Balancing of assembly lines with collaborative robots: comparing approaches of the Benders' decomposition algorithm

Celso Gustavo Stall Sikora[a*], **Christian Weckenborg**[b]

[a] *Institute for Operations Research,*
University of Hamburg
Moorweidenstraße 18, 20148 Hamburg, Germany
Tel: +49 40 42838-5518
e-mail: celso.sikora@uni-hamburg.de

[b] *Institute of Automotive Management and Industrial Production,*
Technische Universität Braunschweig,
Mühlenpfordtstr. 23, 38118 Braunschweig, Germany
Tel: +49 531 391-2207
e-mail: c.weckenborg@tu-braunschweig.de

[*] *Corresponding author*

Abstract: In recent years, human workers in manual assembly lines are increasingly being supported by the deployment of complementary technology. Collaborative robots (or cobots) represent a low-threshold opportunity for partial automation and are increasingly being utilized by manufacturing corporations. As collaborative robots can be used to either conduct tasks in parallel to the human worker or collaborate with the worker on an identic task, industrial planners experience an increasingly complex environment of assembly line balancing. This contribution proposes three different decomposition approaches for Benders' decomposition algorithms exploring the multiple possible partitions of the formulation variables. We evaluate the performance of the algorithms by conducting extensive computational experiments using test instances from literature and compare the findings with results generated by a commercial solver and a metaheuristic solution procedure. The results demonstrate the Benders' decomposition algorithms' efficiency of finding exact solutions even for large instances, outperforming the benchmark procedures in computational effort and solution quality.

Keywords: assembly line balancing, collaborative robots, cobots, Benders' decomposition, collaboration

This is an original manuscript of an article published by Taylor & Francis in International Journal of Production Research on July 7th 2022, available at: https://www.tandfonline.com/doi/full/10.1080/00207543.2022.2093684

Declarations of interest: none

ORCID iD of Celso Gustavo Stall Sikora: https://orcid.org/0000-0001-9180-1206
ORCID iD of Christian Weckenborg: https://orcid.org/0000-0003-3598-4508

Introduction

In the past centuries, the role of automation in industrial manufacture consistently increased. Consequently, automated assembly lines utilizing industrial robots dominate in the manufacturing of standardized products in high volumes and at low costs. In the current state, however, manufacturers cannot efficiently automate assembly tasks requiring a high degree of flexibility. To this end, they rely on manual assembly lines utilizing the advantages of human workers.

In the past years, new technologies with the capability to complement human workers in manual assembly lines emerged. In this field, collaborative robots are discussed as a particularly promising technology (Calzavara et al. 2020; BMAS/BAuA 2018; Fraunhofer IAO 2016). Collaborative robots (cobots) are characterized by inherent security mechanisms eliminating the necessity to install external safety devices, e.g., physical barriers. Therefore, cobots and workers can share common stations yielding the advantages of both manual and automated assembly at low costs (Antonelli, Astanin, and Bruno 2016; Krüger, Lien, and Verl 2009). This human-robot collaboration is characterized by human and cobot sharing the same working place and time, where the required assembly tasks can be processed by either the human worker or the cobot individually, or in collaboration of both (Chen et al. 2011; Krüger, Lien, and Verl 2009; Helms, Schraft, and Hägele 2002). Additionally, cobots are assumed to be positioned and set up quickly, which is beneficial in cases of their frequent redeployment in changing production environments (Bosch 2021; Universal Robots 2021; Hashemi-Petroodi et al. 2020a). Consequently, manufacturing corporations already successfully implemented this technology in a broad range of applications (IFR 2018; Brigl 2017; Volkswagen 2017; Schillmoeller 2013). An overview of applications is given by the commercial cobot provider Robotiq (2021).

To effectively utilize this technology, it has to be considered already in the phase of assembly line balancing. An assembly line consists of $k, l \in K = \{1, \dots, |K|\}$ stations which are connected by a material handling device. The overall work required to assemble a product is split into individual tasks $i, j \in I = \{1, \dots, |I|\}$ which are characterized by their processing times t_i. The basic version of assembly line balancing is referred to as the simple assembly line balancing problem (SALBP). In SALBP, tasks need to be allocated among the stations such that precedence relations $(i, j) \in E$ existing between tasks are considered and the cycle time c of the line is not exceeded. Several objectives can be pursued within assembly line balancing, e.g., minimizing the cycle time for a given number of stations, or vice versa. The reader finds an extensive introduction to assembly line balancing in Scholl (1999). The availability of cobots extends SALBP by three additional characteristics. First, the line planner has to decide about the allocation of cobots to stations. While human workers are assumed to be available in each station, cobots represent additional resources to complement the worker in selected stations. SALBP is therefore extended by an *equipment selection problem*. Second, the problem expands toward a *scheduling problem* for stations with both a human worker and a cobot as they can conduct different tasks simultaneously. To ensure compliance with the precedence relations also within the stations, the start and end times of the tasks within the stations have to be considered. Third, the human worker and the cobot can collaborate on the identic task which requires both resources to be available at the respective station and at the required points in time. Therefore, the *optional collaboration of resources* has to be considered as a separate mode alternative to the individual task execution by a human worker or a cobot.

The firsts to address the described problem are Weckenborg et al. (2020). In their contribution, they motivate the problem and identify the characteristics relevant to the problem. They propose a mixed-integer programming formulation and develop a genetic algorithm to solve the problem. In their study, they find that the cycle time can be reduced substantially by the

effective deployment of cobots. Additionally, they identify the optional collaboration of worker and cobot on a task as a driver of the improvements which is made frequent use of. From a computational perspective, however, they were not able to prove the optimality of the generated results for the majority of instances they consider.

For the described problem, decomposition-based solution approaches may be particularly suitable to exploit the structure of the problem consisting of decisions relating to the line and decisions relating to individual stations. In the contribution at hand, we develop a Benders' decomposition algorithm with three different decomposition approaches. We evaluate our approaches using the original test set of Weckenborg et al. (2020). The solution approaches developed in the contribution at hand outperform the previous approaches in both solution quality and computational effort.

To this end, the remainder of the contribution at hand is structured as follows. In Section 2, we provide a review of related literature. Subsequently, we introduce the problem setting we consider in Section 3. The developed Benders' decomposition algorithm and the associated decomposition approaches are presented in Section 4. In Section 5, we evaluate the performance of the different decomposition approaches. The article concludes in Section 6.

1. Literature review

In this section, we review literature related to the characteristics of the problem we consider (Section 2.1.) and discuss previous Benders' decomposition approaches applied on assembly line balancing problems (Section 2.2). Within the past decades, a rich body of research was published on assembly line balancing problems. For a general overview of scientific articles on this topic, please refer to the reviews of Becker and Scholl (2006), Scholl and Becker (2006), Boysen, Fliedner, and Scholl (2008), and Battaïa and Dolgui (2013).

1.1. Assembly line balancing with collaborative robots

Contributions with characteristics relevant to the problem under consideration have to consider the *equipment selection problem*, *scheduling problem*, and *optional collaboration of resources* to cover the availability of cobots in assembly lines. The equipment selection problem is frequently considered whenever alternative resources can be selected for the execution of tasks. Frequently, this appears in approaches on the robotic assembly line balancing problem (RALBP) in the case of automated assembly lines and the assembly line worker assignment and balancing problem (ALWABP) in the case of manual assembly lines. The scheduling problem arises if more than one resource can be assigned to a station at the same time. Therefore, we are particularly concerned with two-sided and multi-manned approaches in RALBP and ALWABP. As more than one resource is required for an optional collaboration of resources, this characteristic is also to be expected in those approaches or in approaches explicitly addressing collaboration in assembly line balancing.

In RALBP, the balancing of automated assembly lines is considered, in which heterogeneous robots can be allocated among the stations of the line and alternatively conduct assembly tasks (Rubinovitz and Bukchin 1991; Rubinovitz, Bukchin, and Lenz 1993). A recent review on RALBP is provided by Chutima (2020). In the majority of contributions, however, only a single robot can be assigned to each station. The first approach to RALBP with more than one robot per station is proposed by Aghajani, Ghodsi, and Javadi (2014). In their contribution, the authors consider the scheduling problem arising in stations with two robots. Lopes et al. (2017) and Michels et al. (2018) address problems arising in robotic spot welding in the automotive industry, in which multiple robots may simultaneously perform welding points at the same car body. However, they assume no or simplified precedence relations to be applicable within the stations and therefore do not consider the related scheduling problem. Neither of the aforementioned approaches considers the optional collaboration of resources.

Approaches to ALWABP are motivated by the inclusion of disabled workers in sheltered work centres, in which heterogeneous workers with limited capabilities are allocated among the stations of the assembly line (Miralles et al. 2007; Miralles et al. 2008). As in RALBP, the majority of contributions assume the number of workers to be limited to one per station. Araújo, Costa, and Miralles (2012) overcome this limitation and allow for the allocation of more than one worker to the stations to promote the allocation of workers with limited capabilities. Janardhanan, Li, and Nielsen (2019) assume two workers per station. Weckenborg and Spengler (2019) and Weckenborg (2021) assume a setting, in which multiple heterogeneous workers and cobots can be allocated in each station. Further contributions address human-robot collaboration and can therefore be accounted to the RALBP or ALWABP literature streams (e.g., Çil et al. (2020), Rabbani, Behbahan, and Farrokhi-Asl (2020), Koltai et al. (2021)). Promising extensions incorporate further characteristics of the used resources into their evaluation, e.g., variable task times of human workers (Sotskov et al. 2015; Lai et al. 2016). Among the aforementioned approaches, however, neither considers the optional collaboration of resources.

In a recent survey, Hashemi-Petroodi et al. (2020b) emphasize the collaboration between humans and cobots as an important characteristic of future manufacturing systems. In their outlook, the authors propose the definition of different modes for the execution of tasks by human workers, cobots, or in collaboration of both as a promising chance to include dual resource-constrained considerations into the design of hybrid workstations. In assembly line balancing, this characteristic is known as a *common task*, which enforces multiple resources to be available for its execution. In these cases, the collaboration is given as an external assignment restriction and therefore remains mandatory. Few approaches in the assembly line balancing literature consider this characteristic (Yazgan et al. 2011; Sikora, Lopes, and Magatão 2017). Further articles consider a mode which they interpret as the collaboration of multiple

resources on an identic task (Samouei and Ashayeri 2019; Li, Janardhanan, and Tang 2021; Dalle Mura and Dini 2019; Yaphiar, Nugraha, and Ma'ruf 2020). These contributions, mainly motivated by human-robot collaboration, however, neglect that resources can work in parallel when not occupied with a common task. Therefore, they assume the respective other resources to be idle while one resource executes a task individually. Stecke and Mokhtarzadeh (2022) adapt the approach of Weckenborg et al. (2020) and extend it to cover mobile robots and ergonomics.

