search also becomes more intensive, resulting in a computational time that may increase as well. Epoch lengths of 10, 75, and 150 were tested. The reward mechanism affects the heuristic selection probabilities within the hyper-heuristic frameworks. Three reward configurations representing different learning reinforcements were tested. The first configuration applied a weak reinforcement strategy, where better solution, worse solution but accepted, and worse selection but

rejected, were rewarded 1, 0.3, -0.2, respectively. Stronger levels of reinforcement strategy, 5, 1.5, -1, and 10, 3, -2, were also tested.

Combining the four cooling factors, three epoch lengths, and three reward configurations resulted in a total of 36 parameter combinations. Each combination was evaluated through multiple independent runs on three chosen benchmark datasets representing small to large-scale problems, and the resulting solution quality and computational performance were compared to identify the most suitable parameter settings for each benchmark instance. Result of the parameter testing can be seen on Table 4. For both the small and large datasets, the primary objective value, which is the number of workers, remained the same across all tests. However, the medium-scale problem highlighted a clear trade-off behaviour between the primary and secondary objective based on parameter tuning. In several configurations, such as cooling ratio 0.97, epoch 10, and reward of 10, 3, -2, the algorithm achieved its lowest smoothness index but did so at the expense of the primary objective requiring 31 workers instead of 30. Because of the way the problem is modelled, the algorithm should prioritize worker optimization before smoothness optimization. In the case of the algorithm prioritizing smoothness optimization instead, it represents a case where the instance is trapped in a local optimum.

To achieve near-true optimal balance, the parameter must allow and balance exploration and exploitation during the search process. The results of parameter tuning show that the configuration of cooling ratio (Λ) = 0.95, Epoch = 75, and Reward = 1, 0.3, −0.2 consistently produced high-quality solutions across the Mitchell, Warnecke, and Barthold datasets.

