A SURVEY ON BENDERS DECOMPOSITION METHODS APPLIED TO ASSEMBLY LINE BALANCING PROBLEMS

Adalberto Sato Michels[1], Alysson M. Costa[1], and Celso Gustavo Stall Sikora[2*]

*Corresponding author
[1] School of Mathematics and Statistics, University of Melbourne, Melbourne, Australia
[2]Institute for Operations Research, University of Hamburg, 20148 Hamburg, Germany / sikora.gustavo@gmail.com

ABSTRACT. This survey presents a comprehensive review on the application of classical and logic-based Benders decomposition (BD) approaches to Assembly Line Balancing Problems (ALBP). The core decisions in assembly lines involve assigning a partially ordered set of tasks to (work)stations to maximise productivity or minimise resource usage in flow-oriented systems. ALBPs may contain stochastic data, multiple workers per station, or depend on the production sequence. Hence, they require further scheduling decisions - various stochastic and sequence-dependent problems have successfully applied this approach. This paper contributes to the current body of knowledge by providing a literature review. It introduces each article's particular aspects and ideas and brings forth a summary and discussions on existing gaps. We offer insights into the BD method's efficiency for combinatorial problems such as ALBPs from a managerial perspective, suggest potential directions for future research, and highlight the increasing number of contributions using this technique.

Keywords: Assembly line design and balancing; Production planning and scheduling; Benders decomposition; Literature review; Stochastic optimisation.

This is the peer-reviewed, accepted version of the paper published by Pesquisa Operacional (2026): 46, 1–31.

DOI: 10.1590/0101-7438.2026.046.00309400

1 Introduction

Assembly line balancing problems (ALBP) are flow oriented manufacturing systems. The core decisions of the problem consist in assigning a partially ordered set of tasks to (work)stations with the general goal of maximising productivity given a set of resources (e.g. stations and workers) or minimising the number of resources needed to achieve a certain productivity level.

The most basic ALBP is known as the *simple assembly line balancing problem* (SALBP) and was originally modelled by Salveson (1955) with several simplification hypotheses.

Later, Baybars (1986) introduced the idea of *general assembly line balancing problems* (GALBP). The GALBP removes some simplification hypotheses found in the SALBP, such as the need for a serial line layout or the deterministic nature of task execution times, allowing for the modelling of more realistic industrial settings.

Efforts to develop exact and heuristic solution procedures were made on both SALBP and GALBP research streams. Scholl and Becker (2006) documented methods for the SALBP, whereas Becker and Scholl (2006) summarised models and solution approaches for GALBP. Boysen et al. (2007) provided a comprehensive classification scheme for the most relevant GALBP extensions, whilst Boysen et al. (2008) laid out some guidelines for deciding on the most appropriate models for each application. Most recently, Boysen et al. (2022) updated these reviews, surveying the scientific literature on ALBPs published in the 15 years prior to their article. In their survey, they hinted that decomposition approaches are becoming the solution method of choice for a number of different problems.

One of the most efficient and studied decomposition methods for combinatorial problems such as GALBPs is Benders Decomposition (BD) proposed by Benders (1962). The method relies on a reformulation of mixed-integer programming models that projects all continuous variables into a single variable, at the expense of introducing an exponential number of new constraints. A computational approach – which is what came to become known as Benders decomposition – solves a relaxed version of this extensive model (known as master problem, or MP) to generate tentative partial solutions. Every time a tentative solution is obtained, a subproblem (SP) is used to generate one of the relaxed constraints (known as Benders cuts) until optimality is obtained.

The classical BD method as describe above relies on duality theory to generate Benders cuts. Therefore, it requires the SPs to be linear programming models. An extension of the method was proposed by Geoffrion (1972), which allows SPs to take the more general form of convex problems. Still, this requirement is often too restrictive to allow for the tackling of GALBPs. A relatively recent version of Benders for general SPs was proposed by Hooker and Ottosson (2003). The technique is known as *logic-based Benders decomposition* (LBBD). The idea of LBBD is to generate cuts in the SPs via a specific logic to the problem at hand, which is known as an *inference dual*. Without the need of convex SPs, the LBBD can be applied to many different versions of GALBPs. The most common implementations of LBBD for GALBPs decide on task assignments in the MP and relegate other problem characteristics (the G of GALBP) to the SPs.

Despite the growing number of BD contributions to GALBPs over the last decade, no dedicated survey exists to guide practitioners and researchers in navigating this literature. The diversity of decomposition strategies (ranging from classical BD with linear subproblems to LBBD with integer subproblems and combinatorial cuts) combined with the variety of GALBP extensions addressed, makes it difficult to identify best practices or transferable design principles. This gap motivates the present survey.

This systematic survey contributes to the current body of knowledge by providing a literature review on the application of classical and logic-based BD approaches to GALBPs as an extension of a conference paper by Michels and Sikora (2022). It introduces each article's particular aspects and ideas and brings forth a summary and discussions on ex-

isting gaps. We offer insights into the BD method's efficiency for combinatorial problems such as ALBPs from a managerial perspective, suggest potential directions for future research, and highlight the increasing number of contributions using this technique.

The remainder of this survey is organised as follows. Section 2 introduces the core SALBP models, which serve as the foundation for more complex formulations. In Section 3, we present a detailed explanation of Benders decomposition, including a generic framework of its application to GALBPs and improvement techniques. Then, in Section 4, we review the literature introducing the particular aspects and ideas of each article. Section 5 provides a summary and discussions on existing gaps in the literature for future research avenues. Finally, Section 6 ends this paper with concluding remarks.

2 Assembly line balancing models

This first version of the problem proposed by Salveson (1955) considers a task set $N = \{1, 2, ..., n\}$. Each task in $i \in N$ has a deterministic execution time t_i. Associated with tasks in N there is also a set of partial precedence relations $P = \{(i, j) \in (N \times N) \,|\, i \prec j\}$, where $i \prec j$ indicates that task i must be executed before task j. Let also $S = \{1, 2, ..., m\}$ be a set of ordered stations and c be the cycle time of the line, i.e. the workload of the most loaded station. The two main versions of the SALBP consist in minimising the number of stations m needed to respect a given cycle time (type-1, or SALBP-1) and minimising the cycle time c given a fixed number of stations (type-2, or SALBP-2).

We herein present mathematical models for both SALBP versions. For type-1, which aims at minimising the number of stations for a given cycle time $\overline{c}$, we use a known upper bound for the number of stations, $\overline{m}$, to define the cardinality of set S. The model (1)–(6) uses binary variables x_{si} representing task-to-station assignments, that is, x_{si} assumes value 1 if task i is assigned to station s and 0 otherwise. Furthermore, y_s are binary variables indicating the usage of station s:

$$\text{Minimise} \sum_{s \in S} s y_s, \tag{1}$$

subject to:

$$\sum_{s \in S} x_{si} = 1, \quad \forall\, i \in N, \tag{2}$$

$$\sum_{s \in S} s x_{si} \leq \sum_{s \in S} s x_{sj}, \quad \forall\, (i, j) \in P, \tag{3}$$

$$\sum_{i \in N} t_i x_{si} \leq \overline{c} y_s, \quad \forall\, s \in S, \tag{4}$$

$$x_{si} \in \{0, 1\}, \quad \forall\, i \in N;\ s \in S, \tag{5}$$

$$y_s \in \{0, 1\}, \quad \forall\, s \in S. \tag{6}$$

The type-2 problem can be written in a similar way, now minimising the cycle time

continuous variable c for a fixed number of stations m:

$$\text{Minimise } c, \tag{7}$$

subject to:

$$\sum_{i \in N} t_i x_{si} \leq c, \quad \forall\ s \in S, \tag{8}$$

Constraints (2)–(3) and (5),

$$c \in \mathbb{R}_{\geq 0}. \tag{9}$$

Both models contain occurrence constraints (2), which state that all tasks must be assigned, precedence constraints (3) ensuring that the partial ordering of tasks is respected, and cycle time constraints (4) or (8) to ensure workload limitations. The type-1 problem still requires the logical link checking how many stations of the set S are actually needed. The enforcement of an open station if any task is performed there happens in constraints (4). Additionally, the s multiplier in its objective function (1) is an optional symmetry breaking feature. It requires the optimal solution to use the first available stations, thus avoiding an equally good solution that could, for instance, use the last available stations.

Models for type-1 (1)–(6) and type-2 (7)–(9) SALBPs were both explored in the first ALBP survey (Baybars, 1986), in which the idea of GALBP was introduced.

3 Benders decomposition

Benders decomposition (Benders, 1962) is a well-established strategy for solving large optimisation problems. In the following, we present a brief explanation of the method.

Consider the following general optimisation problem:

$$\text{Minimise } z(x, y), \tag{10}$$

subject to:

$$g(x, y) \geq b, \tag{11}$$

$$h(x) \geq d. \tag{12}$$

The classical Benders decomposition method considers linear functions $z(x, y)$, $g(x, y)$ and $h(x)$ on variables $x \in \mathbb{R}^n$ (or $\mathbb{Z}^n$) and $y \in \mathbb{R}^m$. The approach first relaxes model (10)–(12) into a master problem as:

$$\text{Minimise } z(x, 0) + \theta, \tag{13}$$

subject to:

$$h(x) \geq d. \tag{14}$$

Where $\theta \in \mathbb{R}$ represents the cost of the subproblem. At each iteration k of the method, the relaxed master problem is solved to obtain a partial solution $\overline{x}^k$. This tentative

solution is then used in the subproblem, written as:

$$\text{Minimise } z(0, y), \tag{15}$$

$$\text{subject to:}$$

$$g(\overline{x}^k, y) \geq b. \tag{16}$$

The solution of this subproblem is used to generate missing constraints for the master problem (13)–(14). When the subproblem is infeasible, these constraints are called feasibility cuts and indicate that the tentative solution $\overline{x}^k$ proposed by the master is actually infeasible for the original problem (10)–(12). When the subproblem has an optimal solution, $\overline{y}^k$, a complete solution $(\overline{x}^k, \overline{y}^k)$ for the original problem has been found. This information is sent to the master with a constraint that states that the value of θ in (13) must be equal to $z(\overline{x}^k, \overline{y}^k)$ if solution $x = \overline{x}^k$ is found again in the master problem. These latter constraints are called optimality cuts.

