IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 9 July 2024, accepted 30 July 2024, date of publication 2 August 2024, date of current version 14 August 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3437410

RESEARCH ARTICLE

A Mathematical Model and Ant Colony Algorithm for Assembly Line Balancing Problem With Human–Robot Collaboration and Alternative Subgraphs

ANAS MA'RUF, R. CAHYADI NUGRAHA, ANDI CAKRAVASTIA, AND ABDUL HAKIM HALIM
Faculty of Industrial Technology, Bandung Institute of Technology, Bandung 40132, Indonesia
Corresponding author: Anas Ma'ruf (maruf@itb.ac.id)

ABSTRACT In recent years, the use of collaborative robots in assembly lines has promised productivity improvement. It provides more alternatives for the assembly line design, which are the alternative resources of the human, robot, or human-robot collaboration (HRC), and the alternative subsets of processes, termed alternative subgraphs, taking advantage of the variety of robotic tools or end-effectors. However, more alternatives make the assembly line balancing problem more complex. This situation is encountered frequently in modern electronics and automotive assembly lines. The contribution of this study is to provide a mathematical model and solution to the assembly line balancing problem that has both HRC and alternative subgraphs, which has not been discussed as an integrated problem in previous literature. To accomplish this optimization problem, a mixed-integer linear programming (MILP) model has been developed to assign tasks to stations and determine the type of resources required while minimizing the cycle time. Practical constraints such as the available number of robots and robotic end-effector types are also considered. Owing to the complexity of the problem, the exact method for MILP is extremely time-consuming for real-world applications. Therefore, a metaheuristic algorithm based on the ant colony optimization (ACO) approach has been developed to solve the problem more efficiently. The results show that the MILP model can obtain optimal solutions for small-sized problems, whereas the ACO algorithm has proven to be a practical solution for medium- to large-sized problems, providing good solutions within an acceptable computation time. The results also show that the presence of alternative subgraphs can give opportunities for better solutions.

INDEX TERMS Assembly line balancing, human-robot collaboration, mathematical model, metaheuristic.

I. INTRODUCTION

Human-robot collaboration (HRC) is an emerging technology in the fourth industrial revolution [1] and continues to become one of the key enabling technologies in the new paradigm of Industry 5.0 [2]. It uses collaborative robots (cobots), which are robots that can interact safely with nearby humans by using integrated sensors and artificial intelligence (AI) applications [3], [4]. The sensors and AI can be enhanced with a trust model to make the correct decision regarding the dynamic interaction between humans and robots [5], [6].

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Liu.

Many HRC applications, as reported in [7], involve assembly-production systems. Although a fully robotic assembly line has been implemented for welding and painting-related processes, many assembly tasks still require the dexterity and flexibility of humans. For such manual assembly tasks, the power and accuracy of cobots can improve task efficiency and quality and reduce health and safety risks. Consequently, the introduction of HRC to assembly processes has become increasingly common, facilitated by advancements in sensors and actuator technology [8].

In the design of assembly lines with high-volume production, the assembly line balancing problem (ALBP) is important. The main issue of ALBP is the assignment of

assembly tasks to a set of workstations, subject to precedence and other constraints, to achieve a specific objective function. ALBP has been extensively investigated [9], [10], ranging from simple to generalized ALBP with various aspects. However, the use of cobots in assembly lines has introduced a new kind of problem, such as the assembly line balancing problem with human-robot collaboration (ALBP-HRC), which has received significant attention in recent years. In the rest of this paper, the term ''robot'' refers to ''cobot'' in the context of ALBP-HRC. Since the first papers discussing ALBP-HRC were published [11], [12], basic mathematical formulations for ALBP-HRC have been proposed [13], [14]. This problem extends the ALBP by providing alternatives for the resources that perform each task, whether by a human only, a robot only, or a human and a robot simultaneously. It is important to note that ALBP-HRC is different from robotic assembly line balancing problems (RALBP) [15], [16]. The RALBP involves a fully robotic assembly line, where each station is assigned a robot as its single resource. In contrast, ALBP-HRC starts with conventional manual lines that can be improved using HRC. It can decide whether a station has a single resource, whether a human or a robot, or whether it has two resources, namely a human and a robot, in the same station. This implies that a task can be performed by a single resource or by two resources, such as a human and a robot simultaneously.

A robot can be installed with various tools or end-effectors, which are devices attached to the end of the robot arm to perform its main tasks. For example, if a robot is required to hold a component, it can be installed with a gripper tool. The automatic tool-changing technology enables a single robot to handle multiple tools [17], [18]. This feature offers the possibility for alternative processes. For example, for a set of tasks of installing four screws, one method is to use a single screwdriver tool and perform the tasks sequentially, whereas the other method is to use multiple screwdrivers in parallel and perform the function simultaneously. In ALBP literature, such alternative processes are termed alternative subgraphs [19]. This term comes from the representation of precedence constraints in ALBP as a precedence graph. Each node of the precedence graph represents a task or process. Hence, a subset of processes is called a subgraph. The alternative subgraphs assembly line balancing problem (ASALBP) is formulated using mathematical programming in [20] and is more recently discussed in [21].

The use of cobots in assembly lines has added decision alternatives concerning the type of resources to use for each task, whether by a human only, a robot only, or a human and a robot simultaneously in collaboration [14]. The possibility of alternative subgraphs further expands the opportunities to achieve higher system performance at the cost of increased complexity in the problem formulation. The main scientific contribution of this research is the development of a mathematical model and solution for ALBP with HRC and alternative subgraphs (ALBP-HRC-AS) in an integrated manner, which has never been discussed in previous literature. This kind of problem frequently occurs in current industries, such as electronic goods and automotive, which adopt HRC for their assembly lines. The objective function is to minimize the cycle time while [14] minimizing the total cost and [20] minimizing the number of stations. The ALBP-HRC-AS is modeled as a mixed-integer linear programming (MILP) formulation, which can be solved optimally using standard mathematical programming software. This approach usually requires a large amount of computation time for medium- to large-sized problems. For this reason, subsequently, a metaheuristic algorithm is developed to find good solutions with a reasonable computation time.

II. LITERATURE REVIEW

ALBP has evolved since its first introduction [22] from the simple assembly line balancing problem (SALBP) to more advanced problems categorized as the generalized assembly line balancing problem (GALBP) [9], [23]. ALBP-HRC falls into the category of resource selection problems in GALBP, which is initiated by [24]. The resource selection problem in ALBP-HRC involves deciding the type of resources (human, robot, or HRC) and the assignment of specific resources (the operator and robot to perform the task) depending on the system characterization. Table 1 shows the related research on the ALBP and the contribution of this research.

The basic assumption of the ALBP-HRC is that there can be one human operator and one robot in a station. This idea is borrowed from the two-sided assembly line balancing problem (TSALBP) introduced in [25]. The mathematical programming formulation of the TSALBP can be found in [26] and [27]. Although there are similarities between TSALBP and ALBP-HRC, they exhibit fundamental differences. In TSALBP, two resources (left-side and right-side operators) always work for different tasks, whereas in ALBP-HRC, there is a possibility of the simultaneous operation of two resources (human and robot).

The first study on ALBP-HRC was published in [12], which used a cost-oriented approach while considering ergonomics. Within the same year of publication, [11] defined the ALBP-HRC with a multi-objective function and considered multiple types of robots. In subsequent years, several studies on ALBP-HRC have been published. Both [13] and [28] minimized the cycle time but used different approaches to system characterization and solution procedures. In [29], a compound objective function derived from both the minimization of the cycle time and the number of resources was used. In [30], a makespan minimization objective was used with a different approach to the system characterization. In [14], a cost-minimization objective function was used, and the notion of robotic tool types was considered.

ASALBP is an extension of ALBP with processing alternatives [34] that include sequence-dependent ALBP [35]. After its introduction in [19], the heuristic for ASALBP was discussed in [20] while [31] formulating a comprehensive mathematical programming and optimal solution method. Further research, such as [32] and [33], constructed

TABLE 1. Related research in assembly line balancing problem with human-robot collaboration and alternative subgraphs.

Research	ALBP HRC	AS ALBP	Objective function				Solution approach					
			NS	CT	Cost	MO / Other	Exact	SH	GA	SA	ACO	Other MH
Capacho & Pastor [19]		√	√				√					
Capacho et al [31]		√	√					√				
Capacho & Pastor [32]		√	√									√
Scholl et al [20]		√	√				√					
Palamut & Akpınar [33]		√	√									√
Leiber et al [21]		√			√				√			
Weckenborg & Spengler [12]	√				√		√					
Mura & Dini	√					√			√			
Weckenborg et al [13]	√			√			√		√			
Çil et al [28]	√			√			√					√
Nourmohammadi et al [29]	√					√	√			√		
Boschetti et al [30]	√					√	√					
Nugraha et al [14]	√				√		√					
This research	√	√		√			√				√	

Notes:
NS = number of stations; CT = cycle time; MO = multiple objective; SH = simple heuristic; GA = genetic algorithm; SA = simulated annealing; ACO = ant colony optimization; MH = metaheuristic

metaheuristic approaches and recently [21] integrated ASALBP with a general resource selection problem and parallel stations using the same approach.

