Combinatorial Benders Cuts for Assembly Line Balancing Problems with Setups

Sener Akpinar*[1], Atabak Elmi[2], and Tolga Bektaş[3]

[1] *Dokuz Eylul University, Faculty of Engineering, Department of Industrial Engineering, Izmir, Turkey*

[2] *Gaziantep University, Faculty of Engineering, Department of Industrial Engineering, Gaziantep, Turkey*

[3] *Southampton Business School, Centre for Operational Research, Management Science and Information Systems (CORMSIS), University of Southampton, Southampton, United Kingdom*

April 6, 2016

Abstract

The classical assembly line balancing problem consists of assigning assembly work to workstations. In the presence of setup times that depend on the sequence of tasks assigned to each workstation, the problem becomes more complicated given that two interdependent problems, namely assignment and sequencing, must be solved simultaneously. The hierarchical nature of these two problems also suggest a natural decomposition of the problem. This paper adopts such an approach and describes an exact algorithm based on Benders decomposition to solve both simple and mixed-model assembly line balancing problems with setups. The algorithm is tested on a set of benchmark instances and numerically compared against a mixed-integer linear programming formulation of the problem solved using an off-the-shelf optimizer.

Keywords: Combinatorial optimization, type-I assembly line balancing problem, sequence-dependent setup times, Benders decomposition, combinatorial Benders cuts

1 Introduction

Assembly lines (ALs) are special flow-based production systems. The design of such systems gives rise to the assembly line balancing problem (ALBP), which consists of assigning assembly tasks to a number of workstations in order to optimize a given objective. Early designs of assembly lines were for a single product to be produced in high volumes with the corresponding problem known as the simple assembly line balancing problem (SALBP). Single-model assembly lines did not prove efficient for products with a high variety, required by a consumer-centric market, necessitating a high degree of flexibility in the manufacturing system [29]. To address this need, and bearing in mind the high investment and maintenance cost of assembly lines, manufacturers started to produce a single model of

*Corresponding author's Email: sener.akpinar@deu.edu.tr; Tel: +90 232 301 76 32

product with different features or several models on a single assembly line. These developments gave rise to mixed-model assembly lines with the corresponding design problem named as the mixed-model assembly line balancing problem (MMALBP).

AL design problems come in two flavors: (Type-I): design of a new assembly line for which the demand is known or assumed to be easily forecasted, and (Type-II): redesign of an existing assembly line to accommodate any changes in the assembly process or in the product range. Type-I (resp. Type-II) problems deal with the assignment of tasks to workstations with the aim of minimizing the number of workstations (resp. the cycle time) for a predetermined cycle time (resp. predetermined number of workstations) by respecting the precedence relations of the tasks involved in the assembly. The rest of this paper is concerned with Type-I problems, for which reason we will not explicitly spell out the type unless otherwise stated.

The MMALBP is NP-hard [9], complex, and CPU time-consuming to solve to optimality [5] (see [6] for a recent survey on solution approaches on ALBPs). The computational difficulty of solving the MMALBP with setups to optimality using a commercial software was shown in [2].

In order to optimally solve the MMALBP with setups in an efficient manner, we describe, in this paper, a Benders decomposition algorithm that exploits the two subproblems, in particular assignment of tasks to workstations and sequencing of tasks within each workstation. We reformulate the original problem as a master assignment problem with an exponential number of feasibility constraints, following which we search for infeasible assignments using the sequencing problem as a slave problem, and forbid any such assignments through infeasibility cuts in the master problem until an optimal solution is identified. The proposed algorithm is numerically shown to be much faster than an off-the-shelf optimizer and is able to solve larger-scale instances than the state-of-the art.

The remainder of the paper is organized as follows. A summary of the relevant literature on ALBPs with setups is given in Section 2. A formal problem definition and formulations are presented in Section 3. The Benders decomposition algorithm is described in Section 4. Computational results are given in Section 5. The paper concludes with some remarks in Section 6.

2 A Summary of Relevant Literature on ALBPs with Setups

Setup times considerations for assembly line balancing problems have first appeared in [4] and [21], where the SALBP was extended to incorporate sequence dependent setup times between tasks, for which the authors described a mathematical programming model of the problem and proposed several heuristics including GRASP. Priority rules based solution procedures were developed in [17] to solve the problem, however these procedures were not effective enough in solving large-size test problems with more than 100 tasks. Mixed-integer programming formulations of a similar problem were developed in [23] where the authors stated that solving the problem with standard MIP solvers is not an effective solution method. The problem was extended in [24] by introducing different backward and forward setups, for which the authors developed a mixed-binary linear formulation and proposed some effective solution procedures. Another mathematical formulation of the SALBP was proposed in [14], along with a combination of a particle swarm optimization algorithm and variable neighbourhood search. A hybrid genetic algorithm was proposed in [31] for a related problem. For the Type-II version of the problem, [25] described a mathematical model similar to that in [4], and a simulated annealing (SA) algorithm. A mixed-integer programming formulation for another version of the problem where setup times were considered for a two sided assembly line was presented in [20], for which the authors proposed a heuristic. The mixed-model version of the assembly line balancing problem with setups was studied in [18], and the variant with sequence dependent setup times between tasks was studied in [2], and hybrid meta-heuristic algorithms, including a combination of ant colony optimization and genetic algorithm and a multiple colony hybrid bees algorithm were described in [1] and [3].

3 Problem Definition and Formulations

This section presents a formal definition of the main problem considered in this study, followed by a mathematical programming formulation.

3.1 Formal problem description

Mixed-model assembly lines are used to either produce a single model of product with different features or several models on a single assembly line. Each model comes with a specific set of precedence relations between its tasks which can be combined into a precedence diagram for all models. Hence, the combined precedence diagram has N tasks that must be assigned to a maximum of S workstations under a capacity constraint defined by the cycle time C of the assembly line. The assignment is subject to a precedence constraint defined by a parameter P_{ij} derived from the combined precedence diagram that equals 1 if task i must precede task j, and 0 otherwise. Each task i for model m has a processing time T_{im}, and this may vary between the M models assembled on the line. The MMALBP consists of assigning N tasks associated with M models to workstations so as to minimize the number of the workstations used. The SALBP is a special case of the MMALBP where $M = 1$.

3.2 Mathematical programming formulations

This section presents two formulations, the first for the MMALBP, which then forms the basis for the second model of the MMALBP with setups.

3.2.1 Basic formulation of the MMALBP

To model the MMALBP, we use a binary variable Y_{is} that is equal to 1 if task i is assigned to a workstation s, and 0 otherwise, and minimize the number of workstations used. Using these variables the MMALBP can be formulated as the following binary programming model.

$$\text{Minimize} \qquad \sum_{s=1}^{S} sY_{Ns} \tag{1}$$

subject to

$$\sum_{s=1}^{S} Y_{is} = 1 \qquad i \in \{1, ..., N\} \tag{2}$$

$$\left(\sum_{s=1}^{S} sY_{is} - \sum_{s=1}^{S} sY_{js}\right) P_{ij} \leq 0 \qquad i, j \in \{1, ..., N\}; i \neq j \tag{3}$$

$$\sum_{i=1}^{N} Y_{is}T_{im} \leq C \qquad s \in \{1, ..., S\}; m \in \{1, ..., M\}. \tag{4}$$

The objective function (1) minimizes the number of used workstations. Constraints (2) ensure the assignment of each task to exactly one workstation. Constraints (3) are used to guarantee the precedence relations between tasks. Capacity constraints (4) ensure that the workload of each used workstation does not exceed the cycle time.

