Hindawi
Journal of Mathematics
Volume 2021, Article ID 4196228, 18 pages
https://doi.org/10.1155/2021/4196228

Research Article

A Branch-and-Price Algorithm for Balancing Two-Sided Assembly Lines with Zoning Constraints

Qidong Yin[1,2] **Xiaochuan Luo**[1,2] **and Julien Hohenstein**[3]

[1]*College of Information Science and Engineering, Northeastern University, Shenyang 110819, China*
[2]*State Key Laboratory of Synthetical Automation for Process Industries, Northeastern University, Shenyang 110819, China*
[3]*iFactory Munich, BMW AG, Munich 80809, Germany*

Correspondence should be addressed to Qidong Yin; yqdmail@163.com

Received 20 June 2021; Accepted 25 November 2021; Published 14 December 2021

Academic Editor: Basil Papadopoulos

Two-sided assembly lines are widely used in the large-size product manufacturing industry, especially for automotive assembly production. Balancing the assembly line is significant for assembly process planning and assembly production. In this study, we develop a novel and exact method to optimize the two-sided assembly line balancing problem with zoning constraints (TALBz), in which the aim is to minimize the number of mated-stations considering the task restrictions. A mixed-integer programming model is employed to exactly describe the TALBz problem. To strengthen the computational efficiency, we apply Dantzig–Wolfe decomposition to reformulate the TALBz problem. We further propose a branch-and-price (B&P) algorithm that integrates the column generation approach into a branch-and-bound frame. Both the benchmark datasets with zoning constraints and without zoning constraints are tested to evaluate the performance of the B&P algorithm. The numerical results show that our proposed approach can obtain optimal solutions efficiently on most cases. In addition, experiments on the real-world datasets originating from passenger vehicle assembly lines are conducted. The proposed B&P algorithm shows its advantage in tackling practical problems with the task restrictions. This developed methodology therefore provides insight for solving large-scale TALBz problems in practice.

1. Introduction

In the final production process in automotive manufacturing, parts and components are installed on the painted body in the assembly shop. Assembly production is very labour-intensive in a passenger vehicle plant, with more than half of workforces assigned on the assembly line. In the automotive industry, the assembly line balancing (ALB) problem involves searching for the optimum assignment of assembly tasks to workstations given precedence constraints according to a predefined single or multiobjective goal [1]. Based on the structure of the workstation, the ALB problem can be categorized into the simple assembly line balancing (SALB) problem and the two-sided assembly line balancing (TALB) problem. In the simple assembly line, there is one worker working at each workstation. This pattern is common in the preassembly line where vehicle parts are produced. In contrast, workers can be arranged to both the left and right sides of workstations on the two-sided assembly line. Workstations with the left and right positions are called mated-stations, which construct most sections in the main assembly line. Workers install parts located on one side of the vehicle at the corresponding position. Spaces on both sides of the assembly line can be efficiently used to store materials and parts. Walking time spent taking the parts is reduced on the two-sided assembly line. However, the complexity of the TALB problem increases considerably compared to the SALB problem. In the TALB problem, tasks can be operated on both sides of a mated-station. The positional characteristic of the task and work station results in more variables in the TALB problem model. For different objective functions, the TALB problem can also be classified into the TALB type-I (TALB-I) problem and the TALB type-II (TALB-II) problem. The TALB-I problem minimizes the number of workstations under a given cycle time, while in the TALB-II problem, the objective is to obtain a minimum cycle time for the assembly line with a fixed length.

In practice, there exist assignment restrictions related to the features of assembly process, the allocation of instruments, and the production resource. One kind of these assignment restrictions is the task restrictions which are also called zoning constraints [2]. Zoning constraints define which tasks must be performed at the same workstation and which tasks must be assigned to different workstations [3]. The former situation is often found in the workstations which have tools shared by different tasks. The process, equipment, or quality requirements may also define this restriction of assignment. On the other hand, some tasks are forbidden to assign to the same workstation. The zone constrained type-I two-sided assembly line balancing (TALBz-I) problem is studied in this work to minimize the number of mated-stations.

The SALB problem is a demonstrated NP-hard problem. Considering the additional sequence and positional and zone constraints, the TALBz problem is harder to optimally solve than the SALB problem. Because of these constraints, the TALBz problem is a unique type of mixed-integer programming problem. The column generation method is proven effective for solving large-scale integer programming. However, to the best of the authors' knowledge, no research has thus far reported the development of a column generation approach to solve the TALBz problem. Thus, the proposed branch-and-price algorithm reformulates the mathematical models, designs the column generation procedure, and develops the branching strategy for the TALBz problem.

The remainder of this article is stated as follows: In Section 2, the authors mention the literature review based on methods, constraints, and characteristics of the TALB problem. In Section 3, the mathematical model of the TALBz-I problem and D-W decomposition of the original model are presented. Reformulation of the TALBz-I problem is constructed through the master problem and subproblems. In Section 4, the overall structure of the branch-and-price algorithm is first presented. Details of the initial solution generation and the branching scheme are also described in this section. In Section 5, the most commonly used benchmark datasets as well as four new datasets from the real production line, which happen at a premier brand automaker's assembly shop, are tested. The results are compared with the original models and the state-of-the-art algorithms. A summary is given in Section 6.

2. Literature Review

The assembly line balancing problem has been widely studied by many researchers, as reviewed by Erel and Sarin [4], Scholl and Becker [5], and Boysen et al. [6]. Due to its utilization for large-size product manufacturing, especially for automotive assembly, the two-sided assembly line becomes the key element in the structure of the assembly production system. The study of the TALB problem has gained importance in the last two decades [7]. Methodologies for solving the TALB problem can be categorized into three main approaches: exact algorithms, heuristic algorithms, and metaheuristics. Bartholdi [8] first introduced the TALB problem and designed a heuristic algorithm based on the "first-fit" rule. The general structure of the TALB problem was defined. Additionally, he contributed test data of the 148-task problem for a real assembly line. Lee et al. [9] designed a heuristic procedure for the TALB problem and considered work relatedness and work slackness as the decision criteria. The heuristic rules of the TALB problem were then discussed, and two problems (65-task and 205-task) from a truck assembly line were presented in the experimental results. Hu et al. [10] proposed a heuristic algorithm based on a station-oriented enumerative assignment procedure for the TALB problem. Their tests were performed on small-sized problems (P9 to P24). Wu et al. [11] presented a branch-and-bound algorithm to minimize the total length of the assembly line and the number of opened stations as the objective of the TALB problem. Hu et al. [12] continued their works to develop a branch-and-bound algorithm that integrated a new lower bound, dominance rule, and reduction rules for the TALB-I problem. Sepahi and Naini [13] proposed a new heuristic algorithm for the TALB problem that allows for a parallel performance of tasks. Li and Coit [14] proposed new priority rules that are specific to the TALB problem. Michels et al. [15] applied Benders' decomposition to solve the multi-manned assembly line balancing problem. These published exact and heuristic methods are efficient only on medium-sized instances, since computational demands and memory requirements grow exponentially for large-scale problems [4]. Özcan and Toklu [16] proposed an exact solution approach, which is the goal programming, to solve the small-sized line balancing problem with zoning constraints. Li, Kucukkoc, and Zhang [17] developed new exact methods applying the Hoffman heuristic and branch bound and remember (BB&R) algorithm, to minimize the number of mated-stations in the TALB problem. New dominance rules, lower bounds, and a novel task enumeration procedure were designed in their research. The proposed BB&R algorithm achieved better solutions than the current best heuristic algorithm.

Various metaheuristic algorithms have been applied to solve the TALB problem. The genetic algorithm was applied to solve the ALB problem by some researchers [18–20]. Baykasoglu and Dereli [21] proposed an ant colony-based and group assignment heuristic algorithm to solve the TALB problem with zoning constraints. Simaria et al. [22] developed an ant colony algorithm named 2-ANTBAL to solve the TALB problem to optimality. A mathematical programming model was built, whose major objective was to minimize the number of workstations. A simulated annealing algorithm was presented by Özcan and Toklu [23] to tackle the mixed-model two-sided assembly line balancing problem. A new mixed-integer programming model that involves some specific additional constraints such as zoning constraints, positional constraints, and synchronism constraints was illustrated. Özcan [24] studied the TALB problem with stochastic task times. A simulated annealing algorithm was selected as the approach to solve the problem, which was modelled as chance-constrained, piecewise-linear, and mixed-integer programming. The bees algorithm

was adopted to solve the TALBz problem by Özbakır and Tapkan [3]. Tapkan et al. [25] also applied the bees algorithm and artificial bee colony algorithms to solve the TALB problem, which is fully constrained.

Tapkana et al. [26] studied the TALB problem considering the walking time in parallel lines and developed the bees algorithm and artificial bee colony algorithm. Tang et al. [27] proposed an improved discrete artificial bee colony (DABC) algorithm to solve the type-II TALB problem. The proposed method found 22 brand-new results for large-size problems and outperformed nine other metaheuristic algorithms in the experiments. Li et al. [28] illustrated a new local search method called the iterated greedy (IG) approach to solve the type-I TALB problem based on an encoding scheme of task permutation. Some recent works studied TALB problems with specific line configurations or additional constraints, such as underground workstations [29], robotic workstations [30], real-world constraints [31], worker assignment [32], space and resource constraints [33], varying or uncertain task times [34], and multiworker workstations [35]. Although metaheuristic algorithms have been widely used, some disadvantages still exist. Since the search of solutions proceeds randomly, metaheuristic algorithms generate nondeterministic results. In addition, metaheuristics require additional training to specify suitable parameters, which is time-consuming and not deterministic [14].

3. Description and Modelling of the TALBz-I Problem

The two-sided assembly line, which is shown in Figure 1, can represent the key structure of the main assembly line. Automotive parts and components are delivered to material areas on the line side and installed onto the semifinished vehicle according to the assembly sequence. Workers can perform the installation operations in both the left and right positions in the workstation. Some basic concepts that occur in the TALB problem are first introduced to better understand the assembly production process:

(i) Cycle time: cycle time is determined by the production rate of the assembly line. It defines how long the vehicle can stay in one workstation for installations.

(ii) Idle time: apart from assembly operations in one cycle time, the spare time without any activity occurring is called the idle time.

