RAIRO - Operations Research

A constraint programming approach to type-2 assembly line balancing problem with assignment restrictions

--Manuscript Draft--

Manuscript Number:	ro170139R2
Article Type:	Research article
Full Title:	A constraint programming approach to type-2 assembly line balancing problem with assignment restrictions
Short Title:	Constraint Programming Approach to Assembly Line Balancing
Section/Category:	
Corresponding Author:	Mehmet PINARBAŞI, Ph.D. Independent Researcher Çorum, Merkez TURKEY
Order of Authors:	Mehmet PINARBAŞI, Ph.D.
	Hacı Mehmet ALAĞAŞ, M.Sc.
	Mustafa Yüzükırmızı, Ph.D.
Manuscript Region of Origin:	TURKEY
Keywords:	Assembly line balancing; Type-2 problem; Assignment restrictions; Constraint programming; Mixed integer programming
Abstract:	Main constraints for an assembly line balancing problem (ALBP) are cycle time/number of stations and task precedence relations. However, due to the technological and organizational limitations, several other restrictions such as task assignment, station, resource and distance limitations can be encountered in real production systems. In this study, we evaluate the effect of these restrictions on ALBP. A Constraint Programming (CP) model is proposed and compared to Mixed-Integer Programming (MIP) as a benchmark. The objective is to minimize the cycle time for a given number of stations. We provide a more explicit analogy of the effects of assignment restrictions on line efficiency, the solution quality and computation time. Furthermore, the proposed approach is verified with a real-life problem from a furniture manufacturing industry. Computational experiments show that, despite additional assignment restrictions are problematic in mathematical solutions, CP is a versatile exact solution alternative in modelling and solution quality.

RAIRO Operations Research
Will be set by the publisher

A CONSTRAINT PROGRAMMING APPROACH TO TYPE-2 ASSEMBLY LINE BALANCING PROBLEM WITH ASSIGNMENT RESTRICTIONS

Mehmet Pinarbasi[1], Haci Mehmet Alagas[2] and Mustafa Yuzukirmizi[3]

Abstract. Main constraints for an assembly line balancing problem (ALBP) are cycle time/number of stations and task precedence relations. However, due to the technological and organizational limitations, several other restrictions such as task assignment, station, resource and distance limitations can be encountered in real production systems. In this study, we evaluate the effect of these restrictions on ALBP. The objective is to minimize the cycle time for a given number of stations. A Constraint Programming (CP) model is proposed, and compared to Mixed-Integer Programming (MIP) as a benchmark. We provide a more explicit analogy of the effects of assignment restrictions on line efficiency, the solution quality and the computation time. Furthermore, the proposed approach is verified with a real-life problem from a furniture manufacturing industry. Computational experiments show that, despite additional assignment restrictions are problematic in mathematical solutions, CP is a versatile exact solution alternative in modelling and solution quality.

Keywords: Assembly line balancing, Type-2 problem, Assignment restrictions, Constraint programming, Mixed integer programming

---

The date should precede \maketitle in AMS documentclasses; reported.

[1] Independent Researcher, Ulukavak Mahallesi, Mavral Sokak Yılmaz Apt., 19030 Çorum, Turkey; e-mail: mehmetpinarbasi71@hotmail.com

[2] Kırıkkale University, Department of Industrial Engineering, Faculty of Engineering, 71450 Yahşihan, Kırıkkale, Turkey;e-mail: hmalagas@kku.edu.tr

[3] Independent Researcher, Bahcelievler Mah., Tepebaglar Cad., No:51/26 38280 Talas, Kayseri, Turkey;e-mail: myuzukirmizi@hotmail.com

1. Introduction

Assembly line (AL) is a serial production system in which stations are organized according to task precedence relations, task completion times and cycle time constraints. The decision problem of finding the optimal task assignments to stations within a set of precedence relations is defined as Assembly Line Balancing Problem (ALBP). ALBP was first mathematically formulated by Salveson [44], upon then called simple ALBP. As a solution approach, the author proposed a linear programming model to minimize the total idle time at stations. Later, 0-1 integer programming model was developed by Bowman [8]. Hence forth, ALBP studies became one of the major topics in manufacturing research.

ALBP has different classifications in terms of objective functions, task times, layout, product model type, etc. The main classification of ALBP is according to their objectives functions, and is as follows [18, 47]

- Type 1: Minimization of the number of stations for a given cycle time.
- Type 2: Minimization of the cycle time for a given number of stations.
- Type E (Effectiveness): Minimization of both number of workstation and cycle time.
- Type F (Feasible): Obtaining a feasible solution for a given number of workstation and cycle time.

While extremizing above objectives, the main consideration in ALBP studies are precedence relations of tasks with general assumption of all stations are equally equipped with respect to machines and workers. However, various assignment restrictions can be encountered due to the technological, operational and location decisions. Also, these assignment restrictions arise by the time the line needs to be re-balanced. For example, tasks must be assigned to different stations which are performed at the interactive two machines (like press and precision machining). In addition, certain assignment restrictions of tasks need to be considered depending on the production conditions, operators' skills, space of the workstations and requirements of the equipments. These restrictions are necessary for astute design and/or redesign of assembly lines. However, the literature on ALBP Type-2 problems only deal with the basic constraint of precedence relations. Also, the solution approaches while reconfiguring an assembly lines are insufficient in terms of including assignment restrictions [39].

2. Assignment Restrictions in The Assembly Line Balancing Problem

Mainly, simple ALBP constraints are cycle time/number of stations constraints, precedence constraints and occurrence constraints. In addition to these basic model requirements, the following constraints can further restrict the assignment of tasks to the stations [10]:

Task restrictions (zoning restrictions): There are two types of task assignment restrictions: linked tasks and incompatible tasks restrictions. In linked tasks

restrictions, a set of tasks has to be assigned to the same station due to the resource requirement. Incompatible tasks, in contrast, require different equipments and should not be assigned to the same stations [39, 55].

Resource (attribute) restrictions: When the required machine at a station needs a special space to place in line, the resource restrictions limit the cumulative assignments of the tasks [9, 26].

Station restrictions: The station constraints restrict the possible assignment of tasks to stations. There are two types of station restrictions: tasks must be assigned to a certain station, such as requiring certain equipment, and tasks cannot be assigned to certain stations [17, 23, 55].

Distance restrictions: Due to the production process needs, the tasks should be assigned to stations to observe minimum or maximum distance between tasks. For a minimum distance example, a task succeeding the colouring task may have to be performed after the color on the work piece dries. Maximum distance can be observed in the case where melted metal must be prevented from cooling [35, 39].

Adopting from the renowned study of Scholl et al. [48] and adding recent studies, assignment restricted ALBP research can be listed as in Table 1.

