Annals of Operations Research (2024) 335:491–516
https://doi.org/10.1007/s10479-023-05809-1

ORIGINAL-COMPARATIVE COMPUTATIONAL STUDY

Chance-constrained stochastic assembly line balancing with branch, bound and remember algorithm

Zixiang Li[1,2] · Celso Gustavo Stall Sikora[3] · Ibrahim Kucukkoc[4]

Received: 27 November 2022 / Accepted: 19 December 2023 / Published online: 1 February 2024

Abstract

Assembly lines are widely used mass production techniques applied in various industries from electronics to automotive and aerospace. A branch, bound, and remember (BBR) algorithm is presented in this research to tackle the chance-constrained stochastic assembly line balancing problem (ALBP). In this problem variation, the processing times are stochastic, while the cycle time must be respected for a given probability. The proposed BBR method stores all the searched partial solutions in memory and utilizes the cyclic best-first search strategy to quickly achieve high-quality complete solutions. Meanwhile, this study also develops several new lower bounds and dominance rules by taking the stochastic task times into account. To evaluate the performance of the developed method, a large set of 1614 instances is generated and solved. The performance of the BBR algorithm is compared with two mixed-integer programming models and twenty re-implemented heuristics and metaheuristics, including the well-known genetic algorithm, ant colony optimization algorithm and simulated annealing algorithm. The comparative study demonstrates that the mathematical models cannot achieve high-quality solutions when solving large-size instances, for which the BBR algorithm shows clear superiority over the mathematical models. The developed BBR outperforms all the compared heuristic and metaheuristic methods and is the new state-of-the-art methodology for the stochastic ALBP.

Keywords Assembly line balancing · Stochastic assembly line · Branch and bound · Heuristic algorithms · Meta-heuristics · Chance-constraint

✉ Ibrahim Kucukkoc
ikucukkoc@balikesir.edu.tr

Zixiang Li
zixiangliwust@gmail.com

Celso Gustavo Stall Sikora
celso.sikora@uni-hamburg.de

[1] Key Laboratory of Metallurgical Equipment and Control Technology of Ministry of Education, Wuhan University of Science and Technology, Wuhan, Hubei, China

[2] Hubei Key Laboratory of Mechanical Transmission and Manufacturing Engineering, Wuhan University of Science and Technology, Wuhan, Hubei, China

[3] Institute for Operations Research, University of Hamburg, Hamburg, Germany

[4] Industrial Engineering Department, Balikesir University, Cagis Campus, Balikesir 10145, Turkey

1 Introduction

Assembly lines are the production system of choice for several industries that deal with the mass production of complex homogenous product families. A set of workstations (or stations, shortly) is connected via a transportation system such as a conveyor or a moving belt. The operations required to assemble a product are divided among the workstations, which are specialized for a small subset of the tasks required in the assembly. This flow-oriented process can be very efficient if the workload is well distributed among the workstations since the production cycle time is limited by the system bottleneck.

The optimization problem of assigning tasks to workstations is a classic problem known as the assembly line balancing problem (ALBP) and several variants and extensions of the problem exist (Battaïa & Dolgui, 2013). Recent surveys by Boysen et al. (2022) and Battaïa and Dolgui (2022) provide a comprehensive review and classification of the assembly line balancing literature and identify potential future research directions. In its most basic version, the simple ALBP can be described as distributing a set of tasks to a set of stations to minimize the number of stations under capacity and precedence relationship constraints. The capacity constraint ensures that the workload of each workstation is less than or equal to the cycle time (CT) (Pitakaso et al., 2021). The precedence relationship constraint requires that a task can only be performed after all its predecessors have been completed (Kucukkoc et al., 2018; Sivasankaran & Shahabudeen, 2014).

In the traditional ALBP, the task times are deterministic and fixed in advance (Li, Kucukkoc, & Tang, 2021). Nevertheless, in many real-world applications, the task times are stochastic due to fatigue, breakdowns, erroneous entries, and workforce with insufficient qualifications (Ağpak & Gökçen, 2007; Sarin et al., 1999). Furthermore, real assembly lines only seldom assemble a single product and must be adaptable to support the production of a family of similar products. As the processing time of tasks belonging to different products may vary, assuming constant and deterministic times is not realistic. More advanced approaches either consider the sequence of products as part of the decision process (Lopes et al., 2020) or utilize uncertain processing times. This paper focuses on the latter formulation, which was first introduced by (Vrat & Virani, 1976), who reduce the multiple-product problem to a single-product problem with stochastic processing times.

Uncertainties in assembly lines can be found in the cycle time (Lopes, Michels, Sikora, Brauner, & Magatão, 2021), and the product demand mix (Yuchen Li et al., 2023; Sikora, 2021b), although the great majority of the literature models uncertainty on the processing time of tasks (Boysen et al., 2022). The uncertain task times might be defined as probability distributions, where the normal distribution is usually assumed in most studies (Ağpak & Gökçen, 2007; Diefenbach & Stolletz, 2022; Urban & Chiang, 2006). The complication of considering an uncertain processing time is noticed in the cycle time condition. Several approaches were proposed to achieve a feasible or acceptable assignment when the workload of stations is not deterministic. Such formulations may either be defined by general conditions (such as a high completion probability) or application-oriented considerations or remedial actions (such as the stoppage of the assembly line if a workstation requires longer than the cycle time).

A recent survey on ALBP under uncertainty (Sikora, 2021a) classifies the literature on stochastic single-product assembly lines in formulations that minimize product or sums of non-completion probabilities and those that consider the cost of remedial actions. Examples of remedial actions are the use of utility workers (Sikora, 2021b), stopping the line (Silverman & Carter, 1986), or considering incompletion costs or rework at the end of the line (Kottas &

Lau, 1976). The methods that do not explicitly model a remedial action often use a chance constraint approach for the cycle time. In the most common variation of the problem (Battaïa & Dolgui, 2013), tasks assigned to a workstation are required to be completed within the given CT with a probability equal to or larger than the predetermined limit. One advantage of such a chance constraint-based stochastic programming approach is that a method developed for the deterministic ALBP can often be extended to deal with the stochastic version. Other approaches that have been used in literature to deal with uncertain task times are *robust optimization*, *stability analysis* and *uncertain programming* (Li et al., 2020a). Brief information on each of these methods will be provided below.

The stochastic programming approach is more favorable when past data is available on task processing times. Conversely, robust optimization may be preferred when implementing a new product or utilizing a new set of tasks with no historical data. Robust optimization may be employed to calculate the most adverse scenario when the execution times of tasks are known within specific intervals (Hazır & Dolgui, 2013). Among others addressing the robust optimization approaches for ALBP, Pereira and Miranda (2018) developed a formulation and a branch and bound (referred to as BB hereafter) algorithm for the simple ALBP to minimize the number of workstations. Moreira et al. (2015) considered the assignment of heterogeneous workers as well as ALBP with worker-dependent and uncertain task execution times, respecting a robustness measure. Hazir and Dolgui (2015) defined the robust U-shaped ALBP and proposed an iterative approximate solution algorithm.

Stability analysis determines and evaluates the effect of uncertain task times on the optimality of line balance (Gurevsky et al., 2012). Gurevsky et al. (2012) addressed the type-E ALBP (aiming to maximize the line efficiency) and proposed two heuristics seeking a compromise between the efficiency and the desired stability measure. Gurevsky et al. (2013) addressed the generalized formulation for an ALBP with several workplaces in a workstation and proposed a stability measure for feasible and optimal solutions when task processing times may vary. Lai et al. (2019) also addressed the type-E ALBP and performed the stability analysis of an optimal line balance determining sufficient and necessary conditions that the line balance to stay stable. More recently, Gurevsky et al. (2022) proposed a mixed-integer linear program to maximize the stability factor in the general case and investigated the relationship between the stability factor and stability radius.

Uncertain programming has emerged due to the mathematical intractability of stability analysis. It relies on the belief degrees of experts, based on the uncertainty theory invented by Liu (2007). Following the first uncertain programming model by Liu (2009), it has been applied to various optimization problems, including machine scheduling (Li & Liu, 2017), capacitated facility location-allocation (Wen et al., 2014), vehicle routing (Ning & Su, 2017) and production control (Liu & Yao, 2015). Its applications on the ALBPs are still limited. Li et al., (2019a) applied the uncertainty theory to model task time uncertainties and introduced the belief reliability measure to the assembly line production to minimize cycle time. Li et al., (2020a) considered two types of uncertainties caused by task times through a new reliability metric and proposed a mathematical model as well as neighborhood search methods to maximize the reliability and efficiency of the line.

As mentioned above, the chance constraint-based stochastic programming approach is the most studied method in the literature. Several exact, heuristic and metaheuristic methods have been developed. The exact methods include the dynamic programming approach (Carraway, 1989), mixed-integer programming models (Ağpak & Gökçen, 2007; Fathi et al., 2019; Urban & Chiang, 2006), and BB (Diefenbach & Stolletz, 2022). Mixed-integer programming models can solve small-size problems optimally, whereas they might not achieve high-quality or feasible solutions within acceptable computation time when solving large-size instances.

Hence, heuristic and metaheuristic methods have been developed to tackle this problem in a short time. Several heuristics and metaheuristics were developed, including the multiple single-pass heuristic algorithm (Gamberini et al., 2009), beam search (Erel et al., 2005), simulated annealing algorithm (Suresh & Sahu, 1994), the single-run optimization algorithm (JrJung, 1997), genetic algorithm (Baykasoğlu & Özbakır, 2007), imperialist competitive algorithm (Bagher et al., 2011), ant colony optimization algorithm (Celik et al., 2014), hybrid evolutionary algorithm (Zhang et al., 2017), modified evolutionary algorithm (Zhang et al., 2018), and particle swarm optimization (Aydoğan et al., 2019). Meanwhile, the modeling of chance constraints for variations, and extensions of the stochastic ALBP have also attracted the attention of many research papers. Specifically, the literature on other variants of the stochastic ALBP includes the type-II stochastic ALBP (Liu et al., 2005; Pınarbaşı & Alakaş, 2020), cost-oriented stochastic ALBP (Foroughi & Gökçen, 2019), multi-objective stochastic ALBP (Cakir et al., 2011), stochastic U-shaped ALBP (Aydoğan et al., 2019; Bagher et al., 2011; Baykasoğlu & Özbakır, 2007; Celik et al., 2014; Chiang & Urban, 2006; Serin et al., 2019; Urban & Chiang, 2006; Zhang et al., 2018), stochastic two-sided ALBP (Chiang et al., 2015; Delice et al., 2016; Tang, Li, Zhang, & Zhang, 2017; Özcan, 2010), stochastic parallel ALBP (Özbakır & Seçme, 2020; Özcan, 2018), mixed-model stochastic ALBP (McMullen & Frazier, 1997; Zhao et al., 2007) and others. Efficient methods for the stochastic ALBP with one product can be extended to deal with more realistic variants.