Table 4. Parameter tuning result

Λ	Epoch	Reward	Mitchell			Warnecke			Barthold		
			Z	*SI*	Time (second)	*Z*	*SI*	Time (second)	*Z*	*SI*	Time (second)
0.90	10	1, 0.3, -0.2	9.00006	6.164	1.309	30.00180	179.989	9.289	15.00632	631.794	121.368
0.90	10	5, 1.5, -1	9.00014	14.248	1.134	30.00263	262.633	9.808	15.00665	665.408	50.260
0.90	10	10, 3, -2	9.00006	6.164	2.571	30.00230	230.404	5.683	15.00639	638.797	62.253
0.90	75	1, 0.3, -0.2	9.00008	8.185	1.184	30.00217	216.596	15.498	15.00649	649.149	64.716
0.90	75	5, 1.5, -1	9.00009	8.660	0.843	30.00210	209.833	9.697	15.00613	613.339	54.340
0.90	75	10, 3, -2	9.00006	6.164	2.737	30.00282	282.276	11.236	15.00667	666.504	72.500
0.90	150	1, 0.3, -0.2	9.00014	14.248	1.885	31.00138	137.732	6.787	15.00594	593.630	66.622
0.90	150	5, 1.5, -1	9.00006	6.164	1.814	30.00182	182.499	7.672	15.00650	649.813	63.442
0.90	150	10, 3, -2	9.00006	6.164	2.489	30.00230	229.648	13.161	15.00632	631.657	62.744
0.95	10	1, 0.3, -0.2	9.00014	14.248	1.369	30.00312	312.170	14.370	15.00665	665.131	46.887
0.95	10	5, 1.5, -1	9.00014	14.248	1.134	31.00150	150.043	5.934	15.00649	648.757	64.524
0.95	10	10, 3, -2	9.00006	6.164	2.620	30.00171	170.874	17.383	15.00636	635.716	53.242
0.95	75	1, 0.3, -0.2	9.00006	6.164	1.296	30.00155	155.016	8.898	15.00580	580.085	84.874
0.95	75	5, 1.5, -1	9.00006	6.325	1.870	30.00173	172.841	11.920	15.00634	634.132	78.262
0.95	75	10, 3, -2	9.00008	8.307	0.948	31.00131	130.694	9.470	15.00687	687.015	66.471
0.95	150	1, 0.3, -0.2	9.00006	6.164	1.533	30.00215	214.513	7.051	15.00692	691.576	59.301
0.95	150	5, 1.5, -1	9.00014	14.248	1.123	30.00173	172.662	10.958	15.00667	666.562	87.195
0.95	150	10, 3, -2	9.00014	14.248	1.204	30.00175	175.317	7.811	15.00603	603.214	92.305
0.97	10	1, 0.3, -0.2	9.00006	6.164	1.288	30.00229	229.024	8.166	15.00652	652.175	93.234
0.97	10	5, 1.5, -1	9.00014	14.248	1.144	30.00162	162.241	9.477	15.00669	669.375	95.969
0.97	10	10, 3, -2	9.00014	14.248	1.218	31.00129	129.395	13.054	15.00683	683.144	84.510
0.97	75	1, 0.3, -0.2	9.00006	6.164	1.233	30.00161	161.146	14.492	15.00656	656.443	59.507
0.97	75	5, 1.5, -1	9.00014	14.248	1.125	30.00228	227.987	11.597	15.00634	633.698	79.779
0.97	75	10, 3, -2	9.00014	14.248	1.196	30.00171	171.038	18.670	15.00633	632.945	108.495
0.97	150	1, 0.3, -0.2	9.00009	8.775	1.028	30.00206	205.879	11.094	15.00694	693.628	52.996
0.97	150	5, 1.5, -1	9.00014	14.177	1.434	30.00166	165.517	11.011	15.00694	694.243	57.026
0.97	150	10, 3, -2	9.00014	14.248	1.211	30.00206	205.869	7.189	15.00655	655.276	88.898
0.99	10	1, 0.3, -0.2	9.00014	14.248	1.750	30.00235	234.768	10.764	15.00669	669.388	76.824
0.99	10	5, 1.5, -1	9.00008	8.307	1.416	30.00214	214.210	14.758	15.00633	632.988	110.939
0.99	10	10, 3, -2	9.00008	8.307	1.003	30.00180	180.427	11.957	15.00643	643.030	47.714
0.99	75	1, 0.3, -0.2	9.00006	6.164	1.501	30.00174	173.603	9.316	15.00670	669.877	64.537
0.99	75	5, 1.5, -1	9.00014	14.248	1.144	30.00183	183.396	7.460	15.00678	677.939	47.161
0.99	75	10, 3, -2	9.00014	14.248	1.216	30.00246	246.183	9.132	15.00597	597.080	63.285
0.99	150	1, 0.3, -0.2	9.00014	14.248	1.739	30.00214	213.659	6.442	15.00655	655.361	59.601
0.99	150	5, 1.5, -1	9.00009	8.66025	0.9482	30.00229	229.399	8.578	15.00624	623.701	78.744
0.99	150	10, 3, -2	9.00014	14.248	1.217	30.00170	169.765	19.385	15.00581	581.096	82.325

Table 5. Result comparison table

No	Dataset	N	C_T (second)	M	ACO				SA				SA-HH (Proposed model)			
					Z	w_s	SI	Computation time (second)	Z	w_s	SI	Computation time (second)	Z	w_s	SI	Computation time (second)
1	Bowman	8	17	2	5.000122	5	12.25	0.049	5.000073	5	7.35	0.044	5.000073	5	7.35	0.241
2	Mitchell	21	13	2	9.000253	9	25.27	0.189	9.000130	9	12.98	0.150	9.000062	9	6.16	1.210
3	Heskiaoff	28	116	2	10.000025	10	246.20	0.423	10.000006	10	57.78	0.327	10.000005	10	51.34	1.919
4	Sawyer	30	25	2	14.000639	14	63.86	0.402	14.000236	14	23.57	0.315	14.000187	14	18.65	3.321
5	Kilbridge	45	55	4	11.000994	11	99.42	0.812	11.000593	11	59.31	0.642	11.000436	11	43.65	3.171
6	Warnecke	57	53	4	31.002600	31	259.96	1.430	30.002195	30	219.50	1.153	30.001550	30	155.02	8.984
7	Tonge	70	294	4	23.000065	23	653.55	2.030	23.000045	23	446.09	1.661	23.000034	23	344.05	9.848
8	Arcus	83	3691	6	21.131740	21	13174.03	2.552	21.068480	21	6848.02	2.056	21.063291	21	6329.14	9.154
9	Lutz	89	10	6	49.000720	49	72.00	3.305	49.000507	49	50.72	2.563	49.000419	49	41.86	17.071
10	Barthold	148	383	6	15.011628	15	1162.80	12.988	15.007072	15	707.19	10.320	15.005801	15	580.08	84.874
11	Scholl	297	1386	6	51.092151	51	9215.11	78.245	51.055816	51	5581.63	64.225	51.042964	51	4296.42	528.664