ALBP formulations can be solved using either exact methods or approximate methods [36], both are algorithmic in nature, but different in the guarantee to find the global optimal solutions. Exact methods are based on optimization techniques that guarantee finding a global optimal solution if it exists; for example, the simplex method for linear programming and the branch-and-bound method for integer programming. Special techniques such as SALOME [37] also use branch-and-bound; meanwhile, the most recent method is branch, bound, and remember [38]. Mixed-integer linear programming (MILP) integrates the branch and bound for discrete variables and the simplex method for continuous variables.

Owing to the NP-hard nature of the ALBP [39], approximate methods are often used for larger and more realistic problem sizes. Approximate solutions can be achieved by either bounded exact methods, simple heuristics, or metaheuristics. Bounded exact methods are performed by restricting the solution space or limiting the computation time in exact methods. Simple heuristics attempt to find good feasible solutions using simple rules. A simple heuristic is limited to a particular problem. Many types of simple heuristics were discussed in [31] and [39]. Metaheuristics are advanced methods that attempt to seek a better approximation of the global optimal solution.

Metaheuristics are widely used in the field of ALBP. They have many variants or approaches, including 1) neighborhood algorithms such as simulated annealing [29], GRASP [32], and harmony search [40]; 2) evolutionary methods such as genetic algorithms as used in [11] and [13], some new methods including the water flow-like algorithm (WFA) [41] and a specialized evolutionary approach [42]; and 3) swarm intelligence, such as ant colony [43], bee algorithm [28], whale algorithm [44], and artificial immune system [45]. Notably, each metaheuristic approach has advantages and disadvantages that render it suitable for a particular problem.

III. MATHEMATICAL MODELING

This section discusses the mathematical modeling of the problem in detail. The problem is described and a mathematical formulation is presented. Numerical examples are provided to describe the workings of the mathematical model.

A. PROBLEM DESCRIPTION

A set of assembly tasks, denoted by V, are assigned to a set of stations, J. These assignments must not violate the precedence constraints that each task can only be performed after its predecessors have been performed. Each task may have one or more direct predecessors represented by P_i and one or more direct successors denoted by F_i. The precedence constraints can be represented by a precedence graph. Figure 1 shows an example of a precedence graph.

In Figure 1, $P_1 = \{\}$, $F_1 = \{2\}$, $P_2 = \{1\}$, $F_2 = \{3, 7, 9\}$, etc. In this study, a precedence graph must begin with a single node, called the source node, and end with a single node, called the sink node. The source or sink node may be a real task; however, if there is no single task to start or end the graph, an artificial dummy task is added. In Figure 1, Task 1 is a real task as the source node, whereas Task 22 is a dummy

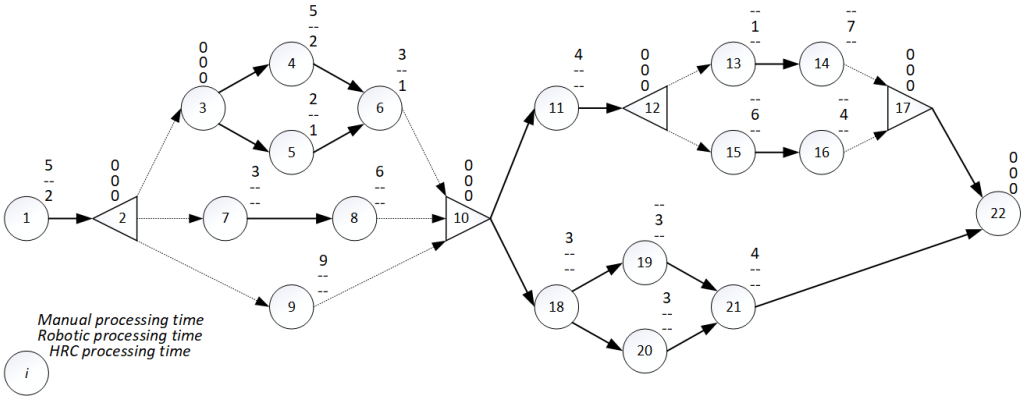


FIGURE 1. An example of a precedence graph (adapted and extended from [20]).

task, identified with zero processing time, as the sink node. Each task has a deterministic processing time but may have alternatives to be performed either by a human alone, by a robot alone, or by a human and a robot simultaneously in collaboration (HRC), each of which has its own processing time. In Figure 1, the processing times are shown by the data attached to each node.

The precedence graph may have some alternative subgraphs, which are some subsets of tasks that represent alternative processes. Each alternative subgraph is flanked by a pair of imaginary tasks, termed entry node and terminal node. Entry nodes and terminal nodes are not real tasks; they have zero processing time. In Figure 1, subgraph {3, 4, 5, 6} has alternatives of subgraph {7, 8} and subgraph {9}. This means that to achieve the same goal as Tasks 3, 4, 5, and 6, we can choose to perform Tasks 7 and 8 or perform Task 9. These alternative subgraphs are flanked by Task 2 as the entry node and Task 10 as the terminal node. Each branch of the alternative subgraphs must begin with a single node and end with another single node so that each node following an entry node represents its branch of the subgraph, and so does each node before a terminal node. Again, if needed, a dummy task may be added. For example, Task 3 in Figure 1 is a dummy task. In Figure 1, there are two sets of alternative subgraphs: subgraphs between Tasks 2 and 10, and subgraphs between Tasks 12 and 17. During the decision process, only one branch of subgraph from each set of alternative subgraphs must be chosen.

A robotic process may have an alternative with a different tool. For example, in Figure 2, the process of tightening four screws using a single robotic screwdriver (the robot performs screwing four times) can alternatively be performed using multiple screwdrivers simultaneously at once. There may be a question like "Why not just decide to restrict the use of parallel tools for the reason of efficiency?" The answer is that these alternatives may be considered due to the number of tools available in the system, the number of tasks that potentially use the tool, etc. Each robot can be installed with one or more tools depending on the capacity of the automatic tool changer (ATC). In this study, the tool-changing time is

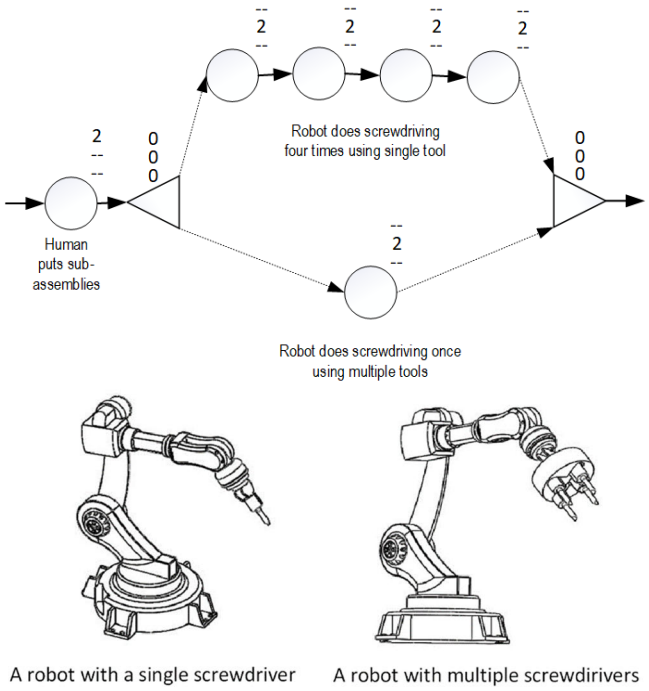


FIGURE 2. An example of alternative processes with similar resources but using different robotic tools.

assumed negligible. Some tools may also have incompatibility to be installed together with other tools in a single ATC. For example, a complex sealing tool cannot be installed together with other standard robotic tools (e.g., grippers, screwdrivers, or pneumatic suction cups) in one robot.

Figure 3 illustrates the ALBP-HRC-AS. The assignment of tasks to stations must consider the availability of resources. There are specified numbers of human operators, robots, and robotic tools available in the system that are fixed, and the company has no option to increase the number of resources. This is a situation in which minimization of the cycle time is considered in the optimization of the assembly line. The assignment of tasks to stations determines the cumulative time of each station and the line cycle time t, which is the maximum of the station cumulative times.