As the master problem is a relaxed version of the original problem, its solution provides a lower bound for the objective function of the original problem. Moreover, an upper bound is obtained at each iteration that finds a complete solution. The method iterates until the lower bound has the same value of the best upper bound, proving the optimality of the best solution found over all iterations.

To exemplify the application of Benders decomposition to assembly line balancing problems, consider the following general model for a GALBP of type-2:

$$\text{Minimise } c, \tag{17}$$

$$\text{subject to:}$$

$$\sum_{s \in S} x_{si} = 1, \tag{18}$$

$$\sum_{s \in S} s x_{si} \leq \sum_{s \in S} s x_{sj}, \quad \forall\ (i, j) \in P, \tag{19}$$

$$\sum_{i \in N} t_i x_{si} \leq c, \quad \forall\ s \in S \tag{20}$$

$$g(x, y) \geq b. \tag{21}$$

In this model, variables x_{si} are binary and indicate that task $i \in N$ is assigned to station $s \in S$. Constraints (11) have been instantiated as classical SALBP assignment constraints (18), precedence constraints (19) and cycle time constraints (20). Constraints (21) are specific to the GALBP at study.

Example (i) Consider an instantiation of model (17)–(21) in which constraints (21) take the form of constraints (23)–(24):

$$\text{Minimise } c, \tag{22}$$
$$\text{subject to:}$$
$$(18)\text{–}(20),$$
$$\sum_{i \in N} t_i x_{si} + y_s \leq c, \quad \forall\ s \in S, \tag{23}$$
$$\sum_{s \in S} y_s = p. \tag{24}$$

An interpretation of this fictitious GALBP is that an additional amount of work, p, must be executed in the product. This extra work can be performed in any of the stations. The amount of extra work assigned to station $s \in S$ is given by $y_s \geq 0$. A possible implementation of Benders decomposition to problem (22)–(24) uses the following relaxed master problem:

$$\text{Minimise } \theta, \tag{25}$$
$$\text{subject to:}$$
$$(18)\text{–}(20).$$

This master problem is solved to obtain tentative assignments $\overline{x}_{si}^k$ at each iteration k of the method. The actual cycle time associated with this solution is computed after the extra work is allocated to the stations in the subproblem:

$$\text{Minimise } c, \tag{26}$$
$$\text{subject to:}$$
$$y_s + \sum_{i \in N} t_i \overline{x}_{si}^k \leq c, \quad \forall\ s \in S, \tag{27}$$
$$\sum_{s \in S} y_s = p. \tag{28}$$

Constraints (27) are constraints (23) rewritten with the assignment of tasks to stations obtained by the master problem in iteration k of the method, $\overline{x}_{si}^k$. We note that problem (26)–(28) is a linear program on variables y_s and therefore its optimal solution can be obtained by solving the associated dual problem. Associating dual variables $\pi_s \geq 0$ and

μ to constraints (27) and (28), respectively, this dual problem can be written as:

$$\text{Maximise } \sum_{s\in S}\sum_{i\in N} t_i \overline{x}^k_{si}\pi_s + p\mu, \tag{29}$$

subject to:

$$\sum_{s\in S}\pi_s \leq 1, \tag{30}$$

$$\mu - \pi_s \leq 0, \quad \forall\ s \in S, \tag{31}$$

$$\pi_s \geq 0, \quad \forall\ s \in S, \tag{32}$$

$$\mu \text{ free.} \tag{33}$$

Let $(\overline{\pi}^k_s, \overline{\mu}^k)$ be the optimal solution of this problem and $\overline{c}^k$ be its objective solution value. That is, for assignment $\overline{x}^k_{si}$, the minimum cycle time that can be obtained is given by $\overline{c}^k$. This yields the optimality cut (34), which is added to the master problem for the next iteration:

$$\theta \geq \sum_{s\in S}\sum_{i\in N} t_i x_{si}\overline{\pi}^k_s + p\overline{\mu}^k. \tag{34}$$

Finally, note that in the first iteration, there is no objective in the master problem as no bound for z has yet been obtained. Also, as problem (26)–(28) is always feasible there are only optimality cuts.

The solution of this example relies on duality theory to obtain the optimality cuts. This is only possible because variables y_s are continuous. The case in which variables $y \in \mathbb{Z}^m$ is considered by Hooker and Ottosson (2003). The authors propose the use of an inference dual, in which cuts are obtained using logical deductions.

Example (ii) Consider the problem in example (i), with the modification that the extra work at stations must be performed in integral chunks. That is, variables y_s must now be integer, as expressed in constraints (38):

$$\text{Minimise } c, \tag{35}$$

subject to:

$$\sum_{i\in N} t_i\overline{x}_{si} + y_s \leq c, \quad \forall\ s \in S, \tag{36}$$

$$\sum_{s\in S} y_s = p, \tag{37}$$

$$y_s \geq 0 \text{ and integer}, \quad \forall\ s \in S. \tag{38}$$

The subproblem now contains integer variables and, therefore, linear programming duality theory can not be used to generate Benders cuts. The approach in this case consists in generating a logical cut that explicitly states that if the master solution is repeated, the

cost of the subproblem must be at least equal to the optimal solution of the respective subproblem.

As before, let $\overline{c}^k$ be the optimal solution value of subproblem (35)–(38) for an assignment $\overline{x}^k_{si}$. In this assignment, let X^0_{ks} and X^1_{ks} be the sets of tasks i for which the associated variable in the tentative solution $\overline{x}^k_{si}$ assumes value 0 and 1, respectively, for the station s in iteration k. The combinatorial cut that can be added to the master reads:

$$\theta \geq \overline{c}^k - \left(\sum_{s \in S} \sum_{i \in X^0_{ks}} x_{si} + \sum_{s \in S} \sum_{i \in X^1_{ks}} (1 - x_{si}) \right) \overline{c}^k. \tag{39}$$

This cut imposes that if a new solution to the master problem repeats solution $\overline{x}^k_{si}$, the cycle time is imposed to be at least the cycle time obtained in the subproblem of iteration k, $\overline{c}^k$. As this cut remains valid if variables in set T^k_1 are set at 0, it can be strengthened as:

$$\theta \geq \overline{c}^k - \left(\sum_{s \in S} \sum_{i \in X^0_{sk}} x_{si} \right) \overline{c}^k. \tag{40}$$

3.1 Improving the Benders decomposition

The LBBD is a widely used technique for solving different versions of large-scale GALBPs. Despite its effectiveness, the method has some limitations that can hinder its performance in certain scenarios. Rahmaniani et al. (2017) have surveyed various modifications and extensions to overcome these limitations and improve its efficiency. These modifications include acceleration strategies, heuristics, and extensions to handle more complex optimisation models. In this context, we provide a comprehensive summary of particular advancements and techniques for improving the LBBD performance when applied to GALBPs. We also discuss the strengths and weaknesses of these enhancement approaches.

3.1.1 Variable set used in the cut

Modifying the variable set used in the cut might be a promising approach to improve the method's performance, as tightening the MP or the SP by generating stronger cuts can be an effective acceleration strategy. For instance, an approach that generates cuts based on the models' most relevant subset of variables can reduce the combinatorial cut size and improve its effectiveness, leading to faster convergence and better solution quality. While standard cuts take the shadow price to extend to other values, a combinatorial Benders cut (CBC) only limits the variable subset that is used in its definition for a combination of values. As these CBCs are known to be relatively weak, Codato and Fischetti (2006) suggests the Minimal Infeasible Subsystem (MIS) usage. However, since it consists of a minimal variable subset that causes the solution to become infeasible, detecting such MIS is not always trivial.

Another way to modify the variable set is by adding continuous variables to valid inequalities (VI) involving only integer variables or using lifting techniques from cuts. By solving the linear relaxation of the original problem with a cutting-plane method, we are able to generate cuts that involve continuous variables. After decomposing the model, these VIs may be moved to the MP/SP, resulting in improved combinatorial cuts.

We hereafter refer to these kinds of improvements as I-1.

3.1.2 Warm start

A warm start refers to the use of an initial feasible solution as a starting point for the optimisation process. This approach can reduce the number of iterations required for a BD to converge to an optimal solution. Therefore, they tend to improve the algorithm's overall efficiency.

Several studies have investigated the use of warm starts in BD, demonstrating that such approach can lead to faster convergence, significantly reduce computational time, and ultimately generate better solution quality when compared to applying a stand-alone BD. For example, heuristics can generate initial feasible solutions, which are then used as a warm start before starting the BD iterations. Analogously, hybrid algorithms that combine metaheuristics with BDs may yield similar results.

As proposed methods in the literature have shown promising results by using heuristic or metaheuristic algorithms to generate initial feasible solutions for BDs, we hereafter refer to these improvements as I-2.

3.1.3 Heuristics for MP solutions

Heuristics are problem-specific algorithms suitable for quickly producing good-quality solutions. Unlike simple warm starts, however, these MP heuristics would iteratively generate better solutions during an optimisation run.

In literature, we can find promising results in this regard by using heuristic algorithms such as tabu or local searches to solve around each integer solution of the MP more efficiently. Thus, we hereafter refer to these kinds of improvements as I-3.