Table 6. Optimized worker's workload distribution

No	Parameter	Current condition	ACO	SA	SA-HH
1	Total worker	19	8	8	8
2	Total station	12	3	3	3
3	Smoothness index	1,504.09	642.316	562.686	538.126
4	Line efficiency	48.37%	88.21%	88.21%	88.21%
5	Computational time (second)	-	1.1498	1.0139	5.477

Table 7. Workload allocation after optimization

Station	worker	Task			Load		
		ACO	SA	SA-HH	ACO	SA	SA-HH
1	1	1, 40, 4, 6, 12, 24, 9, 25, 15, 17, 48	1, 2, 40, 6, 12, 15, 24, 47, 25, 17, 46, 28	39, 40, 4, 6, 13, 15, 25, 8, 26, 17, 28, 10	805	845	859
	2	1, 3, 5, 7, 13, 8, 27, 25, 26, 46, 28	1, 3, 5, 7, 13, 8, 24, 27, 26, 18, 10	1, 3, 5, 12, 14, 24, 25, 8, 9, 18, 46	745	761	782
	3	39, 2, 6, 12, 24, 8, 14, 47, 16, 18, 29	39, 4, 6, 12, 14, 8, 9, 25, 16, 48, 10	1, 2, 6, 12, 7, 24, 47, 16, 27, 48, 10	907	814	779
2	1	10, 11, 32, 23, 22, 35, 36, 37, 44	11, 29, 20, 23, 33, 35, 42, 38, 45	11, 20, 21, 29, 30, 32, 37, 44	894	840	825
	2	10, 30, 19, 23, 33, 35, 36, 42	11, 30, 21, 22, 34, 31, 36, 43, 49	11, 23, 22, 34, 35, 36, 31, 38	839	793	818
	3	11, 30, 20, 21, 34, 31, 41, 44	19, 30, 23, 32, 35, 41, 36, 37, 50	19, 23, 33, 30, 35, 36, 44	733	875	865
3	1	38, 45, 50, 52, 54, 56, 57, 58	44, 51, 53, 55, 56, 58	43, 45, 50, 51, 53, 55, 56, 58	585	430	450
	2	43, 49, 51, 53, 55, 56, 58	44, 52, 54, 56, 57, 58	41, 49, 42, 52, 54, 56, 57, 58	420	570	550
	3	-	-	-	-	-	-

This configuration generated the best objective values for the medium- and large-scale benchmarks while remaining highly competitive on the small-scale benchmark. Thus, this combination of setting parameter was used for all SA-HH executed in this paper.

Benchmark Result

Table 5 shows a comparison of the performance of the three methods tested, namely, ACO, SA, and SA-HH. The methods were evaluated using several datasets, which are intended to represent their performance when applied to problems ranging from small-scale to large-scale. Although the three methods were able to achieve identical solutions in terms of the primary objective, SA-HH produced lower smoothness-index values compared to the others. Lower smoothing-index values, that SA-HH able to achieve indicate that SA-HH able to give solution with more balanced workload distribution among the workers.

However, the improvement in solution quality achieved by the SA-HH method comes at the cost of increased computational time. Based on the results in Table 5, the SA method has the fastest average computational time, at 7.587 seconds. This is faster than the ACO method, which requires an average of 9.311 seconds, or approximately 22.7% longer than SA. Meanwhile, the SA-HH method requires the longest average computational time, at 60.769 seconds, or approximately 701% longer than the fastest method.