3.2.2 An improved formulation for the MMALBP with setups

A mixed integer linear programming formulation of the MMALBP with setups (MMALBPS) was proposed in [2] (see Appendix A) but this formulation suffered from a significant number of "big-M"type constraints. In this section, we present an improved version of this formulation, which has a reduced number of "big-M"type constraints. The existing model in [2] considers workstation parallelization and zoning constraints, which we do not consider in this paper, but this does not detract from the applicability of the new formulation. The assumptions and the notation of the model are given below, and in Table 1, respectively.

- A set of similar models of a product are assembled on a straight line.
- The combined precedence diagram contains N tasks.
- A task can be assigned to exactly one workstation.
- Tasks common to several models must be performed on the same workstation.
- Processing time of a common task may be different among the different models.
- Task processing times and setup times between tasks are deterministic and known in advance.

Table 1: Model notation

	Notation	Definition
Parameters	N	Number of tasks
	M	Number of models simultaneously assembled on the line
	S	Maximum number of workstations
	C	Cycle time
	T_i	Processing time of task $i \in \{1, 2, ..., N\}$ on model $m \in \{1, 2, ..., M\}$
	$Q_{im} \in \{0, 1\}$	Equals 1 if processing time of task $i \in \{1, 2, ..., N\}$ is positive for model $m \in \{1, 2, ..., M\}$, and 0 otherwise
	F_{ijm}	Forward set-up time between task $i \in \{1, 2, ..., N\}$ and $j \in \{1, 2, ..., N\}$ on model $m \in \{1, 2, ..., M\}$
	B_{ijmn}	Backward set-up time between task $i \in \{1, 2, ..., N\}$ of model $m \in \{1, 2, ..., M\}$ and task $j \in \{1, 2, ..., N\}$ of model $n \in \{1, 2, ..., M\}$
	$P_{ij} \in \{0, 1\}$	Equals 1 if task $i \in \{1, 2, ..., N\}$ must precede task $j \in \{1, 2, ..., N\}$, and 0 otherwise
Decision Variables	$Y_{is} \in \{0, 1\}$	Equals 1 if task $i \in \{1, 2, ..., N\}$ is assigned to workstation $s \in \{1, 2, ..., S\}$, and 0 otherwise
	$A_s \in \{0, 1\}$	Equals 1 if station $s \in \{1, 2, ..., S\}$ is active, and 0 otherwise
	$w_{ijs} \in \{0, 1\}$	Equals 1 if task $i \in \{1, 2, ..., N\}$ precede task $j \in \{1, 2, ..., N\}$ at workstation $s \in \{1, 2, ..., S\}$, and 0 otherwise
	$X_{ijms} \in \{0, 1\}$	Equals 1 if task $j \in \{1, 2, ..., N\}$ directly follows task $i \in \{1, 2, ..., N\}$ on model $m \in \{1, 2, ..., M\}$ in the forward direction in workstation $s \in \{1, 2, ..., S\}$, and 0 otherwise
	$Z_{ijmns} \in \{0, 1\}$	Equals 1 if $i \in \{1, 2, ..., N\}$ is the last task of model $m \in \{1, 2, ..., M\}$ and $j \in \{1, 2, ..., N\}$ is the first task of model $n \in \{1, 2, ..., M\}$ in workstation $s \in \{1, 2, ..., S\}$, and 0 otherwise

$$\text{Minimize} \qquad OBJ_1 = \sum_{s=1}^{S} A_s \tag{5}$$

subject to

$$\sum_{s=1}^{S} Y_{is} = 1 \qquad i \in \{1, ..., N\} \tag{6}$$

$$\left(\sum_{s=1}^{S} sY_{is} - \sum_{s=1}^{S} sY_{js} \right) P_{ij} \leq 0 \qquad i, j \in \{1, ..., N\}; i \neq j \tag{7}$$

$$\sum_{i=1}^{N} \left(Y_{is} T_{im} + \sum_{j=1}^{N} (X_{jms} F_{ijm} + Z_{ijmns} B_{ijmn}) \right) \leq C \qquad s \in \{1, ..., S\}; m, n \in \{1, ..., M\} \tag{8}$$

$$A_s \geq Y_{is} \qquad i \in \{1, ..., N\}; s \in \{1, ..., S\} \tag{9}$$

$$A_s \geq A_{s+1} \qquad s \in \{1, ..., S-1\} \tag{10}$$

$$w_{ijs} + w_{jis} + X_{ijms} + X_{jims} + Z_{ijmns} + Z_{jimns} \leq 3(1 - Y_{is} + Y_{js}) \qquad i, j \in \{1, ..., N\}; m, n \in \{1, ..., M\}; s \in \{1, ..., S\} \tag{11}$$

$$w_{ijs} + w_{jis} + X_{ijms} + X_{jims} + Z_{ijmns} + Z_{jimns} \leq 3(1 + Y_{is} - Y_{js}) \qquad i, j \in \{1, ..., N\}; m, n \in \{1, ..., M\}; s \in \{1, ..., S\} \tag{12}$$

$$w_{ijs} + w_{jis} + X_{ijms} + X_{jims} + Z_{ijmns} + Z_{jimns} \leq 3(Y_{is} + Y_{js}) \qquad i, j \in \{1, ..., N\}; m, n \in \{1, ..., M\}; s \in \{1, ..., S\} \tag{13}$$

$$P_{ij}\left(Y_{is}+Y_{js}-1\right) \le w_{ijs} \qquad i,j \in \{1,...,N\} : i \neq j; s \in \{1,...,S\} \tag{14}$$

$$w_{iis} = 0 \qquad i \in \{1,...,N\}; s \in \{1,...,S\} \tag{15}$$

$$(w_{iks} + w_{kjs} - 1 \le w_{ijs}) \qquad i,j,k \in \{1,...,N\} : i \neq j \neq k; s \in \{1,...,S\} \tag{16}$$

$$\left|\sum_{k=1}^{N}\sum_{l=1}^{N} w_{kls} - \sum_{j|j<i}^{N} j\right| \le N\left|i - \sum_{p=1}^{N} Y_{ps}\right| \qquad i \in \{1,...,N\}; s \in \{1,...,S\} \tag{17}$$

$$\sum_{j=1}^{N}\sum_{s=1}^{S} X_{ijms} \le 1 \qquad i \in \{1,...,N\}; m \in \{1,...,M\} \tag{18}$$

$$X_{ijms} + X_{jims} \le 1 \qquad i,j \in \{1,...,N\} : i \neq j; m \in \{1,...,M\}; s \in \{1,...,S\} \tag{19}$$

$$\sum_{s=1}^{S} X_{iims} = 0 \qquad i \in \{1,...,N\}; m \in \{1,...,M\} \tag{20}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} Z_{ijmns} \le 1 \qquad m,n \in \{1,...,M\}; s \in \{1,...,S\} \tag{21}$$

$$X_{ijms} \le 1 - Z_{ijmns} \qquad i,j \in \{1,...,N\} : i \neq j; m,n \in \{1,...,M\}; s \in \{1,...,S\} \tag{22}$$

$$(Y_{is}Q_{im} + Y_{js}Q_{jn} - 1) - \left(\sum_{k=1}^{N}(w_{iks}Q_{km})\right) - \left|\sum_{l=1}^{N}(Y_{ls}Q_{ln}) - \sum_{p=1}^{N}(w_{jps}Q_{pn}) - 1\right| \le Z_{ijmns}$$

$$i,j \in \{1,...,N\}; m,n \in \{1,...,M\}; s \in \{1,...,S\} \tag{23}$$

$$(Y_{is}Q_{im} + Y_{js}Q_{jm} - 1) - \left|\sum_{k=1}^{N}(w_{iks}Q_{km}) - \sum_{l=1}^{N}(w_{jls}Q_{lm}) - 1\right| \le X_{ijms}$$

$$i,j \in \{1,...,N\} : i \neq j; m \in \{1,...,M\}; s \in \{1,...,S\}. \tag{24}$$