(iii) Mated-station: in the two-sided assembly line, a workstation with two facing positions is called a mated-station.

(iv) Precedence relationship: the assembly processes must satisfy the structural relations of products. The assembly sequence is given as the constraints of the line balancing design. It defines which task must be done before one task can begin.

(v) Predecessor: tasks that must be performed before the referenced task are called the latter's predecessor.

(vi) Successor: tasks that follow the referenced task in the precedence diagram are called the latter's successor.

(vii) Linked tasks: these are tasks that must be assigned to the same workstation due to the zoning constraints.

(viii) Incompatible tasks: these are tasks that must not be assigned to the same workstation due to the zoning constraints.

3.1. Problem Description. Given a fixed cycle time, the type-I TALB problem seeks a line balancing result that possesses the minimum number of mated-stations. This value will determine the length of the assembly line. Figures 2 and 3 illustrate an example of the TALB problem with sixteen tasks taken from the study by Lee et al. [9]. Figure 2 presents the precedence diagram. Labels of task numbers are connected by arrows that indicate the sequence relationship between tasks. For example, tasks 4 and 5 must be finished before task 7 starts. The first value in the bracket equals the task's operation time, where the second item in the bracket represents the position information of the task. *L* means the task must be done on the left position in the mated-station, while tasks labelled with *R* belong to the right position. E-type tasks can be assigned to either the left or the right position.

In a simple assembly line, task assignment on the workstation is continuous. The idle time for one worker may only occur when he or she finishes all the operations at the end of one cycle. The capacity constraint requires that the summation of tasks' operation time is less than the cycle time in the SALB problem. However, this is not sufficient in the TALB problem. Task assignment must consider the sequence relations of tasks on both sides of mated-stations and the position constraints in the TALB problem. Continuous working plans can be interrupted by the sequence constraints among tasks that are assigned to opposite positions in a mated-station. An operator may wait for the complement of work in the opposite position. This type of waiting time is called the sequence-induced idle time, which does not exist in the SALB problem. Figure 3 describes a line balancing solution for the above sixteen tasks case under a cycle time of 20 time units. For example, in the assignment of mated-station 2, task 10 must be done in the right position and after task 7. Thus, there is an idle time of 16 units before task 10. Zoning constraints are usually established according to the requirement of assembly process which forces or forbids the distribution of different tasks to the same workstation. These constraints should be satisfied in the line balancing result in addition to cycle time and precedence constraints.

3.2. Mathematical Model of the TALBz-I Problem. To ease presentation, the notations that appear in the models are given as follows:

Indices:

i, h, p: task number

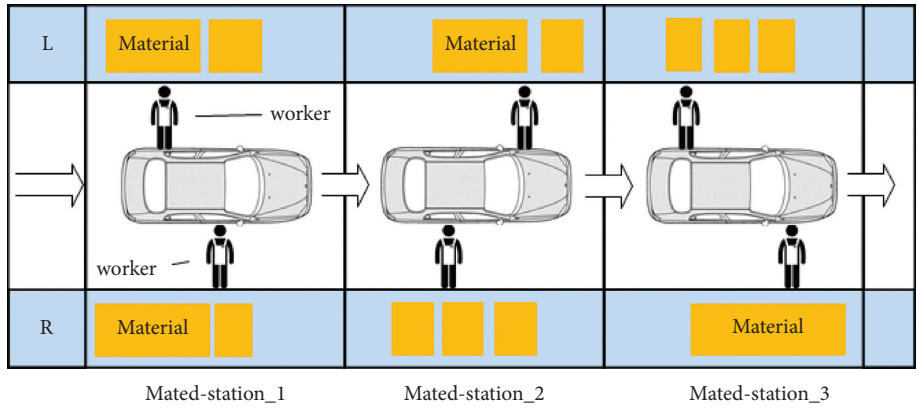

Figure 1: Two-sided assembly line of vehicle manufacturing.

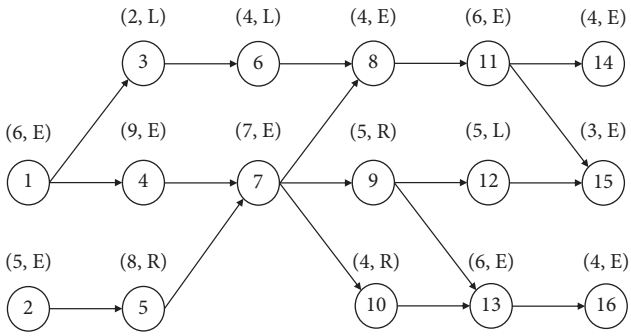

Figure 2: Precedence diagram with sixteen assembly tasks (Lee et al., 2001).

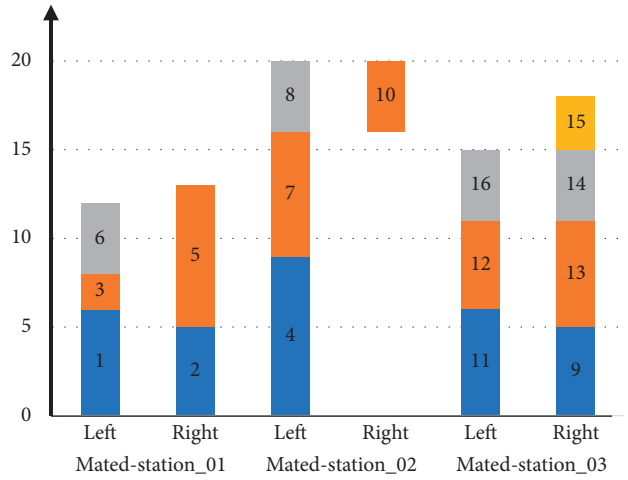

Figure 3: Line balancing solution of sixteen tasks.

j, g: mated-station
k: side of the assembly line

Parameters:

I: set of tasks
J: set of mated-stations
$K(i)$: set of available positions of the task i
P_0: set of tasks which have no immediate predecessor
$P(i)$: set of immediate predecessors of task i
$P_a(i)$: set of total tasks which precede task i
$S_a(i)$: set of total tasks which follow task i
S_L/S_R: lower bound of numbers of left/right workstations
$T_{SumL/R/E}$: sum of the operation time of the set of left/right/either position tasks
X_j: assignment j for one mated-station in a given solution set
B: set of assignments in the master problem
ct/CT: cycle time
t_i: operation time of task i
μ: multiplier as the big M
π: set of values of the dual variable

Variables:

x_{ijk}: if task i is assigned to side k of station j, = 1; otherwise, = 0
N_j: if station j is open for assembly production, = 1; otherwise, = 0
t_i^f: finish time of task i.
z_{ip}: indicator variable to represent assembly sequence of task i and p.
λ_j: if assignment j is included in the solution, = 1; otherwise, = 0.
y_i: if column j is selected in the solution, = 1; otherwise, = 0.
v_i: if task i is taken in the assignment, = 1; otherwise, = 0
s_{ik}: if task i is taken on the k side in the assignment, = 1; otherwise, = 0

We build the mathematical model of the type-I TALBz problem based on the model presented by Kim et al. [20]. They studied the type-II TALB problem in which their objective was to minimize the cycle time given a fixed number of mated-stations. The objective of our model is to minimize the length of the assembly line, which equals the minimization of the number of mated-stations. We use 0–1 variable N_j to represent whether a station j is used.

$$\text{Minimize} \sum_{j \in J} N_j. \tag{1}$$

Subject to

$$\sum_{j\in J}\sum_{k\in K(i)} x_{ijk} = 1 \quad \forall i \in I, \tag{2}$$

$$\sum_{g\in J}\sum_{k\in K(h)} g\cdot x_{hjk} \le \sum_{j\in J}\sum_{k\in K(i)} j\cdot x_{ijk} \quad \forall i \in I - P_0,\ h \in P(i), \tag{3}$$

$$t_i^f - t_h^f + \mu\cdot\left(1 - \sum_{k\in K(i)} x_{ijk}\right) + \mu\cdot\left(1 - \sum_{k\in K(h)} x_{hjk}\right) \ge t_i \quad \forall i \in I - P_0,\ h \in P(i),\ j \in J, \tag{4}$$

$$\begin{aligned} &t_i^f - t_p^f + \mu\cdot\left(1 - x_{ijk}\right) + \mu\cdot\left(1 - x_{pjk}\right) + \mu\cdot z_{ip} \ge t_i \\ &\forall i \in I,\ p \in \left\{r | r \in I - \left(P_a(i)\cap S_a(i)\right) \text{and } i < r\right\},\ j \in J,\ k \in K(i)\cap K(p), \end{aligned} \tag{5}$$

$$\begin{aligned} &t_p^f - t_i^f + \mu\cdot\left(1 - x_{ijk}\right) + \mu\cdot\left(1 - x_{pjk}\right) + \mu\cdot\left(1 - z_{ip}\right) \ge t_p \\ &\forall i \in I,\ p \in \left\{r | r \in I - \left(P_a(i)\cap S_a(i)\right) \text{and } i < r\right\},\ j \in J,\ k \in K(i)\cap K(p), \end{aligned} \tag{6}$$

$$t_i \le t_i^f \le \text{ct} \quad \forall i \in I, \tag{7}$$

$$\sum_{i\in I}\sum_{k\in K(i)} x_{ijk} \le \left\|W_{jk}\right\|\cdot N_j \quad \forall j \in J, \tag{8}$$

$$\sum_{g\in J}\sum_{k\in K(i)} g\cdot x_{igk} - \sum_{g\in J}\sum_{k\in K(j)} g\cdot x_{jgk} = 0 \quad \forall (i,j) \in \text{LT}, \tag{9}$$

$$\sum_{g\in J}\sum_{k\in K(i)} g\cdot x_{igk} - \sum_{g\in J}\sum_{k\in K(j)} g\cdot x_{jgk} \ne 0 \quad \forall (i,j) \in \text{IT}, \tag{10}$$

$$x_{ijk} = 0 \text{ or } 1 \quad \forall i \in I,\ j \in J,\ k \in K(i), \tag{11}$$

$$N_j = 0 \text{ or } 1 \quad \forall j \in J, \tag{12}$$

$$z_{ip} = 0 \text{ or } 1 \quad \forall i \in I,\ p \in \left\{r | r \in I - \left(P_a(i)\cap S_a(i)\right) \text{and } i < r\right\}. \tag{13}$$