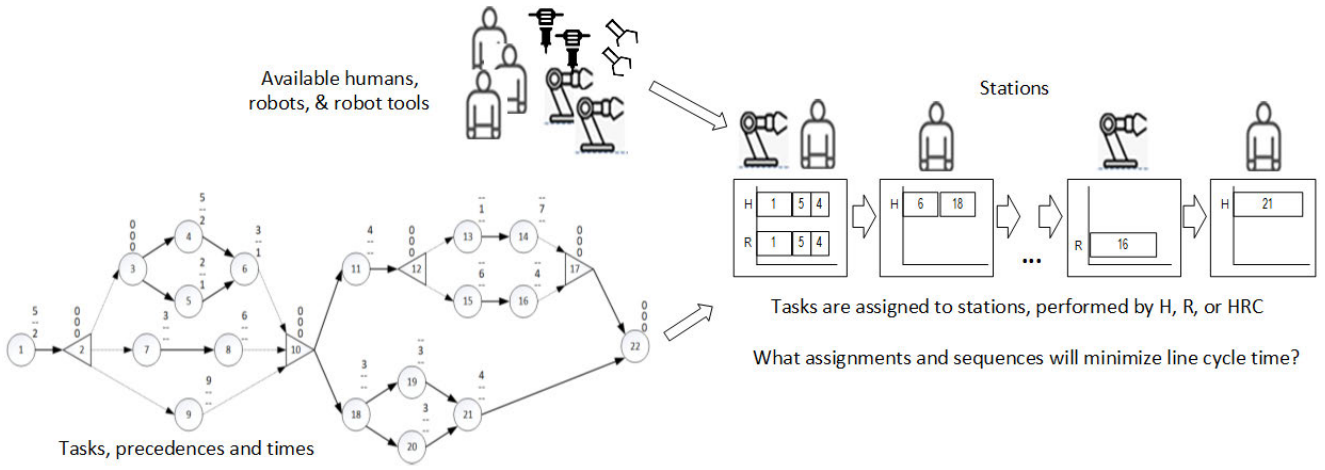


FIGURE 3. Systematic picture of the problem.

TABLE 2. Mathematical notations.

Notation	Definition
Variables	
τ	Assembly line cycle time
x_{ijs}	Assignment variables: 1 if task i is assigned to station j with resource type s, 0 otherwise
f_i	Finish time of task i
z_{ip}	Sequencing variables for pairs of tasks that have no precedence relations if they are assigned in the same stations: 1 if task i is performed before task p, 0 if task p is performed before task i
ξ_i	Auxiliary variables: station number that task i is assigned to
Parameters	
V	Set of all tasks = $\{1, 2, \ldots, n_T\}$
V_e	Set of all tasks as entry nodes
V_t	Set of all tasks as terminal nodes
V_d	Set of dummy tasks
V_z	Set of tasks with zero processing time = $V_e \cup V_t \cup V_d$
V_r	Set of real tasks = $V - V_z$
J	Set of stations = $\{1, 2, \ldots, n_W\}$
S	Set of resource types = {1, 2, 3}, 1 = human only, 2 = robot only, 3 = HRC
P_i	Set of direct predecessors of task i
F_h	Set of direct successors of task h
NP	Set of pairs of tasks that have no precedence relationship
G	Set of tool types
IG	Set of pairs of incompatible tool types that cannot be installed in the same robot
t_{is}	Processing time of task i if performed by resource type s
γ_{gi2}	Binary parameter: 1 if tool type g is required when task i is performed by robot only, 0 otherwise
γ_{gi3}	Binary parameter: 1 if tool type g is required when task i is performed by HRC, 0 otherwise
α_0	Number of human-operators available
ρ_0	Number of robots available
β_{g0}	Number of tools type g available
K	Maximum number of tool types in a robot

B. MATHEMATICAL FORMULATION

The problem that has been described is modeled in an MILP formulation, developed mainly by extending the formulation in [14] and [20]. The notations used for the mathematical formulation are listed in Table 2.

The MILP formulation for the problem is presented in (1) to (29). The equations in the formulation are explained in the subsequent paragraphs following the formulation.

$$\text{Minimize } \tau \tag{1}$$

Subject to:

$$\tau - f_i \geq 0 \; \forall i \in V \tag{2}$$

$$\sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{1js} \right) = 1 \tag{3}$$

$$\sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{njs} \right) = 1 \tag{4}$$

$$\sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{ijs} \right) \leq 1 \; \forall i \in V \setminus \{1, n\} \tag{5}$$

$$\xi_i = \sum_{j=1}^{n_W} \sum_{s \in \{1,2,3\}} j \cdot x_{ijs} \; \forall i \in V \tag{6}$$

$$\xi_h - \xi_i \leq 0 \; \forall i \in F_h, h \in V \setminus V_e \tag{7}$$

$$\xi_h - \xi_i \leq n_W \cdot \left(1 - \sum_{j=1}^{n_W} \sum_{k \in \{1,2,3\}} x_{ijk} \right) \quad \forall i \in F_h, h \in V_e \tag{8}$$

$$\sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{hjs} \right) - \sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{ijs} \right) \geq 0 \quad \forall i \in V \setminus V_t, h \in P_i \tag{9}$$

$$\sum_{i \in F_h} \left(\sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{ijs} \right) \right) - \sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{hjs} \right) = 0 \quad \forall h \in V_e, i \in F_h \tag{10}$$

$$\sum_{h \in P_i} \left(\sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{hjs} \right) \right) - \sum_{j \in J} \left(\sum_{s \in \{1,2,3\}} x_{ijs} \right) = 0 \quad \forall i \in V_t, h \in P_i \tag{11}$$

$$f_i \geq \sum_{j=1}^{n_W} \left(\sum_{s\in\{1,2,3\}} t_{is} x_{ijs} \right) \quad \forall i \in V \tag{12}$$

$$f_i - f_h + M \left(1 - \sum_{s\in\{1,2,3\}} x_{ijs} \right) + M \left(1 - \sum_{s\in\{1,2,3\}} x_{hjs} \right) \geq \sum_{s\in\{1,2,3\}} t_{is} x_{ijs} \quad \forall i \in V,\ \forall h \in P_i, j \in J \tag{13}$$

$$f_i - f_p + M \left(1 - x_{ijk}\right) + M \left(1 - \sum_{s\in\{k,3\}} x_{pjs} \right) + M z_{ip} \geq t_{ik}\ \forall (i,p) \in NP, j \in J, k \in \{1,2\} \tag{14}$$

$$f_p - f_i + M \left(1 - \sum_{s\in\{k,3\}} x_{ijs} \right) + M \left(1 - x_{pjk}\right) + M \left(1 - z_{ip}\right) \geq t_{pk}\ \forall (i,p) \in NP, j \in J, k \in \{1,2\} \tag{15}$$

$$f_i - f_p + M \left(1 - x_{ij3}\right) + M \left(1 - \sum_{s\in\{1,2,3\}} x_{pjs} \right) + M z_{ip} \geq t_{i3}\ \forall (i,p) \in NP, j \in J \tag{16}$$

$$f_p - f_i + M \left(1 - \sum_{s\in\{1,2,3\}} x_{ijs} \right) + M \left(1 - x_{pj3}\right) + M \left(1 - z_{ip}\right) \geq t_{p3}\ \forall (i,p) \in NP, j \in J \tag{17}$$

$$M \left(1 - \sum_{s\in\{1,2,3\}} x_{ijs} \right) + M \left(1 - \sum_{s\in\{1,2,3\}} x_{hjs} \right) \geq M \left| x_{ijk} - x_{hjk} \right| \quad \forall j \in J, k \in \{1,2,3\}, i \in V_z, h \in P_i \tag{18}$$

$$M \left(1 - \sum_{s\in\{1,2,3\}} x_{hjs} \right) + M \left(1 - \sum_{s\in\{1,2,3\}} x_{ijs} \right) \geq M \left| x_{hjk} - x_{ijk} \right| \quad \forall j \in J, k \in \{1,2,3\}, h \in V_z, i \in F_h \tag{19}$$

$$\min \left\{ 1, \left(\sum_{i=1}^{n_T} \sum_{s\in\{1,2,3\}} x_{ijs} \right) \right\} = \min \left\{ 1, \left(\sum_{p\in V_r} \sum_{s\in\{1,2,3\}} x_{pjs} \right) \right\} \quad \forall j \in J \tag{20}$$

$$\min \left\{ 1, \left(\sum_{i=1}^{n_T} \sum_{s\in\{1,2,3\}} x_{ijs} \right) \right\} \geq \min \left\{ 1, \left(\sum_{i=1}^{n_T} \sum_{s\in\{1,2,3\}} x_{i(j+1)s} \right) \right\} \quad \forall j \in J \setminus \{n_W - 1\} \tag{21}$$

$$\sum_{j=1}^{n_W} \left[\min \left\{ 1, \left(\sum_{i=1}^{n_T} \left(x_{ij1} + x_{ij3} \right) \right) \right\} \right] \leq \alpha_0 \tag{22}$$

$$\sum_{j=1}^{n_W} \left[\min \left\{ 1, \left(\sum_{i=1}^{n_T} \left(x_{ij2} + x_{ij3} \right) \right) \right\} \right] \leq \rho_0 \tag{23}$$