3.1.4 Restarting the MP

The idea behind restarting the MP is to periodically reset the instance run by solving a refreshed model with a different set of constraints or variables. This procedure might help the BD to overcome stagnation issues that may arise during the optimisation process.

In current solver implementations, combinatorial Benders cuts are not part of the formulation. Instead, we usually append them as lazy constraints while the optimisation process runs, so they are tested for the solutions but are not included in the model to generate further automatic cuts. Therefore, restarting the MP can be an effective strategy to improve the BD performance. One way to improve it is to store the generated

cuts, and if the solution is taking too long or making little progress, we can restart a new MP with the previously evaluated cuts. Additionally, the choice of constraints or variables added at each restart also depends on various factors such as problem structure, solution quality, and available computational resources.

We hereafter refer to this potential alternative as I-4.

3.1.5 Variable choice for MP and SP

The variable choice for MP and SP can significantly impact the BD performance, as the proper selection of variables and cuts is essential for effective acceleration strategies. In general, formulations include variables that are difficult to handle in the SP, while addressing the more straightforward ones in the MP.

Specifically for GALBPs, balancing variables are typically incorporated in the MP, while other extensions, such as setup times, resource availability, sequencing and scheduling, are delegated to the SP. Nonetheless, some authors may also choose to include some additional model elements or leave relaxed versions of complicating variables in the MP (Gendron et al., 2014).

Naturally, the choice of variable placement also depends on the problem structure and available computational resources. Including additional variables in the MP can help improve solution quality and reduce computational time by decreasing the SP size and complexity. On the other hand, this approach may not always be feasible or effective if the MP cannot find new tentative primal bounds. Nevertheless, we hereafter refer to the deliberate choice of variable placement as I-5.

3.1.6 MP tightness

Although warm starts and local search heuristics can provide good feasible solutions, the tightness of the MP to inform lower bounds on the objective function is a crucial factor in the approach efficiency. If the MP is not tight enough, it can lead to slow convergence (instability) or erratic progression (degeneracy) of lower bounds, which might result in stagnation for long computational processing times.

For GALBPs, we can use the problem's knowledge to apply assignment restrictions and incompatibility constraints. These MP tightening improvements are hereafter referred to as I-6.

3.1.7 Hybrid procedures

Hybrid BD procedures combine optimisation techniques, such as constraint programming, column generation, and dynamic programming. These hybridisations are usually incorporated into the SP solving process to enhance the effectiveness of BDs. In addition, these methods may help address specific GALBP extensions by leveraging their respective strengths at different stages of the BD process.

Within a constraint programming framework, we can more easily formulate some complex constraints. It also provides efficient algorithms for finding feasible solutions to scheduling problems, which are helpful for some GALBP variants.

Column generation is known to be efficient at handling large-scale optimisation problems by generating columns incrementally, so it could be used to dynamically produce the necessary variables based on the MP structure or solution progress. By iteratively adding promising columns (variables) to the MP, a hybrid column generation approach may effectively reduce the problem's initial size, allowing the BD to focus on a subset of variables that contribute the most to the objective function under evaluation.

Finally, dynamic programming can efficiently solve SPs with overlapping structures. It breaks down the SPs into smaller subsets, solves them recursively, and stores their solutions in a hash table for quick reuse. This latter part of the approach can be used independently, avoids redundant computations, and significantly improves computational processing times.

We hereafter refer to hybrid strategies as I-7. Furthermore, we discuss how each strategy has been specialised for different versions of GALBPs in the following Section 4.

4 Literature review

The papers included in this review are identified through a systematic search on Scopus and Google Scholar, using combinations of the terms *Benders decomposition* and *assembly line balancing* applied to titles, abstracts, and keywords. The reference lists of all retrieved papers are additionally screened, and any relevant works not captured by the initial search are included in the pool. Inclusion was restricted to papers in which a BD approach constitutes a core methodological contribution to an ALBP variant.

As presented in Section 2, although the SALBP can be compactly formulated, GALBPs contain variants and extensions that might change every single restriction: namely occurrence, precedence, station opening, and cycle time constraints. This section presents a detailed literature review of BD algorithms proposed for GALBPs. We organise the sections according to current research trends concerning extended and new decision problems, as laid out by the most recent ALBP surveys (Boysen et al., 2022; Battaïa and Dolgui, 2022).

Occurrence, for instance, is not mandatory in disassembly lines, in which the inverse tasks are performed in order to disassemble the product to reuse or recycle its components. Depending on the demand and the profit for each component, it may not be economically viable to disassemble some parts, so tasks are simply not performed (Gungor and Gupta, 2001). The disassembly process is also an example of other forms of precedence relations. The disassembly of inner components often can be performed if one of their sides is free, resulting in an "OR" precedence relation (either one of a set of tasks may suffice for the disassembly) (Gungor and Gupta, 2001). The opening of stations is also usually more complex than the constraint expressed in SALBP. Several GALBP variants integrate the selection of equipment (Michels et al., 2018) or assign multiple workers in a station

(Michels et al., 2019) so that more than the variable y_s is needed for the station definition.

Besides the presented variants, the single most changed and adapted constraint is the cycle time one, i.e. Constraints (4). The assumption that the occupation of a station is the sum of the tasks' processing times is severely limiting, and the modelling of (and solving for) this occupation is the most challenging part of an ALBP. Before we dive into the description of the BD contributions, some ways the cycle time restriction has been modelled are introduced:

- Stochastic processing times: when the processing times t_i are uncertain, the cycle time can be transformed into a chance constraint. For instance, it may be required that the cycle time is within a given upper bound 95% of the time.
- Set-up times: workers may require to change tools or set up the equipment between tasks. Considering set-up times add assignment-dependent load to the station so that the sum of the processing times is not enough to model the station occupation.
- Multiple workers per station: due to precedence relations the scheduling of workers inside a station is interdependent. The cycle time is enforced on the worker with the longest 'makespan'.
- Multiple products: different products require task adaptations with different processing times. The limiting factor is not the sum of the individual tasks or the average of all products but it depends on the line's implementation. Paced lines can, for instance, use stoppage or utility workers to allow for processing time variations. Unpaced lines depend on the scheduling of stations being prone to blockage and starvation.

4.1 Combinatorial Benders decomposition framework: an example

We bring an example from the pioneering work of Akpinar et al. (2017) due to its simplicity to start off the review on decomposition structures. Consider a simplified version of the Set-Up Assembly Line Balancing Problem (SUALBP) (Akpinar et al., 2017), in which set-up times exist between the processing of two tasks. The problem can be defined as an extension of model (1)–(6), requiring extra variables for task sequencing. The cycle time restriction (4) is still valid for the problem, but the left-hand side of the inequality is not tight, as it does not consider the set-up times u_{ij} between tasks i and j. In this section, we develop an MILP model to include the set-up times and show how the decomposition can be applied to the resulting model.

4.1.1 Task sequencing formulation

Let z_{sij} (and w_{sij}) be binary variables to capture whether task i (directly) precedes task j in station s. z_{sij} assumes 1 if task i is performed before j and 0 otherwise, while w_{sij} is 1 only if i is performed directly before j. For simplicity, we leave the set-up between

the last task of a cycle to the first task of the next cycle aside. The cycle time restriction can then be improved to:

$$\sum_{i \in N} t_i x_{si} + \sum_{i \in N} \sum_{j \in N} u_{ij} w_{sij} \leq c, \quad \forall\ s \in S. \tag{41}$$

However, correctly modelling variables w_{sij} requires many logic constraints. Constraints (42)–(47) give a possible formulation.

$$z_{sij} \leq x_{si}, \quad \forall\ i,j \in N,\ s \in S, \tag{42}$$
$$z_{sij} \leq x_{sj}, \quad \forall\ i,j \in N,\ s \in S, \tag{43}$$
$$z_{sij} = 0, \quad \forall\ i,j \in N, (i,j) \in P,\ s \in S, \tag{44}$$
$$z_{sij} + z_{sji} \geq x_{si} + x_{sj} - 1, \quad \forall\ i,j \in N,\ s \in S, \tag{45}$$
$$w_{sij} \leq z_{sij}, \quad \forall\ i,j \in N,\ s \in S, \tag{46}$$
$$\sum_{i \in N} \sum_{j \in N} w_{sij} = \sum_{i \in N} x_{si} - 1, \quad \forall\ s \in S, \tag{47}$$
$$z_{sij} \in \{0,1\}, \quad \forall i,j \in N \mid i \neq j,\ s \in S, \tag{48}$$
$$w_{sij} \in \{0,1\}, \quad \forall i,j \in N \mid i \neq j,\ s \in S. \tag{49}$$

Firstly, it only makes sense to define the order of two tasks if they are assigned to the same station, as enforced in Constraints (42) and (43). Also, the order in which tasks are performed must not contradict the precedence constraints (P), as stated in Constraints (44). If both tasks i and j are assigned to the same station s ($x_{si} = x_{sj} = 1$), one of these tasks must precede the other. This restriction is formulated in Constraints (45). The direct precedence of tasks is modelled with Constraints (46) and (47). The former state that task j can only directly follow task i if i precedes j, i.e. $z_{sij} = 1$. The latter enforces the number of required set-ups: if L tasks are assigned to station s ($\sum_{i \in N} x_{si}$), then there should be $L - 1$ set-ups. Once again, note that the set-up for the new cycle is ignored. The domain of variables z_{sij} and w_{sij} are given in Constraints (48) and (49).

The resulting model (1)–(3), (5)–(6), (41)–(49) is henceforth named as SUALBP. It contains a slightly different cycle time restriction (compare (4) to (41)), but requires several extra binary variables (z_{sij} and w_{sij}) in the formulation.