Although the literature on stochastic ALBP is extensive, not many approaches can efficiently solve large instances exactly. After the publication of the dynamic programming approach (Carraway, 1989), there is a large hiatus in the literature. Recently, a new BB algorithm (Diefenbach & Stolletz, 2022) was proposed for the exact solution of the problem. The BB algorithm is based on a sampling method that draws random realizations for the processing time of the tasks. In the numerical experiments, the authors used 10,000 realizations to model the stochastic processing time as a set of possible scenarios. As it is easy to check whether the cycle time restriction is obeyed for a specific scenario, the chance constraint reduces to checking whether the solution is feasible for a given percentage of scenarios. The approach is flexible enough to deal with any probability distribution (with or without correlation) but showed satisfying results for only small and medium-size instances. The results show that even some instances with 20 tasks cannot be solved within 10,000 s and only one instance with 50 tasks has been solved. Efficient and fast solution procedures for larger instances are, therefore, still required. In this manuscript, we propose a solution method for a specific yet highly studied ALBP: the stochastic ALBP with normally distributed task processing times.

The branch, bound, and remember (BBR) algorithm is an exact method, which combines the dynamic programming method with the BB algorithm (Li, Kucukkoc, & Tang, 2020; Li et al., 2020b; Morrison et al., 2014; Sewell & Jacobson, 2012). BB algorithm utilizes bounds to eliminate partial solutions which could not possibly achieve a better solution. Dynamic programming method utilizes memory to eliminate redundant partial solutions. The main difference between BBR and BB algorithms is that BBR keeps every sub-problem created in the BBR framework in memory. Before branching on a sub-problem, BBR checks all the partial solutions in memory and a sub-problem is not explored when it is dominated by a previously encountered subproblem (i.e. a stored partial solution). This powerful method is capable of achieving all the optimal solutions of Scholl's 269 instances within an average computation time of one second, and it might be considered to be the state-of-the-art methodology for the simple ALBP (Battaïa & Dolgui, 2013; Li, Kucukkoc, & Tang, 2020). Apart from the simple ALBP, BBR algorithm also produces a competing performance in robust ALBP (Pereira & Álvarez-Miranda, 2018), U-shaped ALBP (Li et al., 2018), two-sided ALBP (Li, Kucukkoc,

& Zhang, 2020), integrated worker assignment and line balancing problem (Pereira, 2018; Vilà & Pereira, 2014) and robotic ALBP (Borba et al., 2018).

From the literature review, BBR has not been utilized for the stochastic ALBP, although it has produced a competing performance in several variations of ALBP. Meanwhile, to the best of the authors' knowledge, there is no exact method which is capable of solving the large-size stochastic ALBP effectively in an acceptable computation time. Therefore, a BBR algorithm, which is an exact method, is developed to tackle the stochastic ALBP, where the task times satisfy the normal distribution, to minimize the number of workstations. In short, this study presents two contributions as follows. 1) This study develops new dominance rules and lower bounds for the stochastic ALBP, which differentiate the proposed BBR algorithm from those in the published studies. Regarding the dominance rules, the maximal load rule and the extended Jackson rule (originally developed for the deterministic ALBP) are modified by taking the probability constraint into account as they cannot be directly applied to the stochastic ALBP. Meanwhile, this study also proves the correctness of the new extended Jackson rule. As for the lower bounds, LB1, LB2 and LB3 in the deterministic situation are transferred into LB_1^S, LB_2^S and LB_3^S by taking the stochastic operation time into account. Notice that, the developed LB_1^S, LB_2^S and LB_3^S are different from that in Diefenbach and Stolletz (2022). The LB_1^S, LB_2^S and LB_3^S are very fast; the corresponding lower bounds in Diefenbach and Stolletz (2022) are sampling-based lower bounds and they consume much more computation time. Hence, the proposed BBR is not a simple extension of the deterministic BBR and all the segments of the BBR algorithm are modified to solve this problem. 2) This study re-implements two mixed-integer programming models and twenty heuristics and metaheuristics, including the well-known tabu search algorithm, simulated annealing algorithm, genetic algorithm, particle swarm optimization algorithm and ant colony optimization algorithm, and conducts a comprehensive comparative study to evaluate the performance of the proposed method. The computational results demonstrate that the mathematical models cannot achieve high-quality solutions when solving the large-size instances and the BBR algorithm shows clear superiority over the mathematical models for the large-size instances. The BBR algorithm outperforms all the implemented heuristic and metaheuristic methods and is the state-of-the-art methodology for the stochastic ALBP.

The remainder of this paper is organized as follows. Section 2 presents the problem description and the mathematical formulation. Section 3 illustrates the main procedure of the BBR algorithm along with the main segments. The computational study is provided in Sect. 4 to evaluate the developed BBR algorithm. Conclusions are given in Sect. 5 together with some future research directions.

2 Problem formulation

The stochastic ALBP and its mathematical formulation are provided in this section.

2.1 Problem description

This study addresses the stochastic ALBP to minimize the number of workstations. The chance-constrained formulation is given first to describe the problem. Task times are normally distributed with known means and variances ($t_i \sim N(\mu_i, \sigma_i^2)$) as in most of the studies in the literature, *e.g.* see Urban and Chiang (2006) and Ağpak and Gökçen (2007). For the stochastic ALBP, a set of tasks is partitioned into a set of connected stations under the

Table 1 The task times and precedence relationships of the illustrated example

Tasks	Successors	μ_i	σ_i^2
1	2,4	1	0.0598
2	3,5	5	0.0817
3	–	4	0.1566
4	7	3	0.3090
5	6	5	0.6931
6	–	6	0.1526
7	–	5	0.3806

probability constraint (cycle time constraint) and the precedence constraint. The precedence constraint requires that the predecessors of one task must be allocated to the former station or be operated before the successors when they are allocated to the same station. The probability constraint demands that the tasks on stations can be completed within the given CT with a probability equal to or larger than the predetermined limit α (greater than 0.5).

Assuming the station-load Y ($Y = \sum t_i$) has a completion probability larger than or equal to α, the probability constraint can be expressed with $P\left(\sum t_i \le CT\right) \ge \alpha$. As $t_i \sim N\left(\mu_i, \sigma_i^2\right)$, the operation time of station-load Y can be defined as $Y \sim N\left(\sum \mu_i, \sum \sigma_i^2\right)$ and hence the probability constraint can be rewritten using the normal distribution as presented in Eq. (1). For the normal distribution, the critical value for Z is available numerically for a value of α given. For instance, $Z_{0.95} = 1.6449$, $Z_{0.975} = 1.9600$ and $Z_{0.99} = 2.3263$. Hence, the probability constraint for a station-load is then described utilizing Eq. (2).

$$P\left(Z \le \frac{CT - \sum \mu_i}{\sqrt{\sum \sigma_i^2}}\right) \ge \alpha \tag{1}$$

$$\sum \mu_i + z_\alpha \cdot \sqrt{\sum \sigma_i^2} \le CT \tag{2}$$

To exhibit the features of the stochastic ALBP, one example with 7 tasks is given here, where the precedence relation and task times are illustrated in Table 1. Figure 1 depicts the optimal solutions of this illustrated example on the traditional (deterministic) assembly line and stochastic assembly line. Specifically, Fig. 1a illustrates the optimal solution when the stochastic operation time is not considered ($t_i = \mu_i$) within a cycle time of 7 time-units.

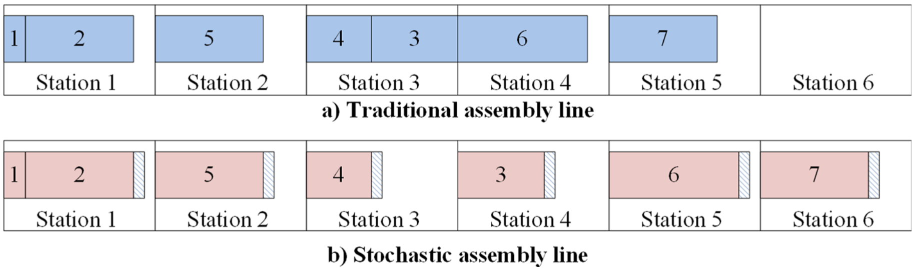

Fig. 1 Optimal assignment of tasks on the traditional and stochastic assembly lines

Figure 1b illustrates the optimal solution when the stochastic operation time is considered ($t_i \sim N\left(\mu_i, \sigma_i^2\right)$) with α equals to 0.9.

As seen in Fig. 1, five stations are utilized when $t_i = \mu_i$ on the traditional assembly line and six stations are utilized when $t_i \sim N\left(\mu_i, \sigma_i^2\right)$ on the stochastic assembly line. The reason lies behind this is that task 4 and task 3 are allocated to station 3 on the traditional assembly line, whereas task 3 cannot be allocated to station 3 on the stochastic assembly line as stochastic operation time is involved. Namely, the cycle time constraint is satisfied on the traditional assembly line when $\sum \mu_i \leq CT$; the probability constraint is satisfied on the stochastic assembly line when the tasks on each station could be completed within the cycle time with a probability equal to or larger than 0.9 ($\sum \mu_i + z_\alpha \cdot \sqrt{\sum \sigma_i^2} \leq CT$).