1.2. Benders' decomposition approaches to assembly line balancing

The assembly line balancing literature has recently received several contributions using some form of Benders' or combinatorial Benders' decomposition (Sikora 2021). The success of such decomposition lays in the separate solution of the allocation variables and the correspondent feasibility problem caused by such an assignment. Akpinar, Elmi, and Bektaş (2017) solve the assembly line balancing problem with set-up times between tasks. The authors divide the problem into a first stage, in which the assignment of tasks to stations is determined, and a second stage, which is responsible for checking the solution feasibility. At the second stage, the scheduling of tasks is solved individually for each station. The information on the feasibility of a solution can be transferred to the first stage by using combinatorial cuts. Zohali, Naderi, and Roshanaei (2021) extend the approach to the type-2 version (cycle time minimization) of the assembly balancing problem with set-up times. A version of the set-up problem considering different costs for worker abilities is explored in Furugi (2022). Another application of the Benders considers multiple workers in the workstations. Huang et al. (2021) consider two-sided assembly lines, while Naderi, Azab, and Borooshan (2019) solve the problem for five-sided ALBPs. A version with multi-manned stations without side assignments can be found in Michels et al. (2019) and Michels, Lopes, and Magatão (2020). A final refer-

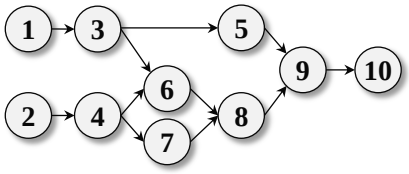
ence on the literature deals with human and robotic allocation and ergonomics (Stecke and Mokhtarzadeh 2022).

Most of the references on Benders' decomposition applied to variations of the assembly line balancing problem propose only one decomposition scheme: variables deciding the assignment of the tasks to stations at the master problem and the specific integrated problem as a subproblem. There are, however, multiple possibilities of splitting variables in a decomposition framework. Including some of the variables of the subproblem in the master problem can provide better bounds and useful information for the master problem (Rahmaniani et al. 2017).

2. Problem setting and illustrative example

We consider an extension to SALBP (cf. Section 1), in which different modes $p \in P = \{\mathrm{p_H}, \mathrm{p_R}, \mathrm{p_C}\}$ with resulting processing times t_{ip} are available for the tasks' manual execution by human workers ($\mathrm{p_H}$ with $t_{i\mathrm{p_H}}$), automated execution by cobots ($\mathrm{p_R}$ with $t_{i\mathrm{p_R}}$), and collaborative execution by cobots and workers ($\mathrm{p_C}$ with $t_{i\mathrm{p_C}}$). However, not each task may be feasible for automated or collaborative execution and thus not each mode might be available for each task. While human workers are assumed to be available at each station k, we have to decide about the allocation of cobots to stations ($r_k \in \{0,1\}$). The number of available cobots may be limited to q units for the assembly line. We decide about the allocation of tasks i to stations k and modes p ($x_{ikp} \in \{0,1\}$). If cobots are involved in a task (i.e., modes $\mathrm{p_R}$ and $\mathrm{p_C}$), they have to be available at the respective station. If a task is conducted collaboratively, both a human worker and a cobot have to be available at the respective points in time in the associated station. To this end, we decide about the scheduling of tasks and determine their starting time within the stations ($s_i \geq 0$). We aim to minimize the cycle time c for a given number of stations.

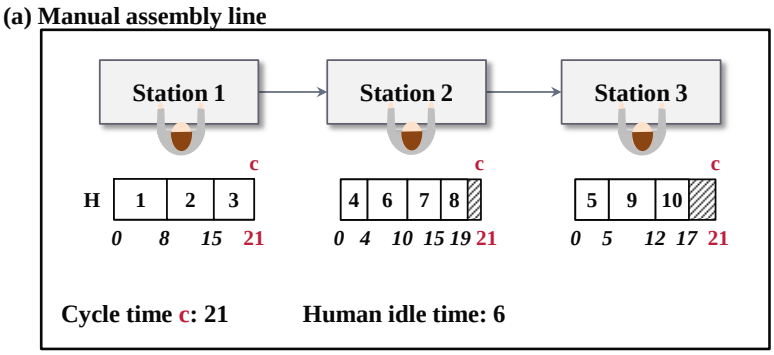
The problem setting is further illustrated in Figure 1 for an example product consisting of $|I| = 10$ tasks and $|K| = 3$ stations. Tasks i infeasible with either automated or collaborative execution are indicated by a processing time of $t_{ip_R}, t_{ip_C} = \infty$, respectively. In a manual assembly line (no cobots available, cf. Figure 1(a)), tasks are executed serially. The optimal cycle time results in $c = 21$ time units. If one collaborative robot is available (cf. Figure 1(b)), the cycle time can be reduced to $c' = 17$ time units taking advantage of the simultaneous execution of tasks by worker and cobot ($i = \{1, 2\}$) or their collaboration on identic tasks ($i = \{4,7\}$).



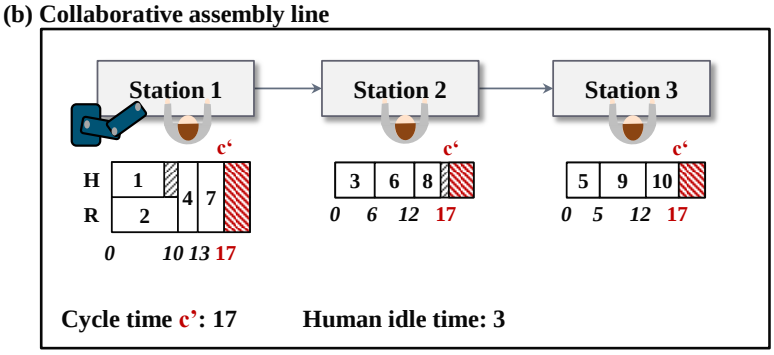
Precedence graph of the example problem

Task i	Processing times t		
	t_{ip_H}	t_{ip_R}	t_{ip_C}
1	8	∞	6
2	7	10	5
3	6	∞	∞
4	4	∞	3
5	5	11	4
6	6	∞	∞
7	5	11	4
8	4	∞	∞
9	7	∞	5
10	5	11	4

Processing times of manual (t_{ip_H}), automated (t_{ip_R}), and collaborative (t_{ip_C}) modes



Optimal line balance for manual assembly line ($q = 0$ cobots available)



Optimal line balance for collaborative assembly line (q =1 cobots available)

Cycle time reduction: Idle time:

Figure 1 – Illustrative example (adapted from Weckenborg et al. (2020))

[Figure 1 Alt-Text: Illustrative example describing the problem setting. Precedence graph and processing times are presented. If one collaborative robot is available, one task is exe-

cuted automated and worker and robot collaborate on two other tasks. The cycle time can be reduced from 21 to 17 time units.]

Our modeling approaches are based on further assumptions:

1. The line is balanced for a single product or the products' common (mixed) precedence graph is available.
2. The precedence relations between the tasks are known.
3. Stations are arranged serially. The parallelization of workplaces within stations is of particular importance to the considered problem but the parallelization of tasks, stations, or lines is neglected.
4. Processing times are known, deterministic, constant, and sequence-independent for each of the modes. However, not each mode must be available for each task.
5. A worker and necessary equipment are available in each station. For each station, one collaborative robot may be assigned additionally.
6. No further assignment restrictions apply.

3. Benders' decomposition algorithm

The success of the Benders' decomposition in recent years lays in the formulation of assembly line balancing problem variations with two levels of decisions. In the first level, the balancing of the assembly line is decided, while the second part of the decision is relegated to the subproblem. The subproblems may deal with the sequencing of tasks with set-up times (Akpinar, Elmi, and Bektaş 2017), multiple workers (Michels et al. 2019), or multiple products (Sikora 2021). The division of decisions, however, is not straightforward for the balancing with cobots since there are multiple possibilities to split the variables between master and subproblems. The decision whether tasks are performed individually or in the collaborative mode could be a part of either decision level. According to Rahmaniani et al. (2017), the de-

signers of Benders' algorithms should balance out the size and difficulty of master and sub-problems to obtain better results. To this end, we explore the different possibilities of splitting the formulation and identify the potential advantages and disadvantages of each decomposition approach in Section 4.1. In Section 4.2., we describe the general procedure of the algorithm. Additionally, an optional acceleration technique based on local search is introduced in Section 4.3.

3.1. Decomposition approaches

The notation common to our decomposition approaches is presented in Table 1.

Table 1 – Definitions of sets, parameters, and variables.

Sets and parameters	
I	Set of tasks $i, j \in I = \{1, \dots, \lvert I \rvert\}$
K	Set of stations $k, l \in K = \{1, \dots, \lvert K \rvert\}$
P	Set of modes $p \in P = \{\mathrm{p_H}, \mathrm{p_R}, \mathrm{p_C}\}$, in which tasks are processed by human ($\mathrm{p_H}$), robot ($\mathrm{p_R}$) or in collaboration ($\mathrm{p_C}$), respectively
E	Set of direct precedence relations (i, j)
t_{ip}	Execution time of task $i \in I$ with processing alternative $p \in P$
$t_i^{\min}$	Minimal required resource time of task $i \in I$ considering both worker and robot. $t_i^{\min} = \min\{t_{i\mathrm{p_H}}, t_{i\mathrm{p_R}}, 2 \cdot t_{i\mathrm{p_C}}\}$
$\overline{c}$	Upper bound on cycle time. Initially, $\overline{c} = \max\{t^{\max}, 2 \cdot \lfloor t^{\mathrm{sum}}/\lvert K \rvert \rfloor\}$, where $t^{\max} = \max\{t_{ip} \mid i \in I, p \in P\}$ and $t^{\mathrm{sum}} = \sum_{i \in I} \max\{t_{ip} \mid p \in P\}$
q	Maximum number of robots to be allocated
N_m	Set containing pairs $(i, k) \in (I, K)$ of assignments of a given feasible solution m
M	Set of balancing solutions m
$\overline{c}_m$	Known upper bound for the cycle time for a given feasible solution m
I_k	Subset of I containing the tasks assigned to a station k in a solution m
Decision and auxiliary variables	
x_{ikp}	Binary variable with value 1, if task $i \in I$ is assigned to station $k \in K$ with processing alternative $p \in P$
z_{ik}	Binary variable with value 1, if task $i \in I$ is assigned to station $k \in K$ independent of the processing alternative
s_i	Continuous variable for encoding the start time of task $i \in I$ in the station it is assigned to
r_k	Binary variable with value 1, if a cobot is assigned to station $k \in K$
c	Non-negative variable for encoding the cycle time
y_{ij}	Binary variable with value 1, if task $i \in I$ starts before task $j \in I$ $(s_i \leq s_j)$

3.1.1. Selecting task mode in the subproblem

In the first approach to decomposing the problem, the master problem is responsible for the assignment of tasks into stations (variable z_{ik}), while the mode for each task is decided in the subproblem. We therefore propose the following decomposition in master problem (expressions (1) to (8)) and subproblem (expressions (9) to (20)).