4.1.2 Decomposition structure for the example

In order to introduce the decomposition structure, we first compare the solution space of the SALBP to its extension considering setup times (SUALBP).

All feasible solutions of the SUALBP are also feasible for the SALBP since the setup times are not considered in the latter. On the other hand, there are feasible SALBP solutions that are infeasible for the SUALBP. Such solutions occur when the sum of processing times of the assigned tasks is within the cycle time, but the total work required including setup times surpasses the cycle time. If one could enumerate all these solutions and exclude them from the solution space of a SALBP formulation, the solution space of the resulting formulation would be equivalent to the SUALBP.

The exclusion of an assignment can be performed using a combinatorial cut (Codato and Fischetti, 2006). Suppose F is a set containing all task assignments feasible for the SALBP and infeasible for the SUALBP. Each element of F contains a list of tasks assigned to each station and F_s contains the assignments of station s (variables x_{si} assuming the value 1), which are denoted as X^1_{sf}. As all assignments are given by binary variables, cuts (50) enforce the assignment contained in X^1_{sf} to be infeasible. According to this cut, it is not possible that all x_{si} variables in the set are set to 1 simultaneously.

$$\sum_{i \in X^1_{sf}} x_{si} \leq |X^1_{sf}| - 1, \quad \forall\ s \in S,\ f \in F. \tag{50}$$

Although the solution space of formulation (1)–(6), (50) is equivalent to the SUALBP, set F can be extremely large since it contains combinations of multiple tasks. The idea of the (combinatorial) BD (Benders, 1962; Codato and Fischetti, 2006; Akpinar et al., 2017) is to start with a relaxed version of the problem and add cuts (such as the ones in (50)) when necessary in an iterative fashion.

The decomposition structure presented in Akpinar et al. (2017) relegates all sequence decisions to subproblems. In this way, the master problem consists of only assembly-line balancing decisions and is initially equivalent to the SALBP model presented in (1)–(6). The MP is solved and task assignment decisions of its solution are then tested at the SP level. The SP consists of sequencing tasks within the stations. The sequencing problem is defined by the model (41)–(49) considering x_{si} as parameters. Note that the sequencing problem of each station is independent, thus they can be solved separately for each $s \in S$. If the subproblem of any station s and iteration k is infeasible, the set of tasks assigned to such station X^1_{sk} must be prohibited. A cut in the form of (50) is defined and added to the master problem. After appending the cuts, the MP keeps searching for a solution. The process is repeated until an optimal solution to the MP is also feasible after considering its task sequencing in all stations.

As the BD is an iterative process, it is important for its convergence rate to intelligently define the subset of F that is included in each iteration. According to Codato and Fischetti (2006), the user should aim at using the MIS to define X^1_{sf}, which consists of a minimal set of variables that cause the solution to become infeasible. For instance, a solution with the assignment of 5 tasks with index 1 to 5 in station s is discovered as infeasible. The cut $x_{s1} + x_{s2} + x_{s3} + x_{s4} + x_{s5} \leq 4$ can be used to eliminate the solution. However, if assigning tasks 1, 2, and 3 alone already cause an infeasibility, the cut $x_{s1} + x_{s2} + x_{s3} \leq 2$ is stronger and eliminates several solutions that the previous constraint does not cut off. In the next sections, the importance of the number of variables within the cut is discussed for different ALBP extensions.

4.2 ALBP with sequence-dependent set-up times

The framework presented in the example of Section 4.1 is implemented by Akpinar et al. (2017) in a less simplified problem. Differently from the example, Akpinar et al. (2017) considered the assembly of multiple products and the setup of tasks between products as well. The decomposition structure is identical to the one shown in Section 4.1.2.

An extension of the problem solved in Akpinar et al. (2017) considering worker wages is given in Furugi (2022). In this extension, each task requires a minimal level of skill to be performed. Therefore, the wages paid in each station are determined by the most cost-intensive tasks assigned to it. The definition of MP and SP for the division of the assembly-line balancing and setup time sequencing is the same as in Akpinar et al. (2017).

Note that for both Akpinar et al. (2017) and Furugi (2022), the variables of the SP do not play a role in the objective function, as only the number of stations is minimised. A structural adaptation is required when the quality of a solution depends on the result of the SP. For instance, Zohali et al. (2022) minimises the cycle time for a given number of stations (ALBP Type-2). When setup times are present, the cycle time depends on the sequencing solution and cannot be obtained directly from the relaxed MP.

For the type-2 problem, CBCs as cuts (51) can be used to correct the objective function of the MP in each iteration $k \in K$.

$$c \geq \overline{c_s^k} - \sum_{i \in X_{sk}^1} \beta_i (1 - x_{si}), \quad \forall\ s \in S,\ k \in K. \tag{51}$$

Cuts (51) are defined based on a set of tasks X_{sk}^1 assigned to station s found in iteration k of the algorithm. A known upper bound on the cycle time $\overline{c_s^k}$ is enforced if all variables x_{si} within the set X_{sk}^1 assume 1. Nonetheless, these cuts must only enforce the upper bound for this specific assignment. If a task i is not selected, the upper bound $\overline{c_s^k}$ is reduced by β_i. β_i is defined as the maximal time increment that can be saved by leaving task i aside. This increment is equal to the task processing time plus the maximal setup time required from any task that can be assigned before and after task i.

4.3 ALBP with multiple workers in stations

The ALBP extensions that flourished the most with respect to BD approaches are the ones considering multiple workers in the same station. Allowing more than one worker can greatly shorten the assembly line length at the cost of coordination among workers. Due to precedence relations and interference between workers, a task scheduling problem arises in each multi-manned station to assure the workload completion within the cycle time. The objective function of these problems is usually the minimisation of the number of workers as the primary target and the number of stations as a secondary objective. In this section, we discuss the use of decomposition for assembly lines with 2 sides (left and right), 5 sides (left, right, rear, front, inside), general multi-manned stations, and the collaboration of human and robotic workers. Besides the task-station assignment variable x_{si}, we extend the nomenclature to include task-worker (or task-side) variables a_{ir} for a set of workers $r \in R$, and worker-station variables b_{rs}. Note that we use a simplified notation for decision variables and cuts to solely focus on the decomposition core aspects.

We start the survey of this section with the simplest problem with respect to the station scheduling. The two-sided ALBP (TALBP) described in Huang et al. (2022) is modelled by deciding the task assignments in each station and side in the MP. In the SP, the

sequencing and scheduling of the tasks in each station are performed to check whether the assignment is feasible. Assume that XL^1_{sk} and XR^1_{sk} are two sets containing the tasks assigned to the left and right sides of a station in an infeasible solution. The trivial combinatorial cuts (52) can be used to exclude this solution from the MP.

$$\sum_{i \in XL^1_{sk}} x_{si} + \sum_{i \in XR^1_{sk}} x_{si} \leq |XL^1_{sk}| + |XR^1_{sk}| - 1, \quad \forall\ s \in S,\ k \in K. \tag{52}$$

These cuts can be further improved by finding the MIS or, at least, subsets XL^1_{sk} and XR^1_{sk} containing fewer elements for a stronger cut. Huang et al. (2022) propose four heuristic methods to reduce the number of variables accounted in the cut (improvement I-1). In their results, the use of these heuristic methods defines whether some instances with instance sizes of 65 and 148 are solved within the time limit of 7200 seconds or not.

A similar approach is considered by Naderi et al. (2019) in the balancing of an assembly line considering 5 sides. In this approach, the assignments of tasks and workers to stations (x_{si} and b_{rs}), as well as the task-worker allocations (a_{ir}), are decided in the MP. In the SP, the scheduling in each station is solved by sequencing the assigned tasks. The feasibility check is performed in two steps. First, the solutions of all three variables (x_{si}, a_{tr}, and b_{rs}) obtained in the MP are fixed in the SP. If the SP is proven infeasible, a cut prohibiting the assignments of all three variables can be defined. This cut contains several variables and is highly probably very weak. Therefore, Naderi et al. (2019) propose a second SP, which is solved by relaxing the task-worker assignments allowing a_{tr} to be decided in the SP. If the SP in the second phase is infeasible, the assignment of tasks (x_{si}) with the allocated amount of workers (b_{rs}) is not feasible independent of the internal division of work. Therefore, the cut (53) can be added to the MP.

$$\sum_{i \in X^1_{sk}} (1 - x_{si}) + \sum_{r \in R^0_{sk}} b_{rs} \geq 1, \quad \forall\ s \in S, k \in K. \tag{53}$$

Here, R^0_{sk} is the set of workers that are not assigned to station s in solution k. The constraint cuts off the solution by requiring that either one of the assigned tasks is removed or that one extra worker is assigned to the station in question. The authors also propose cuts (54) to inform the MP if the first SP is infeasible but there is a form to reassign tasks in a feasible solution of the second-phase SP.

$$\sum_{(i,r) \in TR^1_{sk}} a_{ir} \geq |X^1_{sk}| \left(1 - \sum_{r \in R^1_{sk}} (1 - b_{rs}) - \sum_{i \in X^1_{sk}} (1 - x_{si}) - \sum_{i \in X^0_{sk}} x_{si} \right), \quad \forall\ s \in S,\ k \in K. \tag{54}$$

These cuts (54) enforce a known feasible solution for the a_{ir} for a known task- and worker assignment to station s (represented in set TR^1_{sk}). This way, all other task-worker assignments are eliminated.