2.2 Mathematical model

The notations used in the mathematical model are first explained as follows.

i,h	Index of tasks, $i \in \{1, \cdots, nt\}$, where nt is the number of tasks
I	Set of tasks, $I = \{1, \cdots, nt\}$
j,k	Index of stations, $j \in \{1, \cdots, m_{max}\}$, where m_{max} is the maximum number of stations that can be utilized
J	Set of stations, $J = \{1, \cdots, m_{max}\}$
t_i	Operation time of task i, where $t_i \sim N\left(\mu_i, \sigma_i^2\right)$
P	Set of precedence relations of tasks, where $(i, h) \in P$ when task i is the immediate predecessor of task h
CT	Cycle time
x_{ij}	1, if task i is assigned to station j; 0, otherwise
s_j	1, if there is at least one task assigned to station j; 0, otherwise

The model by Ağpak and Gökçen (2007) is formulated utilizing Eqs. (3)–(7) here using the notation given above. Equation (3) minimizes the number of workstations. Equation (4) ensures that each task is allocated to exactly one workstation. Equation (5) is the precedence constraint, which indicates that the predecessors of one task should be allocated to the former or the same station in which the task has been assigned. Equation (6) is the probability constraint, indicating that the sum of the processing times of tasks on each workstation does not exceed the cycle time with a probability of α. Equation (7) defines the domain of each variable.

$$\text{Min} \sum_{j \in J} s_j \tag{3}$$

$$\sum_{j \in J} x_{ij} = 1 \quad \text{for} \quad i \in I \tag{4}$$

$$\sum_{j \in J} j \cdot x_{ij} \leq \sum_{k \in J} k \cdot x_{hk} \quad \text{for} \quad (i, h) \in P \tag{5}$$

$$\sum_{i\in I}\mu_i\cdot x_{ij}+Z_\alpha\cdot\sqrt{\sum_{i\in I}\left(\sigma_i^2\cdot x_{ij}\right)}\leq CT\cdot s_j \quad \text{for} \quad j\in J \tag{6}$$

$$x_{ij},s_j\in\{0,1\} \quad \text{for all} \quad i,j \tag{7}$$

As the above model is a non-linear integer programming model, this study proposes two integer programming models. The first model is the direct linear approach (LA) by Ağpak and Gökçen (2007) to provide an approximate solution. It utilizes Eq. (8) to replace Eq. (6). Notice that the achieved solution by the LA might not be the true optimal solution, but the achieved solution is a valid upper bound.

$$\sum_{i\in I}(\mu_i+Z_\alpha\cdot\sigma_i)\cdot x_{ij}\leq CT\cdot s_j \quad \text{for} \quad j\in J \tag{8}$$

The second model is the model with pure linear transformation (MLTT) by Ağpak and Gökçen (2007). The MLTT consists of Eqs. (3)–(5) and the following, i.e., Eqs. (9)–(13). The MLTT has a limited capacity to solve large-size instances optimally based on our preliminary experiments.

$$CT^2-2\cdot CT\cdot\sum_{i\in I}\mu_i\cdot x_{ij}+\sum_{i\in I}\mu_i^2\cdot x_{ij}+2\cdot\sum_{i\in I}\sum_{h|h\in I\bigwedge h>i}\mu_i\cdot\mu_h\cdot u_{ihj}$$
$$-Z_\alpha^2\cdot\left(\sum_{i\in I}\sigma_i^2\cdot x_{ij}\right)\geq 0 \quad \text{for} \quad j\in J \tag{9}$$

$$\sum_{i\in I}\mu_i\cdot x_{ij}\leq CT\cdot s_j \quad \text{for} \quad j\in J \tag{10}$$

$$x_{ij}+x_{hj}-u_{ihj}\leq 1 \quad \text{for} \quad j\in J, i\neq h \tag{11}$$

$$x_{ij}+x_{hj}-2\cdot u_{ihj}\geq 0 \quad \text{for} \quad j\in J, i\neq h \tag{12}$$

$$x_{ij},s_j,u_{ihj}\in\{0,1\} \quad \text{for all} \quad i,h,j \tag{13}$$

3 Proposed method (BBR)

BBR is an exact algorithm which integrates the dynamic programming method with the BB algorithm (Li, Kucukkoc, & Tang, 2020; Li et al. 2020b; Morrison et al., 2014; Sewell & Jacobson, 2012). This exact method shows a competing performance in solving the simple ALBP (Battaïa & Dolgui, 2013; Li, Kucukkoc, & Tang, 2020) and many others (Li et al., 2018; Li, Kucukkoc, & Zhang, 2020). Nevertheless, there is no application of this powerful exact method in solving the stochastic ALBP. Meanwhile, there is no exact method which can tackle the large-size stochastic ALBP effectively within an acceptable amount of computation time. Therefore, this section develops the BBR algorithm to deal with the large-size stochastic ALBP optimally for the first time. The procedure and the main components of the proposed BBR method are explained in this section.

3.1 The main procedure applied by the BBR algorithm

Algorithm 1 illustrates the procedure of the developed BBR. The proposed BBR algorithm consists of three phases: Phase I, Phase II and Phase III. Firstly, Phase I achieves a high-quality upper bound (UB) utilizing the modified Hoffman heuristic (called MHH hereafter). Here, the lower bound at the root LB$_{root}$ is calculated with LB$_{root}$ = max$\{LB_1^S, LB_2^S, LB_3^S, \text{BPLB}\}$, where LB_1^S, LB_2^S and LB_3^S are three lower bounds and BPLB is the bin packing lower bound (see Sect. 3.3 for further explanation). If the achieved solution is not optimal or is not verified to be optimal, Phase II conducts the modified cyclic best-first search (MCBFS) with the developed new lower bounds and dominance rules (see Sect. 3.3 for further explanation). Phase II aims at achieving the solution with the smaller station number and proving the optimality of the achieved solution. Finally, Phase III conducts the breadth-first search (BrFS) when Phase II is unable to prove the optimality of the solution achieved. Phase III proves the optimality by enumerating all the possible station-loads and it is relatively slow for the large-size instances. However, on the basis of the strict upper bound by Phase II, Phase III is utilized mainly for proving the optimality of the achieved solutions only for some "hard" instances.

Algorithm 1 The main procedure of the BBR method

```
Phase I     1  Obtain UB with the modified Hoffman heuristic;
            2  Calculate the lower bound at the root  ($LB_{root} = \max\{LB_1^S, LB_2^S, LB_3^S, BPLB\}$);
Phase II    3  If  UB > $LB_{root}$  and termination criterion is not satisfied
                 Conduct the modified cyclic best-first search and update UB when the smaller
                 station number is achieved;
               Endif
Phase III   4  If  UB > $LB_{root}$  and termination criterion is not satisfied
                 Conduct the breadth-first search and update UB when the smaller station number
                 is achieved;
               Endif
```

The differences between the proposed BBR method and the reliability-based BB algorithm in Diefenbach and Stolletz (2022) can be clarified as follows:

(1) The proposed BBR method utilizes the MCBFS whereas the BB algorithm utilizes the depth-first search (DFS) strategy.
(2) The proposed BBR method utilizes the LB_1^S, LB_2^S and LB_3^S, which are very fast. The BB algorithm utilizes the sampling-based lower bounds which consume much more computation time.
(3) The proposed BBR method stores all the searched partial solutions in memory, and it utilizes the memory-based dominance rule, whereas the BB algorithm does not store all the searched partial solutions in memory.
(4) The proposed BBR method also utilizes the extended Jackson rule and the no-successor rule, which are not utilized in the BB algorithm.

This study also tests various BBR methods with different search strategies, lower bounds, and dominance rules in Sect. 4.2 to prove the advantages of the differences used in the developed BBR method.

3.2 Branching method and upper bound

There are two branching methods in the published BB algorithms. They are station-oriented branching and task-oriented branching (Li, Kucukkoc, & Tang, 2020). Task-oriented branching obtains partial solutions by assigning one task to the current station when all the constraints are met or to a newly opened station when the probability constraint is conflicted. Station-oriented branching creates partial solutions by assigning a station-load consisting of several tasks to one newly opened station. Suppose that a partial solution utilizes m stations. This partial solution can be expressed with $\wp = (A, U, S_1, S_2, \cdots, S_m)$, where A and U are the set of assigned tasks and unassigned tasks, respectively. S_m holds the set of tasks assigned to workstation m. As station-oriented branching assigns a station-load together, the new partial solution at the deeper depth can be described with $\wp\prime = (A\prime, U\prime, S_1, S_2, \cdots, S_m, S_{m+1})$. This study selects the station-oriented branching due to its superior performance demonstrated in the published study (Li, Kucukkoc, & Tang, 2020). To avoid tremendous time to achieve all the station-loads, the maximum number of generated station-loads is set to 10,000 in Phase II as in Sewell and Jacobson (2012).

On the basis of the station-oriented branching, this study utilizes the MHH (Sewell & Jacobson, 2012) to obtain a high-quality initial solution as UB. MHH generates a feasible solution from station to station and the main procedure of the proposed MHH is described as follows. Firstly, MHH generates a number of station-loads for the first station and selects the most promising station-load utilizing the selection criterion as the selected station-load. This procedure is applied to the latter station subsequently and terminates when the complete solution is achieved. Specifically, for a selected partial solution $\wp = (A, U, S_1, S_2, \cdots, S_m)$, MHH generates a set of new partial solutions $\wp' = \left(A', U', S_1, S_2, \cdots, S_m, S_{m+1}\right)$ for station $m + 1$. Subsequently, the most promising station-load with the maximum value of $\sum_{i \in S_{m+1}} (\mu_i + \alpha \cdot w_i + \beta \cdot |F_i| - \gamma)$ is selected as the station-load for station $m+1$, where S_{m+1} is the task set assigned to station $m + 1$, F_i is the set of immediate successors of task i, $|F_i|$ is the number of tasks in set F_i,F_i^* is the set of all successors of task i and w_i is the positional weight of task i calculated with $w_i = \mu_i + \sum_{h \in F_i^*} \mu_h$. The terms α, β and γ are three parameters, where the values of them are set to $\alpha \in \{0, 0.005, 0.010, 0.015, 0.020\}$, $\beta \in \{0, 0.005, 0.010, 0.015, 0.020\}$ and $\gamma \in \{0, 0.01, 0.02, 0.03\}$. To obtain a high-quality UB, this study tests all the combinations of these three parameters following Sewell and Jacobson (2012) and the minimal station number among the ones by all the combinations is selected as the output of MHH.