Minimize c (1)

Subject to:

$$\sum_{k \in K} z_{ik} = 1 \qquad \forall\, i \in I \tag{2}$$

$$\sum_{l=1}^{k} z_{il} \leq \sum_{l=1}^{k} z_{jl} \qquad \forall\, (i,j) \in E, k \in K \tag{3}$$

$$\sum_{k \in K} r_k \leq q \tag{4}$$

$$\sum_{i \in I} t_i^{\min} z_{ik} \leq c + \overline{c} \cdot r_k \qquad \forall\, k \in K \tag{5}$$

$$\sum_{i \in I} t_i^{\min} z_{ik} \leq 2c \qquad \forall\, k \in K \tag{6}$$

$$z_{ik} \in \{0,1\} \qquad \forall\, i \in I, k \in K \tag{7}$$

$$r_k \in \{0,1\} \qquad \forall\, k \in K \tag{8}$$

The objective function is to minimize the cycle time of the line, estimated by the variable *c*. Each task must be assigned to a single station (Expression (2)), while the assignment must obey the precedence relations (Expression (3)). The allocation of the cobots is also decided in the master problem, for which the maximal amount of cobots is limited in Expression (4). As the scheduling of the tasks is not known in the master problem, the resulting cycle time is estimated using a lower bound. Expression (5) covers the case in which no cobot is assigned to the workstation. The left-hand side considers the sum of the processing times of the assigned tasks at the station, which must be smaller or equal to *c*. The term $\overline{c} \cdot r_k$ serves as a

Big-M restriction to not enforce the bound if a cobot is assigned. The general case is modelled in Expression (6). Accordingly, the processing times of the tasks must be considered as the minimal capacity requirement ($t_i^{\min}$), as the inequality may not be valid if the cobots were faster than the human workers. $t_i^{\min}$ is determined by the least resource intensive mode of performing task i. Its value can be either determined by the time required from a human worker (t_{ip_H}), from a robot (t_{ip_R}), or the combined time resources used in a collaborative task ($2 \cdot t_{ip_C}$). The sum of the processing times of the assigned tasks must be smaller or equal to the equivalent of two cycle times, i.e., one for the human worker and one for the cobot.

For the dataset used in Weckenborg et al. (2020), the automated times are modelled as 200% of the manual time, while the collaborative time corresponds to 70% of the manual time. For these specific settings, Expression (6) can be improved by using

$$\sum_{i \in I} t_{ip_H} z_{ik} \leq 1.5\, c \quad \forall\, k \in K$$

since a cobot can contribute at most 50% to the 100% of human productivity when working individually. When human and cobot collaborate, the resulting productivity is about 143% (100% / 70%) of the human productivity, which is within the improved expression's bound.

To check the real cycle time of the solution, the modes for the tasks previously assigned to station k (subset of tasks I_k) are selected and the tasks are sequenced and scheduled using the following formulation of the subproblems for the individual stations.

Minimize c (9)

Subject to:

$$\sum_{p \in P} x_{ikp} = 1 \qquad \forall\, i \in I_k \qquad (10)$$

$$c \geq s_i + \sum_{p \in P} x_{ikp} \cdot t_{ip} \qquad \forall\, i \in I_k \qquad (11)$$

$$s_i + \sum_{p \in P} x_{ikp} \cdot t_{ip} \leq s_j \qquad \forall\, i,j \in I_k, (i,j) \in E \qquad (12)$$

$$s_i + t_{i\mathrm{p_C}} \cdot x_{ik\mathrm{p_C}} \leq s_j + \overline{c} \cdot \left(1 - x_{ik\mathrm{p_C}}\right) + \overline{c} \cdot \left(1 - y_{ij}\right) \qquad \forall\, i,j \in I_k, i \neq j \qquad (13)$$

$$s_i + \sum_{p \in P} x_{ikp} \cdot t_{ip} \leq s_j + \overline{c} \cdot \left(1 - x_{jk\mathrm{p_C}}\right) + \overline{c} \cdot \left(1 - y_{ij}\right) \qquad \forall\, i,j \in I_k, i \neq j \qquad (14)$$

$$s_i + t_{ip} \cdot x_{ikp} \leq s_j + \overline{c} \cdot \left(1 - x_{ikp}\right) + \overline{c} \cdot \left(1 - x_{jkp}\right) + \overline{c} \cdot \left(1 - y_{ij}\right) \qquad \forall\, i,j \in I_k, i \neq j, p \in \{\mathrm{p_H}, \mathrm{p_R}\} \qquad (15)$$

$$y_{ij} + y_{ji} = 1 \qquad \forall\, i,j \in I_k, i \neq j, (i,j) \notin E \qquad (16)$$

$$y_{ij} = 1 \qquad \forall\, i,j \in I_k, (i,j) \in E \qquad (17)$$

$$x_{ikp} \in \{0,1\} \qquad \forall\, i \in I_k, p \in P \qquad (18)$$

$$y_{ij} \in \{0,1\} \qquad \forall\, i,j \in I_k, i \neq j \qquad (19)$$

$$s_i \geq 0 \qquad \forall\, i \in I_k \qquad (20)$$

The objective function addresses the minimization of the makespan of the considered station and therefore supports the minimization of the line's cycle time (Expression (9)). The choice of modes is enforced in expression (10). The makespan must be larger than the finishing time of all tasks within the station (expression (11)). Expressions (12) to (17) are responsible for the sequencing and scheduling of tasks. For tasks with precedence relations, the sequence is predetermined (expression (17)). For the other cases, the sequence can be decided based on the variables y_{ij} (expression (16)). If tasks are performed collaboratively, expression (13) assures that all succeeding tasks are started after these tasks both for human and cobot. Similarly, expression (14) assures all preceding tasks to be finished beforehand. Tasks that are not performed collaboratively must obey expression (15), which allows for exclusively one task to be conducted by each worker or cobot at the same time.

The subproblem must only be solved if a cobot is assigned to workstation *k*. Otherwise, the scheduling is trivial, since the worker processes all tasks sequentially without idle times.

Whenever a subproblem is solved, the optimal makespan and the assignment of tasks to stations are stored in $\overline{c}_m$ and N_m, respectively. The information can then be added to the master problem in the form of the Cut (21).

$$c \geq \overline{c}_m \cdot \left(\sum_{(i,k) \in N_m} z_{ik} - |N_m| + 1 \right) \qquad \forall\, m \in M \qquad (21)$$

This expression assures that the cycle time *c* is corrected to the real value if the variables z_{ik} assume the value of an already solved subproblem.

3.1.2. Selecting task mode in the master problem

A second possibility of decomposing the problem is to include the mode decisions in the master problem. The formulation of the master problem consists of expressions (1)–(8) and (22)–(27).

$$\sum_{p \in P} x_{ikp} = z_{ik} \qquad \forall\, i \in I, k \in K \qquad (22)$$

$$x_{ik\mathrm{p_R}} + x_{ik\mathrm{p_C}} \leq r_k \qquad \forall\, i \in I, k \in K \qquad (23)$$

$$\sum_{i \in I} (t_{i\mathrm{p_H}}\, x_{ik\mathrm{p_H}} + t_{i\mathrm{p_C}}\, x_{ik\mathrm{p_C}}) \leq c \qquad \forall\, k \in K \qquad (24)$$

$$\sum_{i \in I} (t_{i\mathrm{p_R}}\, x_{ik\mathrm{p_R}} + t_{i\mathrm{p_C}}\, x_{ik\mathrm{p_C}}) \leq c \qquad \forall\, k \in K \qquad (25)$$

$$\sum_{i \in I} t_i^{\min} z_{ik} \leq 2c \qquad \forall\, k \in K \qquad (26)$$

$$x_{ikp} \in \{0,1\} \qquad \forall\, i \in I, k \in K, p \in P \qquad (27)$$

By selecting the mode in the master problem, the lower bound on the cycle time *c* is tighter since the processing times of humans and cobots are computed separately. The disadvantage

of this approach is that the master problem contains more variables, requiring more time to solve.

As the processing alternatives are determined in the master problem, these decisions can be fixed in the subproblem. In this case, the subproblem consists of Expressions (9) to (20) but variable x_{ikp} must be treated as a parameter. The information of the optimal solution can be added to the master problem in the form of Cut (28).

$$c \geq \overline{c}_m \cdot \left(\sum_{(i,k,p) \in N_m} x_{ikp} - |N_m| + 1 \right) \qquad \forall\, m \in M \qquad (28)$$

Although the subproblem with fixed modes is easier to solve, Cut (28) is very weak: A solution of the master problem containing the same task-to-station assignment but with one difference in one mode would result in the cut $c \geq 0$.

One alternative to this decomposition scheme is to solve the subproblem defined in Section 4.1.1. assuming the processing alternatives as variables. This way, the combinatorial cut is created considering all possible processing alternatives (Cut (21)).

3.1.3. Using relaxed decision variables in the master problem

A third option for the decomposition scheme is an intermediary solution between the ones presented in Sections 4.1.1. and 4.1.2. Here, the master problem is defined with both z_{ik} and x_{ikp} variables as in Section 4.1.2. Variables x_{ikp}, however, are defined as continuous variables within the bounds [0,1]. This setting allows for the formulation of the expressions (24) and (25) in the master problem without adding any binary variables to the formulation. Although continuous variables for the assignments will probably not be an integer within feasible solutions, these expressions can improve the bound on the cycle time. For this variant, the subproblem and combinatorial cut are the same as the one presented in Section 4.1.1.

3.2. The combinatorial Benders' decomposition algorithm

The original algorithm of Benders (1962) is designed for subproblems containing only continuous variables since the cuts formulated based on the subproblem use the information of the shadow price of such variables. If the subproblem contains integer or binary variables, cuts based on the linear relaxation of the subproblem or combinatorial cuts can be used instead (Rahmaniani et al. 2017).

The algorithm is explained in Figure 2. Every solution of the master problem is used to formulate subproblems. For every station with a cobot, the scheduling subproblem is solved to obtain the real cycle time of the assignment. If the cycle time is equal to the optimal solution of the master problem, the algorithm converges and the found solution is optimal. If the cycle time checked in the subproblem is larger than the estimated in the master problem, a combinatorial cut correcting the estimation is added to the master problem.