In the Multi-manned Assembly Line Problem (MALBP), multiple workers can be assigned to a station without the restriction of specific sides. In the decomposition structure of Michels et al. (2019), the MP is responsible for the assignments of tasks to stations

and the number of workers in each station. In the SP, the scheduling of the stations is solved to check the solution's feasibility. In Michels et al. (2019), the authors consider a spatial limit on the number of workers allowed in each station (2 or 4, depending on the instance). The derived Benders cuts for a balancing solution X^1_{sk} and R^1_{sk} depend on the number of workers assigned. If an SP is infeasible even when all workers are assigned, cuts (55) prohibit the assignment.

$$\sum_{i \in X^1_{sk}} x_{si} \leq |X^1_{sk}| - 1, \quad \forall\ s \in S,\ k \in K. \tag{55}$$

If there is still possible to assign more workers to the station, a weaker version of the cut is required. Cuts (56) exclude the task assignments in set X^1_{sk} if and only if the next worker $r^* + 1$ is not assigned to the station, where r^* is the number of workers assigned to the station s in solution k.

$$\sum_{i \in X^1_{sk}} x_{si} \leq |X^1_{sk}| - 1 + b_{(r^*+1)s}, \quad \forall\ s \in S,\ k \in K. \tag{56}$$

The version of the MALBP with the cycle time minimisation for a given number of workers is named MALBP Type-2 (Michels et al., 2020) and can also be solved using the BD. The cuts from Michels et al. (2019) are further extended to correct the cycle time variable based on the resulting cycle time of the SP.

There are also ALBP extensions considering heterogeneous workers at the same station. In special, three contributions using the BD apply the technique for lines using collaborative robots (Sikora and Weckenborg, 2023; Stecke and Mokhtarzadeh, 2022; Huang et al., 2024). Collaborative robots or cobots have sensors and/or cameras that allow them to work safely along and with humans at the same station. The related optimisation problem is then based not only on the assignment of tasks to stations but has to consider where to allocate the robots and how the tasks are performed.

In the BD from Sikora and Weckenborg (2023), not only one decomposition structure is proposed, but three ways of splitting the variables between MP and SP are tested (improvement I-5). In ALBP with cobots, tasks can be performed by one individual (either human or robotic) or in collaboration with a reduced processing time. In the first decomposition strategy from Sikora and Weckenborg (2023), robot and task-station assignments are selected in the MP, while the division of tasks between worker and cobot and their schedule is performed in the subproblem. Note that each station is manned by one human and a single robot can be allocated. Therefore, if an assignment is infeasible for a station with a cobot, the assignment is infeasible and a cut as Constraint (50) can be used. The second decomposition strategy delegates more decisions to the MP, which also selects whether the task is performed by a human or a robot. A possible combinatorial cut for this solution would cut off the combined task-station and task-worker assignment solution. Such a cut is, however, very weak since it cuts off a very specific combination of variables. For the second decomposition, the SP considers the assignment of tasks to workers also as a variable and the same cut as the first strategy is used. Such a strategy is also seen in Naderi et al. (2019). One could ask why variables are being modelled in both

stages of an algorithm. Do these variables not increase the solution time of both MP and SP? Indeed, more effort is required to solve the SPs. There is, however, an advantage to the MP: using the task-worker assignment variables improves the formulation of the MP. Without the knowledge of how tasks are assigned between the human and robot worker, Constraints (57) are used to limit the amount of processing time that can be assigned to a station, where r_s is the variable for the assignment of a cobot in station s.

$$\sum_{i \in N} t_i^* x_{si} \leq c + r_s c, \quad \forall\ s \in S. \tag{57}$$

The limit for the processing time is the cycle time c plus an extra cycle time if a robot is assigned. Note that this cut is a simplification of the one in Sikora and Weckenborg (2023) and t_i^* is an adapted task processing time required for the feasibility of the constraint. If, however, the assignment of tasks to workers is decided in the MP, it is possible to write the cycle time constraints as Ineqs. (58) and (59). In these expressions, the superscript/index h in t_i^h and a_{ht} refers to the human worker, r stands for the robotic worker, and l stands for the collaborative task mode that requires the assignment of both of them.

$$\sum_{i \in N} t_i^h a_{hi} + \sum_{i \in N} t_i^l a_{li} \leq c, \quad \forall\ s \in S, \tag{58}$$

$$\sum_{i \in N} t_i^r a_{ri} + \sum_{i \in N} t_i^l a_{li} \leq b_s c, \quad \forall\ s \in S. \tag{59}$$

The advantage of using Constraints (58) and (59) over Constraint (57) is that fewer assignments are feasible in the MP, potentially requiring fewer iterations of the BD. The third decomposition strategy is an intermediary of both: the variables a_{ri} are used in the MP but are relaxed as linear variables. This way, the more specific formulation of Constraints (58) and (59) is possible but the disadvantage of adding more binary variables to the MP is avoided.

Sikora and Weckenborg (2023) provide a long computational study to compare the efficiency of the three decomposition strategies. The results, however, cannot rule out which of the partition of variables performs best for all cases. The first strategy outperformed the others for small and medium instances, while the second strategy is best for the large instance of their dataset. More comparative tests are, therefore, required to devise the best form of structuring MP and SP in the BD.

An alternative variable splitting for the ALBP with cobots that is not explored in Sikora and Weckenborg (2023) is dealt with in Stecke and Mokhtarzadeh (2022). The authors of the latter assign all assignment variables to the MP along with the variables determining the sequence of tasks inside the station. This division of variables is different to all other contributions described in this section. For a fixed sequence of tasks, the SP of Stecke and Mokhtarzadeh (2022) is a simple linear program similar to a project scheduling problem. The linearity of the SP allows the use of the standard Benders cuts based on the shadow prices of the solution. The disadvantage of this method is that the MP is much harder to solve. Not only the assignments but also part of the scheduling are decided at the MP level, while the cut based on the SP brings little information to the MP. The results from

Stecke and Mokhtarzadeh (2022) show that a Constraint Programming (CP) approach and an MILP model often outperform their BD for small and medium instances. For large instances, the BD is, however, often the only method that finds a feasible solution. Finally, Huang et al. (2024) proposed an MILP model with enhancement techniques and tighter bounds, developed an improved BD algorithm with local search strategies, combinatorial cuts, and acceleration procedures, and verified the effectiveness of their approaches with computational experiments.

A last contribution is due to Şahin and Kellegöz (2023) and models the use of walking workers in multi-manned assembly lines. In this setting, workers can perform tasks in multiple stations moving between them as necessary. Such lines are typical of applications with long cycle times such as the production of buses or trucks. In their MP, the assignment of tasks to workers is solved. In the SP, the division of tasks to stations is addressed. For the workers whose tasks are assigned to multiple stations, their scheduling also has to be considered. If no feasible schedule is found, the cut (60) is used to eliminate the solution, where X^1_{rk} is the set of all tasks assigned to worker r in iteration k.

$$\sum_{r \in R} \sum_{i \in X^1_{rk}} a_{ir} \leq |N| - 1, \quad \forall\, k \in K. \tag{60}$$

As the feasibility of the assignment depends on the scheduling of multiple workers, the cut (60) must only prohibit the assignments as a whole. This results in a very weak cut. Therefore, Şahin and Kellegöz (2023) develop lower bounds and the integration with heuristic methods to find good solutions (improvement I-2). In their results, however, no instance in which the best solution found is larger than the initial lower bound is solved, showing the weakness of their cut.

4.4 ALBP with multiple products

When an assembly line is used for multiple products, the resulting cycle time does not depend on the assignment of tasks alone but also on the sequence in which products are assembled. One example can be seen in Fig. 1 in which each row represents the processing of a product in a station. If a sequence of products with long processing times appears (P3 and P4), production may be not possible. The violation illustrated in the figure could represent stoppage time or rework, which reduces the capacity of the assembly line.

The contributions in this section (Sikora, 2021; Saadi et al., 2022; Sarı et al., 2023) consider the production sequencing problem. In Sikora (2021), the demand is considered stochastic and modelled as a finite set of scenarios $q \in Q$ with probabilities p_q. The assignment of tasks must be decided at an early stage, representing the strategic decision of designing the assembly line. The short-turn decisions are the production sequencing, which represents the operational level of the production system. For each realization of the demand, the sequencing of the products can be individually optimised. The BD is used to separate balancing decisions (MP) and sequencing decisions (SP) for each demand scenario minimising a cost-related objective function here abstractly represented

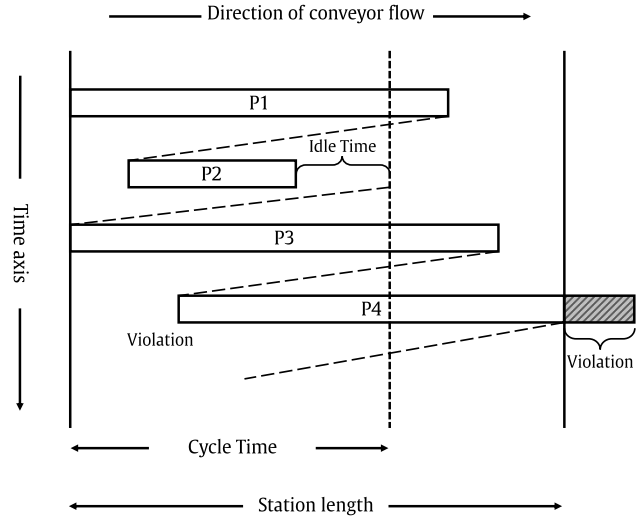

Figure 1: Scheduling example of a paced assembly line station – adapted from Sikora (2021).

by $Z = \sum_{q \in Q} p_q Z_q$ as the expected cost of the line, for which Z_q is the cost of a specific scenario q.