To signify milestone studies; Klein and Scholl [21] presented a comprehensive study for type-2 ALBP without restrictions. They also proposed a branch and bound procedure consisted of a local lower bound method and a new enumeration technique. In a similar study, Uğurdağ et al. [56] provided a two-stage heuristic. They generated an initial solution in stage 1 and improved it using a Simplex method in stage 2. A bidirectional heuristic procedure was proposed for stochastic type-2 ALBP in [27]. Nearchou [31] improved a heuristic using differential evolution method for large size assembly line. Also a particle swarm optimization was proposed by Nearchou [32] and an ant colony optimization was developed by Zheng et al. [61] to solve the type-2 ALBP. Unlike the traditional solution methods to solve the ALBP, Kilincci [19] presented a heuristic method integrating with forward, backward and bidirectional procedures based on the Petri-nets using reachability analysis.

As it is seen in Table 1, while task, resource and station restrictions have widely studied, distance restrictions have not attracted considerable attention. However several industrial firms (e.g. auto mobile, electronics, machine construction etc.) need to implement all-encompassing restriction types for balancing their assembly lines [48]. As the solution methods of AR-ALBP, they are mainly mathematical models and heuristic or meta-heuristic techniques [30]. Mathematical models use enumeration procedures such as branch and bound to seek optimum solution. Constructive or greedy procedures which utilize a priority rule to assign the tasks are two main approaches for heuristic procedures. Although heuristics procedures reach a prime solution in reasonable computing time, the optimal solution is not assured. There is a lack of research that focuses on the exact solution methods for ALBP in the literature. Accordingly, CP as an exact approach which considers several types of assignment restrictions to solve the assembly line balancing problem, has to be explored. Thus, this study is also novel in its solution method

using constraint programming and establishing its advantages on AR-ALBP for simultaneously considering several types of assignment restrictions.

TABLE 1. Literature review and solution methods for ARALBP

Restriction	Paper	Solution Method
Linked	Pastor and Corominas [35]	M,H
	Lapierre and Ruiz [22]	H
	Rekiek et al. [40, 41]	H
	Miralles [29]	M
	Vilarinho and Simaria [57, 58]	M, H
	Boysen and Fliedner [9]	M
	Purnomo et al. [39]	H
	Tuncel and Topaloğlu [55]	M
Incompatibles	Bautista et al. [5]	H
	Pastor and Corominas [35]	M,H
	Rekiek et al. [41]	H
	Bautista and Pereira [3]	H
	Boysen and Fliedner [9]	M
	Lapierre and Ruiz [22]	H
	Vilarinho and Simaria [57, 58]	M, H
	Purnomo et al. [39]	H
	Tuncel and Topaloğlu [55]	M
Resource	Carnahan et al. [13]	H
	Liu and Chen [26]	M,H
	Pastor et al. [34]	H
	Sawik [45]	M
	Wilhelm and Gadidov [59]	M
	Miralles [29]	M
	Ağpak and Gökçen [2]	M
	Bautista and Pereira [4]	H
	Boysen and Fliedner [9]	M
	Corominas et al. [14]	M
Station	Kim et al. [20]	H
	Gadidov and Wilhelm [17]	M
	Lee et al. [24]	H
	Pastor et al. [34]	H
	Rekiek et al. [40, 41]	H
	Lapierre and Ruiz [22]	H
	Vilarinho and Simaria [58]	M, H
	Lapierre et al. [23]	H
	Purnomo et al. [39]	H
	Tuncel and Topaloğlu [55]	M
Distance	Deckro [16]	M
	Pastor and Corominas [35]	H,M
	Purnomo et al. [39]	H

M: Mathematical Model, H: Heuristic

In this study, the problem of balancing a line with assignment restrictions is considered. Principally, Type-2 ALBP with zoning, station, resource and distance assignment restrictions is studied. The applicability of contemporary method of Constraint Programming which performs well to solve many combinatorial problems is examined as a solution approach, and innovative models are developed. While, CP has been widely applied to solve the several combinatorial problems (e.g. scheduling, sequencing, assignment problem etc.) [53], the applications to ALBP are rare. As a benchmark approach, comparison to Mixed Integer Programming (MIP) is also carried out. After validating the performance of the

proposed CP model from literature examples, assembly line balancing problem of a furniture firm in Turkey is solved for various assignment restrictions. Results reveal that the proposed CP model is very effective to obtain the optimal balance for assignment restricted ALBP.

This paper is organized as follows. In the following section an overview of the constraint programming is given. In Section 4, Proposed CP model for AR-ALBP is presented. Mathematical model for AR-ALBP is given in Section 5. A comparison of CP and MIP is presented in Section 6. Section 7 provides the numerical results, case by case for each assignment restriction. Discussion of the results is given in Section 8. Conclusion and future research are addressed in the last section.

2.1. Model assumptions and notations

Basic assumptions of the AR-ALBP considered in this study are as follows:

(1) Each task must be assigned to a station.
(2) Serial assembly line layout.
(3) Task durations are deterministic and known.
(4) Tasks are not divisible.
(5) Total number of stations is fixed and known.
(6) A single model assembly line without buffers.
(7) Precedence constraints are known and must be stable on task-station assignments.
(8) At least one task must be assigned to each station.
(9) Incompatible and linked tasks are considered between some pairs of tasks.
(10) Resource and station restrictions may exist which limits assignments.
(11) Distance restrictions between tasks are considered.

Accordingly, the following notations in Table 2 are used in the description of models:

3. Constraint Programming

The main aim of this research is to utilize problem specific approaches for AR-ALBP. Constraint programming (CP) is an alternative programming technique to MIP. It combines the effectiveness of linear programming and easy definition property of logical expressions in computer programming. CP is *an exact method* among combinatorial optimization methods -i.e. branch and bound, dynamic programming etc.- with single or multi objective functions. [51]. Main concept of CP is constraints. Each constraint is defined as relation between some variables related to their domains. Domain is a set of values that can be assigned to variables. The constraints may be of various different types: linear, non-linear, logical or global constraints.

CP has been widely used to solve several NP-hard combinatorial problems. Some recent studies by subjects can be counted as following: scheduling problems

TABLE 2. Notations of the model

Symbol	Meaning
c	Cycle time
W	Set of workstations
$\|W\|$	Number of elements of set W
T	Set of tasks
$\|T\|$	Number of elements of set T
t_i	Operation time of task i
Pr	The set of precedence relations between task pairs
P_i^*	The set of all predecessors of task i
F_i^*	The set of all followers of the task i
d_{ij}^-	Minimum distance between task i and j
d_{ij}^+	Maximum distance between task i and j
IT	Set of all incompatible task pairs
LT	Set of all linked task pairs
LD	Set of task pairs with minimum (lower bound) distance
UD	Set of task pairs with maximum (upper bound) distance
x_{ik}	$= 1$ if task i assigned to station k
	$= 0$ otherwise

[33, 52, 54, 60], supply chain configuration problems [25], timetabling [1], the car sequencing problem [49]. Brailsford et al. [11] can be referred as a literature review for other applications of CP.