Three types of common constraints of the TALB problem (i.e., occurrence constraints, precedence constraints, and cycle time constraints) and extended zoning constraints are included in the type-I TALBz model. Constraint set (2) describes the type of occurrence constraint that ensures that each task is assigned one and only one time. Constraints sets (3)–(6) define the sequence constraints in the TALB problem. Constraint set (3) restricts the sequence relationships of tasks between different mated-stations, and it can also be applied to the SALB problem; that is to say, tasks cannot be assigned to the former stations before their predecessors. Constraint set (4) is applied to tasks that have precedence relationships and assigned to the same mated-station. Since there is sequence-induced idle time in the TALB problem, the assembly sequence must be determined for tasks assigned to the same station. We introduce the variable t_i^f to define the finishing time of task i. Predecessors must be finished before their reference task begins when they are done in the same mated-station. For tasks without precedence relationships and appearing in the same workstation, constraint sets (5) and (6) become active. z_{ip} is set as the indicator variable which is a 0–1 variable. When $z_{ip} = 1$ task i is arranged before task p, constraint set (6) is effective; when task p is planned before task i, $z_{ip} = 0$ and constraint set (5) becomes active. Constraint set (7) is the set of cycle time constraints. The finish time of one task must be no greater than the cycle time, which is larger than the task's operation time. $||W_j||$ represents the number of works assigned to station j. Once there is one task i assigned to workstation j on side k, the value of variable x_{ijk} is 1; then, station j is counted as the total number of open stations, and the value of N_j is equal to 1. Additionally, zoning constraint sets (9) and (10) describe the restrictions of tasks' relationships. For tasks (i, j) which belong to the linked task (LT) set, they must be assigned to the same mated-station (9). Meanwhile, constraint set (10) guarantees that tasks (i, j) defining the incompatible task (IT) set are finished at different mated-stations.

3.3. Decomposition of the Mixed-Integer Programming Model. A key to implementing the branch-and-price algorithm is the use of D-W decomposition for the original problem. It is a standard approach to obtain the linear programming

model used in the master problem of the column generation method.

Appelgren [36] first applied D-W decomposition to solve an integer programming-based ship scheduling problem. To address the fractional solution, Appelgren [37] adopted a branch-and-bound searching tree to seek the integer solution based on D-W decomposition. Lubbecke and Desrosiers [38] provided remarks on the column generation method and discussed the theoretical details of Dantzig–Wolfe decomposition and integer programming column generation algorithms. In the mathematical model introduced in the former section, decision variables are set to assign task to different stations. We apply D-W decomposition to the original model of the TALBz problem and present a reformulation of the original model. This method is used to determine which solution in the assignment set is adopted for a mated-station in D-W decomposition.

$$\text{Minimize} \sum_{j \in J} \lambda_j. \tag{14}$$

Subject to

$$\sum_{j \in J} \sum_{k \in K(i)} \lambda_j X_j^{ik} = 1 \quad \forall i \in I, \tag{15}$$

$$\sum_{g \in J} \sum_{k \in K(h)} g \cdot \lambda_g \cdot X_g^{hk} \leq \sum_{j \in J} \sum_{k \in K(i)} j \cdot \lambda_j \cdot X_j^{ik} \quad \forall i \in I - P_0,\ h \in P(i), \tag{16}$$

$$t_i^f - t_h^f + \mu \cdot \left(1 - \sum_{k \in K(i)} \lambda_j X_j^{ik}\right) + \mu \cdot \left(1 - \sum_{k \in K(h)} \lambda_j X_j^{hk}\right) \geq t_i \quad \forall i \in I - P_0,\ h \in P(i),\ j \in J, \tag{17}$$

$$\begin{aligned} &t_i^f - t_p^f + \mu \cdot \left(1 - \lambda_j X_j^{ik}\right) + \mu \cdot \left(1 - \lambda_j X_j^{pk}\right) + \mu \cdot z_{ip} \geq t_i \\ &\forall i \in I,\ p \in \{r | r \in I - (P_a(i) \cap S_a(i)) \text{ and } i < r\},\ j \in J, k \in K(i) \cap K(p), \end{aligned} \tag{18}$$

$$\begin{aligned} &t_p^f - t_i^f + \mu \cdot \left(1 - \lambda_j X_j^{ik}\right) + \mu \cdot \left(1 - \lambda_j X_j^{pk}\right) + \mu \cdot \left(1 - z_{ip}\right) \geq t_p \\ &\forall i \in I,\ p \in \{r | r \in I - (P_a(i) \cap S_a(i)) \text{ and } i < r\},\ j \in J,\ k \in K(i) \cap K(p), \end{aligned} \tag{19}$$

$$t_i \leq t_i^f \leq \text{ct} \quad \forall i \in I, \tag{20}$$

$$\sum_{g \in J} \sum_{k \in K(i)} g \cdot \lambda_g \cdot X_g^{ik} - \sum_{g \in J} \sum_{k \in K(j)} g \cdot \lambda_g \cdot X_g^{jk} = 0 \quad \forall (i, j) \in \text{LT}, \tag{21}$$

$$\sum_{g \in J} \sum_{k \in K(i)} g \cdot \lambda_g \cdot X_g^{ik} - \sum_{g \in J} \sum_{k \in K(j)} g \cdot \lambda_g \cdot X_g^{jk} \neq 0 \quad \forall (i, j) \in \text{IT}, \tag{22}$$

$$\lambda_j = 0 \text{ or } 1 \quad \forall j \in J, \tag{23}$$

$$z_{ip} = 0 \text{ or } 1 \quad \forall i \in I,\ p \in \{r | r \in I - (P_a(i) \cap S_a(i)) \text{ and } i < r\}. \tag{24}$$

Given a group of line balancing solutions X, the decision variable λ_j means that if assignment j is selected in the final solution, then $\lambda j = 1$; otherwise, when assignment j is not included in the solution, $\lambda j = 0$. The objective function is to obtain the minimum number of mated-stations. Constraint set (15) is the occurrence constraint which is transferred from (2). Constraint set (16) describes the sequence constraints of tasks that are assigned to different mated-stations, which are constructed as (3) in the original model. For tasks that are distributed in the same workstation, constraint sets (17)–(19) ensure that the assembly sequence can satisfy the precedence graph. The use of these constraints can be referred to (4)–(6) in the former model. Constraint set (17) and constraint sets (18) and (19) are effective for tasks with and without precedence relationships, respectively. Indicator variables z_{ip} are still utilized in the sequence constraint sets (18) and (19) in the Dantzig–Wolfe decomposition model. The decision variable of the task's finish time is the same as that of the original model. The cycle time constraints (20) also remain the same as (7). Constraint sets (21) and (22) define the zoning constraints (9) and (10) that established in the original model.

3.4. Master Problem and Pricing Subproblem of Reformulation. In the model of Dantzig–Wolfe decomposition, each column constitutes task assignments for one workstation. Since there are an enormous number of

possible assignments related to a workstation, it is not feasible to enumerate all the assignment solutions of the model. Therefore, we develop a column generation approach to solve the reformulation model obtained from Dantzig–Wolfe decomposition.

Gilmore and Gomory first used the column generation method to solve the cutting stock problem [39, 40]. Bin packing and cutting stock problems became one classic application for the column generation method, as studied by Vance et al. [41], Chen et al. [42], and Vanderbeck [43]. Degraeve et al. [44] embedded the column generation procedure within the branch-and-bound frame for the cutting stock problem. Peeters and Degraeve [45] proposed a column generation algorithm to calculate the lower bound of type-I SALB problem. Column generation has been proven to be one of the most successful approaches for solving large-scale integer programming, especially for problems that contain an enormous number of columns in its mathematical model. Rather than enumerating all columns explicitly, the column generation method solves this problem by bringing the column into the basis only when needed. The main advantage of column generation is that the original problem is formulated in the master problem, which only has a selected set of columns and subproblems. Thus, the master problem is defined as the restricted master problem (RMP).

The column generation procedure deals with the reformulation model by iteratively solving RMP and pricing problems. The simple idea in column generation is not to represent all variables (columns) explicitly; it works with a reasonably small subset of columns. Similar to looking for new basic variables in the simplex algorithm, column generation deals with adding columns to the master problem. A column with the most negative reduced cost is added to the master problem after solving the pricing problem.

The RMP model for the TALBz problem is

$$\text{Minimize} \sum_{j \in J} y_j. \tag{25}$$

Subject to

$$\sum_{j \in J} B_{ij} y_j = 1 \quad \forall i \in I, \tag{26}$$

$$0 \le y_j \le 1 \quad \forall j \in J. \tag{27}$$

The objective is to minimize the sum of mated-stations which are represented as y_i in (25). Variables are set to decide the values of the columns in the assignment set and are relaxed into continuous variables. We keep task occurrence constraint set (26) in the master problem. The set of B_{ij} consists of task assignments at different stations. Each column in this set represents a pattern for one mated-station's assignment solution.

Column generation is an extension of the simplex algorithm. The RMP only contains a small subset of possible columns at the beginning, and then other columns are generated by solving pricing subproblems. In each iteration of column generation, a column of task assignment with the most negative reduced cost is appended to the solution set B_{ij} of the RMP. Since the reduced cost (RC) can be calculated as

$$\text{RC} = \text{Min}\left(1 - \sum_{i=1}^{n} \pi_i v_i\right) = 1 - \text{Max} \sum_{i=1}^{n} \pi_i v_i \quad \forall i \in I, \tag{28}$$

where v_i is the task variable in the new assignment solution and π represents the dual cost vector, which consists of dual variables of the RMP. We can design the pricing subproblem to seek the minimum RC.