3.3 New lower bounds and dominance rules

Lower bounds are utilized to reduce the number of nodes explored in the enumeration. All the lower bounds of the simple ALBP with the μ_i as the operation time are the lower bounds of the stochastic ALBP. Nevertheless, these lower bounds might be weak, and this study extends LB1, LB2 and LB3 (the three well-known lower bounds), to the stochastic ALBP with normal distributed processing times. The developed LB_1^S, LB_2^S and LB_3^S in the stochastic situation are calculated in Eqs. (14)–(17). If σ_i is equal to 0.0 for all tasks, the LB_1^S, LB_2^S and LB_3^S are equal to the LB1, LB2 and LB3 in the deterministic situation. Optimally, this study also utilizes the bin packing lower bound (BPLB) as in Sewell and Jacobson (2012) with the μ_i as the operation time. This is because BPLB is more powerful than LB1, LB2 and LB3. BPLB helps further prone the sub-problem when the variances of the task times are not large and hence it is utilized here.

$$LB_1^S = \left\lceil \frac{\sum_{i\in I} t_i + z_\alpha \cdot \sqrt{\sum_{i\in I}\sigma_i^2}}{CT} \right\rceil^+ \tag{14}$$

$$LB_2^S = \left|\left\{ i \in I | t_i > \frac{CT}{2} \text{or}(t_i = \frac{CT}{2} \quad \text{and} \quad \sigma_i > 0) \right\}\right| + \left\lceil \frac{\left|\left\{ i \in i | t_i = \frac{CT}{2} \quad \text{and} \quad \sigma_i = 0 \right\}\right|}{2} \right\rceil^+ \tag{15}$$

$$LB_3^S = \left\lceil \sum\nolimits_{i\in I} v_i \right\rceil^+ \tag{16}$$

$$v_i = \begin{cases} 1 \text{ if } t_i > 2 \cdot \frac{CT}{3} \; or \; (t_i = 2 \cdot \frac{CT}{3} \; and \, \sigma_i > 0) \\ 2/3 \text{ if } t_i = 2 \cdot \frac{CT}{3} \; and \, \sigma_i = 0 \\ 1/2 \text{ if} CT/3 < t_i < 2 \cdot CT/3 \, or \, (t_i = \frac{CT}{3} \; and \, \sigma_i > 0) \\ 1/3 \text{ if} t_i = \frac{CT}{3} \; and \, \sigma_i = 0 \\ 0 \quad otherwise \end{cases} \tag{17}$$

Dominance rules determine whether a generated new partial solution is dominated, and whether the dominated partial solution should be pruned. This study utilizes and modifies four dominance rules, which are maximal load, extended Jackson, no-successor and memory-based dominance rules.

3.3.1 Maximal load rule

One partial solution is pruned when 1) this partial solution contains a station-load S_j and an unallocated task i; 2) task i can be allocated to station j under the precedence constraint and probability constraint. Suppose that the load on station j be expressed as $Y \sim N\left(\mu_j, \sigma_j^2\right)$, then maximal load rule is applied only when $\mu_j + \mu_i + Z_\alpha \cdot \sqrt{\sigma_j^2 + \sigma_i^2} \leq CT$.

3.3.2 Extended Jackson rule

One partial solution is pruned when 1) there is a task i assigned to the last station j; 2) there is an unallocated task h which potentially dominates task i; 3) task h can replace task i without violation of the precedence constraint and probability constraint. Here, task h dominates task i only when task i and task h has no precedence relation, $\mu_i \leq \mu_h$, $\mu_i + Z_\alpha \cdot \sqrt{\sigma_i^2} \leq \mu_h + Z_\alpha \cdot \sqrt{\sigma_h^2}$ and $F_i^* \subseteq F_h^*$.

3.3.3 No-successor rule

One partial solution is pruned when 1) the assigned tasks on the last station have no successors; 2) there exists an unassigned task which can be allocated to the last station and has at least one successor.

3.3.4 Memory-based dominance rule

One partial solution is pruned when 1) the assigned task set of this partial solution is equivalent to the task set of a previously identified partial solution; 2) the current partial solution requires no smaller station number than that of a previously identified partial solution.

When the maximal load rule is applied to two sub-problems with the same station number (X $= S_1 \bigcup S_2 \bigcup \cdots \bigcup S_j \cdots$, Y $= S_1 \bigcup S_2 \bigcup \cdots \bigcup S'_j \cdots$ and $\bigcup S_j \in \bigcup S'_j$), only the sub-problem Y with more tasks is preserved. Hence, the maximal load rule never prunes an optimal solution. When the memory-based dominance rule is applied to two sub-problems with the same number of tasks assigned, only the sub-problem with an equivalent or better solution is preserved. Hence, the memory-based dominance rule never prunes an optimal solution (Morrison et al., 2014). Hence, this study only proves the correctness of the extended Jackson rule and the no-successor rule here.

Lemma 1: *For a given partial solution, a sub-problem containing this partial solution can be pruned if* (1) *there is a task i assigned to the last station j*; (2) *there is an unallocated task h which potentially dominates task i*; (3) *task h can replace task i without violation of the precedence constraint and probability constraint.*

Proof 1: Let $X = S_1 \bigcup S_2 \bigcup \cdots \bigcup S_j \bigcup S_{j+1} \bigcup \cdots$ be a feasible solution, where $i \in S_j$ and $h \in \bigcup S_{j+1} \bigcup \cdots$. One new solution is achieved by exchanging the positions of task i and task h ($Y = S_1 \bigcup S_2 \bigcup \cdots \bigcup S_j \bigcup S_{j+1} \bigcup \cdots, h \in S_j$ and $i \in \bigcup S_{j+1} \bigcup \cdots$) when 1) task i and task h have no precedence relation, $\mu_i \le \mu_h$, $\mu_i + Z_\alpha \cdot \sqrt{\sigma_i^2} \le \mu_h + Z_\alpha \cdot \sqrt{\sigma_h^2}$ and $F_i^* \subseteq F_h^*$; 2) task h can replace task i without violation of the probability constraint and precedence constraint. Suppose that task h is allocated to station k in solution X and the remaining station-load after removing task h is expressed as $Y \sim N\left(\mu_k, \sigma_k^2\right)$, it is satisfied that $\mu_k + \mu_h + Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_h^2} \le CT$. After exchanging the positions of task i and task h, solution Y is a feasible solution when $\mu_k + \mu_i + Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_i^2} \le CT$. When $\mu_i \le \mu_h$ and $\sigma_i^2 \le \sigma_h^2$, clearly it is satisfied that $\mu_k + \mu_i + Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_i^2} \le \mu_k + \mu_h + Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_h^2} \le CT$. When $\mu_i \le \mu_h$ and $\sigma_i^2 > \sigma_h^2$, $\mu_i + Z_\alpha \cdot \sqrt{\sigma_i^2} \le \mu_h + Z_\alpha \cdot \sqrt{\sigma_h^2}$ can be transferred into $0 \le Z_\alpha \cdot \sqrt{\sigma_i^2} - Z_\alpha \cdot \sqrt{\sigma_h^2} \le \mu_h - \mu_i$. Subsequently, it can be obtained that $0 \le Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_i^2} - Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_h^2} \le Z_\alpha \cdot \sqrt{\sigma_i^2} - Z_\alpha \cdot \sqrt{\sigma_h^2} \le \mu_h - \mu_i$. After transferring, it is clear that $\mu_i + Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_i^2} \le Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_h^2} + \mu_h$ and $\mu_k + \mu_i + Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_i^2} \le \mu_k + \mu_h + Z_\alpha \cdot \sqrt{\sigma_k^2 + \sigma_h^2} \le CT$. Namely, solution Y is a feasible solution after exchanging the positions of task i and task h under the given condition. The two solutions X and Y have the same station number and hence deleting solution X by pruning the corresponding partial solution with the extended Jackson rule will never prevent the discovery of the optimal solution.

Lemma 2:

For a given partial solution, a sub-problem containing this partial solution can be pruned if 1) the assigned tasks on the last station have no successors; 2) there exists an unassigned task which can be allocated to the last station and has at least one successor.

Proof 2: Let $X = S_1 \bigcup S_2 \bigcup \cdots \bigcup S_j \cdots \bigcup S_k \bigcup \cdots$ be a feasible solution, where S_j and S_k are task sets on station *j* and station *k*, respectively. There must be one solution by exchanging the positions of task sets S_j and S_k: $Y = S_1 \bigcup S_2 \bigcup \cdots \bigcup S_k \cdots \bigcup S_j \bigcup \cdots$ if 1) the tasks in S_j have no successors; 2) there exists an unassigned task in S_k with at least one successor. The two solutions X and Y have the same station number and hence deleting solution X by pruning the corresponding partial solution with the no-successor rule will never prevent the discovery of the optimal solution.

3.4 Search strategy and selection criterion

A search strategy is utilized to decide the order of the explored partial solutions and it has a big effect on the consumed time of the BBR algorithm. Among the studies, there are four main search strategies: depth-first search (DFS), best-first search (BFS), breadth-first search (BrFS) and cyclic best-first search (CBFS) (Li, Kucukkoc, & Tang, 2020). DFS starts with selecting one partial solution at depth 1 and later selects one partial solution at a deeper depth in sequence until reaching the deepest depth. After exhausting the solutions at the deepest depth, DFS returns to the second deepest depth and finally returns to the top of the search tree after exhausting all the partial solutions at deeper depths. BFS selects the most promising partial solution utilizing one selection criterion, and the effectiveness of the BFS mainly depends on the selection criterion. BrFS generates all the partial solutions at depth 1, depth 2,$\cdots$, and the deepest depth and it terminates only when the complete optimal solution is obtained. As BrFS tests all the possible nodes, BrFS is very slow when solving the large-size instances and it is utilized to prove the optimality of some instances in Phase III with the strict upper bound provided.

CBFS is a relatively new search strategy by combing the DFS and the BFS. CBFS starts with selecting one most promising station-load at depth 1, and later selects the most promising station-load at depth 2, depth 3 and the deepest depth. Once the deepest depth is reached, CBFS comes back to depth 1 and this procedure is repeatedly carried out until the termination criterion is met. To avoid generating too many partial solutions, Li et al. (2018) proposed a modified CBFS (namely, MCBFS), where a sub-problem at depth l is not selected when there are many unsearched partial solutions at depth $l + 1$. Due to the superiority of the MCBFS (Li, Kucukkoc, & Tang, 2020), this study utilizes the MCBFS in Phase II of the BBR method as presented in Algorithm 2. Here, the MCBFS utilizes the selection criterion of $b(\wp) = LB(U) + IT/m - \lambda|U|$, where $\wp = (A, U, S_1, S_2, \cdots, S_m)$, LB is the minimal value of the LB_1^S, LB_2^S, LB_3^S and BPLB of the remaining task set U, IT is the idle time on the former m stations, $|U|$ is the number of unassigned tasks and λ is an input parameter (λ is set to 0.02 as in Li, Kucukkoc, & Tang (2020)).