First formulation of ALBP as CP has been proposed by Bockmayr and Pisaruk [7]. They offered a combined method of integer programming with CP. Pastor et al. [36] presented a comparative study in the performance of CP and MIP for type 1 and type 2 simple ALBP. Although the authors stated that these methods cannot be compared exactly in terms of efficiency, they conclude that CP model was better and have a faster formulation than MIP even for large size instances. Schaus and Deville [46] have modeled the ALBP as a bin packing problem with precedence constraints where the bins were workstations and the items were tasks. Also, Topaloğlu et al. [53] have proposed a solution procedure with rule-based CP modelling to solve ALBP. They presented a comparative result of CP and integer programming in terms of modelling capacity, solution quality and CPU time. Bukchin and Raviv [12] developed different CP models for various ALBP such as straight type I and type II, U-shaped type I and the task assignment and equipment selection problem. Their results indicated that CP performs better than MILP to solve the ALBPs. These studies did not consider the assignment restrictions. Since the assignment restrictions reduce the domain intervals for the decision variables in CP model, CP could quickly reach a solution.

3.1. Constraint programming approach

A CP model is generally expressed by Constraint Satisfaction Problem (CSP). CSPs are the problems with constraint sets and no objective function in the model. The formal definition of a CSP is as follows:

Definition 1. *A CSP is notated as a triple such that (X,D,C), where;*

X is a finite set of n variables $X = \{x_1, ..., x_n\}$

D which is called a domain is the set of possible values that may be assigned to each variables, $v_i \in D(x_i)$ $i = 1, ..., n$.

$C = \{C_1, ..., C_m\}$ *is a finite set of constraints. C is a relation between some variables of X and* $var(C_j) = \{x_1, ..., x_k\}$ *is referred this subset of variables* S_i*;* $C_j \subseteq D(x_1) \times \cdots D(x_k)$,$S_i \subseteq X$.

A solution of a CSP consists of assigned values to the variables when all constraints are satisfied. An assignment is a set of variable/values couples such that $A = \{(x_1, v_1),\ (x_2, v_2), ..., (x_r, v_r)\}$. If an assignment satisfies all constraint, it is called consistent; otherwise it violates one or more constraints and it is inconsistent. If all the variables take a value from their domain then an assignment is *complete*, otherwise it is *partial* [50].

Briefly, a CSP is formulated as [28]:

$$C_i(x_1, x_2, ..., x_n) \ = \ 1, 1 \leq i \leq m$$

$$x_j \in D_j, 1 \leq j \leq n$$

In this model, if $C_i(x_1, x_2, ..., x_n)$ is satisfied, i.e. is equal to 1, a consistent solution has been achieved. Any CSP solver consists of two main aspects: variable selection and value selection procedures. CSP approach begins by selecting an unassigned variable and then a value is assigned to that variable by using propagation and domain reduction when all the constraints are satisfied.

Furthermore, CP solvers have two fundamentals concepts in general: search tree and propagation-domain reduction. In search tree, a decision variable is declared as a node and a possible assignment is the branch related to the variable. The search starts with an empty assignment and proceeds until there aren't any variables that can be assigned a value. If the search could not reach a possible solution, backtracking mechanism is executed to try some other branches. Iteratively, the search tree has been constructed. Many search strategies can be used to guide the backtracking for obtaining an assignment: depth first, multi-point, restart and automatic [42].

The propagation (or consistency) is used to filter the variable domains by eliminating the inconsistent values. As soon as a variables' domain is changed, constraints related to that variable are propagated. Domain reduction is a process which removes the non-assigned variable values that do not satisfy the constraints. Hence, we obtain a consistent assignment at every iteration of the domain reduction. Let an illustrative example on the point of ALBP to explain the concepts of

the constraint propagation. Assume that, tasks 2 and 5 are two selected tasks with their domain intervals $D_2 = [4, 7]$ and $D_5 = [2, 5]$ respectively and the variables defined as the station number assigned to the tasks are x_2 and x_5. The precedence relation between these tasks is as task 2 should be preceded the task 5 ($x_2 \leq x_5$). Since the constraint states that x_5 should be greater than or equal tox_2, $x_5 = 2$ and $x_5 = 3$ is removed by reducing domain of the task 5. So, D_5 becomes $[4, 5]$. This process is applied for all of precedence relations between tasks, so that all of the variable domains will be modified.

The overall solution procedure of the CSP can be defined as branch and propagate which is similar to branch-and-bound for combinatorial optimization problem. For more detailed information about fundamental concepts of CP, the reader may review [15, 38]. Algorithm 1 is summarizes the CP solution framework.

Algorithm 1: CP solution procedure adapted from [37] and configured for AR-ALBP

Step 1. (Task Selection)
Define variables
Initialize variable domains
Generate the set of constraints
Select a task to assign
Assign a value for selected task
Go to Step 2
Step 2. (Propagation and Domain Reduction)
Apply propagation and domain reduction procedure to the partial assignment
Adjust decision variables
If a feasible solution found then go to Step 3
Otherwise go to Step 2.1.
Step 2.1. (Inconsistency Check)
There is no consistent assignment then Backtrack
Otherwise go to Step 3
Step 3. (Termination)
There exists any unassigned task to the station then go to Step 1(Bactracking)
Otherwise Stop and return the result.

In the above algorithm, a search tree which is the main idea is built. Firstly, variables, domains and constraints set are defined and generated. While there are tasks still remaining to be assigned to stations, one of these jobs is selected with the partial assignment. After a value is assigned to the selected task, the propagation and domain reduction procedure is applied to the current partial assignment and then decision variables take a value. In this step, if a feasible solution found the algorithm ends and the solution is reported, if not go to inconsistency checking step. In the Step 2.1, inconsistency checking is performed and a consistent solution is found then go to Step 3. If the inconsistency checking is not verified, the algorithm backtracks. If there is an unassigned task to the station, the algorithm

returns the Step 1 (i.e. backtracking), in other cases the algorithm terminates and the results are reported.

Whereas CSP is used to find a feasible solution, constraint optimization problem (COP) allows the use of a predefined objective function. The objective f is a function which is related to an assignment launched by the backtracking. Objective function can be minimized or maximized by the assignment A.