The subproblem is to obtain one assignment solution given a list of dual variables of the current RMP. The objective of the subproblem derived from (28) is displayed in (29), where the minimum RC is equal to the maximum sum of the products of variable v_i and π.

$$\text{Maximize} \sum_{i \in I} \pi_i v_i. \tag{29}$$

The constraints can be divided into five blocks: position assigning constraints (30), sequence (among different stations) constraints (31), sequence (in one station) constraints (32)–(34), and cycle time constraints (35), and zoning constraints (36)–(37).

$$\sum_{k \in K(i)} s_{ik} = v_i \quad \forall i \in I, \tag{30}$$

$$v_{i1} + \sum_{i_3 \in F^*(i_2)} v_{i3} - \left\| F^*(i_2) \right\| \cdot v_{i2} \le 1 \quad \forall i_1 \in P(i_2),\ i_2 \in I - P_0, \tag{31}$$

$$t_i^f - t_h^f + \mu \cdot (1 - v_i) + \mu \cdot (1 - v_h) \ge t_i \quad \forall i \in I - P_0,\ h \in P(i), \tag{32}$$

$$\begin{aligned} &t_i^f - t_p^f + \mu \cdot (1 - s_{ik}) + \mu \cdot (1 - s_{pk}) + \mu \cdot z_{ip} \ge t_i \\ &\forall i \in I,\ p \in \{r | r \in I - (P_a(i) \cap S_a(i)) \text{ and } i < r\},\ k \in K(i) \cap K(p), \end{aligned} \tag{33}$$

$$t_p^f - t_i^f + \mu \cdot (1 - s_{ik}) + \mu \cdot (1 - s_{pk}) + \mu \cdot (1 - z_{ip}) \geq t_p \quad \forall i \in I,\ p \in \{r | r \in I - (P_a(i) \cap S_a(i)) \text{ and } i < r\},\ k \in K(i) \cap K(p), \tag{34}$$

$$t_i \leq t_i^f \leq \text{ct} \quad \forall i \in I, \tag{35}$$

$$v_i - v_j = 0 \quad \forall \{i, j\} \in \text{LT}, \tag{36}$$

$$v_i - v_j \neq 0 \quad \forall \{i, j\} \in \text{IT}, \tag{37}$$

$$s_{ik} = 0 \text{ or } 1 \quad \forall i \in I,\ k \in K(i), \tag{38}$$

$$v_i = 0 \text{ or } 1 \quad \forall i \in I, \tag{39}$$

$$z_{ip} = 0 \text{ or } 1 \quad \forall i \in I,\ p \in \{r | r \in I - (P_a(i) \cap S_a(i)) \text{ and } i < r\}. \tag{40}$$

The solution of the subproblem represents the task assignment for one mated-station. Since there are two positions in one mated-station, decision variables s_{ik} are established to determine on which side (k = left or right) task i is distributed. If task i is assigned on any side of the station, then the variable v_i is recorded as a '1' element in the solution; otherwise, it is equal to '0'. This relationship is expressed in constraint (30). Constraint set (31) ensures the precedence sequence for tasks among different stations. In the original model, these are realized through constraint set (3). We reformulated this constraint into a nonredundant constraint set (31). After reducing the dimensions, it is more parsimonious compared to the established expressions. Constraint (31) prevents that situation from occurring: immediate predecessors and successors of task i are selected in the solution, but task i does not appear in the assignment. This situation would violate the sequence constraints among different mated-stations. Constraint (32) inherits from (17), which defines the precedence relationship for tasks assigned in the same mated-stations. For tasks without any precedence sequence requirement, constraint sets (33) and (34) become active when two tasks occur in the same position in one mate-station. The indicator variable z_{ip} is created to control the appearance sequence of two tasks: when task i is before task p, z_{ip} equals 1, while if the sequence reverses, z_{ip} is 0. Cycle time constraints (35) are kept as constraint set (20) in the D-W decomposition model. Constraint sets (36) and (37) represent the zoning constraints. Tasks in the set of LT are required to install at the same workstation as guaranteed by constraint (36), while tasks in the set of IT must be assembled at different workstations as limited by constraint (37).

4. Proposed Branch-and-Price Algorithm

4.1. General Procedure of Column Generation. The branch-and-price algorithm is usually integrated into the framework of the branch-and-bound approach. We apply the column generation method to solve the linear model of the TALB problem at each node in the branch-and-price searching tree.

We start from the root node in the branch-and-price tree. The TALB problem is first resolved through an initial heuristic algorithm. Based on this solution, the branch-and-price algorithm can enter the branching node calculation stage. The column generation method is designed in the branching node calculation part. The initial solution of the RMP is derived from the heuristic algorithm. Then, the TALBz problem is solved through the iterations between computing the RMP and subproblems. Dual values of the RMP can be utilized as parameters of objective function (29) in the model of subproblems. The subproblem is solved after obtaining a set of optimal dual values of the RMP. The new column, which represents a work station's line balancing solution, is added into the RMP if its reduced cost value is negative, because a column with a negative reduced cost value can improve the RMP.

The column generation iterations stop when the value of the RMP is not larger than the lower bound. Two conditions indicate that the column generated by the subproblem cannot improve the RMP anymore, and the column generation loop breaks:

(1) If the value of the reduced cost from the current subproblem is nonnegative, no better solution could be obtained through further iterative calculations.

(2) When the reduced cost is negative, its absolute magnitude is so small that the RMP cannot reach a better result even when absorbing that new column. This is because the final solution of the line balancing problem contains an integer number of workstations. The rounded-up value of the improved RMP equals the solution in the current loop when the reduced cost is located within a range where $\lceil Z_B \rceil = \lceil Z_B/(1 - RC) \rceil$.

When the column generation finishes at an active node, an integer assessment is made to examine the results of the RMP model. If the RMP solution satisfies the integer condition and is better than the best value, we update the current best value with this solution and fathom the active node. The latest updated value is compared with the lower bound. If it equals the value of the lower bound, then the algorithm

breaks and returns this result as the optimal value. If the RMP delivers a fractional result, options are taken for different cases:

(i) For fractional node i where $Z_i > Z_{best} - 1$, the node i is fathomed for bounding.

(ii) For fractional node j where $Z_j \leq Z_{best} - 1$, branching takes place on the result derived from the node calculation process. New child nodes are generated according to the result of the active node.

The search tree is formed from nodes that are unsolved. Then, the algorithm selects and enters into the next node in the branch-and-price searching tree. This procedure continues until all nodes in the searching tree are detected or the termination condition is invoked during the calculation.

$$S_L = \lceil \frac{T_{SumL}}{\mathrm{CT}} \rceil, \tag{41}$$

$$S_R = \lceil \frac{T_{SumL}}{\mathrm{CT}} \rceil, \tag{42}$$

$$S_E = \lceil \frac{\max\left(\left(T_{SumL} - \left(\left(S_L + S_R\right) \times \mathrm{CT} - \left(T_{SumL} + T_{SumR}\right)\right)\right), 0\right)}{\mathrm{CT}} \rceil, \tag{43}$$

$$\mathrm{LB}_{NM} = \max\left(S_L, S_R\right) + \lceil \frac{\max\left(\left(S_E - \left|S_L - S_R\right|\right), 0\right)}{2} \rceil. \tag{44}$$

The lower bound and the upper bound also play important roles in the branch-and-price algorithm. In this algorithm, the lower bound of the number of mated-stations LB_{NM} is calculated as (41)–(44), which is taken as *Lower bound 1* from Wu et al. [11] and the initial upper bound is set as the value of the heuristic solution obtained at the root node.

In addition to measuring via the reduced cost value, the node calculation will stop when its solution meets the lower bound. The upper bound is updated according to the latest integer solution. The algorithm keeps narrowing the gap between the lower bound and the upper bound until the search is finished. Thus, a high-quality initial upper bound can facilitate the resolving process. Although heuristic methods for SALB and TALB problems were studied, these methods cannot be directly applied in the branch-and-price algorithm, since initial solutions must satisfy the Ryan–Foster branching rule implemented in the branch-and-price algorithm. We therefore design a novel heuristic procedure to obtain this type of solution based on the "earliest start time" priority rule.

4.2. Earliest Start Time Calculation. In our research, an initial solution of the TALBz problem is needed for the column generation process at each node in the branch-and-price tree. Both the sequence and position constraints affect the final result of the TALBz problem. To minimize the idle time induced by the sequence constraints and the direction constraints on the two-sided assembly line, one strategy is to schedule task assignments continuously. Thus, tasks are recommended to start at the earliest possible time. In the enumeration procedure, one task can start if all of its predecessors have been finished. One task's finish time can be calculated through the time transfer function:

$$T_j^f = T_j^s + t_j \quad \forall i \in \mathrm{Pool},\ j \in P_i^s, \tag{45}$$

A set of "*unassigned tasks*" is created in the heuristic algorithm. This set includes tasks that are not distributed to workstations. The heuristic process also creates a "*Pool*" set that contains items in the "*unassigned tasks*" set for which preceding tasks are finished. When there is any predecessor j for task i at the current mated-station, the finish time of j is equal to its start time plus task j's operation time (45). If predecessor j does not appear in the current mated-station, its finish time is set to zero. Then, the earliest start time of a task on the current station can be determined by its predecessors' finish time. For the tasks that have a fixed position (left or right) attribute:

$$\mathrm{EST}_i = \max\left(\max\left(T_j^f\right), C_w\right) \quad \forall i \in \mathrm{Pool},\ j \in P_i^s,\ w \in D_i, \tag{46}$$

The earliest start time (EST) chooses the maximum value among the predecessors' finish time and the current capacity time of one position C_w (depending on which side the task is allowed to perform).

For tasks allowed to be done on both the left and right sides of the assembly line, the earliest start time is calculated by

$$\mathrm{EST}_i = \mathrm{median}\left(\max\left(T_j^f\right),\ \min\left(C_w\right),\ \max\left(C_w\right)\right) \quad \forall i \in \mathrm{Pool},\ j \in P_i^s,\ w \in D_i, \tag{47}$$

Since tasks without a positional preference are flexible for assignment on both sides of a station, we prefer the position that provides an earlier start time. Thus, the earliest start time of task i, EST_i, selects the median value of three parameters: (1) the maximum finish time of task i's predecessors, which are assigned in the same mated-station; (2) the minimum capacity of the left position and right position; and (3) the maximum capacity of the left position and right position.

Once we complete an assignment, the start time and finish time of that task will be updated. The just assigned task is removed from the "*Pool*" and "*Unassigned*" sets, and new tasks are added into the "*Pool*" set according to their predecessors' status.

4.3. Heuristic Rules. Based on the time transfer function and the method of the earliest start time calculation, we develop the mated-station-oriented heuristic algorithm for the TALBz problem. Both sides of a mated-station are simultaneously checked to determine the current best assignment. Several rules are discussed here to minimize the total idle time:

(i) For tasks whose predecessors have already been finished, we select the task with the minimum value of the earliest start time. This could reduce the sequence-induced idle time among tasks.