Algorithm 2 Modified cyclic best-first search (MCBFS)

```
Initialize level 0 with the root problem and set $l \leftarrow 1$;
**While** (there exists unexplored partial solutions)
    **While** (the number of unsearched partial solutions at depth $l + 1$ is equal to or larger than 10,000)
        Update $l \leftarrow l + 1$;
    **Endwhile**
    Select the most promising and unexplored partial solution at depth $l$ with the minimum value of $b(\wp)$;
    Store all the generated non-dominated children at depth $l + 1$ of the selected partial solution in memory;
    Set $l \leftarrow (l + 1) \% (UB - 1)$;
**Endwhile**
```

Another important issue is the utilization sequence of the lower bounds and dominance rules when conducting the MCBFS and BFS. The procedure of conducting the search strategy is provided in Algorithm 3. In this procedure, for any partial solution $Y\left(A', U', S_1, S_2, \cdots, S_m, S_{m+1}\right)$; LB_1^S, LB_2^S and LB_3^S are applied first. After that, the maximal load rule, extended Jackson rule, no-successor rule and memory-based dominance rule are applied in sequence. Here, for the memory-based dominance rule, the maximally loaded partial solutions (where the sub-problems not maximally loaded are pruned by the maximal load rule), are compared and the sub-problem with equivalent or better solutions is preserved. In short, for any partial solution $Y\left(A', U', S_1, S_2, \cdots, S_m, S_{m+1}\right)$, it is stored when it is not dominated by the lower bounds LB_1^S, LB_2^S and LB_3^S and dominance rules. Recall that, in Phase II the maximum number of generated station-loads is set to 10,000 whereas in Phase III there is no limit on the maximum number of generated station-loads. Namely, the complete enumeration is utilized in Phase III and the proposed algorithm is an exact method.

Algorithm 3 The procedure of conducting the search strategy

```
1  While there exists a non-explored partial solution and the termination criterion is not satisfied
2      Select one non-dominated partial solution $X(A, U, S_1, S_2, \cdots, S_m)$ utilizing the search strategy;
3      Do
4          Obtain one partial solution $Y(A', U', S_1, S_2, \cdots, S_m, S_{m+1})$ at deeper depth, where $Y(A', U', S_1, S_2, \cdots, S_m, S_{m+1})$ is the child of $X(A, U, S_1, S_2, \cdots, S_m)$;
5          Update UB when $Y$ is a complete solution and smaller station number is achieved;
6          Delete partial solution $Y$ when $\max\{LB_1^S, LB_2^S, LB_3^S\} \geq UB$;
           % The $LB_1^S$, $LB_2^S$ and $LB_3^S$ are applied here.
7          Delete partial solution $Y$ when it is dominated by maximal load rule, extended Jackson rule, no-successors rule or memory-based dominance rule;
           % The dominance rules are applied here in sequence (the maximal load rule, extended Jackson rule, no-successors rule and memory-based dominance rule).
8          Store the partial solution $Y$ in memory;
9          Calculate the BPLB of partial solution $Y$ and mark partial solution $Y$ as explored when $BPLB \geq UB$;
10     Until (all the nodes have been searched or the generated station-loads in deeper depth is equal to or larger than 10,000)
11 Endwhile
12 Output the achieved UB
```

4 Computational results

This section aims at testing the performance of the developed algorithm. Section 4.1 presents the utilized instances, compared algorithms and the running environments. Section 4.2 conducts the structural parameter evaluation to evaluate the proposed structural parameter values. Section 4.3 compares the performance of the BBR method with the mathematical formulations. Section 4.4 provides a comparative study between BBR method and the re-implemented heuristic and metaheuristic methods. Finally, Sect. 4.5 conducts the comparison on multiple instance characteristics to clarify differences according to multiple instance characteristics.

4.1 Experimental design

To evaluate the implemented method, this study generates a set of instances on the basis of the Scholl's 269 instances utilizing the method in Carraway (1989). To observe the performance of the proposed method on instances with different variances, this study tests two types of task variances (low task variance and high task variance) and three types of α levels ($\alpha \in \{0.90, 0.95, 0.975\}$), where the corresponding values of z_α are 1.28, 1.645 and 1.96, respectively. All the two variances and three different α levels are tested, leading to a total number of $269 \times 2 \times 3 = 1614$ instances. Specifically, the means (μ_i) of task times are set to the original task times and the variances are randomly generated within [0, $(t_i/4)^2$] for the low task variance and [0, $(t_i/2)^2$] for the high task variance. The detailed precedence relations and the original operations times of tasks are available at the website (http://www.assembly-line-balancing.de). Nevertheless, the feasible instances cannot be achieved for many instances in the preliminary experiments and this study sets that $\mu_i + z_\alpha \cdot \sqrt{{\sigma_i}^2} \le CT$ for any task i when generating the instances, where α is set to 0.975. This modification ensures that there are feasible solutions for all the generated instances. The instances are divided into two sets: small-size instances with 70 tasks at maximum ($135 \times 6 = 690$ instances in total) and large-size instances with more than 70 tasks.

To observe the performance of the proposed methodology on the different instances, this study also generates a set of instances based on the instance from Otto et al. (2013) utilizing the above method. The selected cases are Otto-20 with 20 tasks, Otto-50 with 50 tasks, Otto-100 with 100 tasks and Otto-1,000 with 1,000 tasks. The number of test instances in each set Otto-20, Otto-50, Otto-100 and Otto-1,000 is 525. For each instance, this study generates six new instances for the stochastic ALBP with two variances and three different α levels. In total, a total number of $525 \times 4 \times 2 \times 3 =$ 12,600 instances are generated based on the instance in Otto et al. (2013). This large set of instances could help to clarify differences according to multiple instance characteristics. All the generated instances are available upon request.

To evaluate the performance of the proposed BBR method, BBR is compared with the two mathematical models given in Sect. 2: direct linear approach (LA) and the model with pure linear transformation (MLTT). In addition, the proposed BBR method is also compared with twenty heuristic and metaheuristic methods. These methods include two heuristics in ALBP, including the random search (RS) and random task priority search (RTPS). Here, RS creates solutions by selecting tasks randomly (Pape, 2015) and RTPS utilizes the roulette wheel selection to select a task based on the task priorities (Pape, 2015). The metaheuristic methods contain four recent and effective evolutionary algorithms: teaching–learning-based optimization (TLBO) (Rao et al., 2011), migrating birds optimization (MBO) (Duman et al., 2012), grey wolf optimizer (GWO) (Mirjalili et al., 2014), and whale optimization algorithm (WOA) (Mirjalili & Lewis, 2016). Meanwhile, this study also includes some algorithms in solving variants of ALBP, including late acceptance hill-climbing algorithm (LAHC) (Yuan et al., 2015), simulated annealing algorithm (SA) (Suresh & Sahu, 1994), tabu search algorithm (TS) (Özcan & Toklu, 2009), genetic algorithm (GA) (Baykasoğlu & Özbakır, 2007), particle swarm optimization algorithm (PSO) (Hamta et al., 2013), discrete particle swarm optimization algorithm (DPSO) (Li et al., 2016), artificial bee colony algorithm (ABC) (Li et al., 2019b), improved artificial bee colony algorithm-1 (IABC1) (Li et al., 2019b), improved artificial bee colony algorithm-2 (IABC2) (Li et al., 2019b), bees algorithm (BA) (Li et al., 2019b), cuckoo search algorithm (CS) (Li et al., 2019b), discrete cuckoo search algorithm (DCS) (Li et al., 2019b), improved migrating birds optimization algorithm (IMBO) (Li et al., 2019b) and ant colony optimization algorithm (ACO) (Celik et al., 2014). All the algorithms (including BBRs) have been coded in C + + programming language on Microsoft Visual Studio 2015 and run utilizing the same configuration. The main procedures of all these methods are not presented due to page limits but they are available upon request.

The proposed BBR and mathematical models terminate when the optimal solution is achieved and verified, or the computation time reaches 500 s (s). The implemented algorithms terminate when the achieved station number is equal to the lower bound at the root or the computation time reaches 30 s, 180 s, 300 s and 500 s. The utilization of four computation times allows the observation of the algorithms' performance under different computation times. The tested models are solved utilizing the CPLEX solver of the IBM ILOG CPLEX Optimization Studio 12.6.1. The experiments are conducted on a tower type of server with two Intel Xeon E5-2680 v2 processors and 64 GB RAM. The real experiments have been run on a set of virtual computers and each of them has one processor with 2 GB of RAM.

4.2 Structural parameter evaluation

This section evaluates the structural parameters by solving all the instances generated based on Scholl's 269 dataset. Table 2 provides the results by the proposed BBR and the variants of the BBR method. In this table, *#OPT* is the number of optimal solutions achieved and *#ARPD* is the average gap or the average relative percentage deviation (RPD) values for all the tested instances in one run. The RPD for one instance is calculated utilizing $RPD = 100 \cdot (UB_{some} - LB)/LB$, where UB_{some} is the achieved number of stations by a method and LB is the theoretical lower bound on the number of stations. Time is the average computation time for all the tested instances in seconds. In addition, for the dual feasible solution method, this study utilizes the values of k = 1, 2, …, 100 as proposed in Fekete and Schepers (2001). Sampling LB1, sampling LB2, sampling LB3 and sampling LB6 are the new LB1, LB2, LB3 and LB6 for the stochastic ALBP based on the sampling approach with a sample size of 10,000 in Diefenbach and Stolletz (2022).

For the lower bounds, it is observed that the proposed BBR outperforms the variants of the BBR method with other lower bounds. Although the sampling LB1, sampling LB2, sampling LB3, sampling LB6 and sampling dual feasible solution methods could possibly prune a greater number of partial solutions, their utilization leads to poor performance as more computation time is needed to calculate the sampling lower bounds. For the dominance rules, it is observed that the maximal load rule, extended Jackson rule, and no-successor rule obtain minor improvements whereas the memory-based dominance rule could achieve great improvement. For the search strategy, the proposed search strategy outperforms the original CBFS and BFS strategies.