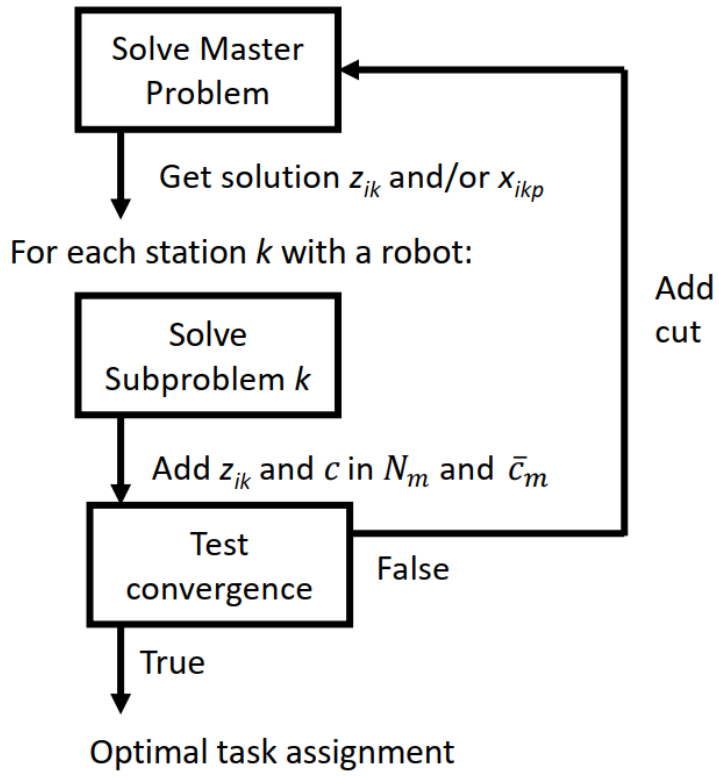


Figure 2 – Diagram of the algorithm

[Figure 2 Alt-Text: Flow chart describing the procedure of the decomposition algorithm. The algorithm has three steps: solve master problem, solve subproblem k, and test convergence. The solution of the master problem is used for the subproblem, which is then checked for convergence, and used to generate combinatorial cuts for the master problem.]

Instead of solving the master problem in each iteration, an alternative implementation considers adding combinatorial cuts "on the fly" (Codato and Fischetti 2006). This option is possible in commercial solvers via callbacks. After every feasible solution found in the master problem, the subproblems are formulated and solved. The cut is then added dynamically during the solution of the master problem.

3.3. Accelerating Benders' decomposition with local search

According to the survey on Benders' decomposition by Rahmaniani et al. (2017), the solution of the master problem can take a considerable amount of time. In the computation experiments of Sikora (2022), the time spent by the algorithm on the master problem is often more than 90% of the algorithm's running time.

To deal with this drawback, some authors consider finding solutions to the master problem heuristically (Costa et al. 2012). Caserta and Voß (2021) propose one approach in which a local search is used for the master problem. Instead of resuming the master problem solution after adding a cut, some time is spent on the neighbourhood of the incumbent solution. If any better solution for the master problem is found, the corresponding subproblems are solved and new cuts are added. This way, feasible solutions are found earlier, and more cuts are added on average for the same amount of time.

For the implementation of this acceleration technique, a model containing the same variables and restrictions of the master problem is solved with additional restrictions (29) and (30), in which N_m is set containing the last solution found.

$$\sum_{(i,k)\in N_m} z_{ik} \leq |N_m| - MinDiff \tag{29}$$

$$\sum_{(i,k)\in N_m} z_{ik} \geq |N_m| - MaxDiff \tag{30}$$

Restriction (29) assures that the solution is not identical to the previous solution. Expression (30) controls the size of the solution neighbourhood. The values of *MinDiff* and *MaxDiff* can be set accordingly. At least one assignment difference is required to obtain a different solution, while the parameter *MaxDiff* must be set not very high so that the local search can be solved quickly.

4. Computational experiments

4.1. Data and computational setting

A dataset for the assembly line balancing with collaborative robots is proposed in Weckenborg et al. (2020) and is used in this section to test the performance of the algorithm. This is the only available test set requiring all relevant information. To facilitate future research on the optional collaboration on identic tasks in assembly line balancing, the used test instances are available online in the supplementary material to this article.

The dataset is divided into three parts: small, medium, and large-sized instances containing 20, 50, and 100 tasks, respectively. The dataset contains a total of 1,500 instances, 500 for each instance size. The instances are enriched with specific parameters on the number of stations, the number of available cobots, and the feasibility of automated and collaborative execution of the tasks. The number of stations is defined to 5 and 10 for small instances (where either of these is used for half of the corresponding instances), 13 and 25 for medium instances, and 25 and 50 for large instances. The number of cobots is chosen to result in a robot den-

sity (i.e. the share of cobots to station) of 0%, 20%, or 40%. From the 1,500 instances, 300 instances represent a robot density of 0%, 600 consider a robot density of 20%, and the remaining 600 consider cobots to be available for 40% of the workstations. The tasks that can be performed by a cobot and/or collaboratively were selected randomly during the instance generation process. The number of tasks that can be performed by cobots (and collaboratively) is set to 20% for half of the instances and 40% for the other half.

For the computational experiments, the Benders' decomposition algorithm is initialized with a feasible solution by solving the problem using the SALOME algorithm for SALBP (Scholl and Klein 1997). The solution without a cobot is an upper bound for the cycle time since the processing time can only get smaller by considering the additional assignment of cobots. From the Weckenborg et al. (2020) dataset, only 1,200 of the 1,500 instances are used, as the remaining do not consider cobots. The search for an initial solution is limited to 300 seconds, which are accounted for in the total solving time of the algorithm. Small and medium-size instances are solved with a time limit of 7,200 seconds, while the large-size instances have a time limit of 28,800 seconds. All calculations are performed on machines with 8 virtual cores of a 2.5 GHz Intel Xeon Platinum 8180 processor and 32 GB RAM.

4.2. Algorithm configurations

The three decomposition frameworks described in Section 4.1. are implemented in Visual Studio 2013 using Gurobi 9.0.1 as a solver for the master and subproblems. Based on preliminary tests, the local search improvement described in Section 4.3. is effective only for the framework proposed in Section 4.1.2. Therefore, the four algorithms tested correspond to those described in sections 4.1.1. (Benders 1), 4.1.2. without local search (Benders 2), 4.1.3. (Benders 3), and 4.1.2. with local search (Benders 4).

For the local search, the same parameters are used as in Sikora (2022). After each integer solution found in the master problem, the neighborhood is searched. A time limit of 60 seconds is used for each iteration of the neighborhood search. If no better solution is found within this limit, the local search iteration is aborted. Furthermore, *MinDiff* is set to 1, *MaxDiff* is set to 6. With a limit of at most 6 differences, the local search procedure can often quickly identify good neighboring solutions, if they exist. The local search should help finding integer solutions but not hinder the convergence of the algorithm. As implemented in Sikora (2022), the local search is limited to two iterations only before resuming the master problem solution. Moreover, the time spent in local search is measured and inactivated whenever it exceeds 10% of the total time spent in the algorithm.

4.3. Numerical results

The four versions of the decomposition algorithm (Benders 1 to 4) are compared based on their results on the 400 small instances of the dataset. A comparison of the results is given in Table 2. The instances are sorted based on the number of stations, the number of robots available, and the robot and collaboration flexibility of tasks. For each of the four versions of the algorithm, we report the CPU running time in seconds. Furthermore, column ΔObj MIP expresses the difference in the objective function values obtained by the decomposition methods and the MIP solution. The values of ΔObj MIP are computed firstly by calculating the percentual difference for each of the contained instances individually and then calculating the average over the 50 instances in each row.

For the small instances, all four decomposition strategies achieve similar values for the objective function. The average values reported in Table 2 do not show any difference in terms of solution quality for the proposed methods. There are, however, slight variations when the individual instances are observed. Within the time limit of 7,200 seconds, Benders 1 solves 399

of the 400 instances optimally, while Benders 2, Benders 3, and Benders 4 prove the optimality of only 376, 370, and 381 instances out of 400, respectively. The difference in behaviour can be observed strongly in the average CPU time, since even without the optimality proof for some instances, their upper bounds are optimal or close to optimal. For the small instances, Benders 1 performed best, since it solves more instances and requires lower solution times on average. The second candidate based on these results is Benders 4, which requires less average CPU times for every group compared to Benders 2 and Benders 3.

Another view of the algorithms' performance can be seen in the performance behaviour illustrated in Figure 3. Here, the curves of the number of instances solved to optimality up to a given run time are displayed. In the x-axis, a factor in relation to the minimal solution time is used, in logarithmic scale as described in Moré and Wild (2009). While the profile curves confirm the superiority of Benders1 for the small instances, the behaviours of Benders2, Benders3, and Benders4 are very similar. For further tests with medium and large size instances, only Benders 1 and Benders 4 are explored, as Benders4 dominated the profile curves of Benders2 and Benders3.

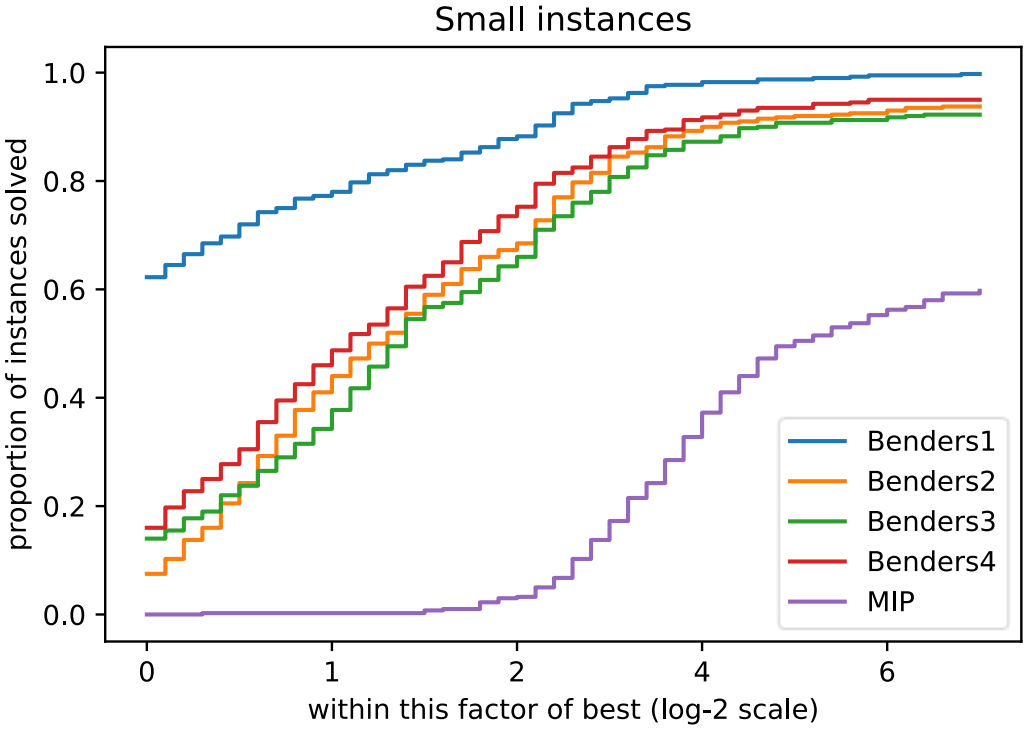


Figure 3 – Performance profile for small instances

[Figure 3 Alt-Text: Performance profile for the small instances. The graph shows the number of instances that can be solved within a factor of the time limit of the lowest solution time. In the figure, Benders1 is above all other methods. Benders4 has the second place, being closely followed by Benders3 and Benders4. The MIP results are way under the other algorithms.]