$$\sum_{j=1}^{n_W} \left[\min \left\{ 1, \left(\sum_{i=1}^{n_T} \left(\gamma_{gi2} x_{ij2} + \gamma_{gi3} x_{ij3} \right) \right) \right\} \right] \leq \beta_{g0} \quad \forall g \in G \tag{24}$$

$$\sum_{g=1}^{n_E} \left[\min \left\{ 1, \left(\sum_{i=1}^{n_T} \left(\gamma_{gi2} x_{ij2} + \gamma_{gi3} x_{ij3} \right) \right) \right\} \right] \leq K \quad \forall j \in J \tag{25}$$

$$\min \left\{ 1, \sum_{i=1}^{n_T} \left(\gamma_{gi2} x_{ij2} + \gamma_{gi3} x_{ij3} \right) \right\} + \min \left\{ 1, \sum_{i}^{n_T} \left(\gamma_{hi2} x_{ij2} + \gamma_{hi3} x_{ij3} \right) \right\} \leq 1 \quad \forall (g,h) \in IG,\ \forall j \in J \tag{26}$$

$$x_{ijs} = 0 \text{ or } x_{ijs} = 1\ \forall i \in I, j \in J, s \in \{1,2,3\} \tag{27}$$

$$f_i \geq 0\ \forall i \in I \tag{28}$$

$$z_{ip} = 0 \text{ or } z_{ip} = 1\ \forall (i,p) \in Q \tag{29}$$

Equation (1) represents the objective of minimizing the cycle time, which is self-explanatory, whereas (2) defines that the cycle time must not be exceeded by the finish time of all tasks.

Equations (3) and (4) are for the assignments of the source and sink nodes, whereas the assignments of the other nodes are modeled by (5) – (11). Some nodes may not be assigned because their subgraph is not chosen (5). Equation (6) defines the assigned station numbers to be used in (7) and (8). Equation (7) determines that all tasks, except entry nodes, must be assigned to the station where its successors are assigned to the same or subsequent station. If a task is an entry node, it must be assigned to the station where its successor from the chosen subgraph is assigned to the same or subsequent station (8). Equation (9) determines the assignments of tasks related to their predecessors, except for terminal nodes. If a task is an entry node, exactly one of its successors must be assigned, as represented in (10). If it is a terminal node, then exactly one of its predecessors must be assigned, as represented in (11).

Equation (12) represents the processing time constraints. Equation (13) indicates the sequencing of tasks that have precedence relationships if they are assigned to the same station. Equations (14) – (17) represent the sequencing constraints for pairs of tasks that have no precedence relationship but are assigned to the same station, (14) and (15) are for human-only or robot-only assignments, while (16) and (17) are for HRC assignments.

Equations (18) and (19) guarantee that each task with zero processing time is assigned to the same resource as its predecessor or successor if they are both assigned to the same station. Equation (20) ensures that all stations

Station 1	Station 2	Station 3	Station 4	Station 5	Station 6
					
1 (0 – 2) **2 (2 – 2)** **3 (2 – 2)** **5 (2 – 3)** **4 (3 – 5)**	6 (0 – 3) 10 (3 – 3) 18 (3 – 6)	11 (0 – 4)	20 (0 – 3)		17 (0 – 0) 21 (0 – 4) 22 (4 – 4)
					
1 (0 – 2) **2 (2 – 2)** **3 (2 – 2)** **5 (2 – 3)** **4 (3 – 5)**		19 (0 – 3)	12 (0 – 0) 15 (0 – 6)	16 (0 – 4)	

Notes: entry i (a – b): i = task number, a = start time, b = finish time
***i* (*a* – *b*)**: task performed by the human and the robot simultaneously.

FIGURE 4. The optimal solution for the example precedence graph in Figure 1.

TABLE 3. Results from sample problems.

Reference problem (scenario#)	No. of tasks	No. of subgraphs (No. of branches in each subgraph)	Optimal cycle time	Computation time (sec.)
Jaesckhe	14	1 (2)	6.0	3
Scholl	22	2 (3, 2)	6.0	5
Mitchell	46	3 (2, 4, 2)	16.25	79
Sawyer	52	3 (2, 3, 3)	18.35*	21,600*
Warnecke	92	5 (3, 2, 2, 2, 2)	201.0*	21,600*

*) Execution terminated at a time limit of computation time
Run on a workstation with Xeon E5-1620 v2 @3.7 GHz & 8 GB RAM

contain real tasks (not only tasks with zero processing time). Equation (21) guarantees that the use of stations starts from the first station and then the next station consecutively.

Equations (22) – (25) consider the available number of human operators, robots, and tools for each tool type, and the maximum number of tools in a robot. If there are pairs of tools that cannot be installed together in the same robot, they are assigned to different stations (26). Equations (27) – (30) define the range of values for all variables.

The MILP formulation has been implemented on a computer workstation using a CPLEX solver, implementing the exact methods. An example of the generated solution for the precedence graph in Figure 1, with the available resources of six humans and four robots, is shown in Figure 4, which shows that the minimum cycle time is six units of time. Only five humans are used; however, all four robots are used. Some sample results obtained using the exact method of the mathematical model are presented in Table 3. The sample problems were generated from the ASALBP reference problem sets on the ALBP research page assembly-line-balancing.de and as described in [20]. The parameterization of HRC components is described in [14].

Further experiments have been conducted using the solver to gain insight into the robustness of the model. First, the effect of the tasks that can be performed by robot or HRC are considered. Experiments on 46-task problems are shown in Figure 5 and Figure 6. In Figure 5, Experiment 1 was done with all tasks performed by humans, Experiment 2 with 12 tasks having robot or HRC alternatives, Experiment 3 with 33 tasks having robot or HRC alternatives, and Experiment 4 with all tasks having robot or HRC alternatives. In Figure 6, Experiment A was done with no alternative subgraphs, Experiment B with a group of tasks having their alternative subgraphs, and Experiment C with three groups of tasks having their alternative subgraphs. The results show that optimal cycle time gets better by providing more alternatives, while computation time increases with more alternatives. These experiments confirm that HRC and alternative subgraphs can improve the assembly line, while the provision of HRC and alternative subgraphs requires more computation time to solve the problem.

IV. METAHEURISTIC ALGORITHM

Based on the sample results presented in Table 3, it was discovered that medium- to large-sized problems require an enormous amount of computation time. This implies that a heuristic approach is required to obtain good results within an acceptable computation time. The NP-hard nature of the ALBP [34] has led to the development of metaheuristic approaches [31].

In this study, the ant colony optimization (ACO) approach was chosen because of its simplicity for the problem. An ACO algorithm can be developed using a construction algorithm only. All the constraints can be evaluated more easily in a construction algorithm; therefore, the ACO is suitable for ALBP-HRC that include many constraints. The construction processes are repeated iteratively, guided by a mechanism to obtain better new solutions.

ACO was first developed in [46], and its use in ALBP can be seen in [43] and more recently in [47]. The contribution of the proposed algorithm in this study is to provide an ACO-based metaheuristic to solve ALBP-HRC with alternative subgraphs. In addition, the objective function in this study is to minimize the cycle time, which differs from [38] and [41]. To search for minimized cycle time, the idea is to set an estimated upper bound of cycle time as a constraint for each loop of construction steps, and then reduce the upper bound step by step in the next loops, until no more feasible solutions can be generated.

The fundamental logic of ACO is as follows: in each step, a virtual ant object constructs a feasible solution. While constructing the solution, it scans for feasible alternatives to be decided in each step and then chooses an alternative using a random approach based on the probability that is influenced by a pheromone trail and a heuristic value. The higher the pheromone trail and the heuristic value of an alternative will result in a higher probability of selecting that alternative. The pheromone trail is a value dropped by each ant in its pathway to construct a solution. Each ant updates the pheromone trail values in each construction step based on the solution quality that it has constructed. The better the solution, the more

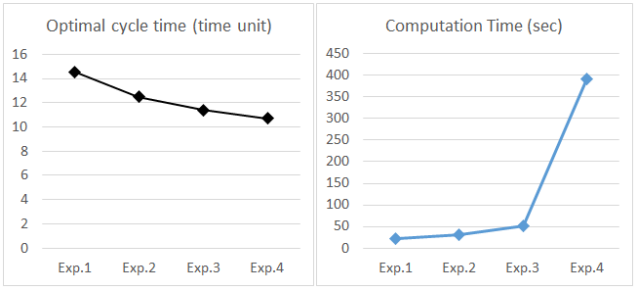


FIGURE 5. Experiments by varying the number of tasks that can be performed by robot or HRC.