As can be seen from Figure 3., due to the nature of the problem (NP-hard), all three methods show an exponential growth trend in computational time as the problem size increases. For smaller problem sizes, the increase in computational time is relatively low. However, as the problem grows larger, a steep increase in the curve can be observed. In the problem, a steep increment can be seen particularly in problems where the size is beyond 80. Among the three methods tested however, SA-HH shows the highest growth.

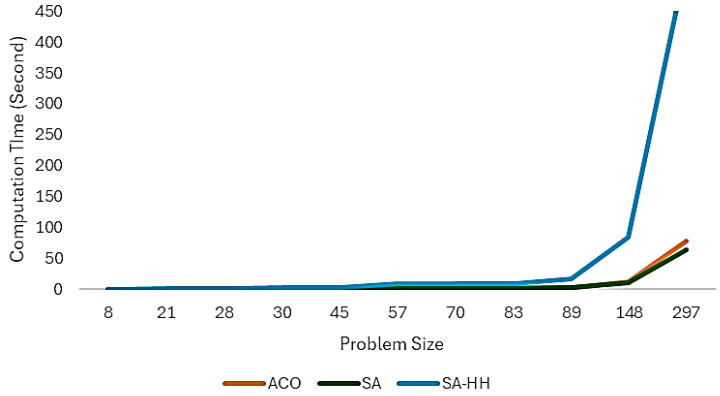


Figure 3. Computational time trend relative to problem size (N)

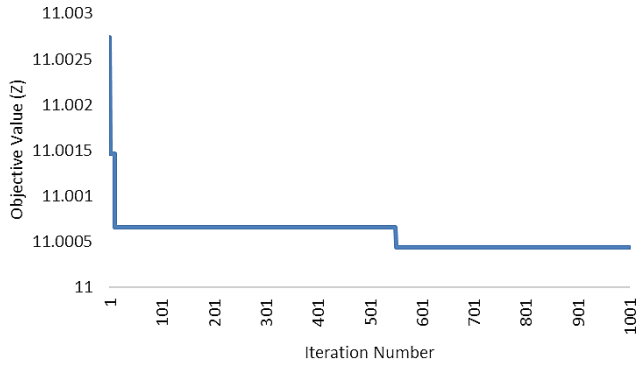


Figure 4. Convergence curves of SA-HH

Next, the algorithm was applied to the case study problem, which the full data can be found in the data availability section of this paper. Unlike the benchmark dataset, for this problem, we set the available production time or takt time as the cycle time input for the model. By using takt time as the cycle time input, optimization should provide an overview of how to set up the assembly line to achieve the daily unit production target in the most efficient way. The current initial condition was 30 units per day or approximately 840 seconds to assemble one unit. Furthermore, for this problem, we set the minimum number of workers for each task following the actual conditions, where some light works were assigned to only one worker, whereas tasks requiring greater physical effort were assigned to two or more workers. Finally, due to the space availability of the assembly line, the maximum number of workers on each station was set to four workers.

Algorithm Performance

Figure 4 presents the convergence behavior of the proposed SA-HH algorithm. The convergence curve shown is based on a randomly selected Kilbridge benchmark dataset. Although only one convergence curve is presented, all benchmark instances exhibited a similar convergence pattern. It can be observed that the objective value decreases rapidly during the early iterations, this behavior indicates that the algorithm is able to quickly identify promising regions of the search space. Subsequently, the solution remains relatively stable for several iterations until another better solutions are obtained. This behavior demonstrates a desirable balance between exploration and exploitation. The early rapid improvements indicate effective exploration of the solution space, whereas the later part reflects the capability of the AHS mechanism to exploit promising research regions to further improve the quality of the solution obtained.

As can be seen on Table 5. SA-HH able to consistently achieve a better solution across all tested datasets. In smaller problems, standard SA can navigate the limited solution space as effectively. However, when faced with problems with larger solution spaces, such as problems with larger tasks, standard metaheuristics often become trapped within local optima. SA-HH able to overcome this scalability issue due to how it searches the space of heuristic combinations rather than the solution space itself. Where instead of swapping individual tasks between stations, SA-HH searches across the permutation of the rule vector. This search method allows the