4. Proposed Constraint Programming Model for AR-ALBP

Our CP model is based on the model introduced by [36]. The model consists of Equations 1, 3 and 5 as mentioned below. However, we propose a new CP model to generate all feasible task assignments by adding problem specific assignment restrictions.

Considering that Definition 1 above, ALBP can be modelled as a CSP without any objective function as follows.

- X is a decision variable set. $x_i \in X; \forall i \in T$: The value of this variable gives the assigned number of station of task i.
- D is domain for each decision variable. Each task can be assigned to a station among station interval$D(x_i) = \{1, ..., \|W\|\}$.
- The set of constraints C for ALBP:
  - The all precedence constraints between tasks can be satisfied:

$$x_i \leq x_j \forall (i,j) \in \Pr$$

Our proposed constraint optimization programming models are as follows:

Objective Function: The objective function in CP model is the minimization of the cycle time.

$$Minimize \quad c \tag{1}$$

Decision variable:

$$stationNumber\,[Task_i] \in W \qquad \forall i \in T \tag{2}$$

Differing from a mathematical model, decision variable is stated as the assigned station number for task and an integer value between $1, ..., \|W\|$. We can easily satisfy all the restrictions by using this decision variable in constraint programming model.

Precedence relations: The precedence relations are satisfied by the following statement:

$$stationNumber\,[Task_i] \leq stationNumber\,[Task_h] \qquad \forall\,(i,h) \in Pr \quad (3)$$

Occurrence and station restrictions: These constraints express the requirement of at least one task is assigned to a station. If a task i must be assigned to a certain station j, the restriction is stated as $stationNumber[Task_i] = j$.

$$\underset{i \in T}{\exists}\ stationNumber\,[Task_i] : (stationNumber\,[Task_i] = j) \qquad \forall j \in W \quad (4)$$

To satisfy the occurrence restrictions, the statement $count(stationNumber\,[Task_i; \forall i \in T]\,; j) \geq 1 \forall j \in W$ is scripted. The $''count''$ function is a special construct in IBM ILOG Software. Using this form, at least one variable of $stationNumber\,[Task_i]$ takes the station number j.

Cycle time restrictions:

$$\sum_i t_i \leq c \qquad \forall i \in T | \forall j \in W \wedge (stationNumber\,[Task_i] = j) \quad (5)$$

The station time is the sum of task times which are assigned to the station. This time must not exceed the cycle time. Equation 5 ensures that these restrictions are satisfied. We use the *pack* constraints to formulate the Equation 5.

$$pack(stationTime, stationNumber, t_i) \quad (6)$$

where *stationTime* is the total task time assigned to that station.

Task assignment restrictions:

We can easily establish the task assignment restrictions with the above decision variable as follows:

$$stationNumber\,[Task_i] = stationNumber\,[Task_j] \qquad \forall (i,j) \in LT \quad (7)$$

$$stationNumber\,[Task_i] \neq stationNumber\,[Task_j] \qquad \forall\,(i,j) \in IT \quad (8)$$

In this way, all of the task assignment constraints are satisfied clearly. For example $stationNumber\,[Task_1] = stationNumber\,[Task_4]$ ensure that task 1 and task 4 must be assigned into the same station. Accordingly, for incompatible tasks

of $stationNumber\,[Task_6] \neq stationNumber\,[Task_7]$ statement provides that task 6 and task 7 cannot be assigned to the same station.

Distance restrictions: Station intervals between tasks can also be easily defined by using our decision variable in the CP model. It is formulated as follows:

For minimum distance restrictions;

$$|stationNumber\,[Task_i] - stationNumber\,[Task_j]| \geq d_{ij}^{-} \qquad \forall\,(i,j) \in LD \quad (9)$$

For maximum distance restrictions;

$$|stationNumber\,[Task_i] - stationNumber\,[Task_j]| \leq d_{ij}^{+} \qquad \forall\,(i,j) \in UD \quad (10)$$

Cycle time and the line efficiency are performance measures to evaluate the assignments. The procedure finds task assignments that have the minimum cycle time; hence the maximum line efficiency. There aren't any models in the literature for Type-2 AR-ALBP using constraint programming. The above procedure can generate all possible task assignments for relevant data set and compute the objective functions for a given number of stations.

5. Mixed Integer Programming Model for AR-ALBP Type-2

The mathematical program for Type-2 AR-ALBP by adapting formulations for Type-1 AR-ALBP from [48] is as follows:

Objective Function: The aim is also the minimization of the cycle time for a given number of stations.

$$Minimize \quad c \qquad (11)$$

Variable definition: The decision variables are defined as follows (if task $i.$ is assigned to station k,$x_{ik} = 1$; otherwise$x_{ik} = 0$)

$$x_{ik} \in \{0,1\} \qquad \forall (i,k) \in (T \times W) \quad (12)$$

Occurrence and station restrictions: Each task must be assigned to exactly one station.

$$\sum_{k=1}^{\|W\|} x_{ik} = 1 \quad \forall i \in T \quad and \qquad x_{ik} = 1, k \quad \text{predefined station for task } i \quad (13)$$

Precedence restrictions: A task can be assigned to a station only if all its predecessors have been assigned to that station or earlier stations:

$$\sum_{k=1}^{\|W\|} k \cdot x_{ik} \leq \sum_{k=1}^{\|W\|} k \cdot x_{jk} \qquad \forall (i,j) \in Pr \quad (14)$$

Cycle time restrictions: Any station time must be less than the cycle time:

$$c \geq \sum_{i=1}^{\|T\|} t_i \cdot x_{ik} \qquad \forall k \in W \quad (15)$$

Incompatible tasks restrictions: Task i and j, $(i,j) \in IT$, must not be assigned to the same station k.

$$x_{ik} + x_{jk} \leq 1 \qquad \forall\, (i,j) \in IT \quad (16)$$

Linked tasks restrictions: Task i and j, $(i,j) \in LT$, must be assigned to the same station, if the task pair has at least one common station within the possible assignable station sets [48].

$$\sum_{k=1}^{\|W\|} k \cdot x_{ik} = \sum_{k=1}^{\|W\|} k \cdot x_{jk} \qquad \forall (i,j) \in LT \quad (17)$$

Minimum distances: We can state minimum distance restriction for task i and j,$(i,j) \in LD$, as follows:

$$\left| \sum_{k=1}^{\|W\|} k \cdot x_{jk} - \sum_{k=1}^{\|W\|} k \cdot x_{ik} \right| \geq d_{ij}^{-} \qquad \forall (i,j) \in LD \quad (18)$$

Maximum distances: This restriction for task i and j, $(i,j) \in UD$, can also be expressed as follows:

$$\left| \sum_{k=1}^{\|W\|} k \cdot x_{jk} - \sum_{k=1}^{\|W\|} k \cdot x_{ik} \right| \leq d_{ij}^{+} \qquad \forall (i,j) \in UD \quad (19)$$

The *abs* which is a function in the CP solver is used to define the absolute value for distance constraints.