The objective function (5) of the IP model minimizes the total number of active workstations. Constraint set (6) assigns each task to exactly one workstation. Constraint set (7) guarantees that a task can only be assigned to a workstation s if all of its predecessors are assigned to a workstation preceding s on the line or to workstation s. The workload of a workstation, expressed by the summation of the task processing times and the setup times, must not exceed a pre-determined cycle time for all models being assembled in that workstation. This capacity restriction is provided by the constraint set (8). Constraint set (9) allows the model to identify the active workstations. Constraints (10) ensure that the active workstations are in an ordered sequence. Constraint sets (11), (12) and (13) order tasks and assign setup operations between them if they are assigned to the same workstation. Any two tasks have to be ordered within a workstation due to their precedence relations which is ensured by constraints (14). Constraint set (15) prevents ordering a task with itself. The set of constraints (16) determines the proper orderings within any three tasks, i.e., if task i has been performed before task k and task k has been performed before task j, then task i would be performed before task j. The number of necessary performing orders between tasks in any workstation is calculated by the constraint set (17). The set of constraints (18) guarantees that each task in any workstation would have at most one immediate successor. It is possible to do only one forward setup operation between any pair of tasks due to constraints (19) and there would not be any forward setup operation between any task and itself due to constraints (20). Constraint set

(21) ensures that each workstation would have just one backward setup operation. If a backward setup operation has been assigned between any tasks pair then there would not be any forward setup operation, which is modeled by constraints (22). Finally, constraints (23) and (24) determine the backward and forward setup operations in any workstation, respectively. We note that the constraints (17), (23) and (24) are semi-linear due to the absolute value. A way of linearising these constraints are given in Appendix B. The formulation defined by (5)–(24) will be henceforth be referred to as IP.

4 A Benders Decomposition Algorithm

Benders Decomposition [7] is based on reformulating the original problem as a so-called master problem (MP) that has an exponential number of cuts, which are initially relaxed and separated in an iterative fashion using a so-called slave (or sub) problem. Benders Decomposition iterates between the master and slave problems until an optimal solution is identified.

Benders Decomposition and its variants have been successfully used to solve combinatorial optimization problems such as network design, mixed-integer linear programming, travelling salesman, strip packing problem. The application of Benders Decomposition to solve ALBPs is scarce, and the only studies we are aware of are [15] and [16], but they consider different problems to what we study here.

Given a mixed-integer linear program $P : \min\{c^T y + d^T x : Ay + Bx \geq b, y \geq 0, x \in X\}$ the BDA first fixes $\bar{x} \in X$, and then solves the slave problem $SP : \min\{c^T y : Ay \geq b - B\bar{x}, y \geq 0\}$, or alternatively the dual slave problem $SD : \max\{u^T(b - B\bar{x}) : u^T A \leq c, u \geq 0\}$. If SP has an optimal solution $\bar{u}$, then an optimality cut in the form of $z \geq \bar{u}^T(b - Bx)$ is constructed. If SP is unbounded, a feasibility cut in the form of $0 \geq \bar{u}^T(b - Bx)$ is formed. Both are gradually and iteratively introduced into the MP. However, the situation is different if SP is not a continuous problem, as is the case in our application.

In this work, we use the feasibility-seeking variant of the decomposition algorithm proposed by Benders [7] to solve the model (5)–(24). As stated by Côté et al., [11], for a special case of P where $c = 0$, the slave SP can be used as a feasibility check on the system $\{Ay + B\bar{x} \geq b, y \geq 0\}$. In particular, if $\bar{x}$ is not a feasible solution for at least one variable x_j causing infeasibility, then this variable must take a different value from $\bar{x}_j$. This condition can be modelled using a linear constraint and added to the MP. Some implementations look for the possible minimal subsets of variables that induce infeasibility in SP and derive a cut from these subsets rather than adding a cut containing all the x variables [11]. Such constraints are called combinatorial Benders cuts by [10], which do not require that SP is continuous. Otherwise, if $\bar{x}$ is a feasible solution for SP, then it is feasible and optimal for P.

We reformulate the MMALBPS by projecting variables w_{ijs}, X_{ijms} and Z_{ijmns} out of formulation (5)–(24) yielding the following master problem that only models the assignment problem of the assembly lines.

$MP(Initial)$: Objective function (5)

subject to

Constraints (6), (7), (4), (9) and (10).

Here the constraint set (10) is not necessary, but is used as it reduces the solution time of the model significantly. The main reason is that it acts as symmetry-breaking constraint. A computational analysis on the effect of this set of constraints will be given in Section 5.2.

The proposed algorithm is based on the observation that MMALBPS can be formulated by using two sets of assignment (Y_{is} and A_s) and one set of sequencing (w_{ijs}) variables. Two other sets of variables (X_{ijms} and Z_{ijmns}) are then used to determine the necessary setup operations between the tasks of the models assembled on the same line. In other words, the model first assigns the tasks to the active workstations and then determines the setup operations in each workstation by sequencing the tasks assigned to the related workstation.

We start by solving the $MP(Initial)$ to identify a solution $(\bar{Y}, \bar{A}) = \{(\bar{Y}_{11}, \ldots, \bar{Y}_{NS}), (\bar{A}_1, \ldots, \bar{A}_S)\}$, which induces a slave problem to check for feasibility of $(\bar{Y}, \bar{A})$. The slave problem $SP_s(\bar{Y}, \bar{A})$ decomposes for each workstation $s \in \{1, ..., S\}$ such that $\bar{A}_s \geq 0$, and is shown below.

$$\text{Minimize} \qquad OBJ_2 = \sum_{i=1}^{N} \left(\bar{Y}_{is} T_{im} + \sum_{j=1}^{N} (X_{jms} F_{ijm} + Z_{ijmns} B_{ijmn}) \right) \tag{25}$$

subject to

Constraints (11)–(16) with $Y = \bar{Y}$ and $A = \bar{A}$

$$\sum_{k=1}^{N} \sum_{l=1}^{N} w_{kls} = \sum_{j | j < \sum_{p=1}^{N} \bar{Y}_{ps}}^{N} j \qquad i \in \{1, ..., N\} \tag{26}$$

Constraints (18)–(24) with $Y = \bar{Y}$ and $A = \bar{A}$.