(ii) If tasks have the same start time, tasks with a fixed positional attribute (L or R) are proposed to give greater priority than the E type task.

(iii) If parallel tasks still exist, the task with the longest operation time is preferable. This could increase the utilization of the current workstation.

For candidate tasks in the set of "*Pool*," the feasibility of the solution is tested. In the SALB problem, a task-oriented heuristic algorithm only checks the cycle time capacity for the solution. However, initial solutions are needed not only in the root node but also in branching nodes in the branch-and-price algorithm. Due to the Ryan–Foster branching rule used in this work, we follow the branching constraints in the process of generating initial solutions. As required by the zoning and branching constraints, some tasks should be assigned to the same workstation and some to different workstations. Possible assignments of different tasks are created, and some may be pruned if constraints are violated. In general, two conditions exist when the solution can be pruned.

The first condition is that the last finish time of the task's assignment exceeds the cycle time.

$$\begin{aligned} &\mathrm{LFT}_i = \max\left(\mathrm{EST}_j + t_j\right), \\ &\mathrm{LFT}_i > \mathrm{ct} \quad \forall i \in \mathrm{Pool},\ j \in G_i. \end{aligned} \tag{48}$$

In (48), LFT_i is the last finish time of tasks in set G_i, where task i belongs to the "*Pool*" set. Assignments in the heuristic procedure take the branching constraints into consideration. For the task i, which has a linked task pair, we search all unassigned tasks that must be operated in the same mated-station; the set of these tasks is referred to as group "G_i" of task assignment. The set of task groups consists of the candidate task, its linked task pair, and all the predecessors of the candidate task that have not yet been assigned. The procedure will search all of the linked tasks and predecessors of the items in the group. The group is updated as the new related task is added into this set in an iterative way until all of the related tasks are included in this group. Then, assignment of these tasks is taken simultaneously in one solution. We record the longest finish time in the solution and compare it with the cycle time. The solution with the final finish time that exceeds the cycle time is pruned.

The second condition limits the solution which contains contradictable tasks in one mated-station. We need to check whether the following two cases occur in the solution:

(a) The task in a possible assignment, either the candidate task or the grouping task, has a contradictory task that has already been assigned to the current mated-station.

$$\exists v_p = 1 \quad \forall i \in \mathrm{Pool},\ j \in G_i,\ p \in C_j, \tag{49}$$

Task i is the candidate task in the pool, and G_i is the group of task i that should be packaged together. C_j denotes the set of tasks that are incompatible items of task j that cannot be assigned to the same workstation. These tasks are defined according to the zoning and branching constraints. When task p in C_j has already been assigned, the value of variable $v_p$equals one.

(b) The grouping tasks contain incompatible task pairs in the assignment.

$$\begin{aligned} &\mathrm{SC}_{ij} = G_i \cap C_j \quad \forall i \in \mathrm{Pool},\ j \in G_i, \\ &\exists m \in \mathrm{SC}_{ij} \neq \varnothing. \end{aligned} \tag{50}$$

The set SC_{ij} consists of self-contradictory items in the group set of task i in the pool. As the number of nodes grows, branching constraints become complicated. Thus, some linked and incompatible task pairs may be identical in the grouping set. There will not be any feasible result because of contradictory requirements. Assignment design that violates these constraints must be pruned.

4.4. Initial Solution Generation. The key steps in the heuristic are illustrated in Heuristic Algorithm 1.

4.5. Branching Scheme. Since the RMP is a linear relaxation of the Dantzig–Wolfe decomposition model, fractional solutions may appear in the optimal solution. However, an integer result is needed to meet the requirements of practical production. In the common method of handling the linear programming problem, integer solutions can be obtained using the branch-and-bound method. Effective branching based on the fractional variables can result in child nodes with integer solutions. In this proposed branch-and-price method, the branch-and-price parameter combines column generation with a branch-and-bound algorithm. The theoretical principle was illustrated by Yin et al. [46] who adopted this method to solve the SALB problem.

The common approach of branching in the branch-and-bound method is dealing with the fractional variable. Fractional variables are rounded up/down by adding additional constraints to the original model. However, this method does not work in the branch-and-price algorithm. Variables corresponding to columns in the set of line balancing results make up the RMP solution. In addition to the initial part, which originates from the heuristic algorithm, other columns are generated from the pricing subproblem. Since constraints in the RMP model cannot control the pricing subproblem, columns that are eradicated in the RMP may return again in the later process of column generation. Additionally, adding new constraints to the model of the master problem will change its dual problem. Thus, we design a new branching strategy for the TALBz problem based on the Ryan–Foster branching rule. In this work, branching constraints are realized in the pricing subproblem instead of in the RMP.

After determining a pair of tasks from the fractional solution, two child nodes are built in the left branch and the right branch. The following constraints are added to the subproblem model of (29)–(40) at different nodes. For nodes in the left branch,

(1) Initialize $Pool = P_0$
(2) **while** *Pool not empty* **do**
(3) **for** *task i in Pool* **do**
(4) *Pre-assignment of task i*
(5) **end for**
(6) **if** *all task i* $LFT_i > CT$ **then**
(7) *Start a new workstation assignment*
(8) **end if**
(9) *Earliest start time calculation*
(10) *Sorting tasks in Pool*
(11) **for** *task in Pool* **do**
(12) *Assignment of task*
(13) *Workstation capacity update*
(14) **end for**
(15) *Pool set update*
(16) **end while**

First Step. Initialize the task assignment from the first workstation with the set of P_0. Create a candidate *Pool* that copies P_0.

Second Step. Capacity checking is done through the assignment test. To ensure that there is enough time capacity for the tasks in the candidate set, a test run of the assignments on the current workstation is performed. For a task without any linked pair task on the left branch, the operation time of this task is compared with the residual time capacity. For a task that has a linked pair task on the left branch, the largest finish time of the packing group is compared with the current station's remaining capacity. If all tasks' finish time periods are beyond the limit of cycle time, the heuristic begins to plan a new workstation.

Third Step. Calculate the earliest start time for all tasks in the candidate set based on the time transfer function.

Fourth Step. Sort the tasks in the candidate set according to the priority rule discussed in the former section.

Fifth Step. Assign the first task in the sorted list. Assigning strategies are designed according to the different attributes of task branching constraints: (1) For a task that has neither a left pair nor a right pair, task assignment can be done directly. (2) The second type of task has a right pair but not a left pair. The heuristic procedure checks whether there is any contradictory task at the current station. (3) For a task with left pair constraints, a package of tasks related to the left pair constraints needs to be determined. The time capacity constraint and the right pair task constraints are tested for this package. Assignment of the task package is taken if all these constraints are satisfied. The latest capacity of the workstation is updated after the assignment.

Sixth Step. Update the set of candidate tasks, remove the task that was just distributed, and append new tasks into the candidate set. If all predecessors of one task have already been assigned, then list this task in the pool.

Last Step. Return to Step 2 until all tasks have been assigned to the assembly line.

Algorithm 1: Pseudo-code of initial solution.

$$v_l = v_m \quad \forall \{l, m\} \in L. \tag{51}$$

In the left branch, constraints require that task $\{l, m\}$ in the left branch pair L be assigned together to one mated-station. Either $v_l = v_m = 1$, which means that tasks l and m are included in the assignment, or $v_l = v_m = 0$, which means that the current solution does not contain l or m.

For nodes in the right branch,

$$v_l + v_m \leq 1 \quad \forall \{l, m\} \in R. \tag{52}$$

In the right branch, constraints require that the task pair of $\{l, m\}$ in the right branch pair R be distributed to different mated-stations. Thus, assignments of the mated-station may cover only one task ($v_l = 0$, $v_m = 1$ or $v_l = 1$, $v_m = 0$), or none of the tasks $\{l, m\}$ appear in the current mated-station ($v_l = v_m = 0$).

Based on the above approach, branching constraint sets (51) and (52) can match the zoning constraint sets (36) and (37) that appear in the subproblem model, which contains the restrictions among tasks. Our proposed method has more advantages in handling this type of problem than other methods that adopt the original model. However, zoning constraint sets (36) and (37) in the B&P algorithm do restrict these relationships of tasks. Thus, they can be directly applied to define the branching constraints instead of adding additional expressions.

5. Computational Experiments

Three categories of numerical experiments are carried out. In the first part, benchmark datasets with the zoning constraints are tested to evaluate the proposed B&P algorithms for the TALBz problem. In the second part, the benchmark problems without any zoning constraints are resolved to make a comparison with the most recent studies of TALB problem. In the third part, four new datasets from the real-world production line, which happen at a premier brand automaker's assembly shop, are tested to verify the effectiveness of the B&P algorithm for practical problems. The proposed B&P is coded in Python 3. An interface with the *Gurobi* 7.5.2 solver is built in the algorithm. The computer on which the experiments are conducted has an Intel Core (TM) i7-3770 3.4 GHz CPU and 8 GB RAM memory.

5.1. Benchmark Instances with the Zoning Constraints. Six benchmark problems, which originate from the literature, are studied in this part. These instances can be classified into

small-sized problems, P12–P16–P24, and large-sized problems: P65–P148–P205. Problems P12 and P24 are studied with reference to [18]. The data of problems P16, P65, and P205 are taken from [9]. P148 contains 148 tasks and was first designed by Bartholdi [8]. We modified the value of task 79 from 281 to 111 and task 108 from 282 to 43 according to Lee et al. [9]. The zoning constraints of these instances listed in Table 1 are taken from Baykasoglu and Dereli [21] and Özcan and Toklu. [16].

Computational results are shown in Table 2. The performance criterion of the number of mated-stations (NMs) is used to evaluate the proposed algorithm. There are a few studies considering the two-sided assembly line balancing problem with the zoning constraints. Results of the proposed B&P were compared with those reported in the best studies. The ant colony (AC) by Baykasoglu and Dereli [21], the goal programming (GP) by Özcan and Toklu [16], and the bees algorithm (BA) by Özbakır and Tapkan [3] have obtained the best solutions for the criterion of the number of work stations. But, these published results are not available for direct comparison, since their objective is to minimize the number of workstations. An equivalent number of mated-stations of the published results are calculated for comparison. The number of workstations in the published results is used, and it is divided by 2. The equivalent number of mated-stations equals to this value. If it is not an integer value, then round it up. The average number of workstations is taken from Özbakır and Tapkan [3]; the minimum number of workstations is taken from Baykasoglu and Dereli [21] and Özcan and Toklu [16]. Additionally, the original models (1–13) are tested using the *Gurobi* solver to compare with the models reformulated in the proposed B&P algorithm.