Capacity utilization of the line which is used as another performance parameter in this study is measured in Equation 20. While the cycle time is minimized by reducing the idle time in the assembly line, the line efficiency is maximized.

$$Line\,eff. = \frac{\sum\limits_{i \in \|T\|} t_i}{\|W\| \cdot c} \tag{20}$$

6. Constraint Programming versus Mixed Integer Programming

Mathematical models have important characteristics: variables, constraints and a solution space. The number of variables and types (e.g. integer, binary) in the model are directly related to the problem complexity. Solution space consists of all feasible solution obtained by an enumeration procedure. Even it restricts the solution space, the number of constraints may further complicate a solution, These characteristics may lead to an exponential growing solution time while the problem size increases. In order to cope with those problems, new logic programming techniques have been developed by researchers. One of these techniques is CP and now used widely for solving several combinatorial problems.

Puget and Lustig [38] compared CP and MIP in terms of three dimensions: modelling ability, node processing and search strategy. Modelling concepts of two approaches consist of general elements: decision variables, constraints, objective function and search tree. Even theory of search strategies is very similar, two approaches has different approach with regard to node processing.

Lustig and Puget [28] stated that CP differs from MIP in two fundamental manners. Firstly, while MIP divide the search space with non-fractional value of decision variables, CP splits search space by choosing a point that is generated by using any set of constraints. To eliminate the suboptimal solution, MIP computes a lower bound for every node of current search. But CP is concerned with the elimination of infeasible solutions. In the latter manner, variable selection step of branch and bound can be developed by CP framework. CP allows the users to determine his own branching strategy with regards to formulation of the problem. Combining of CP and LP can be very attractive research area in which the researcher can build problem-specific search strategies.

CP has some advantages over the Mixed-Integer Programming (MIP). One of the strong features of CP is the representation language. CP has specialized constraints, logical constraints and non-linear cost functions or constraint that are easily defined in a natural and compact way. In contrast, MIP models support only linearized logical constraints or quadratic convex constraints. CP has some disadvantages as well. One can be counted as variable definition compared to MIP. Although all CP solvers support discrete variables, continuous variables are not supported in some solvers [43]. However, CP can be an alternative to MIP for allocation problems that have slow convergence. Even if the CP is generally needs

more usage, it can be say that this situation is not valid any-more because the CP solvers have improved significantly in the last few years.

Overall, modelling the problem as a constraint programming is simpler due to the flexibility of CP solver formulation [12]. Moreover, CP can find feasible solutions quickly by using specific search and propagation algorithm. Also fewer nodes are explored with CP by means of domain reduction technique. Considering these advantages, CP can be a preferable and efficient approach to solve the assembly line balancing problem.

7. Numerical Results

In order to assess the proposed approach, 9 test instances are solved varying the number of tasks between a minimum of 25 and a maximum of 148 tasks. These test instances are from well-known benchmark data set and can be downloaded from http://www.alb.mansci.de. We emphasize that these experimented instances are noted as moderate to large size instances. The existing data set instances include only precedence constraints. For AR-ALBP-2 evaluations, appropriate task assignments, station and distance restrictions are added by the authors for each test problem. These restrictions are generated in a way that the known optimal solution for ALBP remains feasible for AR-ALBP as well. They are listed in Table 3 and Table 4. For example, in Warnecke problem (P58), there exist a linked restriction between task 43 and 45, and incompatible restriction between task 34 and 41. For station restrictions -in 10 station problem-, tasks 3, 6 and 22 must be assigned to the station 3, tasks 36 and 41 to station 6, and vice versa. Between tasks 16 and 22, a minimum two-station distance must be kept. Tasks 44 and 55 should have maximum one-station distance in between.

MIP models for AR-ALBP-2 problem are solved using IBM ILOG CPLEX version 12.7.1 and constraint programming models are solved by IBM ILOG CP Optimizer version 12.7.1. Automatic search strategy is preferred as default setting in ILOG to improve the search performance and to guide the current solution towards the optimal solution. The time limit for each run is 10,000 seconds for MIP and CP. Cycle time, line efficiency, memory usage and CPU time are reported as the performance measurements. All test instances are run by using a PC with Intel Core i3-2120, 3.30 GHz processor and 4 GB memory.

Table 3. List of linked and incompatibles restrictions

Author	Problem	Linked restrictions	Incompatibles restrictions
Roszieg	P25	(9,13), (23,25)	(1,8), (8,24), (15,21), (19,25)
Gunther	P35	(2,17), (30,34)	(3,19), (8,13), (17,25), (26,34)
Kilbridge	P45	(1,15), (22,33), (38,45)	(7,31), (18,42), (39,43)
Warnecke	P58	(43,45), (46,47)	(13,27), (34,41), (49,52)
Wee-Mag	P75	(28,56), (41,44), (64,74)	(3,24), (6,51), (11,25), (24,41), (39,51), (52,73), (59,75)
Lutz	P89	(8,15), (19,20), (32,34), (44,63)	(10,33), (20,34), (45,80), (83,85)
Mukherje	P94	(22,26), (29,59), (52,78), (81,82), (88,90)	(5,33), (12,17), (23,54), (40,56), (57,65), (64,71), (72,90), (82,88)
Arcus	P111	(14,22), (47,56), (92,93), (107,109)	(10,25), (17,52), (26,33), (45,64), (71,104), (105,108)
Bartholdi	P148	(4,8), (8, 36), (46,47), (57,65)	(4,140), (17,33), (39,123), (81,118), (127,134), (139,142)