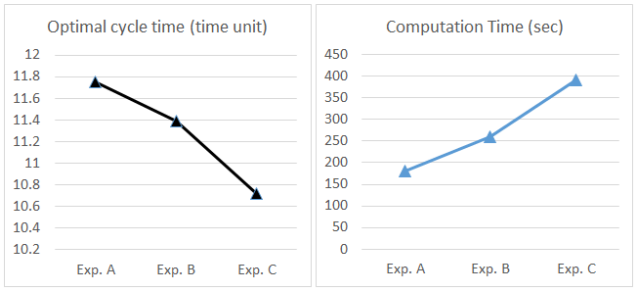


FIGURE 6. Experiments by varying the number of alternative subgraphs.

value it reinforces the pheromones related to the solution. In addition to the reinforcement, all pheromone trails are also subjected to natural decay by the evaporation process. Heuristic value is a simple criterion that predicts the goodness of each alternative. In this study, one of the heuristic values is inversely proportional to the calculated finish time of a task if the task is assigned to the station in the current step. Some additional notations for the proposed algorithm are listed in Table 4.

There are two kinds of pheromone trails in this study:

1) pheromone for the selection of subgraph branches and

2) pheromone for the assignment of tasks to stations and resource types. In the initialization step, all pheromone trails are set to 1.

In the selection of subgraph branches, for each entry node e, each of the alternative subgraph branches b has a probability of selection as in (30).

$$p_{eb} = \frac{\left(\Phi_{eb}^{SG}\right)^{\alpha_A}\left(\frac{1}{t_{eb}^{MAX}}\right)^{\beta_A}}{\sum_{l\in E}\left[\left(\Phi_{el}^{SG}\right)^{\alpha_A}\left(\frac{1}{t_{el}^{MAX}}\right)^{\beta_A}\right]} \tag{30}$$

Based on the selected subgraph branch, the upper bound of the cycle time, τ_{UB}, can be calculated. This is the key to each construction step. Each ant attempts to construct a solution by assigning tasks to stations that do not exceed the upper bound of the cycle time. The upper bound of the cycle time is defined by (31). In this equation, λ_τ is a factor that multiplies the theoretical cycle time. Some studies have used $\lambda_\tau = 2$ [48], [49]. However, in this study, the algorithm defines the search range within the maximum and minimum values of λ_τ,

TABLE 4. Additional mathematical notations for algorithm.

Notation	Definition
a_C	Ant colony size
i_{MAX}	Maximum number of iterations
N_{REP}	Number of replications
E	Set of all branches in entry node e
$\Phi^{SG}{}_{be}$	Pheromone trait for selecting subgraph b of entry node e
Φ_{ijs}	Pheromone trait for assigning task i to station j with resource-type s
p_{eb}	Probability of selecting subgraph b of entry node e
p_{ijs}	Probability of assigning task i to station j with resource-type s
α_A	Importance parameter for pheromone value
β_A	Importance parameter for heuristic value
ρ_A	Evaporation parameter of ACO
$t^{\mathrm{MAX}}{}_{eb}$	The largest task processing time in the subgraph branch b of entry node e
τ_{UB}	Upper bound of cycle time
λ_τ	Multiplication factor for calculating τ_{UB} from theoretical cycle time
$\Delta\lambda_\tau$	Step size for decreasing λ_τ in the algorithm
n_F	Cumulative number of failed ants
$\%F$	Percentage of failed ants compared to total trials of ants performing solution construction
A'	Set of all assignable tasks in the current step
f_i^*	Calculated finish time of task i if assigned in the current step

which can lie somewhere between 0 and 2.

$$\tau_{UB} = \lambda_\tau \cdot \max\left\{\max\left(t_{is}\right), \frac{\sum_{i\in V}\max_{s\in S}\left(t_{is}\right)}{n_W}\right\} \quad \forall i \in V, s \in \{1,2,3\} \tag{31}$$

The developed algorithm constructs and finds the best solution according to the maximum upper bound value of the cycle time. Subsequently, λ_τ is decreased by $\Delta\lambda_\tau$. The construction and determination of the best solution continues according to the current upper bound of the cycle time, up to the minimum value of λ_τ or when the percentage of failed ants exceeds a certain limit. A failed ant is an ant that faces a situation where all stations are already assigned, while some tasks have not yet been assigned; thus, the upper bound of the cycle time in the last station is violated. The percentage of failed ants is calculated by (32). Owing to the use of random numbers, this algorithm runs in replications to increase the chance of obtaining a better solution, whereas iterations are the mechanism to converge the pheromone trails.

$$\%F = \frac{n_F}{a_C \times i_{\mathrm{MAX}} \times N_{\mathrm{REP}}} \tag{32}$$

Algorithm ACO ASALBP-HRC for minimizing cycle time.

Input parameters: N_{REP}, i_{MAX}, a_C, α_A, β_A, ρ_A, range of λ_t, $\Delta\lambda_t$ and maximum limit of $\%F$
Do for each replication:
  Do while upper bound of cycle factor λ_t is within the range of maximum and minimum or $\%F$ has not exceeded the maximum limit
    Initialize pheromone trails Φ^{SG}_{eb} and Φ_{ijs}
    Do iteration 1 to i_{MAX}:
      Do for each Ant a member of ant colony of size a_C
        Choose branch of alternative subgraph for each entry node based on ACO probability of choosing subgraph's branch
        Calculate upper bound of cycle time τ_{UB}
        Open workstation: $j = 1$
        Find assignable tasks
        Do while any task unassigned:
          For each task i meets *precedence*:
            Check taks i by H, R, or HRC, check availability of robot, generate probability of robot assignment in station j
            Calculate finish time f_i^* if assigned in j, if $f_i^* \leq \tau_{UB}$, then i is candidate of assignment; if all of $f_i > \tau_{UB}$ then $j = j+1$, continue inner loop
            Choose task candidate based on ACO probability of choosing tasks
          Update assignable tasks
        Record solution to Ant object; Update pheromone trails using evaporation and reinforcement
        Record best solutions for all Ants, and update record of best solution
      Update record of best solution for each iteration
    Tighten upper bound of cycle time with $\lambda_t = \lambda_t - \Delta\lambda_t$
Choose best solution from all replications

FIGURE 7. The algorithm of ACO ASALBP-HRC for minimizing cycle time.

The assignment of tasks to stations and resource types is based on the selection probability in (33).

$$p_{ijs} = \frac{\left(\Phi_{ijs}\right)^{\alpha_A}\left(\frac{1}{f_i^*}\right)^{\beta_A}}{\sum_{i,l\in A'}\left[\left(\Phi_{ijl}\right)^{\alpha_A}\left(\frac{1}{f_i^*}\right)^{\beta_A}\right]} \tag{33}$$

The ACO-based algorithm for ASALBP-HRC that minimizes the cycle time is represented using the pseudocode in Figure 7.

V. NUMERICAL RESULTS AND ANALYSIS

Parameters setting can affect the performance of the algorithm. As explained in [50], the parameters α_A, β_A, and ρ_A affect the convergence versus stagnation of the solution-searching steps. In this research, the ACO parameters were set to moderate values of $\alpha_A = 0.5$, $\beta_A = 1.0$. The evaporation parameter ρ_A was set to 0.1. The number of replications was 4, the ant colony size was 25, and the maximum number of iterations was 25. A higher number of replications may increase the probability of obtaining a better solution, but proportionally increases the computation time. A similar logic applies to the ant colony size and the maximum iterations. The upper bound of the cycle time was within the range of $0.6 < \lambda\tau < 1.3$. The percentage limit of failed ants was 90%. When the $\%F$ exceeded this limit, the algorithm was terminated.

TABLE 5. Results and performances of ACO ASALBP-HRC algorithm.

Reference problem	Exact method		ACO ASALBP-HRC		Gap (%)
	Optimal solution	Comp. time (s)	Best solution	Comp. time (s)	
Jaesckhe ($n_T = 14$)	6.0	3	6.0	3	0.0
Scholl ($n_T = 22$)	6.0	6	6.0	5	0.0
Mitchell ($n_T = 46$)	16.25	81	16.25	12	0.0
Sawyer ($n_T = 52$)	18.35*	21,600*	20.0	45	9.0
Warnecke ($n_T = 92$)	201.0*	21,600*	224.4	165	10.1

*) Exact method is bounded to a time limit of computation time
Run on a workstation with Xeon E5-1620 v2 @3.7 GHz & 8 GB RAM

Table 5 presents the best solutions for the five sample problems. From the samples presented in the table, it can be concluded that the algorithm can find optimal solutions for some small- to medium-sized problems ($n_T < 50$) with a more efficient computation time compared to the exact method. In large-sized problems ($n_T \geq 50$), the algorithm can find good solutions with gaps to the solutions of the exact method of less than 10% on average, with a much more efficient computation time.

TABLE 6. Best solution findings and termination conditions.