Table 8. LLH frequencies in the SA-HH final iteration for each problem instance

Dataset	LLH1	LLH2	LLH3	LLH4	LLH5	LLH6	LLH7	LLH8	LLH9	LLH10
Bowman	0	0	1	1	1	0	1	2	1	1
Mitchell	0	2	4	3	2	3	2	2	3	0
Heskiaoff	1	4	5	2	4	1	3	2	2	4
Sawyer	2	3	6	1	3	3	2	3	4	3
Kilbridge	6	4	2	6	7	3	4	4	4	5
Warnecke	6	7	4	5	5	3	5	6	12	5
Tonge	11	5	5	6	7	3	10	8	5	10
Arc	8	16	7	10	11	3	9	2	9	8
Lutz	6	12	9	5	7	10	9	14	8	9
Barthold	19	17	11	8	18	14	14	18	18	11
Scholl	37	34	25	31	30	26	33	24	30	27

algorithm to evaluate each task assignment from multiple operational perspectives, such as processing times, predecessor dependencies, and successor dependencies, which was indicated by the low-level heuristic composed the rule vector. This flexibility allows the algorithm to easily escape the local optima that may trap standard metaheuristics.

However, a trade-off in computational time should be expected for the SA-HH algorithm. As shown in Figure 4, the difference in the computational time between the SA-HH and other metaheuristics is particularly distinct in larger-scale problems. This increase occurs because of how the SA-HH operates. Rather than relying on a simple direct permutation and evaluation, SA-HH requires two steps of algorithmic overhead. At each iteration, AHS needs to update the matrix based on its performance score. After the AHS matrix is updated, the decoder must construct the entire line layout again, task by task, while checking precedence matrices and capacity at every single decision point. For larger-scale problems, which involve a greater number of possible task assignment combinations, the time it takes to run both algorithmic overheads will accumulate throughout the iterations. This accumulation ultimately results in longer computational times for hyper-heuristic methods compared to metaheuristic methods.

Table 8 presents the final distribution of each LLH selected by the AHS mechanism at the end of each benchmark instance run. The results indicate that no single LLH dominates all problem instances. A closer examination of the most frequently selected LLH for each dataset reveals that the preferred heuristic varies across problem instances. Through a reward mechanism, the AHS mechanism dynamically increases the selection probability of heuristics that demonstrate better performance for a particular dataset. This adaptive behavior suggests that different problem sizes and structures might benefit from different search strategies and combinations. This sequence-dependent flexibility prevents the decoding process from becoming trapped in local optimum, ultimately contributes to the robustness and effectiveness of proposed SA-HH model in achieving better smoothing index across all problem scales.

Practical Implications

Table 6. shows the workload distribution obtained by the three optimization methods while Table 7. Shows how the task is distributed between workers. As can be seen by the graph in Figure 5, graph profiles from the ACO are the most fluctuated when compared to results from SA and SA-HH. The pronounced peak and valleys on the graph shown indicate that the workload is concentrated on a limited number of workers, while others remain underutilized. The patterns are consistent with the smoothness of the index value achieved by each method. The ACO with the highest smoothness index exhibited the largest fluctuations across workers. SA achieved a better smoothness index compared to ACO, thus showing better fluctuations. However, when comparing SA and SA-HH, both methods achieved smoothness index values that differed only marginally. Because the smoothness index values achieved differed only slightly, the graphs of both SA and SA-HH were slightly similar in terms of the peaks and valleys. These results show that the performance pattern obtained is consistent with previous testing on the 11 benchmark datasets, where SA-HH provides the best results in terms of the smoothing index, followed by SA and ACO.

The real-world case study results highly emphasize the practical value of the SA-HH algorithm to solve multi-manned ALBP with worker-level smoothing. While all three algorithms able to solve the same MMALBP problem, their performance, however, varies from one another. ACO, as can be seen in Figure 5. and Table 6.,