TABLE 4. List of stations and distance restrictions

Problem	#stations	Station restrictions{task, station}	Distance restrictions{(task, task),distance}	
			Minimum	Maximum
P25	4	{11,3}	{(4, 22),1}	{(15,24),1}
	8	{16,5}	{(18,22),1}	{(6,7),1}
P35	9	{4,3}, {(20,21),5}	{(10,16),2}	{(3,25);4}
	14	{(11,25,26),11}, {18,5}	{(16,20),2}	{(25,26),2}
P45	4	{5,3}, {9,4}, {23,2}	{(21,24),1}, {(25,28),1}	{(3,10),2}, {(18,21),1}
	10	{3,7}, {14,2}, {(37,43),4}	{(2,10),2}, {(25,28),1}, {(33,41),3}	{(5,9),1}, {(22,33),1}, {(29,37),4}
P58	10	{(3,6,22),3}, {(36,41),6}, {50,9}	{(16,22),2}, {(14,17),4}	{(18,22),1}, {(44,55),1}
	17	{(2,22),4}, {(13,20),5}, {35,12}	{(2,3),3}, {(14,23),4}	{(7,16), 3}, {(43,48),1}, {(52,56),1}, {(53,57,1)}
P75	15	{(13,22),4}, {29,13}, {44,11}, {46,4}, {72,10}, {56,11}	{(24,64),2}, {(4,16),2}, {(50,68),1}, {(24,63),3}	{(4,42),3}, {(23,69),2}, {(68,75),4}
	22	{(3,7,),5}, {14,9}, {20,7}, {24,6}, {(29,36,40),8}, {37,16}, {(56,75),12}, {70,20}	{(4,16),2}, {(58,63),2}	{(28,41),1}, {(31,40),1}, {(55,59),1}, {(69,71),4}
P89	19	{(8,16),3}, {(26,31),6}, {49,9}, {59,12}, {66,13}	{(1,5),1}, {(23,31),2}, {(40,50),2}, {(43,60),3}	{(74,82),2}, {(20,30),2}, {(49,55),4}
	28	{6,1}, {16,10}, {37,15}, {(60,62,65),19}, {(83,84,87),27}	{(1,5),1}, {(23,31),2}, {(40,50),2}, {(43,60),3}	{(74,82),2}, {(20,30),2}, {(49,55),4}
P94	16	{2,2}, {12,4}, {35,7}, {(42,51,61,71),9}, {53,6}, {77,10}, {11,4}	{(14,21),1}, {(53,57),4}, {(76,77),2}	{(5,12),2}, {(35,37),2}, {(90,93),1}
	26	{12,8}, {(14,18,21), 5}, {(47,57), 16}, {85,12}	{(39,49),3}, {(54,61),2}, {(68,74),2}, {(74,77),4}	{(10,14),1}, {(20,31),4}, {(55,79),2}
P111	13	{(8,12,14,18,30),4}, {(95,100,105),12}	{(8,9),2}, {(34,42),2}, {(76,77),2}, {(91,95),1}	{(15,21),2}, {(54,62),1}, {(103,106),1}
	27	{8,10}, {12,2}, {14,3}, {(18,23,42),9}, {(95,100,105),26}	{(8,9),2}, {(34,42),2}, {(76,77),2}, {(91,95),1}	{(15,21),2}, {(54,62),1}, {(103,106),1}
P148	10	{(14,21,34,79), 4}, {(127,138),6}	{(5,34),2}, {(75,79),2}, {(130,140),1}	{(7,15),1}, {(53,71),2}, {(112,144),2}
	15	{9,4}, {20,12}, {(62,63,64,65),9}, {145,11}	{(5,34),2}, {(75,79),2}, {(130,140),1}	{(7,159,1}, {(53,71),1}, {(112,144),2}

Four experimental cases are designed to evaluate the effects of assignment restrictions on the line balance:

- Case 1: linked and incompatible task assignment restrictions
- Case 2: station and distance restrictions
- Case 3: all assignment restrictions
- Case 4: a real-life problem from a furniture manufacturing industry

In all of the experiments, precedence and cycle time constraints are default restrictions. In the following sections we present the numerical results case by case.

7.1. Case 1: AR-ALBP with task assignment restrictions

In this first case, we consider the linked and incompatible task assignment restrictions as listed in Table 3. CPU time and cycle time are used as the metric for comparison of the solutions. Each test problem is also solved with different station numbers.

As it can be seen in Table 5, MIP and CP found the optimal cycle time for test instances between P25 and P45 (small-size instances). For these instances, CPU times of MIP and CP models are very close. In the medium-size instances (P58-P89),while MIP has attained the optimal for all instances, CP has reached the optimal solutions except in P75 with 15 stations. In terms of CPU times, CP models are lower than MIP models. For large-size instances (P94-P148), CP has reached the optimal solutions for all instances and is significantly faster than MIP. Furthermore, it can be said that CP is more stable than MIP in terms of the memory usage. Fair line efficiency, on the average of 0.93, is achieved except in a few test instances (e.g. P111 with 27 numbers of stations and P148 with 15 numbers of stations).

7.2. Case 2: AR-ALBP with station and distance assignment restrictions

In this second case, the station and distance restrictions as listed in Table 4 are considered. As it can be seen in Table 6, while CP has reached the optimal solutions for all instances, MIP has reached the optimal solution except P111 with 13 station. In terms of the CPU time, although both models generally show almost same performance, it can be said that MIP is faster than CP for large-size instances. The station and distance restrictions have limited the solution space and the models could converge to the optimal solution quickly.

It is worth mentioning that the line efficiencies decrease due to the affect of assignment restrictions. Optimum cycle time of the solutions are also effected. For instance, in the P89 problem, while line efficiencies were 0.95 and 0.91 in Case 1, 0.88 and 0.82 are obtained in Case 2. The optimum cycle times in Case 1 were 26 and 19 time units and in Case 2, 29 and 21 time units, respectively. Overall, assignment restrictions led the line efficiency to decrease, while the cycle time to increase.

TABLE 5. Test results of Case 1

Problem	#stations	MIP			CP			Line eff.
		c	CPU	Memory	c	CPU	Memory	
P25	4	32	0.05	7.12	32	0.17	5.20	0.98
	8	17	1.25	11.82	17	0.32	5.30	0.92
P35	9	55	2.75	12.80	55	0.46	5.90	0.98
	14	40	7.43	16.34	40	0.29	5.90	0.86
P45	4	138	0.22	10.85	138	0.10	5.30	1.00
	10	56	0.98	12.01	56	0.28	6.30	0.99
P58	10	155	7.68	24.13	155	0.90	7.00	1.00
	17	92	64.78	1365.13	92	5.56	8.00	0.98
P75	15	100	43.06	40.60	101	10000.00	109.80	0.98
	22	69	71.34	54.59	69	2.30	8.80	0.97
P89	19	26	34.04	47.79	26	8.20	9.80	0.95
	28	19	46.82	52.60	19	2.77	10.00	0.91
P94	16	268	237.61	161.94	268	2.05	9.20	0.98
	26	171	110.18	86.81	171	1.52	9.20	0.95
P111	13	11574	10000.00	764.48	11571	1590.00	20.10	1.00
	27	9210	55.65	45.09	9210	3.71	12.90	0.60
P148	10	564	14.90	31.19	564	0.81	11.60	1.00
	15	494	23.63	33.20	494	0.68	12.00	0.76