Reference problem	λ_τ when the best solution is found	λ_τ when the algorithm is terminated	%F when the algorithm is terminated
Jaesckhe ($n_T = 14$)	1.2	0.9	100.0
Scholl ($n_T = 22$)	1.0	1.0	95.4
Mitchell ($n_T = 46$)	1.1	1.0	100.0
Sawyer ($n_T = 52$)	0.8	0.8	99.8
Warnecke ($n_T = 92$)	0.7	0.7	91,4

The best solution for each problem is highly dependent on specific parameters such as the structure of the precedence network, processing time values, existence of subgraphs, and alternative resources. This study has introduced a novel mechanism for decreasing the upper bound of the cycle time and termination condition based on the percentage of failed ants, distinguishing it from previous ACO-based algorithms. Table 6 lists the best solution findings and termination conditions for each sample problem to describe this mechanism. The value of λ_τ at which the solution is obtained describes the efficiency of selecting the initial λ_τ. For example, if the initial λ_τ is 1.3, but the best solution is obtained when λ_τ is 0.8, then the algorithm can perform more efficiently by starting from $\lambda_\tau = 1.0$. The discrepancy between λ_τ when finding the best solution and terminating the algorithm also describes the computational efficiency, because iterations between the two are performed without obtaining a better solution. These conditions are likely to pose challenges for future research.

The algorithm has been used to conduct more computational experiments to test the robustness of the developed model. Some variations in the presence of alternative subgraphs were introduced in this research. These experiments were designed to test the advantage of the presence of alternative subgraphs in the precedence graph. They were conducted on the two largest sample problems in this research, which are precedence graphs with 52 nodes (Sawyer reference problem) and 92 nodes (Warnecke reference problem). Due to the limited space, partial precedence graphs of the sample problems are shown in Figure 6, with each graph showing only its source node, sink node, and alternative subgraphs; other nodes in each graph correspond to original base problems in [15]. Refer to Figure 1 and its explanation for the notation of precedence graph. For each alternative subgraph (any nodes in between an entry node and a terminal node), a branch of the alternative subgraph was eliminated in each experiment. For example, in the Sawyer reference problem (Figure 8(a)), Experiment 1a in Table 7(a) eliminated the branch of Node-28 from Subgraph 1; Experiment 1b eliminated the branch of Node-10 from Subgraph 1. Experiment 5b in Table 7(b) for Warnecke problem (Figure 8(b)) eliminated the branch of Node-65 from Subgraph 5.

TABLE 7. Computational experiments for reducing the number of subgraphs' branches.

(a) Reference problem Sawyer ($n_T = 52$)
3 subgraphs; number of alternatives in each subgraph: 3, 2, 3

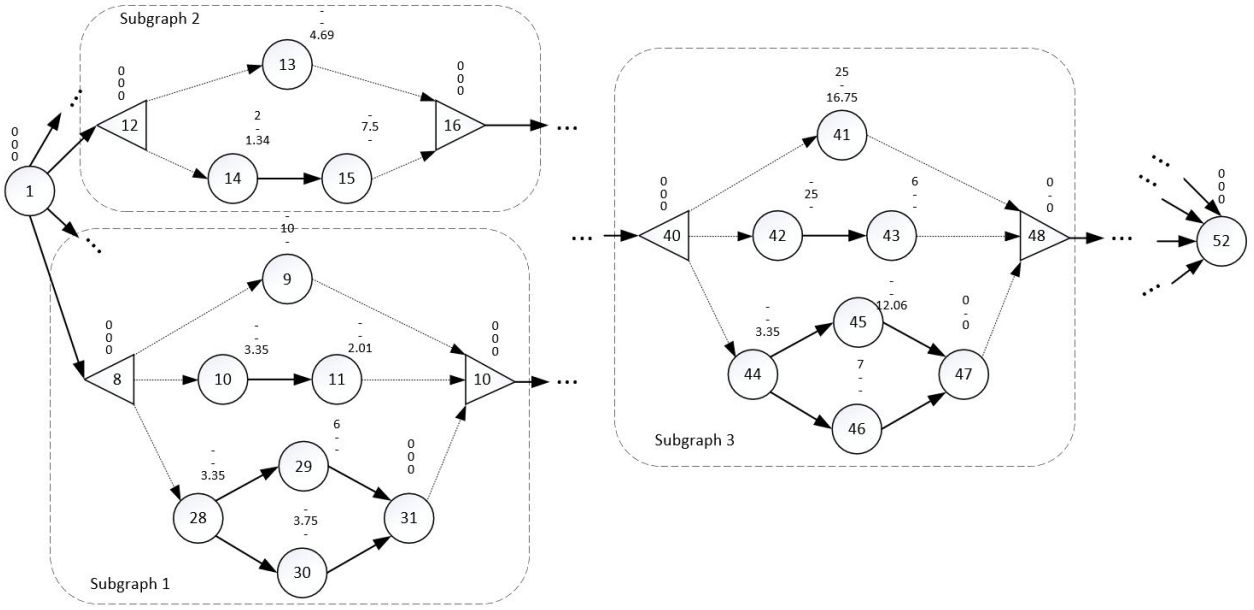
Exp. no.	Description	Best cycle time
0	Original precedence graph	20.0
1a	Subgraph 1 with only branch 1 & 2	21.0*
1b	Subgraph 1 with only branch 1 & 3	20.1
1c	Subgraph 1 with only branch 2 & 3	20.0
2a	Subgraph 2 with only branch 1	21.0*
2b	Subgraph 2 with only branch 2	21.0*
3a	Subgraph 3 with only branch 1 & 2	21.0*
3b	Subgraph 3 with only branch 1 & 3	20.7
3c	Subgraph 3 with only branch 2 & 3	20.0

(b) Reference problem Warnecke ($n_T = 92$)
5 subgraphs; number of alternatives in each subgraph: 3, 2, 2, 2, 2

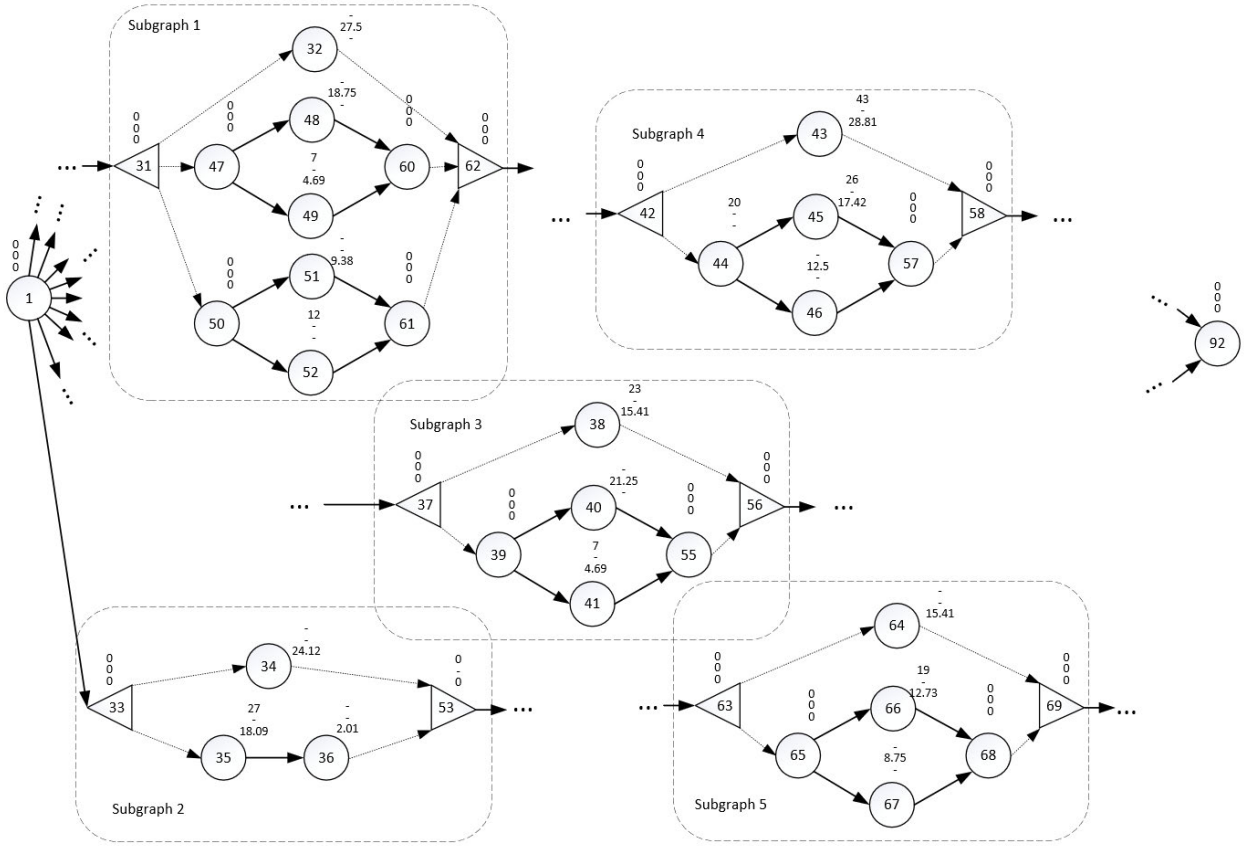
Exp. no.	Description	Best cycle time
0	Original precedence graph	224.40
1a	Subgraph 1 with only branch 1 & 2	231.84
1b	Subgraph 1 with only branch 1 & 3	227.70
1c	Subgraph 1 with only branch 2 & 3	226.22
2a	Subgraph 2 with only branch 1	225.02
2b	Subgraph 2 with only branch 2	259.30*
3a	Subgraph 3 with only branch 1	229.43
3b	Subgraph 3 with only branch 2	319.27*
4a	Subgraph 4 with only branch 1	226.85
4b	Subgraph 4 with only branch 2	320.53*
5a	Subgraph 5 with only branch 1	225.65
5b	Subgraph 5 with only branch 2	253.5*