Tests are conducted under various cycle times (CTs). The value of "computing minimum mated-stations" (CMSs) is calculated directly using the summation of the total tasks' operation time divided by the double value of CT, which is $\mathrm{CMS} = T_{Sum}/(2 * \mathrm{CT})$. This value can measure the slack time between the theoretical minimum number of mated-stations and the lower bound (LB). The lower bound on the mated-station's number is equivalent to the value in [11]. We set a time limit of 300 s for the computation. When the run time of the program exceeds this limit, the algorithm terminates and returns the current best solution. Instances not tested or solved by an algorithm are labelled with "-" for results in Table 2 and in the later results. The proposed B&P obtained the best solutions in 27 of 32 instances considering the zoning constraints. The original models are effective on small-sized problems (P12, P16, P24), and large-sized problems cannot be solved by the *Gurobi* solver within the time limit. The GP only tests the small-sized problems P12, P16, and P24. The BA outperforms other algorithms in the reported research. The BA generated better solutions on 5 instances compared with the B&P algorithm. The proposed B&P generated 3 better solutions than the AC. However, these results cannot strictly evaluate the performance of the algorithms. This is mainly because of the following: First, the real number of mated-stations may be larger than the equivalent value used in this study when some mated-stations only have the task assignments on the single-side workstation. Second, although the average number of workstations for each instance is given, the results generated by the AC and BA are nondeterministic and the values vary for each computation.

5.2. Benchmark Instances without the Zoning Constraints. In order to make a comparison with the most recent research of the general TALB problem, benchmark problems without any zoning constraints are tested. The results are shown in Table 3. The proposed method is compared with the heuristic algorithm of group assignment (GA) by Lee et al. [9], the branch-and-bound (B&B) algorithm of Wu et al. [11], the tabu search (TS) algorithm of Özcan and Toklu [47], the reported best metaheuristic of the modified simulated annealing (mSA) algorithm of Khorasanian et al. [48], and the state-of-the-art branch bound and remember (BB&R) algorithm of Li Z. et al. [17]. Additionally, the original models (OM) with removed zoning constraint sets (9) and (10) are solved using Gurobi solver to evaluate the reformulated models in the B&P algorithm.

The B&P algorithm yielded the best solutions in 25 of 32 instances. The proposed algorithm was quite efficient and effective for small-scale problems. All the tests of P12, P16, and P24 reached the optimal results. Seven cases of the large-size problem, P65 (CT 326), P148 (CT 204), and P205 (CT 1322, 1510, 1699, 2077, 2454) returned the final result which has a one unit gap with the value of the best solution. Lee et al. [9] showed their computational results of P65, P148, and P205 instances. B&P found better solutions in 9 instances than GA did in terms of the number of mated-stations. Wu et al. [11] tested P12, P16, P24, P65, and P148 in their study, where B&B algorithm outperformed the B&P algorithm in only one instance of P148 (CT 204). The B&P algorithm also obtained solutions with fewer mated-stations than the TS algorithm for 7 instances. The mSA presented the results with four better solutions than the B&P in P148 and P205. The BB&R is an effective and efficient method for the TALB problem, and it produced seven better results than the B&P. The original model is still effective at solving the small-scale problems (P12, P16, P24), and it can barely obtain a feasible solution for large-scale problems (P65, P148, P205).

Regarding the running time, we list the computation times of the original models (OM Gurobi) and the proposed B&P algorithm. The CPU times of B&B for P12–P148 were solved between 0.001 and 68.125 s on a Pentium 4 2.0 GHz PC [11]. The CPU times of the TS algorithm were reported to be between 0.06 and 2100 s on a Pentium 4 3.0 GHz PC [47]. The computation times of the mSA algorithm were less than 5 seconds for every problem case on an Intel Core 2 Duo T9600 PC with 2.8 GHz CPU [48]. The BB&R method found optimal solutions for all tested instances in 1.0 s on average with an Intel Core (TM) i7-4790S 3.2 GHZ CPU PC. Due to the great differences among these algorithms in terms of computers, programming compilers, and running systems, it is difficult to accurately distinguish the performance of algorithms based on the reported results. However, it is clear

Table 1: Zoning constraints for the benchmark problems.

Test problem	Set of task pairs	
	LT	IT
P12	{1, 4}	{3, 5}
P16	{3, 6, 7}	{8, 9, 10}
P24	{1, 11}; {7, 10}	{14, 24}
P65	{3, 23, 24}; {31, 32}; {36, 37}	{10, 30}; {46, 56}
P148	{29, 31}; {37, 38}; {40, 41}; {50, 51}; {11, 12, 13}; {90, 111, 112}	{8, 145}; {30, 70}; {48, 110}; {55, 71}; {102, 108}; {122, 125}; {142, 147}
P205	{2, 3}; {7, 8, 9, 10, 11, 12}; {20, 21, 22, 23}; {30, 31, 32}; {37, 38, 39}	{25, 27}; {29, 33}; {35, 110}; {93, 109}; {114, 174}; {144, 154}; {156, 190}; {77, 88}; {87, 100}; {40, 70}

Table 2: Performance comparison of the benchmark instances with zoning constraints.

Problem	CT	CMS	LB_{NM}	AC	GP	BA	OM (Gurobi)		B&P	
				NM	NM	NM	NM	CPU time	NM	CPU time
P12	4	3.13	4	—	—	—	4	0.244	4	0.282
	5	2.5	3	3	3	3	3	0.285	3	0.494
	6	2.08	3	3	3	3	3	0.353	3	0.494
	7	1.79	2	3	2	2	2	0.167	2	1.563
P16	15	2.73	4	—	3	4	4	0.518	4	0.072
	18	2.28	3	—	3	3	3	0.311	3	0.060
	20	2.05	3	—	3	3	3	0.397	3	0.059
	22	1.86	2	—	3	3	3	0.446	3	0.154
P24	25	2.8	3	—	3	3	3	2.397	4	40.479
	30	2.33	3	3	3	3	3	17.231	3	0.122
	35	2.00	2	3	2	2	2	2.924	3	27.771
	40	1.75	2	2	2	2	2	1.443	2	0.121
P65	326	7.82	8	9	—	9	—	302.390	9	244.108
	381	6.69	7	8	—	8	—	302.035	8	0.395
	490	5.20	6	6	—	6	6	301.505	6	0.384
	544	4.68	5	5	—	5	5	301.445	5	0.413
P148	204	12.55	13	13	—	14	—	321.140	14	315.203
	255	10.05	11	11	—	11	—	317.802	11	2.474
	357	7.18	8	9	—	8	—	312.138	8	2.017
	408	6.28	7	8	—	7	—	312.026	8	312.928
	459	5.58	6	7	—	7	—	310.547	7	314.697
	510	5.02	6	6	—	6	—	310.696	6	2.376
P205	1133	10.30	11	13	—	12	—	315.844	12	1.829
	1322	8.83	9	11	—	11	—	315.469	11	1.820
	1510	7.73	8	10	—	9	—	316.281	10	252.363
	1699	6.87	7	9	—	9	—	323.579	9	1.918
	1888	6.18	7	8	—	8	—	316.097	8	1.708
	2077	5.62	6	8	—	8	—	316.238	8	308.554
	2266	5.15	6	7	—	7	—	315.647	8	318.713
	2454	4.76	5	7	—	7	—	315.148	7	1.762
	2643	4.42	5	7	—	7	—	316.396	7	1.660
	2832	4.12	5	6	—	6	—	315.917	6	1.747

that the B&P algorithm is quite fast obtaining optimal solutions for small-sized problems. All instances of P12, P16, P24 and P65 were solved optimally within 20 seconds. For the large-scale problem of P148 and P205, 8 out of 16 instances obtained the optimal result in less than 20 seconds.

5.3. Real Production Cases. Next, four new problems, P56, P58, P81, and P161, which were derived from real production cases at BMW assembly shops, were tested in this paper. Problems P56, P58, and P81 were taken from one powertrain system assembly line at the BMW-Brilliance Plant Dadong and contained 56, 58, and 81 tasks, respectively. The total operation times of the installations of P56, P58, and P81 are 889.46 s, 731.28 s, and 1006.96 s, respectively. These three cases occurred in three different sections in the powertrain assembly line. This powertrain system was installed in the preassembly area with a cycle time of 101 seconds. The production rate (*PR*) of the powertrain part was 33 units per hour. Problem P161 consists of 161 tasks of underbody component assembly that showed at BMW-Brilliance Plant Tiexi. The majority of underbody parts are installed in the tilting line, which is shown in Figure 4. The tilting line is one key part of the

Table 3: Performance comparison of the benchmark instances without zoning constraints.