The combinatorial Benders cut resulting from the solution of a sequencing SP inform the MP about the realized production level for the demand scenario using a task assignment. As the production sequence must be solved for all stations simultaneously, a combinatorial cut in the form (61) corrects the objective value of the MP only for the tested complete assignment (all stations considered). The cut states that the part of the objective function Z_q related to a scenario q has to be larger or equal to the real value obtained in the SP. This is similar to the cut of Şahin and Kellegöz (2023), which is very weak since neighbouring solutions (the change of one task assignment in a single station, for instance) are not affected by the cut. To improve this, Sikora (2021) proposes solving the sequencing problem for a subset of stations. The result of this reduced problem is a lower bound for the complete problem and can be used in more general Benders cuts.

$$Z^q \geq \bar{Z}^q \left(\sum_{s \in S} \sum_{i \in X^1_{sqk}} x_{si} - |N| - 1 \right), \quad \forall\ q \in Q,\ k \in K, \tag{61}$$

$$Z^q \geq \bar{Z}^q \left(\sum_{s \in \bar{S}} \sum_{i \in X^1_{sqk}} x_{si} - \sum_{s \in \bar{S}} |X^1_{sqk}| - 1 \right), \quad \forall\ q \in Q,\ k \in K. \tag{62}$$

The work of Saadi et al. (2022) is here cited because they used the BD in an assembly line context with multiple products. The problem dealt with is, however, not an ALBP

and is restricted to only the sequencing of products. The sequence is defined in the MP, while the SP is a linear problem of scheduling products for a given sequence. Finally, Sarı et al. (2023) included sequence-dependent set-up times into a multi-model assembly line case study conducted in a garment industry and Li and Ji (2025) tackled a circular ALBP with task splitting. Nonetheless, as the cuts for both works are also based on the standard Benders cuts, the problems are not further discussed.

4.5 Uncertain processing times

In the previous subsections, the calculation of the cycle time depends on scheduling decisions. Here, no internal sequencing in the stations is required, but when faced with stochastic or uncertain processing times, the simple cycle time restriction also becomes inadequate. Merely summing up the deterministic processing times fails to capture the true nature of the problem, as the actual cycle time depends on the specific realisations of these processing times.

This subsection delves into the realm of stochastic or uncertain processing times in assembly line balancing problems. Two common scenarios are considered: (1) stochastic processing times, where the duration of tasks follows a certain distribution, and (2) interval processing times, where tasks have a range of possible durations $[a_i, b_i]$. In the latter case, the lower value a_i represents the regular processing time, while the higher value b_i occurs when a task requires additional time, often encountered in robust optimisation approaches.

Stochastic processing times are usually modelled as normally distributed (Boysen et al., 2008), meaning that there are infinitely many possible realizations for the processing time of each task following a continuous distribution. Such continuous distribution can, however, be approximated by a discrete set of possible realizations. A Sample Average Approximation (SAA) approach for stochastic processing times is presented in a disassembly line context by Bentaha et al. (2014), who considered a cost recourse in case the cycle time is not respected. Instead of working with a continuous distribution, the authors sample a set Q of realizations. For each realization $q \in Q$, the processing time of a task is defined as t_{iq}. If the set Q is large enough, the result of the SAA approach approximates the original distribution (Bentaha et al., 2014).

In the problem from Bentaha et al. (2014), the sum of the processing times in each station can be computed for each realisation of Q separately. A variable v_{sq} is defined as a compensation measurement or penalty cost if the cycle time cannot be observed in one of the sampled realisations and is modelled as Eq. (63).

$$\sum_{i \in N} t_{iq} x_{si} \leq c + v_{sq}, \quad \forall\ s \in S,\ q \in Q. \tag{63}$$

As the number of realizations can be large in order to efficiently approximate the original distribution ($|Q| = 500$ in the paper), the BD is used to separate these numerous constraints from the MP. For each solution of the MP, linear SPs minimising the penalty cost related to v_{sq} are solved, and their influence on the original objective function is

informed using a standard Benders cut based on the shadow prices. The problem is solved multiple times to account for the uncertainty generated in the sampling so that the method outputs a confidence interval for the results.

Although the standard Benders decomposition with linear SP is used, it is questionable whether the approach benefits from the decomposition structure. The subproblem (optimisation of variable v_{sq}) is merely a calculation: the r.h.s. of the expression is extended only if the realization requires more time than the cycle time. Therefore, it is questionable whether the division in MP and SP actually reduces the computational burden in the MP. This comparison (monolithic model vs. decomposition) is, however, not explored in the paper.

One extension of the SAA approach considering the disassembly of hazardous tasks along with the stochastic processing times is explored by Bentaha et al. (2018). Hazardous tasks are operations that can be dangerous or require safety measurements (for instance, dismounting the battery) and require larger investments for a station handling them.

The second stream of the literature on uncertain processing times focuses on robust optimisation using interval processing times. Hazir and Dolgui (2013, 2015) minimise the cycle time of an assembly line considering the fact that any Γ tasks from any station can assume the upper bound of its processing time. Consider for instance three tasks with processing times $i = [4, 5]$, $j = [3, 5]$. The cycle time of a station containing these three tasks is 7-time units if the lower bounds of the processing times are considered. If $\Gamma = 1$ task can assume a higher value, the resulting cycle time is 9 ($4 + 5$, as j is the worst case).

The approach proposed by Hazir and Dolgui (2013) consists in solving the ALBP in the MP and calculating the cycle time for the worst-case scenario of each station in the SPs. The corresponding SP is a knapsack problem in which tasks must be selected. Each task has "size" of 1 unit and the capacity of Γ. Due to the structure of the problem, the SP can be solved as an LP even though the variables are binary. The authors then use the shadow prices to form standard Benders cuts. In Hazir and Dolgui (2015), the problem is extended to consider U-shaped lines.

4.6 Uncertain demand

Besides the already described contribution by Sikora (2021) (Subsec. 4.4), demand uncertainty is also the focus of a paper on disassembly lines (He et al., 2022). Dealing with used products introduces more uncertainty since the quality of the internal pieces of the product is unknown before dismantling it. He et al. (2022) consider a stochastic yield factor from the used products, which determines how many of their components can be reused in a refurbished product. This yield factor as well as the demand for the dismantled pieces is considered random and is modelled using a finite set of scenarios $q \in Q$. The MP of their approach consists of designing the disassembly line and the selection of machinery required. In the SP, the operative results of the line are computed for every scenario. The resulting SP computes the amount of pieces disassembled and reused, and the inventory between periods of time, which is modelled as a linear problem. Even though if the SP is an LP, its solution is quite simple: pieces are produced according to

the yield factor and if the production is greater than the demand, the pieces are stored for the next periods. The shadow prices of this LP are used to generate standard Benders cuts for the MP.

He et al. (2022) also consider a Sample Approximation Approach (SAA) in which the set of scenarios Q is approximated by $\bar{Q} \subset Q$. For the approximation, no decomposition is used for solving the MP along with the corresponding SPs in a single optimisation problem. Their results show that no significant difference can be observed in the computing time for small and medium instances and the advantage of the Benders decomposition is only observable for the instances with 48 and 64 machines.

5 Summary and discussion

Figure 2 depicts the number of applications of BD algorithms to ALBPs over this decade after the last survey on the general topic (Rahmaniani et al., 2017). It also distinguishes each contribution in terms of trending classifications discussed in Boysen et al. (2022), showing the subject's relevance in the newest ALBP research fronts. Complementary, Tables 1 and 2 consolidate a summary of all the works that applied BD approaches to ALBP extensions. Table 1 brings a clustered subdivision of the papers based on Boysen et al. (2022)'s classification, a short description on the studied problem, the instance sizes that were treated, and their respective improvement categories (see Subsection 3.1), while Table 2 exhibits components of the four-dimension taxonomy proposed by Rahmaniani et al. (2017), i.e. decomposition strategy, solution generation, optimality/feasibility cut generation (Improved, Classical, or CBC), and MP/SP solution procedure (Advanced or Standard) captured features of each study.

Table 1 evidences how, thanks to the advance of computational power and progress of commercial solvers, numerous new practical applications and more complex combinatorial ALBPs are being formulated and solved with the assistance of BD approaches (Boysen et al., 2022), namely extensions with sequence-dependent task time increments (SU-ALBP), flexible and parallel workplaces (MALBP, TALBP, and cobots), non-deterministic processing times (robust optimisation and DLBP), and stochastic demand (mixed-models and DLBP). Most commonly in these ALBPs, the idea is obtaining the least required number of stations to abide by a given cycle time or the minimum cycle time given a fixed number of stations, all while respecting a set of partial precedence relations among and within these stations, which is, in turn, manifested in the form of scheduling expressions. Thus, formulations for ALBP extensions usually contain a set of binary variables associated with the assignment of tasks to stations, and additional variable sets associated with scheduling requirements. This structure offers a natural framework for the BD approach, which consists of isolating the task-station assignment variables in the MP and relegating scheduling decisions to the station-wise SPs. Indeed, in most surveyed papers, this relative simplicity in solving each SP made the BD an appropriate approach to tackle some GALBP variants – a hypothesis validated by extensive benchmark testing on instance sizes up to 297 tasks. Nonetheless, efficient solution methodologies obtained by combining BD with previous or different techniques is still missing (Magnanti and Wong, 1981; Costa, 2005), even though a rich variety of successful hybrid approaches

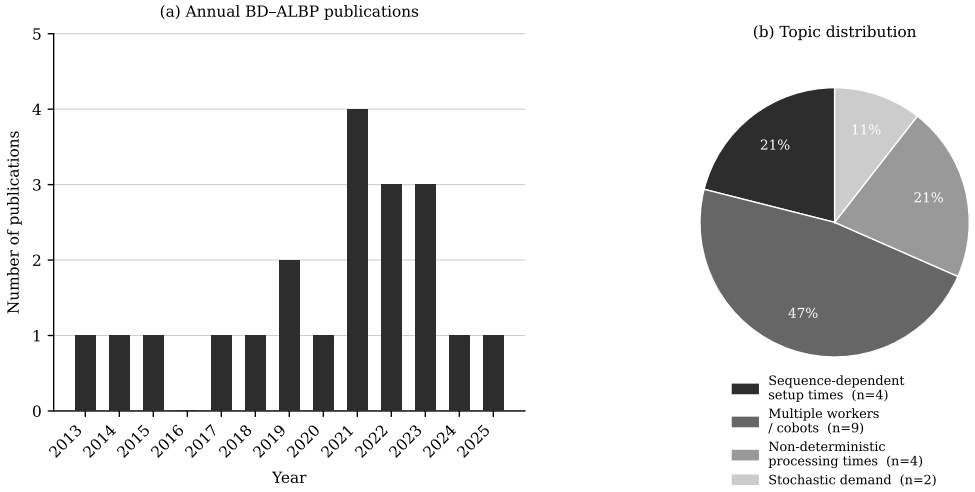

Figure 2: Annual number of publications of BDs applied to ALBPs and their classification count.