Table 2 – Comparison of Benders' decomposition approaches for small instances

Stations	**Robots**	**Flexibility**	**Benders 1**		**Benders 2**		**Benders 3**		**Benders 4**	
			CPU Ø (σ) [s]	ΔObj MIP Ø (σ) [%]	CPU Ø (σ) [s]	ΔObj MIP Ø (σ) [%]	CPU Ø (σ) [s]	ΔObj MIP Ø (σ) [%]	CPU Ø (σ) [s]	ΔObj MIP Ø (σ) [%]
5	1	0.2	28 (88)	-0.1 (0.2)	34 (88)	-0.1 (0.2)	42 (108)	-0.1 (0.2)	15 (25)	-0.1 (0.2)
5	1	0.4	83 (293)	-0.1 (0.2)	79 (236)	-0.1 (0.2)	59 (105)	-0.1 (0.2)	57 (113)	-0.1 (0.2)
5	2	0.2	115 (272)	-0.1 (0.4)	54 (130)	-0.1 (0.4)	75 (235)	-0.1 (0.4)	14 (26)	-0.1 (0.4)
5	2	0.4	76 (222)	-0.2 (0.4)	136 (295)	-0.2 (0.4)	230 (663)	-0.2 (0.4)	151 (354)	-0.2 (0.4)
10	2	0.2	45 (121)	0.0 (0.1)	1191 (2462)	0.0 (0.1)	1401 (2756)	0.0 (0.1)	424 (1425)	0.0 (0.1)
10	2	0.4	20 (44)	0.0 (0.1)	1328 (2683)	0.0 (0.1)	1486 (2821)	0.0 (0.1)	1172 (2537)	0.0 (0.1)
10	4	0.2	173 (1008)	0.0 (0.0)	450 (1445)	0.0 (0.0)	735 (1990)	0.0 (0.0)	197 (1010)	0.0 (0.0)
10	4	0.4	95 (305)	0.0 (0.0)	1442 (2868)	0.0 (0.0)	1474 (2870)	0.0 (0.0)	1391 (2789)	0.0 (0.0)

For the following analyses, we compare the results of the methods for all instance sizes of the dataset. Each of the methods (MIP, GA, Benders 1, and Benders 4) is run with a time limit of 7,200 seconds for the small and medium instances and 28,800 seconds for the large size instances. As the GA randomly selects parents for the genetic replication and additionally terminates after a specific number of replications when not further improving the objective value, up to ten runs of the GA may be conducted as long as the overall run time complies the time limit.

The results are given in Table 3 and Table 4. Both tables contain all 1,200 instances of the dataset. In Table 3, the CPU times are reported along with their solution quality in terms of optimality gap (for the MIP) and percentage difference to the MIP solutions (for GA, Benders 1, Benders 4). The percentage difference to MIP solutions, however, can only be calculated for the instances in which the MIP finds a feasible solution. Therefore, the given values consider only the instances that can be compared and do not represent the average of all 50 instances for each row. The number of instances in each row with feasible solutions and a complete comparison with all instances is given in Table 4.

According to the results in Table 3 and Table 4, the MIP formulation using a commercial solver is not capable to solve a considerable part even of the small instances: 149 out of 400 instances are not solved to optimality. For medium and large instances, the number of stations has a large impact on the solvability of the MIP model. For the instances with shorter lines (13 stations for medium instances and 25 for large instances), only 3 and 0 instances are solved to optimality using the MIP formulation on medium and large instances, respectively. For the longer lines, 116 of the 200 instances of medium size and only 15 of the 200 large size instances are solved optimally. Concerning time, the MIP formulation requires more CPU time than any proposed Benders' decomposition but wins in respect to time for some parameters in comparison to the genetic algorithm.

The number of stations is one of the main drivers of the difficulty to solve an instance. Such a factor is directly related to the size of the subproblem since lines with more stations have fewer tasks per station and, therefore, smaller scheduling problems. Consequently, we find only one large instance with 25 stations being solved by Benders 1 and Benders 4, while 142 and 152 large instances with 50 stations are solved by these decomposition algorithms, respectively. This fact contradicts common results of the simple assembly line balancing literature. According to Otto, Otto, and Scholl (2013), the SALBP instances generated with few tasks per station are the hardest to solve. Since there may not be many combinations of two or three tasks close to a lower bound, SALBP algorithms tend to enumerate more combinations for such instances. In the case of ALBP considering cobots, this effect is less important than the required time for solving the scheduling problems within the stations.

Concerning the average objective value, Benders 4 provides better results in comparison to the genetic algorithm and Benders 1. As shown in Table 3, Benders 4 presents a smaller or the same percentage difference to the MIP solution in all groups of instances. Furthermore, the time comparison between the Benders decompositions and the genetic algorithm is skewed to the decomposition algorithm: Benders 1 requires less CPU time on average for 20 of the 24 instance groups. For the instances individually, Benders 1 solved 977 out of the 1200 instances faster than the genetic algorithm, as displayed in Table 4.

Another meaningful comparison can be drawn for the two decomposition algorithms. Although the solution quality of Benders 4 is superior for the dataset, Benders 1 solved more instances to optimality. By inspecting log files of some instances, it can be observed that Benders 4 finds good upper bounds earlier than Benders 1 due to the integrated local search procedure. These upper bounds of Benders 4, however, do not necessarily result in a better algorithm for proving optimality. As the master problem of Benders 4 has more variables and

more possible combinations, ruling out that a better solution may exist seems to be harder for this version of the decomposition algorithm.

Another view on the comparison of the algorithms is provided by the profile curves in Figure 4 for the medium-size instances, Figure 5 for the large instances, and Figure 6 for all instances. Although Benders1 presents the better profile for the small instances, this behaviour chances for other instance sizes. For the medium-size instances in Figure 4, the profile of Benders1 still dominates the curve of Benders4, although the lead is smaller than the one for the smaller instances. For the large instances, shown in Figure 5, the profile dominance shifts in favour of Benders4. The profiles show, therefore, that the relative performance of the decomposition alternatives depends on the instance size. Finally, the profile curves for all 1,200 instances of the dataset are given in Figure 6. The sum of the contribution of all instance sizes masks the dominance: Benders1 is slightly better than Benders4 for all instances, although both profiles are similar to each other.

Although the MIP approach provided by Weckenborg et al. (2020) is clearly dominated for all instance sizes, the comparison is biased since another version of the solver was used for the solution. To provide a better comparison basis, one instance of all classes (12.5% of the total dataset) is solved using the older version of Gurobi (8.1) for the decomposition algorithms. The profile curve is given in Figure 7. As expected, the newer version of the solver improves the performance of the algorithms. Even with the older version, however, the profile curves of both decomposition versions largely dominate the one of the MIP solutions. It is interesting to observe that the newer version of the solver greatly improves the relative performance of Benders4, while the curve for Benders1 is almost identical.

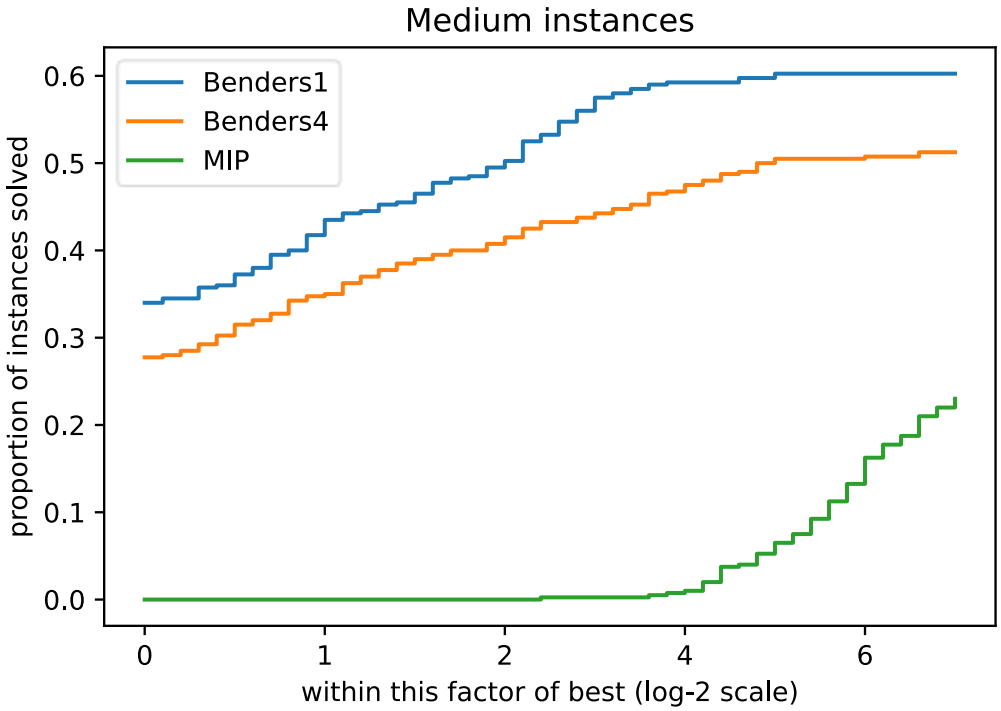


Figure 4 – Performance profile for medium instances

[Figure 4 Alt-Text: Performance profile for the medium instances. The graph shows the number of instances that can be solved within a factor of the time limit of the lowest solution time. In the figure, Benders1 dominates above the curve for Benders4. Both versions of Benders (1 and 4) present much better results than the MIP.]

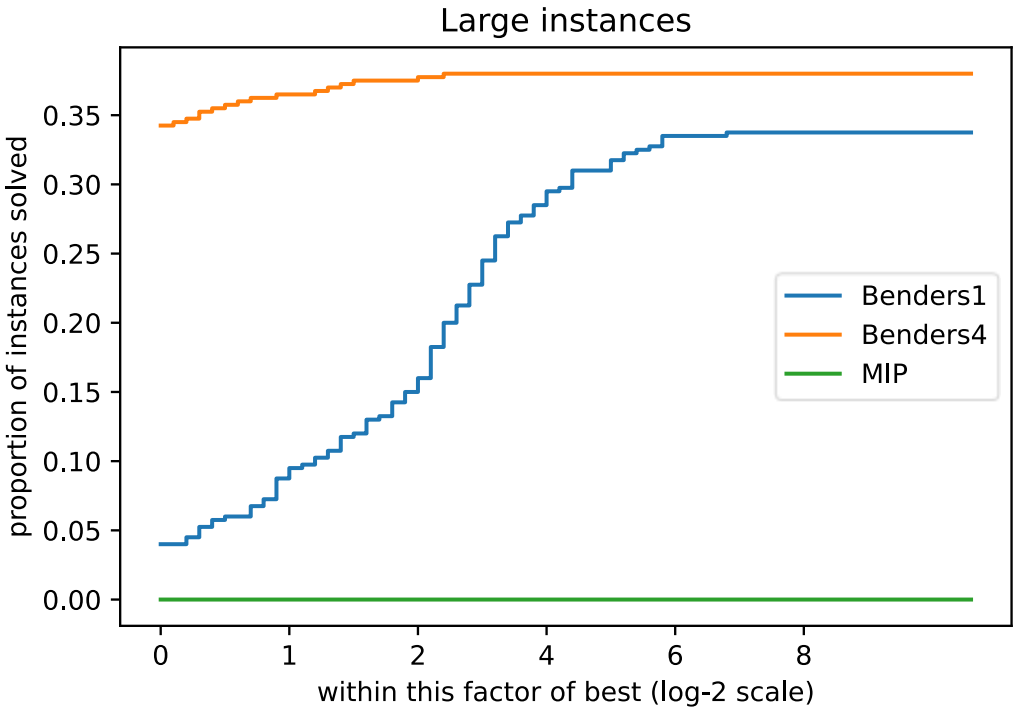


Figure 5 – Performance profile for large instances

[Figure 5 Alt-Text: Performance profile for the large instances. The graph shows the number of instances that can be solved within a factor of the time limit of the lowest solution time. In the figure, Benders4 presents better results than the other methods (Benders1).]