TABLE 6. Test results of Case 2

Problem	#stations	MIP			CP			Line eff.
		c	CPU	Memory	c	CPU	Memory	
P25	4	32	0.06	5.53	32	0.06	4.50	0.98
	8	16	0.25	9.85	16	0.07	4.80	0.98
P35	9	56	0.22	8.23	56	0.23	5.80	0.96
	14	40	0.62	11.94	40	0.09	5.20	0.86
P45	4	138	0.11	6.86	138	0.04	5.20	1.00
	10	56	0.76	13.61	56	0.23	6.10	0.99
P58	10	160	1.64	9.12	160	1.17	6.70	0.97
	17	95	11.36	17.90	95	9.87	7.60	0.96
P75	15	100	55.93	34.05	100	21.63	9.30	1.00
	22	72	1.06	16.29	72	0.39	7.90	0.95
P89	19	29	4.26	17.18	29	0.49	8.30	0.88
	28	21	9.43	23.36	21	68.81	10.90	0.82
P94	16	284	2.98	22.02	284	10000.00	81.30	0.93
	26	171	4.56	21.66	171	1.60	10.10	0.95
P111	13	13716	10000.00	571.30	13714	3506.41	37.00	0.84
	27	11188	1.15	28.25	11188	0.09	8.40	0.50
P148	10	564	3.43	19.01	564	0.15	10.50	1.00
	15	501	1.31	22.06	501	6256.81	59.10	0.75

7.3. CASE 3: AR-ALBP WITH ALL TASK ASSIGNMENT RESTRICTIONS

In this case, all the task assignment restrictions listed in Table 3 and Table 4 are considered concurrently. The experimental results are reported in Table 7. While the optimal solutions for all problems are obtained. MIP is generally superior to CP with respect to the CPU time. As in previous case, solution spaces

of the problems decrease as more assignment restrictions are added. In terms of the memory usage, CP is superior than MIP.

As expected, similar situations of Case 1 and Case 2 arises in respect to the line efficiency and optimal cycle time. As assignment restrictions force several rules, the line efficiency will decrease and cycle time will increase.

TABLE 7. Test results of Case 3

Problem	#stations	MIP			CP			Line eff.
		c	CPU	Memory	c	CPU	Memory	
P25	4	36	0.41	4.62	36	0.20	5.20	0.87
	8	18	0.22	6.51	18	0.17	5.20	0.87
P35	9	56	0.28	7.29	56	0.25	5.80	0.96
	14	42	0.39	9.53	42	0.24	5.90	0.82
P45	4	138	0.22	9.51	138	0.06	5.10	1.00
	10	58	0.20	7.00	58	0.15	6.10	0.95
P58	10	160	0.92	11.67	160	0.35	6.60	0.97
	17	96	32.99	20.01	96	54.83	8.60	0.95
P75	15	104	469.50	314.27	104	10000.00	104.90	0.96
	22	75	0.84	16.05	75	0.80	7.80	0.91
P89	19	32	0.70	16.73	32	0.49	8.30	0.80
	28	21	15.83	23.52	21	78.20	11.30	0.82
P94	16	284	6.71	21.98	284	10000.00	86.20	0.93
	26	171	8.07	28.35	171	2.24	9.30	0.94
P111	13	13716	10000.00	407.50	13714	564.97	16.70	0.84
	27	11188	1.17	28.36	11188	0.10	8.90	0.50
P148	10	564	3.00	19.31	564	0.15	10.00	1.00
	15	581	0.68	18.12	581	181.25	15.40	0.65

7.4. CASE 4: APPLICATION OF THE PROPOSED MODEL TO A REAL-LIFE CASE

In this section, an experimental case study is carried out to apply the procedure in a real manufacturing firm. Motivated by a balancing problem in its assembly line, the company consulted the authors for a solution. The company is one of the main furniture companies in Turkey which produces bed bases for several sectors such as hotels. In a sector with a competitive and intensive demand, thorough and fast production has crucial importance for the company .

After the constructed configuration, re-balancing problems have arisen due to some changes in operational and technical conditions. These conditions are related to new product models, new machine updates, technological developments, labour needs, changing the production method and fluctuations in customer demand. Hence, company's operations managers wanted to minimize cycle time of the line by assigning the appropriate tasks into the stations.

The proposed solution procedure is applied to re-balance the new bed base assembly line. The bed base production process has 37 tasks. Many of these tasks require sensitive production techniques. In this context, the number of stations are given as 5. Also, due to required modifications mentioned above, several assignment restrictions have arose. Bed base production tasks and required task

times are shown in Table 8. Moreover, Figure 1 shows the precedence diagram of bed base process.

TABLE 8. Bed case production tasks and task times for Case 4

Number	Operation	Time (sec)
1	Moving and fixing the bottom body to the assembly table	23
2	Moving of the right frame to be assembled on the bottom body	12
3	Assembling the right frame on the bottom body	35
4	Moving the left frame to be assembled on the bottom body	12
5	Gluing and assembling the left frame to the bottom body	35
6	Quality checking	7
7	Moving the front frame to be assembled on the bottom body	12
8	Gluing the front frame to the bottom body	35
9	Moving the rear frame to be assembled on the bottom body	12
10	Gluing the rear frame to the bottom body	35
11	Gluing the middle bar to be fixed to the bottom body	22
12	Moving the corner fixtures onto the table	16
13	Drilling on the case for assembling the corner fixtures	54
14	Assembling the corner fixtures on the case	36
15	Drilling the caster wheel holes on the bottom body	26
16	Punching of the right frame	42
17	Punching of the left frame	42
18	Punching of the front frame	26
19	Punching of the rear frame	26
20	Drilling the screw hole of the bed base head on the case	13
21	Drilling the stopping holes on right side frame	14
22	Drilling the stopping holes on the left side frame	14
23	Moving the stop springs onto the assembly table	12
24	Assembling the right stopping spring on the wooden frame	38
25	Assembling the left stopping spring on the wooden frame	38
26	Moving the top frame onto the assembly table	10
27	Drilling the holes of the stopping spring	18
28	Assembling the stop spring on the top frame	22
29	Quality checking	15
30	Assembling of the holder on the top frame	19
31	Drilling the holes of the clips on the right, left and front frames	38
32	Assembling of the clips	43
33	Quality checking	10
34	Turning aside the bed base	5
35	Assembling the caster wheels in the caster holes	20
36	Quality checking	16
37	Packaging	55

Essential assignment restrictions are reported in Table 9. For example, linked assignment restriction is needed between task 2 and 3, since these jobs are very similar jobs and require similar equipment to assembly. Task 19 and 22 should have minimum one stations distance restriction in between. While task 19 needs a heavy machine (e.g. punch machine or press) to produce, task 22 needs precision manufacturing techniques to drill the holes sensitively. Hence, these two tasks must be assigned far away from each other. Furthermore, quality checking processes are required at certain stations intervals to produce robust products. Therefore the

FIGURE 1. Precedence diagram of bed base production

firm wants to assign quality checking tasks to the certain stations (e.g. first quality checking task (task six) must be assigned to the station three).