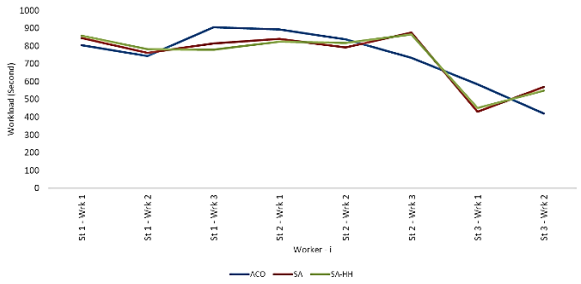


Figure 5. Workload distribution between three methods

struggles to properly balance the secondary objective with the primary objective. All while the other tested algorithm performs a bit more equally, with SA-HH achieving better results. In actual implementation, a slightly unbalanced workload distribution may create a bottleneck. Especially in a multi-manned context, station output speed is restricted by its heavily burdened operator. Therefore, when comparing the algorithm, SA-HH with a smoother graph profile should be able to create fewer bottlenecks because of how the algorithm performs better when distributing the task with the smallest number of workers while maintaining the smoothness index. Proving the superiority of the proposed SA-HH model for the problem addressed in this study.

However, in terms of computational time, SA yields the fastest results, followed by ACO, and lastly, SA-HH. In the context of the case study data, a solution method that provides the best objective results is more preferable. This is due to the nature of the assembly line design, which is typically carried out at the initial stage and does not undergo frequent changes in the short term. Changes in the assembly line configuration would require extensive adjustments within the existing production system, including the reallocation of tasks across workstations, adaptation of operators' working methods, and modifications to the workstations. These adjustments require considerable time, cost, and coordination efforts. Therefore, the trade-off between achieving a better configuration and slightly longer computational time is still considered acceptable.

The proposed workload balancing approach may provide several benefits from human factors. One of the primary advantages of an assembly line with a lower smoothness index is the creation of a fair workload distribution among workers along the assembly line. In an assembly line, where the operator-level workload smoothing is not a primary concern, one or more operators may be assigned to a substantially heavier workload than other workers. This may lead to the perception of unfairness and dissatisfaction, and in long term, it may finally reduce the motivation of the workers. By balancing the workload across stations and workers, the proposed approach may help to ensure that no worker is consistently required to perform significantly more or less workload than others.

Several practical challenges should also be considered during implementation. First, improvement in labor utilization, such as that achieved through workload balancing, may result in reduced workforce requirements. From an organizational point of view, this is a good thing; however, from the workers perspective, this is often associated with layoffs. However, this is not always true, as organizations may address this by reallocating workers to support other production activities or by increasing the overall production capacity. Therefore, creating a win-win situation between the workers and organization, where workers get to keep their jobs while the organization may increase their production capacity. Second, operators may perceive changes in workload after balancing is done. This is generally not an issue for workers who were previously assigned higher workloads. However, this may become an issue for workers who previously had lighter workloads. These changes may potentially create resistance among workers, from workers who are not able to keep up with the new target or who are reluctant to accept increased responsibilities. To facilitate a smoother transition, strategies such as phased implementation, training, and incentive systems may be implemented. Phased implementation allows operators to gradually adapt to the revised workload. Organizations should also provide training to ensure that their operators are prepared for their new task assignments and responsibilities. In addition, performance-based incentives may be introduced to recognize worker contributions and maintain motivation once the expected performance levels have been achieved.

Limitations

Despite the results obtained, we acknowledge several limitations to provide context for the interpretation. First, data-related assumptions. In our study, we assumed that task times are deterministic and that all workers have identical skill levels. We did not account for potential time variations caused by factors such as skill

inconsistency, fatigue, learning curves, or operator heterogeneity. Regarding task sequencing, the precedence constraints we established only reflect the order of operations and do not consider the spatial feasibility of each process. Second, the proposed SA-HH method has only been validated on a limited dataset. Its performance has not yet been tested on assembly lines with different configurations, such as mixed-model, U-shaped, or other layouts. Finally, ergonomic risk factors, worker fatigue, and workload assessment were not explicitly incorporated into the mathematical model. Therefore, the ergonomic and human-factor implications discussed in this study should only be interpreted as potential benefits.