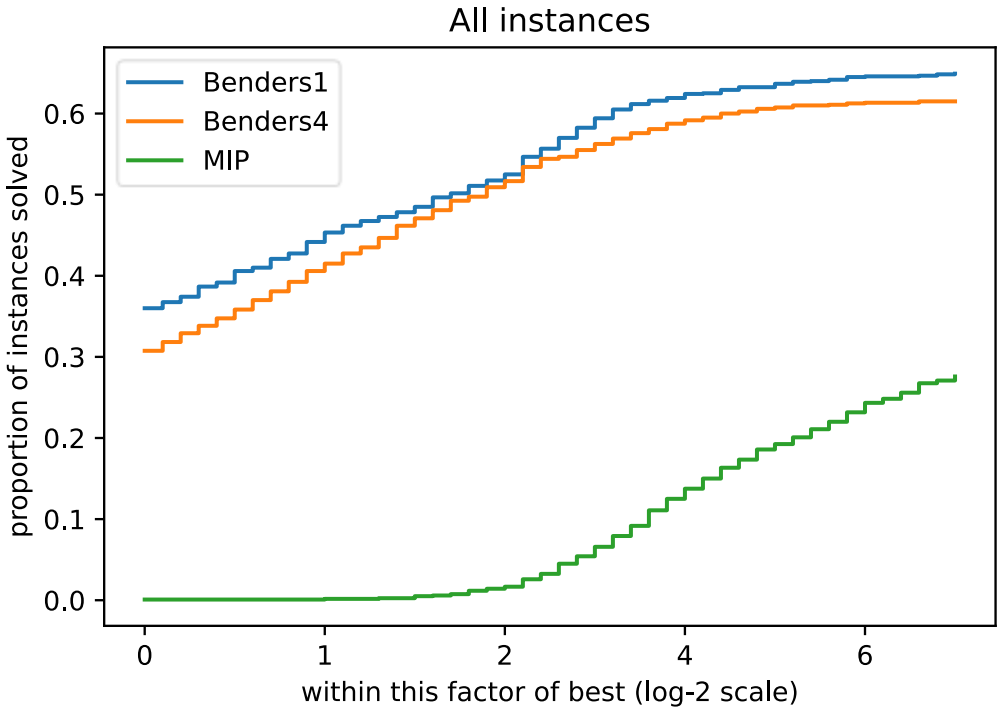


Figure 6 – Performance profile for all instances

[Figure 6 Alt-Text: Performance profile for all instances. The graph shows the number of instances that can be solved within a factor of the time limit of the lowest solution time. In the figure, Benders1 presents better slightly results than Benders4. The MIP is dominated by the decomposition methods.]

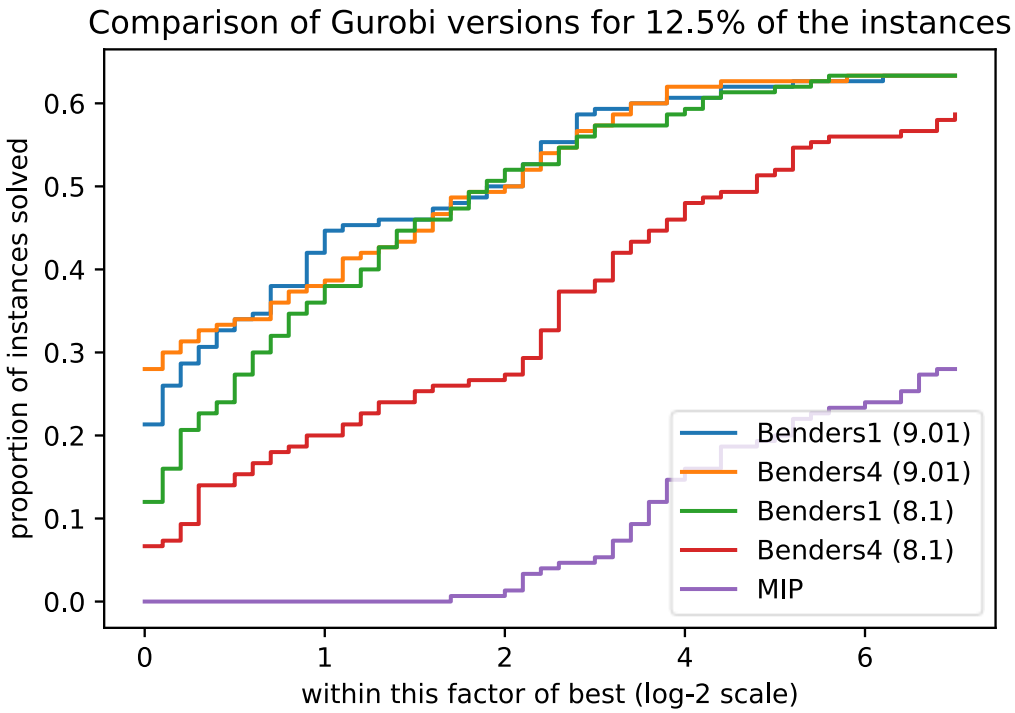


Figure 7 – Performance profile for 12.5% of the instances using different versions of the solver.

*[*Figure 7*Alt-Text: Performance profile for 12.5% of all instances including all sizes. The graph shows the number of instances that can be solved within a factor of the time limit of the lowest solution time. In the figure, Benders1 and Benders 4 using Gurobi 9.01 and Bendes1 using Gurobi 8.1 present similar behaviour. Benders4 with Gurobi 8.1 is clearly dominated. The MIP is the method that solve the lowest number of instances.]*

Table 3 – Results comparison for mixed-integer programming approach (MIP), genetic algorithm (GA), and Benders' decomposition approaches

Instances (tasks)	Stations	Robots	Flexibility	MIP		GA		Benders 1		Benders 4	
				CPU ∅ (σ) [s]	Gap ∅ (σ) [%]	CPU ∅ (σ) [s]	ΔObj MIP ∅ (σ) [%]	CPU ∅ (σ) [s]	ΔObj MIP ∅ (σ) [%]	CPU ∅ (σ) [s]	ΔObj MIP ∅ (σ) [%]
Small (20)	5	1	0.2	3610 (3590)	23.9 (13.8)	558 (161)	0.1 (0.3)	28 (88)	-0.1 (0.2)	15 (25)	-0.1 (0.2)
	5	1	0.4	3624 (3577)	28.3 (13.8)	781 (317)	0.2 (0.3)	83 (293)	-0.1 (0.2)	57 (113)	-0.1 (0.2)
	5	2	0.2	3564 (3563)	18.3 (13.8)	1087 (307)	0.3 (0.8)	115 (272)	-0.1 (0.4)	14 (26)	-0.1 (0.4)
	5	2	0.4	3652 (3551)	25.7 (14.5)	1319 (417)	0.3 (0.8)	76 (222)	-0.2 (0.4)	151 (354)	-0.2 (0.4)
	10	2	0.2	1898 (3144)	16.6 (12.6)	717 (146)	0.3 (0.9)	45 (121)	0.0 (0.1)	424 (1425)	0.0 (0.1)
	10	2	0.4	1910 (3137)	23.2 (10.9)	747 (149)	0.2 (0.8)	20 (44)	0.0 (0.1)	1172 (2537)	0.0 (0.1)
	10	4	0.2	1520 (2858)	9.1 (10.3)	1515 (327)	0.2 (0.6)	173 (1008)	0.0 (0.0)	197 (1010)	0.0 (0.0)
	10	4	0.4	1660 (2956)	23.6 (8.2)	1600 (357)	0.7 (1.3)	95 (305)	0.0 (0.0)	1391 (2789)	0.0 (0.0)
Medium (50)	13	3	0.2	7063 (809)	35.5 (13.8)	4080 (1007)	-0.7 (1.5)	3868 (3476)	-0.6 (1.7)	4260 (3298)	-2.1 (1.5)
	13	3	0.4	7200 (0)	39.5 (12.1)	5153 (1096)	-1.3 (1.6)	3636 (3564)	-1.3 (5.8)	4942 (3014)	-3.0 (1.4)
	13	5	0.2	7074 (882)	30.4 (16.3)	6392 (918)	-0.2 (2.4)	4030 (3348)	-0.3 (1.6)	3990 (3291)	-2.4 (2.1)
	13	5	0.4	7200 (0)	35.2 (13.9)	6945 (536)	-0.9 (2.1)	3921 (3435)	5.5 (31.1)	5544 (2756)	-3.3 (1.6)
	25	5	0.2	3243 (3374)	20.4 (7.8)	4514 (1564)	0.8 (2.0)	2699 (3362)	-0.1 (0.8)	2995 (3513)	-0.7 (1.3)
	25	5	0.4	3480 (3315)	25.2 (8.2)	4911 (1590)	0.2 (1.4)	2503 (3261)	-0.6 (1.0)	2998 (3510)	-1.1 (1.6)
	25	10	0.2	3133 (3347)	18.0 (9.5)	6594 (843)	1.3 (2.0)	1613 (2828)	0.1 (1.0)	2453 (3220)	-0.4 (1.1)
	25	10	0.4	3292 (3339)	21.8 (6.7)	6723 (740)	1.1 (2.1)	2513 (3309)	-0.3 (0.9)	2767 (3430)	-0.9 (1.6)
Large (100)	25	5	0.2	28800 (0)	38.5 (11.2)	16298 (3957)	-1.9 (2.7)	28799 (2)	0.3 (3.9)	28745 (106)	-4.7 (2.5)
	25	5	0.4	28800 (0)	42.6 (13.4)	21962 (4441)	-4.8 (7.4)	28367 (3024)	-5.8 (7.5)	28744 (106)	-7.4 (7.2)
	25	10	0.2	28800 (0)	35.5 (11.0)	27100 (2502)	-1.7 (3.0)	28800 (0)	4.4 (7.7)	28513 (1629)	-6.4 (2.3)
	25	10	0.4	28800 (0)	38.6 (14.9)	28634 (709)	-4.9 (14.7)	28266 (3737)	-5.0 (15.4)	28745 (107)	-9.1 (14.2)
	50	10	0.2	28150 (3018)	49.2 (35.4)	19630 (5884)	-16.1 (34.4)	10453 (13217)	-17.2 (33.6)	6918 (12197)	-18.7 (33.3)
	50	10	0.4	28506 (1416)	60.5 (0.0)	21710 (5768)	-15.1 (26.2)	8462 (12633)	-15.1 (26.2)	6958 (12177)	-15.1 (26.2)
	50	20	0.2	28325 (2151)	9.6 (0.0)	27592 (2690)	0.0 (0.9)	10654 (13217)	2.2 (9.8)	6921 (12195)	-1.6 (3.6)
	50	20	0.4	28699 (535)	22.5 (15.1)	27802 (2679)	-9.7 (13.9)	8032 (12175)	-11.3 (15.5)	6982 (12164)	-11.3 (15.5)