Problem	CT	CMS	LB_{NM}	GA	B&B	TS	mSA	BB&R	OM (Gurobi)		B&P	
				NM	NM	NM	NM	NM	NM	CPU time	NM	CPU time
P12	4	3.13	4	—	4		4	4	4	0.244	4	0.037
	5	2.5	3	—	3	3	3	3	3	0.223	3	0.017
	6	2.08	3	—	3	3	3	3	3	0.376	3	0.015
	7	1.79	2	—	2	2	2	2	2	0.198	2	0.584
P16	15	2.73	4	—	4		4	4	4	1.241	4	0.043
	18	2.28	3	—	3		3	3	3	0.525	3	7.884
	20	2.05	3	—	3		3	3	3	0.504	3	2.715
	22	1.86	2	—	2	2	2	2	2	0.483	2	10.574
P24	25	2.8	3	—	3	3	3	3	3	6.895	3	0.234
	30	2.33	3	—	3	3	3	3	3	59.807	3	0.225
	35	2.00	2	—	2	2	2.45	2	2	16.762	2	8.549
	40	1.75	2	—	2	2	2	2	2	2.305	2	0.231
P65	326	7.82	8	9	-	9	9	8	-	305.460	9	15.643
	381	6.69	7	8	7	8	7	7	-	304.837	7	0.904
	490	5.20	6	6	6	6	6	6	6	304.146	6	0.713
	544	4.68	5	6	5	5	5	5	6	303.641	5	2.301
P148	204	12.55	13	14	13	13	13	13	—	302.533	14	300.00
	255	10.05	11	11	11	11	11	11	—	308.973	11	18.330
	357	7.18	8	8	8	8	8	8	—	306.029	8	5.605
	408	6.28	7	7	7	7	7	7	—	301.672	7	5.703
	459	5.58	6	7	6	6	6	6	—	307.293	6	2.283
	510	5.02	6	6	6	6	6	6	—	306.425	6	5.797
P205	1133	10.30	11	12	—	12	11	11	—	304.414	11	45.277
	1322	8.83	9	11	—	11	9.7	9	—	305.732	10	300.00
	1510	7.73	8	10	—	9	8	8	—	306.836	9	300.00
	1699	6.87	7	8	—	9	7.1	7	—	306.261	8	300.00
	1888	6.18	7	8	—	8	7	7	—	304.959	7	88.521
	2077	5.62	6	7	—	7	6	6	—	304.935	7	300.00
	2266	5.15	6	7	—	7	6	6	—	304.328	6	7.540
	2454	4.76	5	6	—	6	5	5	—	304.302	6	300.00
	2643	4.42	5	6	—	6	5	5	—	303.677	5	7.173
	2832	4.12	5	5	—	5	5	5	—	303.620	5	1.134

main assembly line that it is operated in the two-sided mode. Using the tilting line improves the ergonomics of assembly production. The main parts installed in the tilting line include the harness, brake lines, fuel supply system, roof antenna, and numerous plugs on the vehicle body. The working process consists of 1615.82 s of operations in total. To simplify the line balancing problem, some continuous operations are combined as one task according the standard referenced instructions. For example, preparation of the fuel tank assembly involves three key processes: fuel task scanning (operation time 10.26 s), adhesive pad bonding (operation time 15.66 s), and fuel tank installation (operation time 10.15 s). These three processes are performed continuously in practice and integrated as task 132 (operation time 36.00 s) in P161 problem. Finally, 161 tasks are counted in the P161 problem. These works are completed by 38 workers at 19 mate-stations in the current production line we studied. The current *PR* of the main assembly line in the BMW Tiexi plant is 60 units of cars per hour with a cycle time of 55.38 seconds. Zoning constraints are involved in practical production problems, and the linked task pairs (LT) and the incompatible task pairs (IT) are designed for each problem.

After optimization using the B&P algorithm, the number of mated-stations is reduced for all cases compared to that in previous production schedules. For the middle-sized problems P56 and P58, the NM reached the lower bound. For the large-sized problem P81, the value of CMS is almost equal to the lower bound; we obtained the current best solution with six mated-stations. In problem P161, the number of mated-stations is reduced to 15 in total. This value also equals to the lower bound. Since these instances are new datasets for the TALBz problem, we test the original models (OM) of these problems through the *Gurobi* solver to compare their performances. The optimization results are shown in Table 4.

To test our proposed method for more practical cases, we set various lengths for the cycle time in experiments. Instances are extended for situations when the production rate (*PR*) increases. The passing time T_p of the vehicle on each station is calculated as

$$T_p = \frac{3600}{\mathrm{PR}}. \tag{53}$$

The recommended CT for assembly production is in the range of (0.92~0.95) $*$ T_p. For problems P56, P58, and P81 involving the powertrain system assembly, we set *PR* to

Figure 4: Assembly line of vehicle underbody parts.

Table 4: Performance comparison of vehicle assembly production instances with zoning constraints.

Problem	Zoning constraints		CT	CMS	LB_{NM}	OM (*Gurobi*)		B&P	
	LT	IT				NM	CPU time	NM	CPU time
P56	{[9, 11], [16, 17], [51, 52]}	{[38, 42]}	80.57	5.52	6	7	302.597	6	0.373
			84.6	5.26	6	6	301.711	6	142.117
			89.05	4.99	5	6	301.413	6	301.227
			94.0	4.73	5	6	301.445	5	0.369
			99.53	4.47	5	5	171.507	5	0.396
			101	4.40	5	6	301.756	5	0.372
P58	{[18, 23], [33, 35], [43, 45]}	{[4, 11], [25, 38]}	80.57	4.54	5	5	301.660	5	0.358
			84.6	4.32	5	5	186.620	5	0.378
			89.05	4.12	5	5	301.671	5	0.375
			94.0	3.89	4	5	301.629	4	0.378
			99.53	3.67	4	4	301.414	4	0.389
			101	3.62	4	4	186.485	4	0.352
P81	{[3, 9], [23, 31], [35, 45], [60, 63], [70, 73]}	{[5, 17], [28, 33], [48, 64]}	80.57	6.25	7	8	304.927	7	0.834
			84.6	5.95	6	7	304.312	6	304.779
			89.05	5.65	6	7	304.376	6	0.807
			94.0	5.36	6	-	304.303	6	0.813
			99.53	5.05	6	6	304.561	6	0.811
			101	4.98	5	6	303.881	6	0.998
P161	{[16, 17], [46, 47], [76, 80], [131, 132], [137, 138]}	{[14, 37], [71, 100]}	55.48	14.56	15	-	304.136	15	5.394
			53.71	15.04	16	-	304.254	16	5.314
			52.06	15.52	16	-	304.449	16	6.398
			50.51	16.00	16	-	304.297	17	308.492
			49.04	16.47	17	-	304.661	17	10.627
			55.38	14.59	15	-	304.227	15	5.241

34~42 (34, 36, 38, 40, 42) units. The cycle time is taken as $0.94 * T_p$, which ranges from 80.57 s to 99.53 s. For problem P161, we determined that the cycle time is $0.94 * T_p$ as the *PR* lifts from 61 to 69 (61, 63, 65, 67, 69) units per hour. Thus, the cycle time was taken from 54.48 s to 49.04 s. Table 4 summarizes the computing results.

As presented, *Gurobi* can deal with middle-sized TALBz problems. However, *Gurobi* was unable to resolve any case of

the P161 problem within 300 s or returned a second-best solution when the computation finished. The B&P algorithm provided 15 better solutions out of 24 cases than the *Gurobi* solver. The B&P algorithm gave results equal to the lower bound for 21 of 24 cases. These instances were solved optimally. Three cases (P56-CT89.05, P81-CT101, and P161-CT50.51) obtained a final solution that has one unit gap with the lower bound. For these instances, their CMS values were equal to the lower bound which means that no slack time remained. This explains why the lower bounds are difficult to reach. Fewer instances were solved optimality by *Gurobi* dealing with the TALB problem with zoning constraints. These additional restrictions have resulted in higher complexity. However, the B&P algorithm can still return optimal solutions in a very short time on most cases. To summarize, the proposed B&P algorithm is effective for solving the TALBz problems, while the *Gurobi* is not competitive. The B&P algorithm clearly outperformed *Gurobi* which adopts the original models in terms of the computation time for all cases.

From the test results of all instances above, the performance of the branch-and-price algorithm is affected by several factors. (1) The scale of the problem size: for small- and middle-sized problems (P12, P16, P24, P56, P58, and P65), the proposed algorithm gave almost the optimal solution for all cases. The proceeding time of the program for the small- and middle-sized problems was relatively short: 40 of 44 cases were solved optimally within 15 seconds. As the size grew, the complexity increased, which resulted in longer running times (from 0.811 to 318.713 s) for large-scale problems (P81, P148, P161, and P205). It was difficult to reach the optimal result for large-sized problems; 31 of 44 cases (P81, P148, P161, and P205) obtained the optimal solution. (2) Slack time: the slack time is small when the CMS is near the lower bound. It depends on various cycle times and lower bounds of different cases. Although the CMS is not calculated as rigorously as the value of the lower bound, it can measure the possibility of reaching the lower bound because of the inevitable sequence-induced idle time in the TALB problem. The smaller the slack time is, the more difficult it is for the algorithm to reach the optimal solution. This limits the algorithm to obtain the optimal combination of solutions in a short amount of time.

6. Summary

This study verifies the effectiveness of using the column generation method to solve the TALBz problems. A B&P algorithm is presented to deal with the TALBz problem for the first time. The published models cannot be directly applied to implement a column generation procedure. Thus, Dantzig–Wolfe decomposition of the original model is conducted. Models of the master problem and subproblems are reformulated. The proposed branching strategy is demonstrated to be effective in obtaining an integer solution. Since the initial solution is used not only at the root node in the branch-and-bound tree but also at branching nodes, the heuristic procedure is able to generate feasible solutions which are in accordance with constraints at different nodes.

This novel exact algorithm shows advantages over previous exact approaches and metaheuristic methods. The solution procedure is more efficient than traditional exact algorithms, especially for large-scale TALBz problems. Different from the metaheuristic methods, the column generation procedure is based on deterministic models. The specific restrictions of the TALBz problems, such as the zoning constraints of tasks, can be directly embedded into the mathematical model. This approach makes it feasible to implement more practical constraints, such as process restrictions, resource restrictions, and line configuration restrictions, in real-world assembly production. An automotive company can minimize the production cost based on the deterministic results of the TALBz problem.

Experiments were performed on 32 instances of the general TALB problem and 56 instances with additional task restrictions. The proposed method obtained the optimal results in more than 80% of cases. Although the computation time became longer as the task number increased, especially for instances with a short slack time, promising results were obtained in most cases. The B&P algorithm has more advantages in solving real-world problems that have complex restrictions compared with all other methods. Comparing the performance of reformulated models and original models reveals that the new models are more effective and efficient to obtain optimal solutions (80.68% vs 39.77%). For middle- and large-size problems of practical assembly production, the B&P method outperformed the *Gurobi* solver. In most cases, the B&P algorithm can generate optimal solutions faster than the *Gurobi*.

Further studies will be performed to extend this approach to mixed-model assembly lines, which are more complex but more common than single-model assembly lines in automotive manufacturing plants. Another interesting direction could be designing a specific algorithm to resolve the pricing problems that are handled by the solver in this study. Designing some heuristic rules for the generated columns in that algorithm might improve the performance of the B&P algorithm in instances with little slack time.