Table 2 Results by different BBR methods

Methods	*#OPT*	*#ARPD*	*Time(s)*
Proposed BBR	911	8.7414	216.13
BBR with lower bounds of the simple ALBP (such as LB1, LB2 and LB3)	911	8.7445	229.23
BBR with dual feasible solution method of the simple ALBP	911	8.7536	217.22
BBR with sampling LB1, sampling LB2 and sampling LB3	897	9.6029	346.48
BBR with sampling LB6	875	9.7023	350.24
BBR with sampling dual feasible solution method	863	9.7359	283.35
BBR without maximal load rule	911	8.8179	231.85
BBR without extended Jackson rule	911	8.7644	233.51
BBR without no-successor rule	911	8.7448	235.53
BBR without memory-based dominance rule	840	9.3346	258.11
BBR with the original CBFS strategy	911	8.8022	215.27
BBR with BFS strategy	907	9.3821	214.86

Table 3 Results by BBR and two models

Method	*#OPT*	*#Feasible*	*#Worse-than-BBR*	*#Equal-to-BBR*	*#Better-than-BBR*
BBR	683	690	–	–	–
LA	154	690	536	154	0
MLTT	343	463	347	343	0

In short, the comparative study shows that the proposed BBR outperforms all the variants of the BBR method, indicating that the proposed lower bounds, dominance rules and search strategy are effective and efficient for the stochastic ALBP.

4.3 Comparison with mathematical formulations

A comparative study between BBR and two mathematical models (LA and MLTT) is presented in this section. As LA and MLTT cannot obtain satisfying or feasible solutions for most large-size problems within the given computation time, this section mainly tests the problems generated based on Scholl's 269 instances with 70 tasks at maximum ($135 \times 6 = 690$ instances in total). Table 3 illustrates the results by the BBR and two models under 500 s, where *#OPT* is the number of optimal solutions achieved, *#Feasible* is the number of feasible solutions achieved within the given computation time, *#Worse-than-BBR* is the number of instances where the BBR outperforms the other method, *#Equal-to-BBR* is the number of instances where the other method shows the same performance with the BBR, and *#Better-than-BBR* is the number of instances where the other method outperforms the BBR.

The results presented in this Table 3 indicate that the proposed BBR outperforms LA and MLTT in terms of both *#OPT* and *#Feasible*. Specifically, BBR obtains 683 optimal solutions, whereas LA and MLTT achieve 154 and 343 optimal solutions, respectively. For the *#Feasible*, BBR and LA achieve 690 solutions, whereas MLTT only obtains 463 solutions. It is also observed that the BBR outperforms LA and MLTT for 536 and 347 instances, respectively. LA and MLTT cannot show better performance than BBR for any of the instances tested. Notice that, for the large-size instances, LA and MLTT cannot obtain satisfying or feasible solutions within the given computation time and the proposed BBR is capable of achieving high-quality feasible solutions and shows the same or superior performance than LA and MLTT in all large-size instances. This comparative study shows the superiority of BBR which can reach much more optimal solutions. Therefore, it outperforms the two mathematical models by a significant margin.

4.4 Comparison with heuristics and metaheuristics

This section provides the experimental study to compare the performances of BBR and other implemented heuristics and metaheuristics, where all the instances generated based on Scholl's 269 instances are solved here. The results by BBR and other implemented methods under different computation time limits are presented in Table 4. In this table, *#ARPD* is the average gap or the average relative percentage deviation (RPD) values for all the tested instances in 10 runs. As can be seen in the table, BBR is the best performer under all the termination criteria. Specifically, BBR is the best performer, ACO is the second-best performer and RTPS is the third-best performer. Despite the superiority of the BBR in terms

Table 4 Results by BBR and other heuristics and metaheuristics

Method	*#ARPD*			
	30 s	180 s	300 s	500 s
BBR	**9.133**	**8.866**	**8.821**	**8.769**
RS	11.373	11.051	10.968	10.892
RTPS	9.581	9.446	9.415	9.386
TLBO	9.907	9.772	9.726	9.679
MBO	10.787	10.694	10.678	10.665
GWO	10.149	9.882	9.846	9.830
WOA	10.420	10.139	10.091	10.039
LAHC	10.543	10.251	10.194	10.134
SA	10.664	10.292	10.203	10.118
TS	10.624	10.325	10.251	10.176
GA	10.927	10.634	10.580	10.548
PSO	10.073	9.874	9.846	9.813
DPSO	10.697	10.456	10.406	10.324
ABC	10.742	10.400	10.331	10.248
IABC1	10.571	10.324	10.252	10.195
IABC2	10.544	10.288	10.216	10.139
BA	10.632	10.410	10.338	10.293
CS	10.493	10.289	10.217	10.158
DCS	10.571	10.295	10.211	10.145
IMBO	10.548	10.305	10.228	10.166
ACO	9.520	9.439	9.396	9.376

*Best in bold

of the *#ARPD*, BBR is an exact method, and it can achieve the optimal solution and verify the optimality of the achieved solution.

To have a better assessment of the BBR and other methods, Table 4 presents a comparison between the BBR and other implemented heuristics and metaheuristics in one run with the termination criterion of 500 s. In this table, the meanings of *#OPT*, *#Feasible*, *#Worse-than-BBR*, *#Equal-to-BBR* and *#Better-than-BBR* are the same as that presented above in Table 3.

Results presented in Table 5 indicate that BBR performs the best in terms of *#OPT*, and so it outperforms all the implemented twenty methods. Regarding *#Worse than BBR*, it is observed that BBR outperforms the other methods in many instances. Specifically, BBR outperforms RS, RTPS, TLBO, MBO, GWO, WOA, LAHC, SA, TS, GA, PSO, DPSO, ABC, IABC1, IABC2, BA, CS, DCS, IMBO, and ACO for 702, 301, 408, 676, 460, 514, 529, 525, 544, 636, 464, 577, 560, 542, 530, 585, 551, 536, 535 and 297 instances, respectively. So, BBR shows superior performance over all the methods tested here. For *#Better-than-BBR* results, BBR is only outperformed by RTPS and ACO for 1 and 2 instances, respectively, and the remaining methods cannot outperform BBR for any of the instances tested. In short, this comparative study demonstrates that the proposed BBR outperforms all the implemented heuristic and metaheuristic methods. Therefore, BBR is the new state-of-the-art methodology for the stochastic ALBP.

Table 5 Comparison between BBR and other implemented heuristics and metaheuristics

Method	*#OPT*	*#Feasible*	*#Worse-than- BBR*	*#Equal-to- BBR*	*#Better-than- BBR*
BBR	**911**	1614	–	–	–
RS	656	1614	702	912	0
RTPS	823	1614	301	1312	1
TLBO	808	1614	408	1206	0
MBO	670	1614	676	938	0
GWO	777	1614	460	1154	0
WOA	751	1614	514	1100	0
LAHC	790	1614	529	1085	0
SA	790	1614	525	1089	0
TS	780	1614	544	1070	0
GA	705	1614	636	978	0
PSO	775	1614	464	1150	0
DPSO	743	1614	577	1037	0
ABC	754	1614	560	1054	0
IABC1	779	1614	542	1072	0
IABC2	785	1614	530	1084	0
BA	725	1614	585	1029	0
CS	747	1614	551	1063	0
DCS	784	1614	536	1078	0
IMBO	784	1614	535	1079	0
ACO	824	1614	297	1315	2

*Best in bold

4.5 Comparison based on multiple instance characteristics

This section utilizes the instances derived from Otto et al. (2013) to clarify differences according to multiple instance characteristics. The proposed BBR solves all the 12,600 instances derived from Otto et al. (2013) as described in Sect. 4.1 and terminates when the optimal solution is achieved and verified, or the computation time reaches 500 s.

Table 6 provides the results for different instance sizes, where *Avg-RPD*, *Min-RPD* and *Max-RPD* are the average value, minimum value, and maximum value of RPD values and *Avg-t*, *Min-t* and *Max-t* are the average value, minimum value and maximum value of computation

Table 6 Results for different instance sizes

Instance size	*nt*	*#OPT*	*Avg-RPD*	*Min-RPD*	*Max-RPD*	*Avg-t*	*Min-t*	*Max-t*
Small	20	3150	0.00	0.00	0.00	0.39	0.10	0.69
Medium	50	2759	2.34	0.00	55.56	93.94	0.28	502.01
Large	100	979	18.92	0.00	79.25	377.60	0.66	502.91
Very large	1000	0	35.26	8.57	81.40	501.07	500.51	505.80

Table 7 Results for different task variances and α levels

Task variance	α *levels*	*#OPT*	*Avg-RPD*	*Min-RPD*	*Max-RPD*	*Avg-t*	*Min-t*	*Max-t*
Low	0.9	1196	8.61	0.00	48.06	231.74	0.10	501.62
Low	0.95	1185	10.44	0.00	56.19	235.91	0.12	505.80
Low	0.975	1166	12.08	0.00	62.30	239.90	0.11	501.31
High	0.9	1132	14.86	0.00	70.14	247.00	0.11	502.91
High	0.95	1111	18.07	0.00	77.12	252.25	0.12	501.56
High	0.975	1098	20.73	0.00	81.40	252.70	0.12	501.76

times in seconds. It is observed that the proposed BBR is capable of solving all the small-size instances with 20 tasks optimally. Whereas the BBR fails to find the optimal solution or verify the optimality of the achieved solution for very large-size instances with 1,000 tasks. In short, it is more difficult for the proposed BBR to solve the instances with larger number of tasks.

Table 7 provides the results for different task variances and α levels. It is observed that the BBR method performs better for the instances with lower task variance. Meanwhile, it is also prominent that BBR achieves higher numbers of optimal solutions with smaller α levels. In short, it is more difficult for the proposed BBR to solve the instances with higher task variance and larger α levels.

Due to page limits, this section mainly investigates the differences in terms of instance size, task variance and α level. All the results are available upon request and interested researchers could also study the differences in terms of other instance characteristics based on the detailed results, such as the order strength and the distribution of task times.

5 Conclusions and future research

This research develops the branch, bound, and remember (BBR) algorithm to solve the stochastic ALBP. The proposed BBR method stores all the searched partial solutions in memory and utilizes the modified cyclic best-first search strategy to achieve high-quality complete solutions quickly. Meanwhile, this study also develops numerous new dominance rules and lower bounds by taking the stochastic task times into account. To better assess the performance of the BBR algorithm, this study generates a tremendous number of test instances based on Scholl's 269 benchmark problems as well as those derived from Otto et al. (2013). The proposed BBR method is compared with two mathematical models and twenty re-implemented heuristics and metaheuristics, including the well-known simulated annealing, tabu search, particle swarm optimization, ant colony optimization and genetic algorithms. The structural parameters of the proposed BBR have also been validated through an experimental study including several variants of the BBR.

Computational results confirm that the BBR method shows superior performance over the two models in terms of both the number of optimal solutions and the number of feasible solutions achieved. The comparative studies between BBR and other implemented heuristics and metaheuristics show that BBR outperforms all the other methods in terms of the number of achieved optimal solutions and the average relative percentage deviation. Especially, the BBR algorithm obtains the same or better results than all these twenty heuristic and metaheuristic methods for almost all the tested instances. As a result, the proposed BBR algorithm can be considered to be the new state-of-the-art methodology for the stochastic ALBP.