Table 4 – Competitive results for MIP, GA, and Benders' decomposition approaches

Instances (tasks)	Stations	Robots	Flexibility	MIP		GA		Benders 1						Benders 4					
				#feas	#opt	Obj ≤ MIP	CPU ≤ MIP	#feas	#opt	Obj ≤ MIP	Obj ≤ GA	CPU ≤ MIP	CPU ≤ GA	#feas	#opt	Obj ≤ MIP	Obj ≤ GA	CPU ≤ MIP	CPU ≤ GA
Small (20)	5	1	0.2	50	24	37	25	50	50	50	50	50	50	50	50	50	50	50	50
	5	1	0.4	50	25	33	25	50	50	50	50	50	48	50	50	50	50	50	50
	5	2	0.2	50	25	31	25	50	50	50	50	50	49	50	50	50	50	50	50
	5	2	0.4	50	25	30	25	50	50	50	50	50	50	50	50	50	50	50	49
	10	2	0.2	50	37	42	14	50	50	50	50	50	50	50	48	50	50	50	45
	10	2	0.4	50	37	40	13	50	50	50	50	50	50	50	43	50	50	50	41
	10	4	0.2	50	39	42	11	50	49	50	50	49	49	50	49	50	50	50	49
	10	4	0.4	50	39	31	12	50	50	50	50	49	49	50	41	50	50	47	40
Medium (50)	13	3	0.2	50	2	37	49	50	24	37	29	50	24	50	23	50	50	50	21
	13	3	0.4	50	0	42	50	50	25	47	40	50	27	50	19	50	49	50	20
	13	5	0.2	50	1	31	49	50	25	39	32	50	41	50	25	50	49	50	40
	13	5	0.4	50	0	34	50	50	24	46	38	50	48	50	14	50	49	50	43
	25	5	0.2	50	29	35	21	50	35	46	48	50	35	50	29	50	50	50	32
	25	5	0.4	50	28	37	22	50	35	50	49	50	39	50	29	50	50	50	34
	25	10	0.2	50	30	29	20	50	40	44	48	50	50	50	34	50	50	50	50
	25	10	0.4	50	29	32	21	50	34	45	48	49	50	50	31	50	50	50	50
Large (100)	25	5	0.2	42	0	42	50	50	0	29	20	50	1	50	0	50	50	50	1
	25	5	0.4	44	0	47	50	50	0	49	34	50	9	50	0	50	50	50	9
	25	10	0.2	44	0	38	50	50	0	21	13	50	29	50	1	50	50	50	30
	25	10	0.4	45	0	41	50	50	1	35	23	50	46	50	0	50	50	50	46
	50	10	0.2	7	4	49	48	50	35	50	38	50	40	50	38	50	50	50	45
	50	10	0.4	4	3	50	49	50	34	50	47	50	45	50	38	50	50	50	47
	50	20	0.2	6	5	49	47	50	33	49	39	49	48	50	38	50	50	50	50
	50	20	0.4	6	3	49	47	50	38	50	45	50	50	50	38	50	50	50	50

5. Conclusions

This paper presents exact methods for the assembly line balancing problem allowing the use of collaborative robots (cobots) to aid in the assembly. The proposed exact methods are based on combinatorial Benders' decomposition, which splits the problem in master and subproblems that are iteratively solved. As the division of the original problem can be performed in multiple forms, three different decompositions are proposed, implemented, and tested. Furthermore, a fourth version of the algorithm combining local search is proposed.

Cobots represent a low threshold means of partial automation of assembly tasks and are, therefore, an important lever for further efficiency improvements in manufacturing industries. The major advantage of cobots is their capability to work closely with human workers. Therefore, they can either conduct assembly tasks individually in parallel to the human worker or collaborate with her on the identic task. However, the associated decisions line planners have to consider induce a high complexity to the assembly line balancing problem. Consequently, solution procedures need to provide high-quality suggestions for the decision-makers. Since cobots are frequently mobile and can be redeployed quickly, the computational effort should additionally be small to facilitate the cobots efficient use in industries.

To this end, we compare our exact procedures to the previously proposed exact and heuristic approaches of Weckenborg et al. (2020). Our proposed algorithms provide better results on average in terms of both solution quality and CPU time. One version of the decomposition algorithm (Benders 1) can solve 784 instances out of the 1200 instance dataset to optimality.

However, the question of which decomposition strategy works best is not clearly answered. The way of dividing variables of the original problem in master and subproblem affects multiple aspects of the algorithm. For the obtained results, one version of the decomposition algorithm (Benders 4) is able to provide most of the best upper bounds. This algorithm, nonethe-

less, can prove the optimality of fewer instances in comparison to other approaches. Furthermore, the profile curves of the algorithms show that the relative performance of the decomposition depends on the size of the instances, implying that no decomposition approach seems to dominate all others.

In the future, research on the comparison of decomposition strategies can be extended to other problems. Moreover, instances with a small number of stations and, therefore, more tasks per station proved to be harder to solve and may be of interest to further algorithm developments. In practice, collaborative robots are also discussed as a promising lever to relieve workers from biomechanical load and therefore serve an ergonomic purpose (Weckenborg, Thies, and Spengler 2022; Stecke and Mokhtarzadeh 2022). However, evaluating the distribution of biomechanical load between human and cobot when tasks are performed collaboratively remains a major challenge for future research.

Funding: This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data availability statement: The authors confirm that the data supporting the findings of this study are available within the article [and/or] its supplementary materials.

6. References

Aghajani, M., R. Ghodsi, and B. Javadi. 2014. “Balancing of robotic mixed-model two-sided assembly line with robot setup times.” *The International Journal of Advanced Manufacturing Technology* 74 (5-8): 1005–16. doi:10.1007/s00170-014-5945-x.

Akpinar, Sener, Atabak Elmi, and Tolga Bektaş. 2017. “Combinatorial Benders cuts for assembly line balancing problems with setups.” *European Journal of Operational Research* 259 (2): 527–37. doi:10.1016/j.ejor.2016.11.001.

Antonelli, Dario, Sergey Astanin, and Giulia Bruno. 2016. “Applicability of Human-Robot Collaboration to Small Batch Production.” In *Collaboration in a Hyperconnected World*. Vol. 480, edited by Hamideh Afsarmanesh, Luis M. Camarinha-Matos, and António Lucas Soares, 24–32. IFIP Advances in Information and Communication Technology. Cham: Springer International Publishing.

Araújo, Felipe F., Alysson M. Costa, and Cristóbal Miralles. 2012. “Two extensions for the ALWABP: Parallel stations and collaborative approach.” *International Journal of Production Economics* 140 (1): 483–95. doi:10.1016/j.ijpe.2012.06.032.

Battaïa, O., and A. Dolgui. 2013. “A taxonomy of line balancing problems and their solution approaches.” *International Journal of Production Economics* 142 (2): 259–77. doi:10.1016/j.ijpe.2012.10.020.

Becker, C., and A. Scholl. 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.” *Numer. Math.* 4 (1): 238–52. doi:10.1007/BF01386316.

BMAS/BAuA. 2018. “Sicherheit und Gesundheit bei der Arbeit - Berichtsjahr 2017.” Accessed July 05, 2020. https://bit.ly/3727bjl.

Bosch. 2021. “Bosch APAS – flexible robots collaborate in Industry 4.0.” Accessed November 03, 2021. https://bit.ly/3GJgaYf.

Boysen, Nils, Malte Fliedner, and Armin Scholl. 2008. “Assembly line balancing: Which model to use when?” *International Journal of Production Economics* 111 (2): 509–28. doi:10.1016/j.ijpe.2007.02.026.

Brigl, Silke. 2017. “BMW Group Harnesses Potential of Innovative Automation and Flexible Assistance Systems in Production.” Accessed November 03, 2021. https://bit.ly/2KwLyjA.

Calzavara, Martina, Daria Battini, David Bogataj, Fabio Sgarbossa, and Ilenia Zennaro. 2020. “Ageing workforce management in manufacturing systems: state of the art and future research agenda.” *International Journal of Production Research* 58 (3): 729–47. doi:10.1080/00207543.2019.1600759.

Caserta, Marco, and Stefan Voß. 2021. “Accelerating mathematical programming techniques with the corridor method.” *International Journal of Production Research* 59 (9): 2739–71. doi:10.1080/00207543.2020.1740343.

Chen, Fei, Kosuke Sekiyama, Hironobu Sasaki, Jian Huang, Baiqing Sun, and Toshio Fukuda. 2011. “Assembly Strategy Modeling and Selection for Human and Robot Coordinated Cell Assembly.” *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems, San Francisco, CA,* 4670–75. doi:10.1109/IROS.2011.6048306.

Chutima, Parames. 2020. “A comprehensive review of robotic assembly line balancing problem.” *J Intell Manuf.* doi:10.1007/s10845-020-01641-7.

Çil, Zeynel A., Zixiang Li, Suleyman Mete, and Eren Özceylan. 2020. “Mathematical model and bee algorithms for mixed-model assembly line balancing problem with physical human–robot collaboration.” *Applied Soft Computing* 93: 106394. doi:10.1016/j.asoc.2020.106394.

Codato, Gianni, and Matteo Fischetti. 2006. “Combinatorial Benders' Cuts for Mixed-Integer Linear Programming.” *Operations Research* 54 (4): 756–66. doi:10.1287/opre.1060.0286.

Costa, Alysson M., Jean-François Cordeau, Bernard Gendron, and Gilbert Laporte. 2012. “Accelerating benders decomposition with heuristicmaster problem solutions.” *Pesqui. Oper.* 32 (1): 3–20. doi:10.1590/S0101-74382012005000005.

Dalle Mura, Michela, and Gino Dini. 2019. “Designing assembly lines with humans and collaborative robots: A genetic approach.” *CIRP Annals - Manufacturing Technology* 68 (1): 1–4. doi:10.1016/j.cirp.2019.04.006.

Fraunhofer IAO. 2016. “Lightweight robots in manual assembly: Best to start simply! .” Examining companies' initial experiences with lightweight robots. Accessed November 03, 2021. https://bit.ly/3bCmLW4.