The $SP_s(\bar{Y}, \bar{A})$ is a sequencing problem, in particular it is a precedence constrained traveling salesman problem [13] that is known to be NP-Hard. If $SP_s(\bar{Y}, \bar{A})$ returns a feasible solution for all the used workstations, then an optimal solution of the original problem is obtained. Otherwise, the slave problem returns an infeasibility for at least one of the workstations, for which the following group of feasibility cuts is added to the master problem.

$$Cut_u^s \equiv \left\{ \sum_{i=1 | \bar{Y}_{is}=1}^{N} Y_{iu} \leq \left(\sum_{i=1}^{N} \bar{Y}_{is} \right) - 1 \right\} \qquad \forall u \in \{1, ..., S\} : SP_s(\bar{Y}, \bar{A}) \text{ is infeasible,} \tag{27}$$

where Cut_u^s is all the cuts that would be added to the MP originating from the infeasible slave problems at each iteration. The group of feasibility cuts (27) relate to a set of tasks that are assigned to a workstation in a given solution $(\bar{Y}, \bar{A})$ and are infeasible with respect to the capacity constraint. The set Cut_u^s contains S cuts, one for each workstation $s \in \{1, ..., S\}$, forbidding such tasks to be assigned to any of these workstations. The algorithm iterates in a similar way, where the master problem $MP(Initial)$, augmented with infeasibility cuts at a given iteration, takes the following form.

$$MP(Cutset)\text{: Minimize} \qquad OBJ_3 = \sum_{s=1}^{S} A_s \tag{28}$$

subject to

$$\sum_{s=1}^{S} Y_{is} = 1 \qquad i \in \{1, ..., N\} \tag{29}$$

$$\left(\sum_{s=1}^{S} sY_{is} - \sum_{s=1}^{S} sY_{js} \right) P_{ij} \leq 0 \qquad i, j \in \{1, ..., N\} \tag{30}$$

$$\sum_{i=1}^{N} (Y_{is} T_{im}) \leq C \qquad s \in \{1, ..., S\}; m \in \{1, ..., M\} \tag{31}$$

$$A_s \geq Y_{is} \qquad i \in \{1, ..., N\}; s \in \{1, ..., S\} \tag{32}$$

$$A_s \geq A_{s+1} \qquad s \in \{1, ..., S-1\} \tag{33}$$

$$Cut_c \in Cutset \qquad c \in \{1, ..., |Cutset|\}, \tag{34}$$

where $Cutset$ is the set of all feasibility cuts. A pseudo-code of the proposed Benders Decomposition Algorithm (BDA) is given in Algorithm 1.

Algorithm 1 Benders Decomposition Algorithm (BDA) for the MMALBPS

$[MP]$: Master Problem; $Cutset$: Set of generated feasibility cuts; AS: Number of Active Stations; $counter$ and $control$: User defined variables
Initialization: $Cutset = \emptyset$, $control \leftarrow 0$
Start
  While $(control == 0)$
    $counter \leftarrow 0$
    $AS \leftarrow 0$
    Solve $MP(Cutset)$. Let the solution be $(\bar{Y}, \bar{A})$
    For $s \in \{1, ..., S\}$
      If $(\bar{A}_s == 1)$
        $AS \leftarrow AS + 1$
        Solve $\left[SP_s(\bar{Y}, \bar{A})\right]$. Let the optimal value be $OBJ_2^{SP_s}$
        If $(OBJ_2^{SP_s} > C)$
          $Cutset \leftarrow Cutset \cup \{Cut_u^s\}$
        Else
          $counter \leftarrow counter + 1$
    End For
    If $(counter == AS)$
      Report AS as the objective function value of the optimal solution of the original problem
      $control \leftarrow 1$
  End While
End

In [10], the authors suggest the use of minimal infeasible subsystems (MIS) in generating combinatorial Benders cuts, which are identified using a linear and continuous slave problem. However, the slave problem we use in this paper is an integer program, to which the approach described in [10] to find a MIS does not necessarily apply. For a given solution to our slave problem $SP_s(\bar{Y}, \bar{A})$ that yields an infeasible solution $(\bar{Y}, \bar{A})$, it is possible to identify a MIS by solving another integer programming formulation. The formulation would be similar to that of a prize-collecting and precedence constrained traveling salesman problem, obtained by relaxing the assignment constraints (6) in $SP_s(\bar{Y}, \bar{A})$ to ensure that at least one task from within an infeasible set is chosen, and by introducing a new set of constraints which ensures that the selected tasks are infeasible with respect to the capacity constraint. However, this would require solving another NP-Hard problem at each iteration and slow down the algorithm. Given the satisfactory computational results reported in Section 5, we chose not to implement the MIS strategy.

5 Computational Study

This section presents a computational study, in three parts, to assess the performance of the proposed algorithm. In the first part, we describe the way in which the instances are generated. The second part analyses the effect of the symmetry-breaking constraint set (10) on the computational run time of the algorithm. The third part presents results to numerically compare the IP and the BDA on the instances.

5.1 Instance generation

There is no standard set of benchmark instances with setup times available in the assembly line balancing literature. For that reason, we construct a set of test instances partly based on the literature, as shown in Table 2, for which the operation and setup times were randomly generated in the same way as in [2] and [3]. For instances numbered 7–24 and 28–30 the precedence diagrams were taken from the existing literature. The precedence diagrams for the other test instances numbered 1–6, 25–27 and 31–57 were taken from http://alb.mansci.de/. The main characteristics of the test instances are presented in Table 2 where N, OS, M, and C denote the number of tasks in the precedence diagram, the order strength of the precedence diagrams, the number of models, and cycle time of the assembly line, respectively. The OS is a measure based on the structure of the precedence diagram and indicative of the computational time required by the solution algorithms as stated by Otto et al. [19]. The higher the OS value, the algorithm requires less time to solve the problem to optimality. The test instances that we used in this current paper have OS values vary between 22 and 84. As Table 2 shows, a total of 57 instances were considered in this study with up to three models. We will use the numbering shown in the last three columns of this table to refer to a particular instance in the rest of this section. All tests presented in this section have been conducted on a personal computer running on a Core(TM) i7-2640 CPU with 2.80 GHz speed. All models and subproblems have been solved using GUROBI 6.0. A time limit of one hour has been imposed on each run of the algorithm and the model.

Table 2: Main characteristic of the test problems

				Instance No		
Problem Name\Source	N	OS	C	Single Model (M=1)	Two Models (M=2)	Three Models (M=3)
Bowman	8	75.00	10	1	20	39
Jackson	11	58.18	10	2	21	40
Ponnambalam et al. [22]	12	69.70	10	3	22	41
Simaria and Vilarinho [26]	14	54.95	10	4	23	42
Buckhin et al. [8]	15	49.52	10	5	24	43
Goncalves and Almeida [12]	16	59.17	10	6	25	44
Su and Lu [27]	17	32.35	10	7	26	45
Thomopoulos [28]	19	25.73	10	8	27	46
Mitchell	21	70.95	10	9	28	47
Vilarinho and Simaria [30]	25	61.00	10	10	29	48
Heskiaoff	28	22.49	10	11	30	49
Buxey	29	50.74	10	12	31	50
Sawyer	30	44.83	10	13	32	51
Lutz1	32	83.47	10	14	33	52
Gunther	35	59.50	10	15	34	53
Kilbridge	45	44.55	10	16	35	54
Hahn	53	83.82	10	17	36	55
Warnecke	58	59.10	10	18	37	56
Tonge	70	59.42	10	19	38	57

5.2 Analysis on the effect of the symmetry-breaking constraint set

As stated above, the constraint set (10) is not an inequality that is not necessary to define the set of integer solutions to the problem, but was introduced as a valid inequality to be able to reduce the CPU time of the algorithm. To numerically confirm whether this is the case, some experiments are conducted by running the $MP(Initial)$ with and without the constraint set (10) on a subset of the test instances. The results are given in Table 3. The feasible solutions given in the third and seventh columns of Table 3 are the objective values of the best solution found by $MP(Initial)$ after one hour. The gap values given in the fourth and eighth columns of Table 3 are the percentage differences between the best solutions found by $MP(Initial)$ and the lower bound value calculated by the solver for the problem.