*Case with the reduction of solution quality ≥ 5%

Table 7 shows that the presence of alternative subgraphs may increase the opportunity to find better solutions. The variation of the experimental results is influenced by random numbers in the ACO-based algorithm. Some results with the reduction of solution quality ≥ 5% may indicate that the elimination of some branches of the alternative subgraphs really affect the best solution. For example, in Experiment 1a of Table 7(a), the cycle time is 21.0, so that the reduction of solution quality is (21.0 – 20.0)/20.0 = 5%. In the Sawyer problem experiments, the solution quality is reduced not more than 5%, but it was shown that some branches of subgraphs, especially those of Node-28 and Node-44 affect the opportunity to find better solution. In the Warnecke problem experiments, the results show more variation. The worst case is in Experiment 4b of Table 7(b), that is the elimination of Node-43 from Warnecke problem, with the reduction of solution quality (320.53 – 224.40)/224.40 = 42.8%. The experiments show that the branches of Node-50, Node-34, Node-38, Node-43, and Node-64 are significant in the finding of better solution.



(a) Reference problem Sawyer (n_T=52)



(b) Reference problem Warnecke (n_T=92)

FIGURE 8. Partial precedence graphs showing alternative subgraphs of reference problems.

The model needs to be extended in further studies. Some practical aspects are not yet considered in the model. One of them is a mixed product variant or mixed-model assembly line balancing problem. This kind of assembly line is adopted in many industries to accommodate flexibility and efficiency altogether. Another aspect is the need for total cost minimization as an objective function. Some practical situations may permit the addition of robots and tools in the short or medium

term through robot rental. This implies the need to evaluate cost terms in the decision. Other future research also can be directed to the monitoring aspect of the execution of the assembly line in real-time. To capture the dynamics of the assembly line, a digital twin concept for the assembly line with HRC can be developed. The research and applications of digital twins have been growing in recent years either in manufacturing or in other sectors [51], [52].

VI. CONCLUSION

A mathematical model for the assembly line balancing problem with human-robot collaboration and alternative subgraphs aiming to minimize cycle time has been presented in this paper. The problem is modeled in the MILP formulation, which has been tested with several sample problems. In small-sized problems, this model obtains exact optimal solutions, and in medium- to large-sized problems, time-bounded exact solutions are achieved, but have limited practical use owing to the computation time required. To address this limitation, a metaheuristic algorithm based on the ant colony optimization approach has been developed.

The metaheuristic algorithm is based on the ant colony optimization, which is extended using the principle of decreasing the upper bound of the cycle time. Using this approach, a good solution is obtained, and in some cases, it produces the same result as the optimal solution obtained by the exact method, but with a more efficient computation time. Solution quality is represented by the gap of about 0 to 10% from the solution of the exact method, with the computation time of less than five minutes compared to hours using the exact method.

Experiments have been conducted to gain insight into the benefits of alternative subgraphs. The experiments have shown that the presence of alternative subgraphs gives better opportunities to find better solutions.

Further study needs to address other practical but important aspects of the assembly line balancing problem with human-robot collaboration, such as the assembly lines that produce mixed product variants (mixed-model assembly line balancing problem), and the objective function that minimizes the total cost. Another study also can be directed to the dynamic aspect of the assembly line.

REFERENCES

[1] K.-D. Thoben, S. Wiesner, and T. Wuest, "'Industrie 4.0' and smart manufacturing—A review of research issues and application examples," *Int. J. Autom. Technol.*, vol. 11, no. 1, pp. 4–16, Jan. 2017.

[2] W. Xian, K. Yu, F. Han, L. Fang, D. He, and Q.-L. Han, "Advanced manufacturing in industry 5.0: A survey of key enabling technologies and future trends," *IEEE Trans. Ind. Informat.*, vol. 20, no. 2, pp. 1055–1068, Feb. 2023.

[3] S. Proia, R. Carli, G. Cavone, and M. Dotoli, "Control techniques for safe, ergonomic, and efficient human–robot collaboration in the digital industry: A survey," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 1798–1819, Jul. 2022.

[4] A. A. Cantone, M. Esposito, F. P. Perillo, M. Romano, M. Sebillo, and G. Vitiello, "Enhancing elderly health monitoring: Achieving autonomous and secure living through the integration of artificial intelligence, autonomous robots, and sensors," *Electronics*, vol. 12, no. 18, p. 3918, Sep. 2023.

[5] Y. Wang, F. Li, H. Zheng, L. Jiang, M. Fooladi Mahani, and Z. Liao, "Human trust in robots: A survey on trust models and their controls/robotics applications," *IEEE Open J. Control Syst.*, vol. 3, pp. 58–86, 2024.

[6] A. F. Abate, P. Barra, C. Bisogni, L. Cascone, and I. Passero, "Contextual trust model with a humanoid robot defense for attacks to smart ecosystems," *IEEE Access*, vol. 8, pp. 207404–207414, 2020.

[7] P. Tsarouchi, A.-S. Matthaiakis, S. Makris, and G. Chryssolouris, "On a human–robot collaboration in an assembly cell," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 6, pp. 580–589, 2017.

[8] S. Robla-Gómez, V. M. Becerra, J. R. Llata, E. González-Sarabia, C. Torre-Ferrero, and J. Pérez-Oria, "Working together: A review on safe human–robot collaboration in industrial environments," *IEEE Access*, vol. 5, pp. 26754–26773, 2017.

[9] N. Boysen, M. Fliedner, and A. Scholl, "A classification of assembly line balancing problems," *Eur. J. Oper. Res.*, vol. 183, no. 2, pp. 674–693, Dec. 2007.

[10] N. Boysen, P. Schulze, and A. Scholl, "Assembly line balancing: What happened in the last fifteen years?" *Eur. J. Oper. Res.*, vol. 301, no. 3, pp. 797–814, Sep. 2022.

[11] M. Dalle Mura and G. Dini, "Designing assembly lines with humans and collaborative robots: A genetic approach," *CIRP Ann.*, vol. 68, no. 1, pp. 1–4, 2019.

[12] C. Weckenborg and T. S. Spengler, "Assembly line balancing with collaborative robots under consideration of ergonomics: A cost-oriented approach," *IFAC-PapersOnLine*, vol. 52, no. 13, pp. 1860–1865, 2019.

[13] C. Weckenborg, K. Kieckhäfer, C. Müller, M. Grunewald, and T. S. Spengler, "Balancing of assembly lines with collaborative robots," *Bus. Res.*, vol. 13, no. 1, pp. 93–132, Apr. 2020.

[14] R. C. Nugraha, A. Ma'ruf, A. C. Nugraha, and A. H. Halim, "A mixed-integer linear programming formulation for assembly line balancing problem with human–robot shared tasks," *J. Phys., Conf. Ser.*, vol. 1858, no. 1, Apr. 2021, Art. no. 012021.

[15] J. Rubinovitz, J. Bukchin, and E. Lenz, "RALB—A heuristic algorithm for design and balancing of robotic assembly lines," *CIRP Ann.*, vol. 42, no. 1, pp. 497–500, 1993.

[16] G. Levitin, J. Rubinovitz, and B. Shnits, "A genetic algorithm for robotic assembly line balancing," *IFAC Proc. Volumes*, vol. 35, no. 1, pp. 79–84, 2002.

[17] Z. Samadikhoshkho, K. Zareinia, and F. Janabi-Sharifi, "A brief review on robotic grippers classifications," in *Proc. IEEE Can. Conf. Electr. Comput. Eng. (CCECE)*, May 2019, pp. 1–4.

[18] H. Jenkins, "Design of robotic end effectors," in *Robotics and Automation Handbook*, T. R. Kurfess, Ed., Boca Raton, FL, USA: CRC Press, 2004.

[19] L. Capacho and R. Pastor, "ASALBP: The alternative subgraphs assembly line balancing problem," *Int. J. Prod. Res.*, vol. 46, no. 13, pp. 3503–3516, Jul. 2008.

[20] A. Scholl, N. Boysen, and M. Fliedner, "Optimally solving the alternative subgraphs assembly line balancing problem," *Ann. Oper. Res.*, vol. 172, no. 1, pp. 243–258, Jun. 2009.