Furugi, Ahad. 2022. “Sequence-dependent time- and cost-oriented assembly line balancing problems: a combinatorial Benders’ decomposition approach.” *Engineering Optimization* 54 (1): 170–84. doi:10.1080/0305215X.2021.1953003.

Hashemi-Petroodi, S. E., Alexandre Dolgui, Sergey Kovalev, Mikhail Y. Kovalyov, and Simon Thevenin. 2020a. “Workforce reconfiguration strategies in manufacturing systems: a state of the art.” *International Journal of Production Research*, 1–24. doi:10.1080/00207543.2020.1823028.

Hashemi-Petroodi, S. E., Simon Thevenin, Sergey Kovalev, and Alexandre Dolgui. 2020b. “Operations management issues in design and control of hybrid human-robot collaborative manufacturing systems: a survey.” *Annual Reviews in Control* 49: 264–76. doi:10.1016/j.arcontrol.2020.04.009.

Helms, E., R. D. Schraft, and M. Hägele. 2002. “rob@work: Robot Assistant in Industrial Environments.” *Proceedings / IEEE ROMAN 2002, 11th IEEE International Workshop on Robot and Human Interactive Communication, Berlin, Germany*, 399–404.

Huang, Dian, Zhaofang Mao, Kan Fang, and Biao Yuan. 2021. “Combinatorial Benders decomposition for mixed-model two-sided assembly line balancing problem.” *International Journal of Production Research*, 1–27. doi:10.1080/00207543.2021.1901152.

IFR. 2018. “SHAD opts for UR robot arms to optimize its manufacturing processes.” Accessed November 03, 2021. http://bit.ly/2CRXuDH.

Janardhanan, Mukund N., Zixiang Li, and Peter Nielsen. 2019. “Model and migrating birds optimization algorithm for two-sided assembly line worker assignment and balancing problem.” *Soft Computing* 23: 11263–76. doi:10.1007/s00500-018-03684-8.

Koltai, Tamás, Imre Dimény, Viola Gallina, Alexander Gaal, and Chiara Sepe. 2021. “An analysis of task assignment and cycle times when robots are added to human-operated assembly lines, using mathematical programming models.” *International Journal of Production Economics* 242: 108292. doi:10.1016/j.ijpe.2021.108292.

Krüger, J., T. K. Lien, and A. Verl. 2009. “Cooperation of human and machines in assembly lines.” *CIRP Annals - Manufacturing Technology* 58 (2): 628–46. doi:10.1016/j.cirp.2009.09.009.

Lai, Tsung-Chyan, Yuri N. Sotskov, Alexandre Dolgui, and Aksana Zatsiupa. 2016. “Stability radii of optimal assembly line balances with a fixed workstation set.” *International Journal of Production Economics* 182: 356–71. doi:10.1016/j.ijpe.2016.07.016.

Li, Zixiang, Mukund N. Janardhanan, and Qiuhua Tang. 2021. “Multi-objective migrating bird optimization algorithm for cost-oriented assembly line balancing problem with collaborative robots.” *Neural Computing and Applications;*, no. 33: 8575–96. doi:10.1007/s00521-020-05610-2.

Lopes, Thiago C., Celso G. S. Sikora, Rafael G. Molina, Daniel Schibelbain, Rodrigues, Luiz Carlos de Abreu, and Leandro Magatão. 2017. “Balancing a robotic spot welding manufac-

turing line: An industrial case study." *European Journal of Operational Research* 263 (3): 1033–48. doi:10.1016/j.ejor.2017.06.001.

Michels, Adalberto S., Thiago C. Lopes, and Leandro Magatão. 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. doi:10.1016/j.orp.2020.100163.

Michels, Adalberto S., Thiago C. Lopes, Celso G. S. Sikora, and Leandro Magatão. 2018. "The Robotic Assembly Line Design (RALD) problem: Model and case studies with practical extensions." *Computers & Industrial Engineering* 120: 320–33.

Michels, Adalberto S., Thiago C. Lopes, Celso G. S. Sikora, and Leandro Magatão. 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. doi:10.1016/j.ejor.2019.05.001.

Miralles, Cristóbal, Jose P. García-Sabater, Carlos Andrés, and Manuel Cardos. 2007. "Advantages of assembly lines in Sheltered Work Centres for Disabled. A case study." *International Journal of Production Economics* 110 (1-2): 187–97. doi:10.1016/j.ijpe.2007.02.023.

Miralles, Cristóbal, José P. García-Sabater, Carlos Andrés, and Manuel Cardós. 2008. "Branch and bound procedures for solving the Assembly Line Worker Assignment and Balancing Problem: Application to Sheltered Work centres for Disabled." *Discrete Applied Mathematics* 156 (3): 352–67. doi:10.1016/j.dam.2005.12.012.

Naderi, Bahman, Ahmed Azab, and Katayoun Borooshan. 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): 643–61. doi:10.1080/00207543.2018.1476786.

Otto, A., C. Otto, and A. Scholl. 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. doi:10.1016/j.ejor.2012.12.029.

Rabbani, Masoud, Seyedeh Z. B. Behbahan, and Hamed Farrokhi-Asl. 2020. “The Collaboration of Human-Robot in Mixed-Model Four-Sided Assembly Line Balancing Problem.” *J Intell Robot Syst* 100 (1): 71–81. doi:10.1007/s10846-020-01177-1.

Rahmaniani, Ragheb, Teodor G. Crainic, Michel Gendreau, and Walter Rei. 2017. “The Benders decomposition algorithm: A literature review.” *European Journal of Operational Research* 259 (3): 801–17. doi:10.1016/j.ejor.2016.12.005.

Robotiq. 2021. “Collaborative Robot: Buyer's Guide.” Accessed November 03, 2021. https://bit.ly/2Tpxjz7.

Rubinovitz, J., and J. Bukchin. 1991. “Design and balancing of robotic assembly lines.” *Proceedings of the Fourth World Conference on Robotics Research: September 17-19, 1991, Pittsburgh, Pennsylvania*.

Rubinovitz, J., J. Bukchin, and E. Lenz. 1993. “RALB - A Heuristic Algorithm for Design and Balancing of Robotic Assembly Lines.” *CIRP Annals - Manufacturing Technology* 42 (1): 497–500.

Samouei, Parvaneh, and Jalal Ashayeri. 2019. “Developing optimization & robust models for a mixed-model assembly line balancing problem with semi-automated operations.” *Applied Mathematical Modelling* 72: 259–75. doi:10.1016/j.apm.2019.02.019.

Schillmoeller, Sandra. 2013. “Innovative human-robot cooperation in BMW Group Production.” Accessed November 03, 2021. http://bit.ly/29NxTA7.

Scholl, A., and C. Becker. 2006. “State-of-the-art exact and heuristic solution procedures for simple assembly line balancing.” *European Journal of Operational Research* 168 (3): 666–93.

Scholl, Armin. 1999. *Balancing and sequencing of assembly lines.* Second revised edition. Heidelberg: Physica.

Scholl, Armin, and Robert Klein. 1997. “SALOME: A Bidirectional Branch-and-Bound Procedure for Assembly Line Balancing.” *INFORMS Journal on Computing* 9 (4): 319–34. doi:10.1287/ijoc.9.4.319.

Sikora, Celso G. S. 2021. “Benders’ decomposition for the balancing of assembly lines with stochastic demand.” *European Journal of Operational Research* 292 (1): 108–24. doi:10.1016/j.ejor.2020.10.019.

Sikora, Celso G. S. 2022. “Balancing Under Full Sequencing Control.” In *Assembly-Line Balancing under Demand Uncertainty*, edited by Celso G. S. Sikora, 65–95. Gabler Theses. Wiesbaden: Springer Fachmedien Wiesbaden.

Sikora, Celso G. S., Thiago C. Lopes, and Leandro Magatão. 2017. “Traveling worker assembly line (re)balancing problem: Model, reduction techniques, and real case studies.” *European Journal of Operational Research* 259 (3): 949–71. doi:10.1016/j.ejor.2016.11.027.

Sotskov, Yuri N., Alexandre Dolgui, Tsung-Chyan Lai, and Aksana Zatsiupa. 2015. “Enumerations and stability analysis of feasible and optimal line balances for simple assembly lines.” *Computers & Industrial Engineering* 90: 241–58. doi:10.1016/j.cie.2015.08.018.

Stecke, Kathryn E., and Mahdi Mokhtarzadeh. 2022. “Balancing collaborative human–robot assembly lines to optimise cycle time and ergonomic risk.” *International Journal of Production Research* 60 (1): 25–47. doi:10.1080/00207543.2021.1989077.

Universal Robots. 2021. “Cobots offer game changing benefits.” Accessed November 03, 2021. https://bit.ly/2vxAEQS.

Volkswagen. 2017. “Human robot cooperation: KLARA facilitates greater diversity of versions in production at Audi.” Accessed November 03, 2021. https://bit.ly/3gesLFy.

Weckenborg, C., C. Thies, and T. S. Spengler. 2022. “Harmonizing ergonomics and economics of assembly lines using collaborative robots and exoskeletons.” *Journal of Manufacturing Systems* 62: 681-702.

Weckenborg, Christian. 2021. *Modellbasierte Gestaltung von Fließproduktionssystemen im Spannungsfeld von Ergonomie und Ökonomie.* Wiesbaden: Springer Gabler.

Weckenborg, Christian, Karsten Kieckhäfer, Christoph Müller, Martin Grunewald, and Thomas S. Spengler. 2020. “Balancing of assembly lines with collaborative robots.” *Business Research* 13 (1): 93–132.

Weckenborg, Christian, and Thomas S. Spengler. 2019. “Assembly Line Balancing with Collaborative Robots under consideration of Ergonomics: a cost-oriented approach.” *IFAC-PapersOnLine* 52 (13): 1860–65.

Yaphiar, Susanto, Cahyadi Nugraha, and Anas Ma’ruf. 2020. “Mixed Model Assembly Line Balancing for Human-Robot Shared Tasks.” In *IMEC-APCOMS 2019: Proceedings of the 4th International Manufacturing Engineering Conference and the 5th Asia Pacific Conference on Manufacturing Systems*, edited by Muhammed N. Osman Zahid, 245–52. Lecture Notes in Mechanical Engineering Ser. Singapore: Springer Singapore Pte. Limited.

Yazgan, Harun R., Ismail Beypinar, Semra Boran, and Ceren Ocak. 2011. “A new algorithm and multi-response Taguchi method to solve line balancing problem in an automotive industry.” *Int J Adv Manuf Technol* 57 (1-4): 379–92. doi:10.1007/s00170-011-3291-9.

Zohali, Hassan, Bahman Naderi, and Vahid Roshanaei. 2021. “Solving the Type-2 Assembly Line Balancing with Setups Using Logic-Based Benders Decomposition.” *INFORMS Journal on Computing.* doi:10.1287/ijoc.2020.1015.