(I-7) is available in the literature for other problems (Rahmaniani et al., 2017).

Correspondingly, Table 2 verifies how there is a growing interest in BD optimisation approaches that include logic-based relations introduced by Hooker and Ottosson (2003). As explained in Section 3, the decomposition strategy known as LBBD is similar to the classical BD method, with a major advantage of having feasibility-checking SPs. Moreover, these SPs can be fathomed by employing either commercial solvers for MILP/CP formulations or specialised algorithms. Each of these SPs is able to impose a tighter bound on the global objective function by passing CBCs back to the MP. During this iterative process, improving the initial MP solution generation with VIs or heuristics is crucial for achieving a faster convergence. Nevertheless, since there is no standardised template to produce CBCs in a LBBD approach, coming up with a cut generation method is not always a bed of roses. Typically, based on the problem's knowledge and structure, these cuts are tailored to the GALBP at hand, with classical/improved (C and I) feasibility/optimality cuts or CBCs. According to Rahmaniani et al. (2017), there are simple cuts for some problems, but we must balance their effectiveness with obtainment easiness. Throughout this survey, we showed how the LBBD method had also been applied to a wide range of ALBPs and can outperform by orders of magnitude commercial solvers running mathematical formulations. Lastly, the solution procedure for MP and SP plays an essential role in a BD approach, which might bring advanced (A) or standard (S) features. In the case of GALBPs, the SP can usually be further divided into several independent SPs (one for each station), which allows for concurrent optimisation in parallel. Moreover, we can easily improve the convergence rate by implementing cuts as lazy constraints in a single search tree framework, as demonstrated by the majority of surveyed papers.

Table 1: Summary of GALBP classifications applying a Benders decomposition approach. For each reference, there is a description of particularities solved by the authors, the size of the problem in terms of number of tasks, and the used improvements listed in Section 3.1.

Classification	Reference	Description	Size	Improvements
Sequence-dependent task time increments	Akpinar et al. (2017)	SUALBP-1, multiple products	$N \leq 70$	6
	Zohali et al. (2022)	SUALBP-2	$N \leq 111$	1, 3, 5, 6
	Furugi (2022)	SUALBP-1, cost-oriented	$N \leq 53$	-
	Sarı et al. (2023)	SUALBP-1, multi-model	$N \leq 100$	-
Flexible and parallel workplaces	Naderi et al. (2019)	MALBP-1, 5-sided, multiple products	$N \leq 30$	1, 3, 5, 6
	Michels et al. (2019)	MALBP-1	$N \leq 148$	6
	Michels et al. (2020)	MALBP-2	$N \leq 83$	1, 2, 6
	Huang et al. (2022)	TALBP-1, multiple products	$N \leq 205$	1, 6
	Stecke and Mokhtarzadeh (2022)	Cobots, ergonomic risks	$N \leq 89$	1
	Sikora (2022)	Cobots	$N \leq 100$	1, 3, 5, 6
	Şahin and Kellegöz (2023)	MALBP-1, walking workers	$N \leq 89$	2, 6
	Huang et al. (2024)	Cobots	$N \leq 100$	1, 2, 6
	Li and Ji (2025)	Cobots, task-splitting	$N \leq 104$	1, 2, 6
Non-deterministic processing times	Hazir and Dolgui (2013)	Robust SALBP-2, interval processing times	$N \leq 70$	3, 4, 6
	Bentaha et al. (2014)	DLBP-1, penalty cost	$N \leq 79$	4
	Hazir and Dolgui (2015)	Robust UALBP-2, interval processing times	$N \leq 75$	3, 4, 6
	Bentaha et al. (2018)	DLBP-1, penalty cost, hazardous tasks	$N \leq 37$	4
Stochastic demand	Sikora (2021)	Multiple products, model-sequencing	$N \leq 297$	1, 6
	He et al. (2022)	DLBP-1, resource-constrained	$N \leq 60$	4, 6

The reviewed literature not only apply BD methods for GALBP variants, but the proposed decomposition structures are also extensively varied. The nature of the SP and the kind of cut used largely depends on the problem's nature and on decisions of the decomposition designer. While linear SPs allow the use of efficient Benders cuts based on the dual values, the amount of meaningful linear problems to be combined with GALBPs is limited. On the other hand, SPs containing binary or integer variables cover a much broader range of relevant GALBPs. These problems, however, require the use of CBCs in a LBBD framework, whose efficiency is highly dependant on the problem at hand, customised cuts, and implementation details on state-of-the-art solvers. From a managerial perspective, this survey yields a valuable insight into resource allocation decisions, indicating there is a trade-off between investing in state-of-the-art commercial solvers for solving real-world large scale problems and developing hyper-specific dedicated algorithms.

Ultimately, the question of how to systematically split the original problem in a MP and one or more SPS was partially answered throughout this literature review. Taking the MALBP as an example, there are publications with integer SPs and others containing only linear variables. For instance, Michels et al. (2019) divide the variables in a way that the scheduling of tasks (an MILP problem) is solved in multiple station-wise SPs, requiring the use of CBCs. On a similar problem considering cobots, Stecke and

Table 2: Summary of Benders decomposition characteristics when applied to GALBPs. For each reference, the decomposition strategy, solution/cut generation, and solution procedure for MP/SP are informed.

Reference	Decomposition strategy	Solution generation	Cut generation	Solution procedure
Hazir and Dolgui (2013)	Classical BD	Improved MP (VIs), Approximated MP solution	C/C for base problem; I/I for variation	A/S - Approximated MP solution
Bentaha et al. (2014)	Classical BD	Regular MP	C/C	S/S
Hazir and Dolgui (2015)	Classical BD	Improved MP (VIs), Approximated MP solution	C/C	A/S - Approximated MP solution
Akpinar et al. (2017)	Classical LBBD	Improved MP (VIs)	Standard CBC (1 type)	A/S - Single search tree
Bentaha et al. (2018)	Classical BD	Regular MP	C/C	A/S - Two-phase MP
Naderi et al. (2019)	Modified LBBD	Improved MP (VIs), Heuristics (solution improvement)	Improved and Standard CBC (1 and 2 types)	A/A - Single search tree / Two-phase SP
Michels et al. (2019)	Classical LBBD	Improved MP (VIs)	Standard CBC (2 types)	A/S - Single search tree
Michels et al. (2020)	Classical LBBD	Improved MP (VIs), Heuristics (warm start)	Standard CBC (4 types)	A/S - Single search tree
He et al. (2022)	Classical BD (L-shaped)	Improved MP (VIs)	C/-	S/S
Huang et al. (2022)	Classical LBBD	Regular MP	Improved CBC (1 type)	A/S - Single search tree
Stecke and Mokhtarzadeh (2022)	Classical BD	Regular MP	I/C - Degeneracy	A/S - Single search tree
Sikora (2021)	Classical LBBD	Improved MP (VIs)	Improved and Standard CBC (1 and 1 type); Linear BC (1 Type)	A/A - Single tree search / Parallel SP
Sikora (2022)	Modified LBBD	Improved MP (VIs), Heuristics (solution improvement)	Improved and Standard CBC (1 and 1 type)	A/A - Single tree search, Heuristics / Parallel SP
Zohali et al. (2022)	Modified LBBD	Improved MP (VIs)	Improved and Standard CBC (1 and 2 types)	A/S - Single search tree, Two-phase MP
Furugi (2022)	Classical LBBD	Regular MP	Standard CBC (1 type)	A/S - Single search tree
Şahin and Kellegöz (2023)	Classical LBBD	Improved MP (VIs), Heuristics (warm start)	Standard CBC (1 type)	A/S - Single search tree
Sarı et al. (2023)	Classical LBBD	Improved MP (VIs)	Standard CBC (1 type)	S/S - Single search tree
Huang et al. (2024)	Classical LBBD	Improved MP (VIs)	Improved and Standard CBC (1 and 1 type)	A/S - Local search
Li and Ji (2025)	Classical LBBD	Improved MP (VIs), Heuristics	Standard CBC (4 types)	A/A - Local branching

Mokhtarzadeh (2022) model all sequence-related variables in the MP. The emerging SP is reduced to a linear scheduling problem for a given sequence. The linear cuts can be employed in the MP, which nonetheless have many more difficult variables to deal with, resulting in an unclear trade-off. The literature presents multiple examples of decomposition strategies, however, until now, no publication on the ALBP field explores and compares decomposition arrangements in a straightforward manner.