[21] D. Leiber, A.-T. Vuong, and G. Reinhart, "Alternative subgraphs assembly line balancing problem with resource selection and parallel stations," *Eng. Optim.*, vol. 54, no. 11, pp. 1903–1918, Nov. 2022.

[22] M. E. Salveson, "The assembly-line balancing problem," *J. Fluids Eng.*, vol. 77, no. 6, pp. 939–947, Aug. 1955.

[23] N. Boysen, M. Fliedner, and A. Scholl, "Assembly line balancing: Which model to use when?" *Int. J. Prod. Econ.*, vol. 111, no. 2, pp. 509–528, Feb. 2008.

[24] J. Bukchin and M. Tzur, "Design of flexible assembly line to minimize equipment cost," *IIE Trans.*, vol. 32, no. 7, pp. 585–598, Jul. 2000.

[25] J. J. Bartholdi, "Balancing two-sided assembly lines: A case study," *Int. J. Prod. Res.*, vol. 31, no. 10, pp. 2447–2461, Oct. 1993.

[26] U. Özcan and B. Toklu, "Balancing of mixed-model two-sided assembly lines," *Comput. Ind. Eng.*, vol. 57, no. 1, pp. 217–227, Aug. 2009.

[27] Y. K. Kim, W. S. Song, and J. H. Kim, "A mathematical model and a genetic algorithm for two-sided assembly line balancing," *Comput. Oper. Res.*, vol. 36, no. 3, pp. 853–865, Mar. 2009.

[28] Z. A. Çil, Z. Li, S. Mete, and E. Özceylan, "Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration," *Appl. Soft Comput.*, vol. 93, Aug. 2020, Art. no. 106394.

[29] A. Nourmohammadi, M. Fathi, and A. H. C. Ng, "Balancing and scheduling assembly lines with human–robot collaboration tasks," *Comput. Oper. Res.*, vol. 140, Apr. 2022, Art. no. 105674.
[30] G. Boschetti, M. Faccio, M. Milanese, and R. Minto, "C-ALB (collaborative assembly line balancing): A new approach in cobot solutions," *Int. J. Adv. Manuf. Technol.*, vol. 116, nos. 9–10, pp. 3027–3042, Oct. 2021.
[31] L. Capacho, R. Pastor, A. Dolgui, and O. Guschinskaya, "An evaluation of constructive heuristic methods for solving the alternative subgraphs assembly line balancing problem," *J. Heuristics*, vol. 15, no. 2, pp. 109–132, Apr. 2009.
[32] L. Capacho and R. Pastor, "A metaheuristic approach to solve the alternative subgraphs assembly line balancing problem," in *Assembly Line*, W. Grzechca, Ed., Rijeka, Croatia: IntechOpen, 2011.
[33] U. Palamut and Ş. Akpınar, "A firefly algorithm for the alternative subgraphs assembly line balancing problem," *J. Turkish Oper. Manage.*, vol. 3, no. 2, pp. 290–297, 2019.
[34] P. A. Pinto, D. G. Dannenbring, and B. M. Khumawala, "Assembly line balancing with processing alternatives: An application," *Manage. Sci.*, vol. 29, no. 7, pp. 817–830, Jul. 1983.
[35] A. Scholl, N. Boysen, and M. Fliedner, "The sequence-dependent assembly line balancing problem," *OR Spectr.*, vol. 30, no. 3, pp. 579–609, Jun. 2008.
[36] O. Battaïa and A. Dolgui, "A taxonomy of line balancing problems and their solutionapproaches," *Int. J. Prod. Econ.*, vol. 142, no. 2, pp. 259–277, Apr. 2013.
[37] A. Scholl and R. Klein, "SALOME: A bidirectional branch-and-bound procedure for assembly line balancing," *INFORMS J. Comput.*, vol. 9, no. 4, pp. 319–334, Nov. 1997.
[38] E. C. Sewell and S. H. Jacobson, "A branch, bound, and remember algorithm for the simple assembly line balancing problem," *INFORMS J. Comput.*, vol. 24, no. 3, pp. 433–442, Aug. 2012.
[39] A. Scholl and C. Becker, "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing," *Eur. J. Oper. Res.*, vol. 168, no. 3, pp. 666–693, Feb. 2006.
[40] X. Zheng, S. Ning, H. Sun, J. Zhong, and X. Tong, "Solving multiobjective two-sided assembly line balancing problems by harmony search algorithm based on Pareto entropy," *IEEE Access*, vol. 9, pp. 121728–121742, 2021.
[41] A. Nourmohammadi, M. Fathi, M. Zandieh, and M. Ghobakhloo, "A water-flow like algorithm for solving U-shaped assembly line balancing problems," *IEEE Access*, vol. 7, pp. 129824–129833, 2019.
[42] H. Zhang, C. Zhang, Y. Peng, D. Wang, G. Tian, X. Liu, and Y. Peng, "Balancing problem of stochastic large-scale U-type assembly lines using a modified evolutionary algorithm," *IEEE Access*, vol. 6, pp. 78414–78424, 2018.
[43] A. Baykasoglu and T. Dereli, "Two-sided assembly line balancing using an ant-colony-based heuristic," *Int. J. Adv. Manuf. Technol.*, vol. 36, nos. 5–6, pp. 582–588, Mar. 2008.
[44] K. Meng, Q. Tang, Z. Zhang, and X. Qian, "An improved lexicographical whale optimization algorithm for the type-II assembly line balancing problem considering preventive maintenance scenarios," *IEEE Access*, vol. 8, pp. 30421–30435, 2020.
[45] M. N. A. Khalid and U. K. Yusof, "Leveraging bio-inspired knowledge-intensive optimization algorithm in the assembly line balancing problem," *IEEE Access*, vol. 9, pp. 117832–117844, 2021.
[46] M. Dorigo and G. Di Caro, "Ant colony optimization: A new metaheuristic," in *Proc. Congr. Evol. Comput.*, 1999, pp. 1470–1477.
[47] I. Kucukkoc and D. Z. Zhang, "Type-E parallel two-sided assembly line balancing problem: Mathematical model and ant colony optimisation based approach with optimised parameters," *Comput. Ind. Eng.*, vol. 84, pp. 56–69, Jun. 2015.
[48] D. Lei and X. Guo, "Variable neighborhood search for the second type of two-sided assembly line balancing problem," *Comput. Oper. Res.*, vol. 72, pp. 183–188, Aug. 2016.
[49] M. N. Janardhanan, Z. Li, G. Bocewicz, Z. Banaszak, and P. Nielsen, "Metaheuristic algorithms for balancing robotic assembly lines with sequence-dependent robot setup times," *Appl. Math. Model.*, vol. 65, pp. 256–270, Jan. 2019.
[50] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, Nov. 2006.
[51] G. Mylonas, A. Kalogeras, G. Kalogeras, C. Anagnostopoulos, C. Alexakos, and L. Muñoz, "Digital twins from smart manufacturing to smart cities: A survey," *IEEE Access*, vol. 9, pp. 143222–143249, 2021.
[52] M. Staffa, E. Izzo, and P. Barra, "Leveraging the RoboMaker service on AWS cloud platform for marine drone digital twin construction," in *Social Robotics*, A. A. Ali, J.-J. Cabibihan, N. Meskin, S. Rossi, W. Jiang, H. He, and S. S. Ge, Eds., Singapore: Springer, 2024, pp. 22–32.

ANAS MA'RUF received the Ph.D. degree from Toyohashi University of Technology, Japan. He is currently an Associate Professor with the Industrial Engineering Department, Bandung Institute of Technology, Indonesia. His research interests include production automation and manufacturing systems design. He is a Board Member of Asia–Pacific Industrial Engineering and Management Society.

R. CAHYADI NUGRAHA received the bachelor's, master's, and Ph.D. degrees in industrial engineering from Bandung Institute of Technology, Indonesia. His research interests include manufacturing systems, system modeling, and industrial automation.

ANDI CAKRAVASTIA received the Ph.D. degree from Hiroshima University, Japan. He is currently an Associate Professor with the Industrial Engineering Department, Bandung Institute of Technology, Indonesia. He has published numerous publications in various international peer-reviewed journals. His research interests include supply chain integration and optimization. He is currently a Board Member and the Former Vice President of Asia–Pacific Industrial Engineering and Management Society.

ABDUL HAKIM HALIM received the B.Sc. and M.Sc. degrees in industrial engineering from Bandung Institute of Technology, Indonesia, and the Ph.D. degree in industrial engineering from the University of Osaka Prefecture, Japan. He is currently a Professor with the Industrial Engineering Department, Bandung Institute of Technology. His research interests include scheduling, inventory, and manufacturing systems design. He has published his research outcomes in the scheduling/inventory areas in international journals, such as IJPR, EJOR, PPC, IJPE, IJTech, and international conference proceedings. He served as one of the guest editors for CAIE. He is the immediate past President of Asia–Pacific Industrial Engineering and Management Society (APIEMS) and a fellow of APIEMS and IEOM.