Table 3: Analysis on the Effect of the Symmetry-Breaking Constraint Set

Instance No	Without Constraint Set (10)				With Constraint Set (10)			
	Opt. Value	Feasible Sol.	Gap (%)	CPU (seconds)	Opt. Value	Feasible Sol.	Gap (%)	CPU (seconds)
5	7	-	0	6.58	7	-	0	0.15
11	16	-	0	80.78	16	-	0	16.25
17	-	30	10.6	3600.00	30	-	0	162.98
24	7	-	0	0.35	7	-	0	0.07
30	-	14	8.06	3600.00	14	-	0	2.34
36	-	32	15.6	3600.00	32	-	0	66.92
43	10	-	0	0.06	10	-	0	0.05
49	13	-	0	1.28	13	-	0	1.16
55	-	34	22.7	3600.00	34	-	0	157.28
Average CPU Time				1609.89				45.24

As can be seen from Table 3, the constraint set (10) has a significant effect on reducing the CPU time as the problem size gets larger. For some cases the $MP(Initial)$ cannot identify an optimal solution without the constraint set (10), however the incumbent solution found by the $MP(Initial)$ is the same as the optimum solution. These results confirm the effectiveness of the constraint set (10) on the CPU time, and for this reason they will be included in the tests in remainder of this section.

5.3 Performance evaluation of the proposed Benders decomposition algorithm

This part of the computational analysis concerns the performance evaluation of the BDA and the IP on the test bed of instances listed in Table 2 in terms of solution time. The BDA is coded in visual studio $C\#$ 2012 and allowed to run for a maximum of one hour for comparison purposes. The results are presented separately in Tables 4, 5 and 6 for the single, two and three-model instances, respectively, for both the IP and the BDA. The columns of the tables are self-explanatory. The tables also report the average solution time for those instances that were solved to optimality by both methods in the row named Average, and the number of such instances over the total of 19 instances in row named Ratio. Additionally, the tables contains the average solution times (AvgCPU) for SP and MP, and the number of added feasibility cuts (NFC) for each instance.

Table 4: Computational results for single model problems

Instance No	IP		BDA				
	Optimum Value	CPU (seconds)	Optimum Value	CPU (seconds)	AvgCPU (seconds)		NFC
					SP	MP	
1	7	0.21	7	0.11	0.001	0.103	0
2	7	5.19	7	0.49	0.001	0.237	22
3	8	3.07	8	0.42	0.001	0.202	12
4	10	11.35	10	0.64	0.001	0.150	56
5	7	181.42	7	0.66	0.001	0.157	75
6	9	30.96	9	0.79	0.001	0.386	16
7	13	358.45	13	1.37	0.001	0.672	17
8	13	481.62	13	11.04	0.001	3.666	38
9	14	46.22	14	0.26	0.001	0.246	0
Average		124.28		1.75			
10	Out of Memory		13	3.06	0.001	0.752	150
11			17	7.32	0.001	3.643	140
12			16	14.65	0.001	3.646	290
13			20	565.51	0.001	141.358	120
14			19	25.15	0.001	12.556	64
15			25	7.92	0.001	1.955	105
16			29	435.51	0.001	87.073	405
17			31	875.63	0.001	218.877	318
18			34	2431.86	0.001	243.152	1334
19			36*	3600.00	-	-	-
Ratio	9/19		18/19				
*Best value found by master model after 1 hour							

As can be seen from Table 4, BDA is able to solve 18 out 19 instances to optimality for the single model instances within an hour of computation time, while the IP is only able to solve nine. There is only one instance, for which the BDA could not identify an optimum solution, and for which we instead report the value of the best solution for this problem after one hour. The average computational time for instances 1–9 is 124.28 seconds for the IP and 1.75 seconds for the BDA.

Table 5: Computational results for two model problems

Instance No	IP		BDA				
	Optimum Value	CPU (seconds)	Optimum Value	CPU (seconds)	AvgCPU (seconds)		NFC
					SP	MP	
20	7	0.68	7	0.17	0.001	0.050	16
21	8	6.04	8	0.14	0.001	0.132	0
22	7	3.78	7	0.11	0.001	0.103	0
23	8	112.56	8	0.13	0.001	0.122	0
24	7	201.98	7	0.44	0.001	0.213	30
25	9	83.20	9	2.18	0.001	1.081	32
26	9	1194.13	9	0.32	0.001	0.098	51
27	12	183.50	12	1.03	0.001	0.194	114
28	12	513.61	12	0.25	0.001	0.238	0
Average		255.50		0.53			
29	Out of Memory		16	1.11	0.001	0.539	25
30			14	26.38	0.001	5.262	336
31			13	19.52	0.001	2.427	232
32			20	10.18	0.001	3.373	120
33			18	5.42	0.001	2.692	96
34			17	38.99	0.001	6.481	280
35			27	256.51	0.001	28.474	675
36			34	430.83	0.001	71.771	795
37			31	1145.80	0.001	163.655	928
38			39*	3600.00	-	-	-
Ratio	9/19		18/19				

*Best value found by master model after 1 hour

Similar to the previous table, Table 5 shows that the BDA is able to identify optimal solutions for 18 out of 19 instances, while the IP is able to only solve nine problems to optimality. The faster solution times of the BDA can also be seen from Table 5, in particular it highly outperforms the IP with an average solution time of 0.53 seconds, as compared with that of the latter which is 255.50 seconds.

Table 6: Computational results for three model problems

Instance No	IP		BDA				
	Optimum Value	CPU (seconds)	Optimum Value	CPU (seconds)	AvgCPU (seconds)		NFC
					SP	MP	
39	5	2.05	5	0.08	0.001	0.075	0
40	8	31.53	8	0.25	0.001	0.242	11
41	7	32.82	7	0.13	0.001	0.123	0
42	11	67.66	11	0.31	0.001	0.144	28
43	10	513.65	10	0.18	0.001	0.170	0
44	12	255.89	12	0.81	0.001	0.393	16
45	11	1153.99	11	2.86	0.001	0.942	51
46	14	736.79	14	0.24	0.001	0.226	0
Average		349.30		0.61			
47	Out of Memory		10	0.71	0.001	0.108	189
48			17	2.40	0.001	2.383	0
49			13	9.24	0.001	1.835	252
50			18	76.23	0.001	25.392	58
51			20	7.96	0.001	2.633	120
52			24	40.44	0.001	40.416	0
53			22	204.79	0.001	102.373	35
54			21	446.22	0.001	49.559	900
55			34	300.59	0.001	150.261	212
56			31*	3600.00	-	-	-
57			39*	3600.00	-	-	-
Ratio	8/19		17/19				
*Best value found by master model after 1 hour							

For three model instances, the BDA is able to yield optimal solutions for 17 out 19 instances, while IP is able to solve eight problems to optimality, as shown in Table 6. Here we conclude that the BDA is superior to the IP in terms of solution time, since the average solution computational time for instances 39–46 is 349.30 seconds for the latter and 0.61 seconds for the former.

As can be seen from Tables 4, 5 and 6, the proposed algorithm is able to solve the instances with up to 58 tasks for the single model and two model instances, and up to 53 tasks for the three model case. The computational times show the efficiency of the BDA, in particular that the optimal solutions were identified for 42 out of 57 instances in less than one minute. The algorithm solved 53 instances to optimality within the pre-defined maximum computational time of one hour. For the other four instances numbered 19, 38, 56 and 57, the algorithm was not able to find optimal solutions within one hour. As far as the OS measure is concerned, the efficiency of the BDA is particularly evident in solving instances with lower OS values to optimality in short time scales.