6 Conclusion

This comprehensive survey underscored the potential benefits of addressing GALBPs with classical and logic-based BD approaches, shedding light on the efficiency gains of applying BDs to this class of combinatorial problems. In addition, we have critically reviewed and analysed the literature, introducing the particular aspects and ideas of each article and providing a summary and discussions on existing gaps. Despite BD's success in other fields (Costa, 2005; Rahmaniani et al., 2017), we acknowledged that its application to GALBP variations was primarily overlooked until the last decade. Nevertheless, as demonstrated in this survey, such tendency is gradually shifting due to the increasing number of researchers adopting this technique.

We also offered a critique of the LBBD solution procedure, noting that while it can provide improved feasible primal bounds, the proof of optimality almost entirely depends on the solver's performance and how it branches the MP. This observation suggests there is room for improvement in applying BD formulations to GALBPs. Finally, from a managerial perspective, we present valuable insights into trade-offs between developing more efficient BD methods for combinatorial problems (e.g. GALBPs) and obtaining better results using state-of-the-art commercial solvers (e.g. Gurobi, CPLEX).

We note that reproducibility across the surveyed literature is limited, as most authors do not publicly share implementation code or solver configurations beyond the use of standard benchmark instances; future contributions to the field would benefit from open-source implementations to facilitate direct comparison.

Looking forward, we suggest promising future research avenues, such as exploring problem classes where BD has not been applied yet. This proposition also highlights the need for decomposing more combined problems while potentially sacrificing optimality proofs with hybrid BD approaches (Subsection 3.1.7). Although challenging to implement computationally, these research directions may yield exciting results and further advance the field on this continuing trend.

FUNDING. This research was partially funded by the Australian Government through the Australian Research Council Industrial Transformation Training Centre in Optimisation Technologies, Integrated Methodologies, and Applications (OPTIMA), Project ID IC200100009.

DATA AVAILABILITY. This research was based on the literature and no data was generated.

AUTHORS' CONTRIBUTIONS. Adalberto Sato Michels: Conceptualization, Data Curation, Investigation, Methodology, Writing – Original Draft Preparation, Writing – Review & Editing. **Alysson M. Costa:** Conceptualization, Writing – Review & Editing. **Celso Gustavo Stall Sikora:** Conceptualization, Data Curation, Investigation, Methodology, Writing – Original Draft Preparation, Writing – Review & Editing.

References

Akpinar, S., Elmi, A., and Bektaş, T. (2017). Combinatorial Benders cuts for assembly line balancing problems with setups. *European Journal of Operational Research*, 259(2):527–537.

Battaïa, O. and Dolgui, A. (2022). Hybridizations in line balancing problems: A comprehensive review on new trends and formulations. *International Journal of Production Economics*, page 108673.

Baybars, I. (1986). Survey of Exact Algorithms for the Simple Assembly Line Balancing Problem. *Management Science*, 32(8):909–932.

Becker, C. and Scholl, A. (2006). A survey on problems and methods in generalized assembly line balancing. *European Journal of Operational Research*, 168(3):694–715.

Benders, J. F. (1962). Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4(1):238–252.

Bentaha, M. L., Battaïa, O., and Dolgui, A. (2014). A sample average approximation method for disassembly line balancing problem under uncertainty. *Computers and Operations Research*, 51:111–122.

Bentaha, M. L., Dolgui, A., Battaïa, O., Riggs, R. J., and Hu, J. (2018). Profit-oriented partial disassembly line design: dealing with hazardous parts and task processing times uncertainty. *International Journal of Production Research*, 56(24):7220–7242.

Boysen, N., Fliedner, M., and Scholl, A. (2007). A classification of assembly line balancing problems. *European Journal of Operational Research*, 183(2):674–693.

Boysen, N., Fliedner, M., and Scholl, A. (2008). Assembly line balancing: Which model to use when? *International Journal of Production Economics*, 111(2):509–528.

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

Codato, G. and Fischetti, M. (2006). Combinatorial benders' cuts for mixed-integer linear programming. *Operations Research*, 54(4):756–766.

Costa, A. M. (2005). A survey on benders decomposition applied to fixed-charge network design problems. *Computers and Operations Research*, 32(6):1429–1450.

Furugi, A. (2022). Sequence-dependent time- and cost-oriented assembly line balancing problems: a combinatorial Benders' decomposition approach. *Engineering Optimization*, 54(1):170–184.

Gendron, B., Lucena, A., da Cunha, A. S., and Simonetti, L. (2014). Benders Decomposition, Branch-and-Cut, and Hybrid Algorithms for the Minimum Connected Dominating Set Problem. *INFORMS Journal on Computing*, 26(4):645–657.

Geoffrion, A. M. (1972). Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10(4):237–260.

Gungor, A. and Gupta, S. M. (2001). A solution approach to the disassembly line balancing problem in the presence of task failures. *International Journal of Production Research*, 39(7):1427–1467.

Hazir, Ö. and Dolgui, A. (2013). Assembly line balancing under uncertainty: Robust optimization models and exact solution method. *Computers and Industrial Engineering*, 65(2):261–267.

Hazir, Ö. and Dolgui, A. (2015). A decomposition based solution algorithm for U-type assembly line balancing with interval data. *Computers and Operations Research*, 59:126–131.

He, J., Chu, F., Dolgui, A., Zheng, F., and Liu, M. (2022). Integrated stochastic disassembly line balancing and planning problem with machine specificity. *International Journal of Production Research*, 60(5):1688–1708.

Hooker, J. N. and Ottosson, G. (2003). Logic-based Benders decomposition. *Mathematical Programming, Series B*, 96(1):33–60.

Huang, D., Mao, Z., amd Enyuan Fu, K. F., and Pinedo, M. L. (2024). An Improved Combinatorial Benders Decomposition Algorithm for the Human-Robot Collaborative Assembly Line Balancing Problem. *INFORMS Journal on Computing*, 0(0).

Huang, D., Mao, Z., Fang, K., and Yuan, B. (2022). Combinatorial benders decomposition for mixed-model two-sided assembly line balancing problem. *International Journal of Production Research*, 60(8):2598–2624.

Li, P. and Ji, C. (2025). Application of enhanced benders decomposition algorithm in circular assembly line balancing problem with task splitting. *PLOS ONE*, 20(10):1–32.

Magnanti, T. L. and Wong, R. T. (1981). Accelerating Benders Decomposition: Algorithmic Enhancement and Model Selection Criteria. *Operations Research*, 29(3):464–484.

Michels, A., Lopes, T., Sikora, C., and Magatão, L. (2019). A Benders' decomposition algorithm with combinatorial cuts for the multi-manned assembly line balancing problem. *European Journal of Operational Research*, 278(3):796–808.

Michels, A. S., Lopes, T. C., and Magatão, L. (2020). An exact method with decomposition techniques and combinatorial Benders' cuts for the type-2 multi-manned assembly line balancing problem. *Operations Research Perspectives*, 7:100163.

Michels, A. S., Lopes, T. C., Sikora, C. G. S., and Magatão, L. (2018). The Robotic Assembly Line Design (RALD) problem: Model and case studies with practical extensions. *Computers and Industrial Engineering*, 120:320–333.

Michels, A. S. and Sikora, C. G. S. (2022). A survey on Benders Decomposition methods applied to Assembly Line Balancing Problems. *IFAC-PapersOnLine*, 55(10):464–469.

Naderi, B., Azab, A., and Borooshan, K. (2019). A realistic multi-manned five-sided mixed-model assembly line balancing and scheduling problem with moving workers and limited workspace. *International Journal of Production Research*, 57(3):643–661.

Rahmaniani, R., Crainic, T. G., Gendreau, M., and Rei, W. (2017). The Benders decomposition algorithm: A literature review. *European Journal of Operational Research*, 259(3):801–817.

Saadi, A., Fattahi, P., Rahmanii, D., Hosseini, S. M. H., and Sana, S. S. (2022). An exact method for job sequencing in a mixed-model assembly line considering feeding lines based on the benders decomposition approach. *International Journal of Management Science and Engineering Management*, 17(2):79–92.

Salveson, M. E. (1955). The assembly line balancing problem. *The Journal of Industrial Engineering*, 6:18–25.

Sarı, E., Paldrak, M., Can, Y., Kuzu, T., Ceylan, S. D., Duran, D., Örnek, M. A., and Yıldız, O. C. (2023). Intelligent Benders' Decomposition Algorithm for Sequencing Multi-model Assembly Line with Sequence Dependent Setup Times Problem: A Case Study in a Garment Industry. *Intelligent and Fuzzy Systems*, 759:659–668.

Scholl, A. and Becker, C. (2006). State-of-the-art exact and heuristic solution procedures for simple assembly line balancing. *European Journal of Operational Research*, 168(3):666–693.

Sikora, C. G. S. (2021). Benders' decomposition for the balancing of assembly lines with stochastic demand. *European Journal of Operational Research*, 292(1):108–124.

Sikora, C. G. S. (2022). Balancing Under Full Sequencing Control. In *Assembly-Line Balancing under Demand Uncertainty*, pages 65–95. Springer Gabler, Wiesbaden.

Sikora, C. G. S. and Weckenborg, C. (2023). Balancing of assembly lines with collaborative robots: comparing approaches of the benders' decomposition algorithm. *International Journal of Production Research*, 61(15):5117–5133.

Stecke, K. E. and Mokhtarzadeh, M. (2022). Balancing collaborative human-robot assembly lines to optimise cycle time and ergonomic risk. *International Journal of Production Research*, 60(1):25–47.

Zohali, H., Naderi, B., and Roshanaei, V. (2022). Solving the type-2 assembly line balancing with setups using logic-based benders decomposition. *INFORMS Journal on Computing*, 34(1):315–332.

Şahin, M. and Kellegöz, T. (2023). Benders' decomposition based exact solution method for multi-manned assembly line balancing problem with walking workers. *Annals of Operations Research*, 321(1):507–540.