One of the main limitations of the chance-constrained approach is the requirement for past data to characterize the uncertain parameter. A sufficient amount of data is needed to accurately represent the distribution of uncertain task processing times. Another limitation could be the increase in the problem complexity when dealing with multiple stochastic constraints. It is also needed to define an appropriate α level, which directly has an impact on the feasibility of the solution obtained eventually.

The proposed methodology might help line managers to obtain high-quality solutions where stochastic operation times are involved. Future studies might thoroughly study the influence of different search parameters on the BBR algorithm to obtain a fine-tuned method in solving the stochastic ALBP. The proposed BBR algorithm can also be applied to other assembly and disassembly line balancing problems (e.g., stochastic U-shaped and multi-manned assembly line balancing problems) with some adaptations.

Author contributions All authors contributed to the conceptual design and writing the first draft of the manuscript. Software development and tests were performed by ZL with comments from all authors and the results were also analyzed by CGSS and IK. All authors have contributed to the revision of the article and approved the final manuscript.

Funding Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK). This project is partially supported by the National Natural Science Foundation of China under grant numbers 62173260 and 61803287.

Data availability All data generated or analyzed during this study are available upon request.

Declarations

Conflict of interests The authors have no relevant financial or non-financial interests to disclose.

Ethical approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

Ağpak, K., & Gökçen, H. (2007). A chance-constrained approach to stochastic line balancing problem. *European Journal of Operational Research, 180*(3), 1098–1115. https://doi.org/10.1016/j.ejor.2006.04.042

Aydoğan, E. K., Delice, Y., Özcan, U., Gencer, C., & Bali, Ö. (2019). Balancing stochastic U-lines using particle swarm optimization. *Journal of Intelligent Manufacturing, 30*(1), 97–111. https://doi.org/10.1007/s10845-016-1234-x

Bagher, M., Zandieh, M., & Farsijani, H. (2011). Balancing of stochastic U-type assembly lines: An imperialist competitive algorithm. *International Journal of Advanced Manufacturing Technology, 54*, 271–285. https://doi.org/10.1007/s00170-010-2937-3

Battaïa, O., & Dolgui, A. (2022). Hybridizations in line balancing problems: A comprehensive review on new trends and formulations. *International Journal of Production Economics, 250*. https://doi.org/10.1016/j.ijpe.2022.108673

Battaïa, O., & Dolgui, A. (2013). A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics, 142*(2), 259–277. https://doi.org/10.1016/j.ijpe.2012.10.020

Baykasoğlu, A., & Özbakır, L. (2007). Stochastic U-line balancing using genetic algorithms. *The International Journal of Advanced Manufacturing Technology, 32*(1), 139–147. https://doi.org/10.1007/s00170-005-0322-4

Borba, L., Ritt, M., & Miralles, C. (2018). Exact and heuristic methods for solving the robotic assembly line balancing problem. *European Journal of Operational Research, 270*(1), 146–156. https://doi.org/10.1016/j.ejor.2018.03.011

Boysen, N., Schulze, P., & Scholl, A. (2022). Assembly line balancing: What happened in the last fifteen years? *European Journal of Operational Research*, pp. 797–814.

Cakir, B., Altiparmak, F., & Dengiz, B. (2011). Multi-objective optimization of a stochastic assembly line balancing: A hybrid simulated annealing algorithm. *Computers and Industrial Engineering, 60*(3), 376–384. https://doi.org/10.1016/j.cie.2010.08.013

Carraway, R. L. (1989). A dynamic programming approach to stochastic assembly line balancing. *Management Science, 35*(4), 459–471. https://doi.org/10.1287/mnsc.35.4.459

Celik, E., Kara, Y., & Atasagun, Y. (2014). A new approach for rebalancing of U-lines with stochastic task times using ant colony optimisation algorithm. *International Journal of Production Research, 52*(24), 7262–7275. https://doi.org/10.1080/00207543.2014.917768

Chiang, W.-C., & Urban, T. L. (2006). The stochastic U-line balancing problem: A heuristic procedure. *European Journal of Operational Research, 175*(3), 1767–1781. https://doi.org/10.1016/j.ejor.2004.10.031

Chiang, W.-C., Urban, T. L., & Luo, C. (2015). Balancing stochastic two-sided assembly lines. *International Journal of Production Research, 54*(20), 6232–6250. https://doi.org/10.1080/00207543.2015.1029084

Delice, Y., Kızılkaya Aydoğan, E., & Özcan, U. (2016). Stochastic two-sided U-type assembly line balancing: A genetic algorithm approach. *International Journal of Production Research, 54*(11), 3429–3451. https://doi.org/10.1080/00207543.2016.1140918

Diefenbach, J., & Stolletz, R. (2022). Stochastic assembly line balancing: General bounds and reliability-based branch-and-bound algorithm. *European Journal of Operational Research*, pp. 589–605.

Duman, E., Uysal, M., & Alkaya, A. F. (2012). Migrating Birds Optimization: A new metaheuristic approach and its performance on quadratic assignment problem. *Information Sciences, 217*, 65–77. https://doi.org/10.1016/j.ins.2012.06.032

Erel, E., Sabuncuoglu, I., & Sekerci, H. (2005). Stochastic assembly line balancing using beam search. *International Journal of Production Research, 43*(7), 1411–1426. https://doi.org/10.1080/00207540412331320526

Fathi, M., Nourmohammadi, A., Ng, A. H. C., & Syberfeldt, A. (2019). An optimization model for balancing assembly lines with stochastic task times and zoning constraints. *IEEE Access, 7*, 32537–32550. https://doi.org/10.1109/access.2019.2903738

Fekete, S. P., & Schepers, J. (2001). New classes of fast lower bounds for bin packing problems. *Mathematical Programming, 91*(1), 11–31. https://doi.org/10.1007/s101070100243

Foroughi, A., & Gökçen, H. (2019). A multiple rule-based genetic algorithm for cost-oriented stochastic assembly line balancing problem. *Assembly Automation, 39*(1), 124–139. https://doi.org/10.1108/aa-03-2018-050

Gamberini, R., Gebennini, E., Grassi, A., & Regattieri, A. (2009). A multiple single-pass heuristic algorithm solving the stochastic assembly line rebalancing problem. *International Journal of Production Research, 47*(8), 2141–2164. https://doi.org/10.1080/00207540802176046

Gurevsky, E., Battaïa, O., & Dolgui, A. (2012). Balancing of simple assembly lines under variations of task processing times. *Annals of Operations Research, 201*(1), 265–286. https://doi.org/10.1007/s10479-012-1203-5

Gurevsky, E., Battaïa, O., & Dolgui, A. (2013). Stability measure for a generalized assembly line balancing problem. *Discrete Applied Mathematics, 161*(3), 377–394. https://doi.org/10.1016/j.dam.2012.08.037

Gurevsky, E., Rasamimanana, A., Pirogov, A., Dolgui, A., & Rossi, A. (2022). Stability factor for robust balancing of simple assembly lines under uncertainty. *Discrete Applied Mathematics, 318*, 113–132. https://doi.org/10.1016/j.dam.2022.03.024

Hamta, N., Fatemi Ghomi, S. M. T., Jolai, F., & Akbarpour Shirazi, M. (2013). A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics, 141*(1), 99–111. https://doi.org/10.1016/j.ijpe.2012.03.013

Hazır, Ö., & Dolgui, A. (2013). Assembly line balancing under uncertainty: Robust optimization models and exact solution method. *Computers and Industrial Engineering, 65*(2), 261–267. https://doi.org/10.1016/j.cie.2013.03.004

Hazır, Ö., & Dolgui, A. (2015). A decomposition based solution algorithm for U-type assembly line balancing with interval data. *Computers and Operations Research, 59*, 126–131. https://doi.org/10.1016/j.cor.2015.01.010

JrJung, L. (1997). A single-run optimization algorithm for stochastic assembly line balancing problems. *Journal of Manufacturing Systems, 16*(3), 204–210. https://doi.org/10.1016/S0278-6125(97)88888-7

Kottas, J. F., & Lau, H.-S. (1976). A total operating cost model for paced lines with stochastic task times. *AIEE Transactions*, pp. 234–240.

Kucukkoc, I., Li, Z., Karaoglan, A. D., & Zhang, D. Z. (2018). Balancing of mixed-model two-sided assembly lines with underground workstations: A mathematical model and ant colony optimization algorithm. *International Journal of Production Economics, 205*, 228–243. https://doi.org/10.1016/j.ijpe.2018.08.009

Lai, T.-C., Sotskov, Y. N., & Dolgui, A. (2019). The stability radius of an optimal line balance with maximum efficiency for a simple assembly line. *European Journal of Operational Research, 274*(2), 466–481. https://doi.org/10.1016/j.ejor.2018.10.013

Li, R., & Liu, G. (2017). An uncertain goal programming model for machine scheduling problem. *Journal of Intelligent Manufacturing, 28*(3), 689–694. https://doi.org/10.1007/s10845-014-0982-8

Li, Z., Janardhanan, M. N., Tang, Q., & Nielsen, P. (2016). Co-evolutionary particle swarm optimization algorithm for two-sided robotic assembly line balancing problem. *Advances in Mechanical Engineering, 8*(9), 1–14. https://doi.org/10.1177/1687814016667907

Li, Z., Kucukkoc, I., & Zhang, Z. (2018). Branch, bound and remember algorithm for U-shaped assembly line balancing problem. *Computers and Industrial Engineering, 124*, 24–35. https://doi.org/10.1016/j.cie.2018.06.037

Li, Y., Fu, Y., Tang, X., & Hu, X. (2019a). Optimizing the reliability and efficiency for an assembly line that considers uncertain task time attributes. *IEEE Access, 7*, 34121–34130. https://doi.org/10.1109/ACCESS.2019.2897730

Li, Z., Janardhanan, M. N., Tang, Q., & Ponnambalam, S. G. (2019b). Model and metaheuristics for robotic two-sided assembly line balancing problems with setup times. *Swarm and Evolutionary Computation, 50*, 100567. https://doi.org/10.1016/j.swevo.2019.100567