On the other hand, the IP found optimal solutions for the single and two model instances with up to 21 tasks, and for three model instances with up to 19 tasks. This is also indicative of the improved computational capability of the IP over the previous mathematical model proposed in [2], which was only able to optimality solve for problems with up to 12 tasks for two and three model cases.

6 Conclusions

In this paper, we described a Benders decomposition algorithm for single and mixed-model Type-I assembly line balancing problems with setups. First, we improved a previously proposed mixed-integer programming formulation for the MMALBP by reducing the number of *bigM* constraints used. The model contains the assignment subproblem of the assembly lines and the sequencing subproblem related to the sequence dependent setup times between tasks. By exploiting this structure we devised a Benders decomposition algorithm, which solves the assignment subproblem as a master problem and the sequencing subproblem as a slave problem in order to generate combinatorial Benders cuts. The performance of the proposed algorithm was tested on a set of literature-based benchmark instances and the results are compared against a mixed-integer linear programming formulation of the problem solved using an off-the-shelf optimizer. The results confirm the superior performance of the proposed algorithm in terms of computational time.

Acknowledgement

This research project was partially supported by the Scientific and Technological Research Council of Turkey (TÜBİTAK). While writing this paper, Dr. Sener Akpinar was a visiting researcher at the Southampton Business School at the University of Southampton.

References

[1] S. Akpinar, G. M. Bayhan, and A. Baykasoglu. Hybridizing ant colony optimization via genetic algorithm for mixed-model assembly line balancing problem with sequence dependent setup times between tasks. *Applied Soft Computing*, 13(1):574–589, 2013.

[2] S. Akpinar and A. Baykasoglu. Modeling and solving mixed-model assembly line balancing problem with setups. Part I: A mixed integer linear programming model. *Journal of Manufacturing Systems*, 33(1):177–187, 2014.

[3] S. Akpinar and A. Baykasoglu. Modeling and solving mixed-model assembly line balancing problem with setups. Part II: A multiple colony hybrid bees algorithm. *Journal of Manufacturing Systems*, 33(4):445–461, 2014.

[4] C. Andres, C. Miralles, and R. Pastor. Balancing and scheduling tasks in assembly lines with sequence-dependent setup times. *European Journal of Operational Research*, 187(3):1212–1223, 2008.

[5] O. Battaïa and A. Dolgui. Reduction approaches for a generalized line balancing problem. *Computers & Operations Research*, 39(10):2337–2345, 2012.

[6] O. Battaïa and A. Dolgui. A taxonomy of line balancing problems and their solution approaches. *International Journal of Production Economics*, 142(2):259–277, 2013.

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

[8] J. Bukchin, E. M. Dar-El, and J. Rubinovitz. Mixed model assembly line design in a make-to-order environment. *Computers & Industrial Engineering*, 41(4):405–421, 2002.

[9] Y. Bukchin and I. Rabinowitch. A branch-and-bound based solution approach for the mixed-model assembly line-balancing problem for minimizing stations and task duplication costs. *European Journal of Operational Research*, 174(1):492–508, 2006.

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

[11] J.-F. Côté, M. Dell'Amico, and M. Iori. Combinatorial Benders' cuts for the strip packing problem. *Operations Research*, 62(3):643–661, 2014.

[12] J. F. Gonçalves and J. R. De Almeida. A hybrid genetic algorithm for assembly line balancing. *Journal of Heuristics*, 8(6):629–642, 2002.

[13] L. Gouveia and P. Pesneau. On extended formulations for the precedence constrained asymmetric traveling salesman problem. *Networks*, 48(2):77–89, 2006.

[14] N. Hamta, S. F. Ghomi, F. Jolai, and M. A. Shirazi. A hybrid PSO algorithm for a multi-objective assembly line balancing problem with flexible operation times, sequence-dependent setup times and learning effect. *International Journal of Production Economics*, 141(1):99–111, 2013.

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

[16] Ö. Hazır and A. Dolgui. A decomposition based solution algorithm for u-type assembly line balancing with interval data. *Computers & Operations Research*, 59:126–131, 2015.

[17] L. Martino and R. Pastor. Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research*, 48(6):1787–1804, 2010.

[18] E. Nazarian, J. Ko, and H. Wang. Design of multi-product manufacturing lines with the consideration of product change dependent inter-task times, reduced changeover and machine flexibility. *Journal of Manufacturing Systems*, 29(1):35–46, 2010.

[19] A. Otto, C. Otto, and A. Scholl. 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, 2013.

[20] U. Özcan and B. Toklu. Balancing two-sided assembly lines with sequence-dependent setup times. *International Journal of Production Research*, 48(18):5363–5383, 2010.

[21] R. Pastor, C. Andrés, and C. Miralles. Corrigendum to "balancing and scheduling tasks in assembly lines with sequence-dependent setup"[european journal of operational research 187 (2008) 1212–1223]. *European Journal of Operational Research*, 201(1):336, 2010.

[22] S. Ponnambalam, P. Aravindan, and G. M. Naidu. A comparative evaluation of assembly line balancing heuristics. *The International Journal of Advanced Manufacturing Technology*, 15(8):577–586, 1999.

[23] A. Scholl, N. Boysen, and M. Fliedner. The sequence-dependent assembly line balancing problem. *OR Spectrum*, 30(3):579–609, 2008.

[24] A. Scholl, N. Boysen, and M. Fliedner. The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectrum*, 35(1):291–320, 2013.

[25] S. Seyed-Alagheband, S. F. Ghomi, and M. Zandieh. A simulated annealing algorithm for balancing the assembly line type II problem with sequence-dependent setup times between tasks. *International Journal of Production Research*, 49(3):805–825, 2011.

[26] A. S. Simaria and P. M. Vilarinho. 2-antbal: An ant colony optimisation algorithm for balancing two-sided assembly lines. *Computers & Industrial Engineering*, 56(2):489–506, 2009.

[27] P. Su and Y. Lu. Combining genetic algorithm and simulation for the mixed-model assembly line balancing problem. In *Natural Computation, 2007. ICNC 2007. Third International Conference on*, volume 4, pages 314–318. IEEE, 2007.

[28] N. T. Thomopoulos. Mixed model line balancing with smoothed station assignments. *Management Science*, 16(9):593–603, 1970.

[29] N. T. Thomopoulos. *Assembly line planning and control*. Switzerland. Springer, 2014.

[30] P. M. Vilarinho and A. S. Simaria. A two-stage heuristic method for balancing mixed-model assembly lines with parallel workstations. *International Journal of Production Research*, 40(6):1405–1420, 2002.

[31] A. Yolmeh and F. Kianfar. An efficient hybrid genetic algorithm to solve assembly line balancing problem with sequence-dependent setup times. *Computers & Industrial Engineering*, 62(4):936–945, 2012.