Conclusions

This study is designed to solve the multi-manned ALBP while considering workload smoothing as a secondary objective, and to see how much improvement can be achieved through the optimization. In this study, we compare two metaheuristic methods, ACO and SA, with a metaheuristic-based hyper-heuristic method, SA-HH, as optimization approaches for solving the problem. The first finding is that when the methods are tested on 11 datasets ranging from small-scale to large-scale problems, SA-HH is able to provide the best objective values compared to the two metaheuristic methods. However, when computational time is considered, among the three methods tested, SA is able to obtain the objective value in the shortest time. Although SA-HH has the most optimal objective value z, the time required to achieve this result is relatively long compared to the SA method. Similar findings were also obtained when the three algorithms were applied to the case study dataset. The SA-HH method was able to produce the lowest smoothness index. However, the computational time required by SA-HH was significantly longer than that of both SA and ACO. Therefore, the use of hyper-heuristic methods in optimization problems can be adjusted according to specific needs. For instance, in the case study considered, achieving a more optimal result with a slightly longer computational time is not a major concern. This is because the problem of being optimized, namely assembly line balancing, does not require frequent or rapid changes, given the associated costs, the need for operator training, and the required adaptation processes. Future work could extend the tested model by incorporating stochastic or fuzzy takt time to account for the uncertainty and variability in human performance. Incorporating spatial, ergonomic, and safety-related constraints would also enhance the realism and complexity of multi-worker configurations. In addition, adaptive hyperparameter tuning techniques can be explored to enable the algorithm to automatically adjust its parameters for different problem scales and operational conditions, which would improve its flexibility and robustness across diverse scenarios.

Acknowledgment

The authors would like to thank the reviewers for their constructive feedback and valuable suggestions, which significantly improved the quality of this manuscript. The authors also gratefully acknowledge the support of the company that provided the assembly line data and facilitated direct observation for this study

Declaration of Generative AI in the Writing Process

The author(s) used AI services to assist during coding tasks, including troubleshooting during code development, as well as for initial language checking. The author(s) reviewed and edited the content as necessary and accepted full responsibility for the published work.

References

[1] M. P. Groover, *Automation, Production Systems, and Computer-Integrated Manufacturing Fourth Edition*, 4th ed. Harlow: Pearson, 2016.

[2] N. Zamzam and A. Elakkad, "Time and space multi-manned assembly line balancing problem using genetic algorithm," *Journal of Industrial Engineering and Management*, vol. 14, no. 4, pp. 733–749, 2021, doi: https://doi.org/10.3926/jiem.3542.

[3] P. Fattahi, A. Roshani, and A. Roshani, "A mathematical model and ant colony algorithm for multi-manned assembly line balancing problem," *International Journal of Advanced Manufacturing Technology*, vol. 53, no. 1–4, pp. 363–378, Mar. 2011, doi: https://doi.org/10.1007/s00170-010-2832-y.

[4] A. S. Michels, T. C. Lopes, and L. Magatão, "An exact method with decomposition techniques and combinatorial Benders' cuts for the type-2 multi-manned assembly line balancing problem," *Operations Research Perspectives*, vol. 7, Jan. 2020, doi: https://doi.org/10.1016/j.orp.2020.100163.

[5] T. Ahmed, N. Sakib, R. M. Hridoy, and A. T. Shams, "Application of line balancing heuristics for achieving an effective layout: A case study," *International Journal of Research in Industrial Engineering*, vol. 9, pp. 114–129, 2020, doi: https://doi.org/10.22105/riej.2020.234612.1134.

[6] E. Andreu-Casa, A. Garcia-Villoria, and R. Pastor, "Multi-manned assembly line balancing problem with dependent task times: A heuristic based on solving a partition problem with constraints," *Eur. J. Oper. Res.*, vol. 1, no. 1, pp. 96–116, 2022, doi: https://doi.org/10.1016/j.ejor.2021.12.002.

[7] F. Pilati, E. Ferrari, M. Gamberi, and S. Margelli, "Multi-manned assembly line balancing: Workforce synchronization for big data sets through simulated annealing," *Applied Sciences*, vol. 11, no. 6, Mar. 2021, doi: https://doi.org/10.3390/app11062523.

[8] D. Thiruvady, A. Nazari, and A. Elmi, "An ant colony optimisation-based heuristic for mixed-model assembly line balancing with setups," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2020. doi: http://dx.doi.org/10.1109/CEC48606.2020.9185757.

[9] M. R. Abdullah Make and M. F. F. Ab Rashid, "optimization of two-sided assembly line balancing with resource constraints using modified particle swarm optimisation," *Scientia Iranica*, vol. 29, no. 4, pp. 2084–2098, Jul. 2022, doi: https://doi.org/10.24200/sci.2020.52610.2797.