Data Availability

The datasets used or analyzed during the current study are available from the corresponding author on reasonable request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (Grant nos. 2019YFB1705002 and 2017YFB0304100), National Natural Science Foundation of China (Grant no. 51634002), and Liaoning Revitalization Talents Program (Grant no. XLYC2002041).

References

[1] S. J. Hu, J. Ko, L. Weyand et al., "Assembly system design and operations for product variety," *CIRP Annals*, vol. 60, no. 2, pp. 715–733, 2011.

[2] A. Scholl, M. Fliedner, and N. Boysen, "Absalom: balancing assembly lines with assignment restrictions," *European Journal of Operational Research*, vol. 200, no. 3, pp. 688–701, 2010.

[3] L. Özbakır and P. Tapkan, "Bee colony intelligence in zone constrained two-sided assembly line balancing problem," *Expert Systems with Applications*, vol. 38, pp. 11947–11957, 2011.

[4] E. Erel and S. C. Sarin, "A survey of the assembly line balancing procedures," *Production Planning & Control*, vol. 9, no. 5, pp. 414–434, 1998.

[5] A. Scholl and C. Becker, "State-of-the-art exact and heuristic solution procedures for simple assembly line balancing," *European Journal of Operational Research*, vol. 168, no. 3, pp. 666–693, 2006.

[6] N. Boysen, M. Fliedner, and A. Scholl, "A classification of assembly line balancing problems," *European Journal of Operational Research*, vol. 183, no. 2, pp. 674–693, 2007.

[7] M. Eghtesadifard, M. Khalifeh, and M. Khorram, "A systematic review of research themes and hot topics in assembly line balancing through the web of science within 1990–2017," *Computers & Industrial Engineering*, vol. 139, p. 106182, 2020.

[8] J. J. Bartholdi, "Balancing two-sided assembly lines: a case study," *International Journal of Production Research*, vol. 31, no. 10, pp. 2447–2461, 1993.

[9] T. O. Lee, Y. Kim, and Y. K. Kim, "Two-sided assembly line balancing to maximize work relatedness and slackness," *Computers & Industrial Engineering*, vol. 40, no. 3, pp. 273–292, 2001.

[10] X. Hu, E. Wu, and Y. Jin, "A station-oriented enumerative algorithm for two-sided assembly line balancing," *European Journal of Operational Research*, vol. 186, no. 1, pp. 435–440, 2008.

[11] E. F. Wu, Y. Jin, J. S. Bao, and X. F. Hu, "A branch-and-bound algorithm for two-sided assembly line balancing," *International Journal of Advanced Manufacturing Technology*, vol. 39, no. 9, pp. 1009–1015, 2008.

[12] X. F. Hu, E. F. Wu, J. S. Bao, and Y. Jin, "A branch-and-bound algorithm to minimize the line length of a two-sided assembly line," *European Journal of Operational Research*, vol. 206, no. 3, pp. 703–707, 2010.

[13] A. Sepahi and S. G. J. Naini, "Two-sided assembly line balancing problem with parallel performance capacity," *Applied Mathematical Modelling*, vol. 40, no. 13-14, pp. 6280–6292, 2016.

[14] Y. Li and D. Coit, "Priority rules-based algorithmic design on two-sided assembly line balancing," *Production Engineering*, vol. 12, no. 1, pp. 95–108, 2018.

[15] 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, p. 100163, 2020.

[16] U. Özcan and B. Toklu, "Multiple-criteria decision-making in two-sided assembly line balancing: a goal programming and a fuzzy goal programming models," *Computers & Operations Research*, vol. 36, no. 6, pp. 1955–1965, 2009.

[17] Z. Li, I. Kucukkoc, and Z. Zhang, "Branch, bound and remember algorithm for two-sided assembly line balancing problem," *European Journal of Operational Research*, vol. 284, no. 3, pp. 896–905, 2020.

[18] Y. K. Kim, Y. Kim, and Y. J. Kim, "Two-sided assembly line balancing: a genetic algorithm approach," *Production Planning & Control*, vol. 11, no. 1, pp. 44–53, 2000.

[19] A. S. Simaria and P. M. Vilarinho, "A genetic algorithm based approach to the mixed-model assembly line balancing problem of type II," *Computers & Industrial Engineering*, vol. 47, no. 4, pp. 391–407, 2004.

[20] Y. K. Kim, W. S. Song, and J. H. Kim, "A mathematical model and a genetic algorithm for two-sided assembly line balancing," *Computers & Operations Research*, vol. 36, no. 3, pp. 853–865, 2009.

[21] A. Baykasoglu and T. Dereli, "Two-sided assembly line balancing using an ant colony-based heuristic," *International Journal of Advanced Manufacturing Technology*, vol. 36, no. 5-6, pp. 582–588, 2008.

[22] A. S. Simaria and P. M. Vilarinho, "2-ANTBAL: an ant colony optimisation algorithm for balancing two-sided assembly lines," *Computers & Industrial Engineering*, vol. 56, no. 2, pp. 489–506, 2009.

[23] U. Özcan and B. Toklu, "Balancing of mixed-model two-sided assembly lines," *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 217–227, 2009.

[24] U. Özcan, "Balancing stochastic two-sided assembly lines: a chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm," *European Journal of Operational Research*, vol. 205, no. 1, pp. 81–97, 2010.

[25] P. Tapkan, L. Ozbakir, and A. Baykasoglu, "Modeling and solving constrained two-sided assembly line balancing problem via bee algorithms," *Applied Soft Computing*, vol. 12, no. 11, pp. 3343–3355, 2012.

[26] P. Tapkan, L. Özbakır, and A. Baykasoğlu, "Bee algorithms for parallel two-sided assembly line balancing problem with walking times," *Applied Soft Computing*, vol. 39, pp. 275–291, 2016.

[27] Q. Tang, Z. Li, and L. Zhang, "An effective discrete artificial bee colony algorithm with idle time reduction techniques for two-sided assembly line balancing problem of type-II," *Computers & Industrial Engineering*, vol. 97, pp. 146–156, 2016.

[28] Z. Li, Q. Tang, and L. Zhang, "Two-sided assembly line balancing problem of type I: improvements, a simple algorithm and a comprehensive study," *Computers & Operations Research*, vol. 79, pp. 78–93, 2017.

[29] I. Kucukkoc, Z. Li, A. D. Karaoglan, and D. Z. Zhang, "Balancing of mixed-model two-sided assembly lines with underground workstations: a mathematical model and ant colony optimization algorithm," *International Journal of Production Economics*, vol. 205, pp. 228–243, 2018.

[30] Z. Li, M. N. Janardhanan, Q. Tang, and S. G. Ponnambalam, "Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times," *Swarm and Evolutionary Computation*, vol. 50, p. 100567, 2019.

[31] M. Gansterer and R. F. Hartl, "One- and two-sided assembly line balancing problems with real-world constraints," *International Journal of Production Research*, vol. 56, no. 8, pp. 3025–3042, 2018.

[32] M. N. Janardhanan, Z. Li, and P. Nielsen, "Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem," *Soft Computing*, vol. 2019, no. 23, pp. 11263–11276, 2019.

[33] Y. Zhang, X. Hu, and C. Wu, "Improved imperialist competitive algorithms for rebalancing multi-objective two-sided

assembly lines with space and resource constraints," *International Journal of Production Research*, vol. 58, no. 12, pp. 3589–3617, 2020.

[34] Y. Li, I. Kucukkoc, and X. Tang, "Two-sided assembly line balancing that considers uncertain task time attributes and incompatible task sets," *International Journal of Production Research*, vol. 1, pp. 1–21, 2020.

[35] D. Kizilay and Z. A. Çil, "Constraint programming approach for multi-objective two-sided assembly line balancing problem with multi-operator stations," *Engineering Optimization*, vol. 53, no. 8, pp. 1315–1330, 2021.

[36] L. H. Appelgren, "A column generation algorithm for a ship scheduling problem," *Transportation Science*, vol. 3, no. 1, pp. 53–68, 1969.

[37] L. H. Appelgren, "Integer programming methods for a vessel scheduling problem," *Transportation Science*, vol. 5, no. 1, pp. 64–78, 1971.

[38] M. E. Lübbecke and J. Desrosiers, "Selected topics in column generation," *Operations Research*, vol. 53, no. 6, pp. 1007–1023, 2005.

[39] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting-stock problem," *Operations Research*, vol. 9, no. 6, pp. 849–859, 1961.

[40] P. C. Gilmore and R. E. Gomory, "A linear programming approach to the cutting stock problem-part II," *Operations Research*, vol. 11, no. 6, pp. 863–888, 1963.

[41] P. H. Vance, C. Barnhart, E. L. Johnson, and G. L. Nemhauser, "Solving binary cutting stock problems by column generation and branch-and-bound," *Computational Optimization and Applications*, vol. 3, no. 2, pp. 111–130, 1994.

[42] Z. Chen, C. A. J. Hurkens, and J. L. de Jong, "A branch-and-price algorithm for solving the cutting strips problem," *Applications of Mathematics*, vol. 12, no. 2, pp. 215–224, 1997.

[43] F. X. O. Vanderbeck, "Computational study of a column generation algorithm for bin packing and cutting stock problems," *Mathematical Programming*, vol. 86, no. 3, pp. 565–594, 1999.

[44] Z. Degraeve and L. Schrage, "Optimal integer solutions to industrial cutting stock problems," *INFORMS Journal on Computing*, vol. 11, no. 4, pp. 406–419, 1999.

[45] M. Peeters and Z. Degraeve, "An linear programming based lower bound for the simple assembly line balancing problem," *European Journal of Operational Research*, vol. 168, no. 3, pp. 716–731, 2006.

[46] Q. Yin and X. Luo, "A three-stage optimization method for assembly line balancing problem," *IEEE Access*, vol. 8, pp. 143607–143621, 2020.

[47] U. Özcan and B. Toklu, "A tabu search algorithm for two-sided assembly line balancing," *International Journal of Advanced Manufacturing Technology*, vol. 43, pp. 822–829, 2009.

[48] D. Khorasanian, S. R. Hejazi, and G. Moslehi, "Two-sided assembly line balancing considering the relationships between tasks," *Computers & Industrial Engineering*, vol. 66, no. 4, pp. 1096–1105, 2013.