Appendix A The MILP Model Proposed in [2]

Table A.1: Model notation and their definitions

	Notation	Definition
Indices	N	Total number of tasks
	M	Total number of models simultaneously assembled at the line
	WS	Maximum number of workstations
	i	Set of tasks; $i \in \{1, 2, ..., N\}$
	s	Set of stations; $s \in \{1, 2, ..., WS\}$
	m, n	Set of models; $m, n \in \{1, 2, ..., M\}$
Parameters	C	Cycle time
	$maxp$	Maximum number of replicas for a workstation (Set as 2)
	α	A pre-defined proportion (α%) of the cycle time
	$bigM$	A very large number
	T_i	Processing time of task i on model m
	$TT_{im} \in \{0, 1\}$	Equals 1 if processing time of task i is greater than zero for model m and 0 otherwise
	FST_{ijm}	Forward set-up time between task i and j on model m
	BST_{ijmn}	Backward set-up time between task i of model n and task j of model m
	$PR_{ij} \in \{0, 1\}$	Equals 1 if task i must precede task j and 0 otherwise
	$ZP_{ij} \in \{0, 1\}$	Equals 1 if task i and task j must be assigned to the same workstation and 0 otherwise
	$ZN_{ij} \in \{0, 1\}$	Equals 1 if task i and task j must be assigned to different workstations and 0 otherwise
DecisionVariables	$Y_{is} \in \{0, 1\}$	Equals 1 if task i is assigned to workstation s and 0 otherwise
	$A_s \in \{0, 1\}$	Equals 1 if station s is active, 0 otherwise
	$R_{sm} \in \{0, 1\}$	Equals 1 if workstation s is duplicated due to model$m$ and 0 otherwise
	$R_s \in \{0, 1\}$	Equals 1 if workstation s is duplicated and 0 otherwise
	$w_{ijs} \in \{0, 1\}$	Equals 1 if task i precede task j at workstation s and 0 otherwise
	$FS_{ijms} \in \{0, 1\}$	Equals 1 if task j directly follows task i on model m in the forward direction in workstation s and 0 otherwise
	$BS_{ijmns} \in \{0, 1\}$	Equals 1 if i is the last task of model n and j is the first task of model m in workstation s and 0 otherwise
	N_{WS}	Total number of workstations including replicas

Minimize
$$N_{WS} = \sum_{s=1}^{WS}(R_s + A_s) \tag{A.1}$$

subject to
$$\sum_{s=1}^{WS} Y_{is} = 1 \qquad i \in \{1, ..., N\} \tag{A.2}$$

$$bigM(1 - Y_{is}PR_{ij}) + \sum_{t|t \geq s}^{WS} Y_{jt} \geq 1 \qquad i,j \in \{1,...,N\}; s \in \{1,...,WS\} \tag{A.3}$$

$$Y_{js} + bigM(1 - (Y_{is}ZP_{ij})) \geq 1 \qquad i,j \in \{1,...,N\}; s \in \{1,...,WS\} \tag{A.4}$$

$$Y_{js} - bigM(1 - (Y_{is}ZN_{ij})) \geq 1 \qquad i,j \in \{1,...,N\}; s \in \{1,...,WS\} \tag{A.5}$$

$$R_{sm} - bigM \sum_{i|(T_{im} > \alpha C)}^{N} Y_{is} \leq 0 \qquad s \in \{1,...,WS\}; m \in \{1,...,M\} \tag{A.6}$$

$$R_{sm} \geq Y_{is} \qquad i \in \{1,...,N\} | T_{im} > \alpha C; s \in \{1,...,WS\}; m \in \{1,...,M\} \tag{A.7}$$

$$R_s - bigM \sum_{m=1}^{M} R_{sm} \leq 0 \qquad s \in \{1,...,WS\} \tag{A.8}$$

$$R_s \geq R_{sm} \qquad s \in \{1,...,WS\}; m \in \{1,...,M\} \tag{A.9}$$

$$w_{iis} = 0 \qquad i \in \{1,...,N\}; s \in \{1,...,WS\} \tag{A.10}$$

$$w_{ijs} + w_{jis} + bigM(2 - Y_{is} - Y_{js}) \geq 1 \qquad i,j \in \{1,...,N\} | i \neq j; s \in \{1,...,WS\} \tag{A.11}$$

$$w_{ijs} + w_{jis} - bigM(2 - Y_{is} - Y_{js}) \leq 1 \qquad i,j \in \{1,...,N\} | i \neq j; s \in \{1,...,WS\} \tag{A.12}$$

$$w_{ijs} + bigM(3 - Y_{is} - Y_{js} - PR_{ij}) \geq 1 \qquad i,j \in \{1,...,N\}; s \in \{1,...,WS\} \tag{A.13}$$

$$w_{ijs} + bigM(2 - w_{iks} - w_{kjs}) \geq 1 \qquad i,j,k \in \{1,...,N\}; s \in \{1,...,WS\} \tag{A.14}$$

$$\sum_{i=1}^{N} \sum_{j=1}^{N} w_{ijs} = \sum_{i|(i < \sum_{k=1}^{N} Y_{ks})}^{N} i \qquad s \in \{1,...,WS\} \tag{A.15}$$

$$FS_{ijms} + BS_{ijmns} - bigM(TT_{im}TT_{jn}) \leq 0 \quad i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.16}$$

$$FS_{ijms} - bigM(1 - BS_{ijmns}) \leq 0 \qquad i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.17}$$

$$\sum_{j=1}^{N} \sum_{s=1}^{WS} FS_{ijms} \leq 1 \qquad i \in \{1,...,N\}; m \in \{1,...,M\} \tag{A.18}$$

$$FS_{ijms} + FS_{jims} \leq 1 \qquad i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m \in \{1,...,M\} \tag{A.19}$$

$$\sum_{s=1}^{WS} FS_{iims} = 0 \qquad i \in \{1,...,N\}; m \in \{1,...,M\} \tag{A.20}$$

$$FS_{ijms} + bigM(4 - Y_{is} - Y_{js} - TT_{im} - TT_{jm}) + bigM \left| \sum_{k=1}^{N} (w_{iks}TT_{km}) - \sum_{l=1}^{N} (w_{jls}TT_{lm}) - 1 \right| \geq 1$$
$$i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m \in \{1,...,M\} \tag{A.21}$$

$$w_{ijs} + w_{jis} + FS_{ijms} + FS_{jims} + BS_{ijmns} + BS_{jimns} - bigM(1 - Y_{is} + Y_{js}) \leq 0$$
$$i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.22}$$

$$w_{ijs} + w_{jis} + FS_{ijms} + FS_{jims} + BS_{ijmns} + BS_{jimns} - bigM(1 + Y_{is} - Y_{js}) \leq 0$$
$$i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.23}$$

$$w_{ijs} + w_{jis} + FS_{ijms} + FS_{jims} + BS_{ijmns} + BS_{jimns} - bigM(Y_{is} + Y_{js}) \leq 0$$
$$i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.24}$$

$$\sum_{i=1}^{N}\sum_{j=1}^{N} BS_{ijmns} \leq 1 \qquad s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.25}$$

$$BS_{ijmns} + bigM(4 - Y_{is} - Y_{js} - TT_{in} - TT_{jm}) + bigM\left(\sum_{t=1}^{N}(w_{its}TT_{tn})\right) +$$

$$bigM\left|\sum_{l=1}^{N}(Y_{ls}TT_{lm}) - \sum_{k=1}^{N}(w_{jks}TT_{jm}) - 1\right| \geq 1$$

$$i,j \in \{1,...,N\}; s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.26}$$

$$\sum_{i=1}^{N}\left(Y_{is}T_{im} + \sum_{j=1}^{N}(FS_{ijms}FST_{ijm} + BS_{ijmns}BST_{ijmn})\right) \leq C(1 + R_s(maxp - 1))$$

$$s \in \{1,...,WS\}; m,n \in \{1,...,M\} \tag{A.27}$$

$$A_s + bigM(1 - Y_{is}) \geq 1 \qquad i,j \in \{1,...,N\}; s \in \{1,...,WS\} \tag{A.28}$$

$$A_s - bigM\sum_{i=1}^{N} Y_{is} \leq 0 \qquad s \in \{1,...,WS\} \tag{A.29}$$

$$\left|\left(\sum_{s=1}^{u} A_s\right) - u\right| - bigM(1 - A_u) \leq 0 \qquad u \in \{1,...,WS\}. \tag{A.30}$$