[10] Ö. Hazır, M. A. N. Agi, and J. Guérin, "A fast and effective heuristic for smoothing workloads on assembly lines: Algorithm design and experimental analysis," *Comput. Oper. Res.*, vol. 115, Mar. 2020, doi: https://doi.org/10.1016/j.cor.2019.104857.

[11] S. Finco, D. Battini, X. Delorme, A. Persona, and F. Sgarbossa, "Workers' rest allowance and smoothing of the workload in assembly lines," *Int. J. Prod. Res.*, vol. 58, no. 4, pp. 1255–1270, Feb. 2020, doi: https://doi.org/10.1080/00207543.2019.1616847.

[12] L. Özbakır and G. Seçme, "A hyper-heuristic approach for stochastic parallel assembly line balancing problems with equipment costs," *Operational Research*, vol. 22, no. 1, pp. 577–614, Mar. 2022, doi: https://doi.org/10.1007/s12351-020-00561-x.

[13] A. Muklason, S. R. Ahlan Robbani, E. Riksakomara, and I. G. A. Premananda, "A comparison of meta-heuristic and hyper-heuristic algorithms in solving an urban transit routing problems," *IAES International Journal of Artificial Intelligence*, vol. 13, no. 3, pp. 2923–2933, Sep. 2024, doi: https://doi.org/10.11591/ijai.v13.i3.pp2923-2933.

[14] A. Roshani and D. Giglio, "A tabu search algorithm for the cost-oriented multi-manned assembly line balancing problem," *International Journal of Industrial Engineering and Production Research*, vol. 31, no. 2, pp. 189–202, 2020, doi: https://doi.org/10.22068/ijiepr.31.2.189.

[15] M. Şahin and T. Kellegöz, "Benders' decomposition based exact solution method for multi-manned assembly line balancing problem with walking workers," *Ann. Oper. Res.*, vol. 321, no. 1, pp. 507–540, 2023, doi: https://doi.org/10.1007/s10479-022-05118-z.

[16] Z. Zhang, Q. Tang, and M. Chica, "Multi-manned Assembly Line balancing with time and space constraints: A MILP model and memetic ant colony system," *Comput. Ind. Eng.*, vol. 150, Dec. 2020, doi: https://doi.org/10.1016/j.cie.2020.106862.

[17] D. Dinler and M. K. Tural, "Exact solution approaches for the workload smoothing in assembly lines," *Engineering Science and Technology, an International Journal*, vol. 24, no. 6, pp. 1318–1328, Dec. 2021, doi: https://doi.org/10.1016/j.jestch.2021.03.013.

[18] F. M. Müller and I. S. Bonilha, "Hyper-heuristic based on aco and local search for dynamic optimization problems," *Algorithms*, vol. 15, no. 1, Jan. 2022, doi: https://doi.org/10.3390/a15010009.

[19] F.M. Sugiarta, "Dataset: Optimization of multi-manned assembly line balancing with workload smoothing using simulated annealing based hyper-heuristic", *Zenodo*, doi: https://doi.org/10.5281/zenodo.17513954

[20] N. Boysen, M. Fliedner, R. Klein, and A. Scholl, "Dataset: Assemly line balancing", ALB, url: https://assembly-line-balancing.de/

[21] N. Dey, *Application of Ant Colony Optimization and its Variants Case Studies and New Development*. Singapore: Springer, 2024.

[22] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated annealing: From basics to applications," in *International Series in Operations Research and Management Science*, vol. 272, Springer New York LLC, 2019, pp. 1–35. doi: https://doi.org/10.1007/978-3-319-91086-4_1.

[23] R. Fauzi, A. Priansyah, P. K. Puspadewa, S. M. Z. Awal, H. T. Nguyen, and A. P. Rifai, "Optimizing vehicle routing for perishable products with time window constraints," *Jurnal Teknik Industri: Jurnal Keilmuan dan Aplikasi Teknik Industri*, vol. 27, no. 1, pp. 1–20, Jan. 2025, doi: https://doi.org/10.9744/jti.27.1.1-20.