Li, Y., Peng, R., Kucukkoc, I., Tang, X., & Wei, F. (2020a). System reliability optimization for an assembly line under uncertain random environment. *Computers and Industrial Engineering, 146*, 106540. https://doi.org/10.1016/j.cie.2020.106540

Li, Z., Kucukkoc, I., & Tang, Q. (2020). A comparative study of exact methods for the simple assembly line balancing problem. *Soft Computing, 24*(15), 11459–11475. https://doi.org/10.1007/s00500-019-04609-9

Li, Z., Kucukkoc, I., & Zhang, Z. (2020). Branch, bound and remember algorithm for two-sided assembly line balancing problem. *European Journal of Operational Research, 284*(3), 896–905. https://doi.org/10.1016/j.ejor.2020.01.032

Li, Z., Çil, Z. A., Mete, S., & Kucukkoc, I. (2020b). A fast branch, bound and remember algorithm for disassembly line balancing problem. *International Journal of Production Research, 58*(11), 3220–3234. https://doi.org/10.1080/00207543.2019.1630774

Li, Y., Kucukkoc, I., & Tang, X. (2021). Two-sided assembly line balancing that considers uncertain task time attributes and incompatible task sets. *International Journal of Production Research, 59*(6), 1736–1756. https://doi.org/10.1080/00207543.2020.1724344

Li, Y., Liu, D., & Kucukkoc, I. (2023). Mixed-model assembly line balancing problem considering learning effect and uncertain demand. *Journal of Computational and Applied Mathematics, 422*, 114823. https://doi.org/10.1016/j.cam.2022.114823

Liu, B., & Yao, K. (2015). Uncertain multilevel programming: Algorithm and applications. *Computers and Industrial Engineering, 89*, 235–240. https://doi.org/10.1016/j.cie.2014.09.029

Liu, S. B., Ong, H. L., & Huang, H. C. (2005). A bidirectional heuristic for stochastic assembly line balancing Type II problem. *The International Journal of Advanced Manufacturing Technology, 25*(1), 71–77. https://doi.org/10.1007/s00170-003-1833-5

Liu, D. B. (2007). Uncertainty Theory. In D. B. Liu (Ed.), *Uncertainty Theory* (pp. 205–234). Berlin, Heidelberg: Springer Berlin Heidelberg.

Liu, B. (2009). *Theory and practice of uncertain programming* (Vol. 239): Springer.

Lopes, T. C., Sikora, C. G. S., Michels, A. S., & Magatão, L. (2020). Mixed-model assembly lines balancing with given buffers and product sequence: Model, formulation comparisons, and case study. *Annals of Operations Research, 286*(1), 475–500. https://doi.org/10.1007/s10479-017-2711-0

Lopes, T. C., Michels, A. S., Sikora, C. G. S., Brauner, N., & Magatão, L. (2021). Assembly line balancing for two cycle times: Anticipating demand fluctuations. *Computers and Industrial Engineering*, 107685.

McMullen, P. R., & Frazier, G. V. (1997). A heuristic for solving mixed-model line balancing problems with stochastic task durations and parallel stations. *International Journal of Production Economics, 51*(3), 177–190. https://doi.org/10.1016/S0925-5273(97)00048-0

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007

Moreira, M. C. O., Cordeau, J.-F., Costa, A. M., & Laporte, G. (2015). Robust assembly line balancing with heterogeneous workers. *Computers and Industrial Engineering, 88*, 254–263. https://doi.org/10.1016/j.cie.2015.07.004

Morrison, D. R., Sewell, E. C., & Jacobson, S. H. (2014). An application of the branch, bound, and remember algorithm to a new simple assembly line balancing dataset. *European Journal of Operational Research, 236*(2), 403–409. https://doi.org/10.1016/j.ejor.2013.11.033

Ning, Y., & Su, T. (2017). A multilevel approach for modelling vehicle routing problem with uncertain travelling time. *Journal of Intelligent Manufacturing, 28*(3), 683–688. https://doi.org/10.1007/s10845-014-0979-3

Otto, A., Otto, C., & Scholl, A. (2013). Systematic data generation and test design for solution algorithms on the example of SALBPGen for assembly line balancing. *European Journal of Operational Research, 228*(1), 33–45. https://doi.org/10.1016/j.ejor.2012.12.029

Özbakır, L., & Seçme, G. (2020). A hyper-heuristic approach for stochastic parallel assembly line balancing problems with equipment costs. *Operational Research*. https://doi.org/10.1007/s12351-020-00561-x

Özcan, U. (2010). Balancing stochastic two-sided assembly lines: A chance-constrained, piecewise-linear, mixed integer program and a simulated annealing algorithm. *European Journal of Operational Research, 205*(1), 81–97. https://doi.org/10.1016/j.ejor.2009.11.033

Özcan, U. (2018). Balancing stochastic parallel assembly lines. *Computers and Operations Research, 99*, 109–122. https://doi.org/10.1016/j.cor.2018.05.006

Özcan, U., & Toklu, B. (2009). A tabu search algorithm for two-sided assembly line balancing. *The International Journal of Advanced Manufacturing Technology, 43*(7), 822–829. https://doi.org/10.1007/s00170-008-1753-5

Pape, T. (2015). Heuristics and lower bounds for the simple assembly line balancing problem type 1: Overview, computational tests and improvements. *European Journal of Operational Research, 240*(1), 32–42. https://doi.org/10.1016/j.ejor.2014.06.023

Pereira, J. (2018). The robust (minmax regret) assembly line worker assignment and balancing problem. *Computers and Operations Research, 93*, 27–40. https://doi.org/10.1016/j.cor.2018.01.009

Pereira, J., & Álvarez-Miranda, E. (2018). An exact approach for the robust assembly line balancing problem. *Omega, 78*, 85–98. https://doi.org/10.1016/j.omega.2017.08.020

Pınarbaşı, M., & Alakaş, H. M. (2020). Balancing stochastic type-II assembly lines: Chance-constrained mixed integer and constraint programming models. *Engineering Optimization, 52*(12), 2146–2163. https://doi.org/10.1080/0305215x.2020.1716746

Pitakaso, R., Sethanan, K., Jirasirilerd, G., & Golinska-Dawson, P. (2021). A novel variable neighborhood strategy adaptive search for SALBP-2 problem with a limit on the number of machine's types. *Annals of Operations Research*. https://doi.org/10.1007/s10479-021-04015-1

Rao, R. V., Savsani, V. J., & Vakharia, D. P. (2011). Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Computer-Aided Design, 43*(3), 303–315. https://doi.org/10.1016/j.cad.2010.12.015

Sarin, S. C., Erel, E., & Dar-El, E. M. (1999). A methodology for solving single-model, stochastic assembly line balancing problem. *Omega, 27*(5), 525–535. https://doi.org/10.1016/S0305-0483(99)00016-X

Serin, F., Mete, S., & Çelik, E. (2019). An efficient algorithm for U-type assembly line re-balancing problem with stochastic task times. *Assembly Automation, 39*(4), 581–595. https://doi.org/10.1108/aa-07-2018-106

Sewell, E. C., & Jacobson, S. H. (2012). A branch, bound, and remember algorithm for the simple assembly line balancing problem. *INFORMS Journal on Computing, 24*(3), 433–442. https://doi.org/10.1287/ijoc.1110.0462

Sikora, C. G. S. (2021a). *Assembly-Line Balancing under Demand Uncertainty*. Weisbaden: Springer Gabler Wiesbaden.

Sikora, C. G. S. (2021b). Benders' decomposition for the balancing of assembly lines with stochastic demand. *European Journal of Operational Research*, pp. 108–124.

Silverman, F. N., & Carter, J. C. (1986). A cost-based methodology for stochastic line balancing with intermittent line stoppages. *Management Science*, pp. 455–463.

Sivasankaran, P., & Shahabudeen, P. (2014). Literature review of assembly line balancing problems. *International Journal of Advanced Manufacturing Technology, 73*(9–12), 1665–1694. https://doi.org/10.1007/s00170-014-5944-y

Suresh, G., & Sahu, S. (1994). Stochastic assembly line balancing using simulated annealing. *International Journal of Production Research, 32*(8), 1801–1810. https://doi.org/10.1080/00207549408957042

Tang, Q., Li, Z., Zhang, L., & Zhang, C. (2017). Balancing stochastic two-sided assembly line with multiple constraints using hybrid teaching-learning-based optimization algorithm. *Computers and Operations Research, 82*, 102–113. https://doi.org/10.1016/j.cor.2017.01.015

Urban, T. L., & Chiang, W.-C. (2006). An optimal piecewise-linear program for the U-line balancing problem with stochastic task times. *European Journal of Operational Research, 168*(3), 771–782. https://doi.org/10.1016/j.ejor.2004.07.027

Vilà, M., & Pereira, J. (2014). A branch-and-bound algorithm for assembly line worker assignment and balancing problems. *Computers and Operations Research, 44*, 105–114. https://doi.org/10.1016/j.cor.2013.10.016

Vrat, P., & Virani, A. (1976). A cost model for optimal mix of balanced stochastic assembly line and the modular assembly system for a customer oriented production system. *International Journal of Production Research*, pp. 445–463.

Wen, M., Qin, Z., & Kang, R. (2014). The $$\alpha $$-cost minimization model for capacitated facility location-allocation problem with uncertain demands. *Fuzzy Optimization and Decision Making, 13*(3), 345–356. https://doi.org/10.1007/s10700-014-9179-z

Yuan, B., Zhang, C., & Shao, X. (2015). A late acceptance hill-climbing algorithm for balancing two-sided assembly lines with multiple constraints. *Journal of Intelligent Manufacturing, 26*(1), 159–168. https://doi.org/10.1007/s10845-013-0770-x

Zhang, W., Xu, W., Liu, G., & Gen, M. (2017). An effective hybrid evolutionary algorithm for stochastic multiobjective assembly line balancing problem. *Journal of Intelligent Manufacturing, 28*(3), 783–790. https://doi.org/10.1007/s10845-015-1037-5

Zhang, H., Zhang, C., Peng, Y., Wang, D., Tian, G., Liu, X., & Peng, Y. (2018). Balancing problem of stochastic large-scale u-type assembly lines using a modified evolutionary algorithm. *IEEE Access, 6*, 78414–78424. https://doi.org/10.1109/access.2018.2885030

Zhao, X., Liu, J., Ohno, K., & Kotani, S. (2007). Modeling and analysis of a mixed-model assembly line with stochastic operation times. *Naval Research Logistics (NRL), 54*(6), 681–691. https://doi.org/10.1002/nav.20241