The objective function (A.1) minimizes the total number of activated workstations included replicas. Constraint set (A.2) ensures the assignment of each task to exactly one workstation. Constraint set (A.3) guarantees the precedence relations among tasks. Constraint sets (A.4) and (A.5) are used to force and forbid the assignment of different tasks into the same workstation, respectively. Constraint sets (A.6), (A.7), (A.8) and (A.9) parallel a workstation if it performs a task with processing time larger than a certain proportion (α%) of the cycle time for at least one the models. Constraint sets (A.10), (A.11), (A.12), (A.13) (A.14) and (A.15) are used to determine tasks processing orders within each workstation. Constraint set (A.16), (A.17), (A.18), (A.19), (A.20) (A.21), (A.22), (A.23), (A.24), (A.25) and (A.26) used to determine the necessary setup operations between tasks within each workstation. Constraint set (A.27) ensures that the workload of a workstation does not exceed the pre-defined cycle time for all models being assembled on the line. Constraint sets (A.28), (A.29) and (A.30) used to determine the activated workstations.

Appendix B Linearization of the Semi-Linear Constraints

The absolute value function is a semi-linear function used in constraint sets (17), (23) and (24) into two different forms. Table B.1 provides these forms and the transformation of absolute value function for these forms that we used in the proposed mathematical model.

Table B.1: Forms of converting absolute value constraints into linear constraints

Constraints Forms	Form (1)	Form (2)
	$x - BigM\|a-b\| \leq 0$	$\|x-y\| - bigM\|a-b\| \leq 0$
Transformed to	$x - bigM(p+q) \leq 0$	$(p'+q') - BigM(p+q) \leq 0$
	$a - b - p + q = 0$	$x - y - p' + q' = 0$
	$p - bigM(e) \leq 0$	$a - b - p + q = 0$
	$q - bigM(1-e) \leq 0$	
$x, y, a, b, p, q, p', q' \geq 0$; $e \in 0, 1$; $bigM$: Sufficiently large value		

Considering these two transformation methods, additional variables as stated in Table B.2 are required to linearize the constraint sets (17), (23) and (24).

Table B.2: Auxiliary variables

Auxiliary binary variables	q, p, g
Auxiliary non-negative variables	g^+, g^-, p^+, p^-, z^+, z^-, g^+, g^-

$$\left|\sum_{k=1}^{N}\sum_{l=1}^{N} w_{kls} - \sum_{j|j<i}^{N} j\right| \leq N\left|i - \sum_{p=1}^{N} Y_{ps}\right| \qquad i \in \{1, ..., N\}; s \in \{1, ..., S\}. \tag{13}$$

The constraint set (13) is replaced with the following three constraint sets (13-B1), (13-B2) and (13-B3), since it has the form (2) as stated in Table B.1.

$$z_{is}^{+} + z_{is}^{-} \leq N(g_{is}^{+} + g_{is}^{-}) \qquad i \in \{1, ..., N\}; s \in \{1, ..., S\} \tag{13-B1}$$

$$\sum_{k=1}^{N}\sum_{l=1}^{N} w_{kls} - \sum_{j|j<i}^{N} j = z_{is}^{+} + z_{is}^{-} \qquad i \in \{1, ..., N\}; s \in \{1, ..., S\} \tag{13-B2}$$

$$\sum_{p=1}^{N} Y_{ps} - i = g_{is}^{+} + g_{is}^{-} \qquad i \in \{1, ..., N\}; s \in \{1, ..., S\}. \tag{13-B3}$$

$$(Y_{is}Q_{im} + Y_{js}Q_{jn} - 1) - \left(\sum_{k=1}^{N}(w_{iks}Q_{km})\right) - \left|\sum_{l=1}^{N}(Y_{ls}Q_{ln}) - \sum_{p=1}^{N}(w_{jps}Q_{pn}) - 1\right| \leq Z_{ijmns}$$
$$i, j \in \{1, ..., N\}; m, n \in \{1, ..., M\}; s \in \{1, ..., S\}. \tag{19}$$

The constraint set (19) is replaced with the following four constraint sets (19-B1), (19-B2), (19-B3) and (19-B4), since it has the form (1) as stated in Table B.1 .

$$q^{+}_{ims} - bigMq_{ims} \leq 0 \qquad i \in \{1,...,N\}; m \in \{1,...,M\}; s \in \{1,...,S\} \tag{19-B1}$$

$$q^{-}_{ims} - bigM(1 - q_{ims}) \leq 0 \qquad i \in \{1,...,N\}; m \in \{1,...,M\}; s \in \{1,...,S\} \tag{19-B2}$$

$$(Y_{is}Q_{im} + Y_{js}Q_{jn} - 1) - \left(\sum_{k=1}^{N}(w_{iks}Q_{km})\right) - q^{+}_{jns} - q^{-}_{jns} \leq Z_{ijmns}$$
$$i, j \in \{1,...,N\}; m, n \in \{1,...,M\}; s \in \{1,...,S\} \tag{19-B3}$$

$$\sum_{l=1}^{N}(Y_{ls}Q_{ln}) - \sum_{p=1}^{N}(w_{jps}Q_{pn}) - 1 = q^{+}_{jns} - q^{-}_{jns}$$
$$j \in \{1,...,N\}; n \in \{1,...,M\}; s \in \{1,...,S\}. \tag{19-B4}$$

$$(Y_{is}Q_{im} + Y_{js}Q_{jm} - 1) - \left|\sum_{k=1}^{N}(w_{iks}Q_{km}) - \sum_{l=1}^{N}(w_{jls}Q_{lm}) - 1\right| \leq X_{ijms}$$
$$i, j \in \{1,...,N\}; m \in \{1,...,M\}; s \in \{1,...,S\}. \tag{20}$$

The constraint set (20) is replaced with the following four constraint sets (20-B1), (20-B2), (20-B3) and (20-B4), since it has the form (1) as stated in Table B.1.

$$p^{+}_{ijms} - bigMp_{ijms} \leq 0 \qquad i, j \in \{1,...,N\}; m \in \{1,...,M\}; s \in \{1,...,S\} \tag{20-B1}$$

$$p^{-}_{ijms} - bigM(1 - p^{-}_{ijms}) \leq 0 \qquad i, j \in \{1,...,N\}; m \in \{1,...,M\}; s \in \{1,...,S\} \tag{20-B2}$$

$$(Y_{is}Q_{im} + Y_{js}Q_{jm} - 1) - p^{+}_{ijms} - p^{-}_{ijms} \leq X_{ijms}$$
$$i, j \in \{1,...,N\}; m \in \{1,...,M\}; s \in \{1,...,S\} \tag{20-B3}$$

$$\sum_{k=1}^{N}(w_{iks}Q_{km}) - \sum_{l=1}^{N}(w_{jls}Q_{lm}) = p^{+}_{ijms} - p^{-}_{ijms}$$
$$i, j \in \{1,...,N\}; m \in \{1,...,M\}; s \in \{1,...,S\}. \tag{20-B4}$$