TABLE 9. List of assignment restrictions of the bed base production for Case 4

Types	Restrictions
Incompatible	(21,22), (21,25), (22,24), (24,25)
Linked	(2,3), (4,5), (7,8), (9,10), (12,14)
Maximum distance	{(26,28),1)}
Minimum distance	{(19,22),1}
Stations	(6,3), (29,5), (33,7), (36,9)

The comparison results for MIP and CP models in terms of the cycle time are reported in Table 10. The CP and MIP models have reached the same optimal solution for first three assignment restrictions cases mentioned in Section 7. Furthermore, assigned tasks to the stations are reported in Table 11. These results state that the proposed CP approach can be easily applied to solve real life problems.

TABLE 10. Comparison of MIP and CP model in terms of the cycle time for Case 4

Case	MIP	CP
Without assignment restrictions	104	104
1	106	106
2	106	106
3	109	109

8. DISCUSSION

As noted in above experimental cases, assignment restrictions affect the convergence of the solution and the computing time. When the assignment restrictions are added to simple ALBP, feasible task assignment combinations reduce. They also influence the problem complexity. Optimal solution and optimum task assignment can change by adding these restrictions into the problem. Addition of these assignment restrictions also reduces the problem constraints and variables [47]. A well-known property of mathematical programming is that reduction of the problem variables can make it easier to solve the problem [6]. CP uses this advantage with fewer variables than their MIP counterparts. Furthermore, CP has more constraints than MIP model even if the problem has no additional (assignments)

TABLE 11. Assigned tasks to the stations for each case with proposed model

Stations	Cases			
	Without assignment restrictions	1	2	3
1	1,2,4,5,7,26	1,2,3,26,27	1,2,3,4,7,9	1,2,3,23
2	3,9,10,23	4,5,7,8,23	5,8,10	4,5,9,10
3	6,8,16,21	6,9,10,18,19	6,16,17,23	6,7,8,17
4	11,17,24	11,16,17	21,22,25,26,27	16,21,24,26
5	18,19,22,31	13,21,24	18,24,28,29	22,25,27,28,29
6	12,15,27,32	12,14,15,20,22	12,31,32	18,31,32
7	13,14,20	25,28,29,30	13,14,33	11,15,19,30,33
8	25,28,29,30,33	31,32,33,34	11,15,19,20,30	12,13,14
9	34,35,36,37	35,36,37	34,35,36,37	20,34,35,36,37

constraints. In terms of the objective functions, it can be claimed that the model without any assignment restrictions is a lower bound for the model with assignment restrictions. These results are shown at Table 12.

TABLE 12. Comparison of size and performance of the CP and MP model without assignment restrictions

Problem	#stations	MIP					CP				
		#variables	#constraints	c	CPU	Memory	#variables	#constraints	c	CPU	Memory
P25	4	102	65	32	0.11	5.92	30	96	32	0.21	5.20
	8	202	73	16	0.67	10.39	34	104	16	0.32	5.50
P35	9	317	98	54	1.09	11.82	45	139	54	0.31	6.00
	14	492	108	40	2.85	12.45	50	149	40	0.57	6.20
P45	4	182	115	138	0.16	10.12	50	166	138	0.09	5.30
	10	452	127	56	1.17	13.14	56	178	56	0.43	6.30
P58	10	582	148	155	10.17	20.79	69	212	155	0.90	7.10
	17	988	162	92	1527.39	109.29	76	226	92	8.15	8.10
P75	15	1127	192	100	21.06	37.56	91	273	100	17.12	9.20
	22	1652	206	69	21.96	50.45	98	287	69	1.95	8.70
P89	19	1693	145	26	28.42	47.14	109	340	26	1.31	9.10
	28	2494	263	18	54.11	47.80	118	358	18	2.57	9.90
P94	16	1506	307	268	3195.85	97.43	111	407	268	2.32	9.30
	26	2446	327	171	24.73	40.10	121	427	171	2.13	10.20
P111	13	1445	313	11571	9504.00	1345.56	125	430	11571	1995.61	21.90
	27	2999	341	5689	90.29	51.41	139	458	5689	21.23	13.60
P148	10	1482	343	564	7.92	32.18	159	497	564	0.85	11.70
	15	2222	353	383	5.32	36.32	164	507	383	1.63	12.40

In general, both MIP and CP have demonstrated very good performance in achieving the optimal solutions for test instances. Line efficiency has decreased as we add more assignment restrictions within the model. On the other hand, the optimal cycle times increase. The reason is that assignment restrictions may force some tasks to be assigned to different stations. This situation arises from Eq. (19).

Both models performed same for small-size instances with regard to CPU time. For other instances, while we can generally say that CPU time decreased by adding assignment restrictions for the MIP model, there is no significant change for the CP model.

Another property is CP models have less variables due to the domain interval for each decision variables. CP models work with far less number of variables than MIP models. However, CP models have more number of constraints than MIP models which is an advantage for CP to converge quickly.

A comparison of whether MIP or CP is quicker in terms of CPU times is displayed in Table 13. The numbers of instances for each case are counted for experiments. Especially in small-size instances, both models exhibit almost equal CPU time behaviour. For other instances particularly in Case 1, CP is quicker in solution time. CP gives a good performance for other cases as well. In general, depending on the problem complexity and size, CP is more efficient for solving AR-ALBPs.

TABLE 13. The numbers of instances for each case in terms of the CPU time comparison

	Case1	Case2	Case3
Almost equal	3	10	9
CP faster than MIP	14	5	4
MIP faster than CP	1	3	5

9. Conclusion and Future Researches

In this paper, straightforward modelling and efficient solution procedures for assembly lines with assignment restrictions are developed. The considered assignment restrictions are: linked and incompatible tasks, station restrictions and distance restrictions. The objective of the problem is to minimize the cycle time for a given number of stations and to maximize the line efficiency. To solve the problem, a constraint programming approach has been developed. This approach is an effective technique to solve several combinatorial problems such as scheduling problems, but merely applied to assembly line balancing field. In the solution procedure, CP generates all possible task assignments as feasible solutions while satisfying all restrictions. Then, considering the cycle time and the line efficiency as the performance measures, the best feasible or optimal assignment is obtained. The performance of the proposed procedure is tested and compared to MIP on nine literature data sets with 18 instances for three cases. The proposed model is also practiced to solve a line balancing problem in a furniture production firm in Turkey.

According to the numerical results, CP models give notable performance compared to MIP models especially for large-size instances. In addition, CP modelling is more practical and user-friendly than MIP modelling in terms of scripting language convenience. Consequently proposed solution approach is an effective alternative modelling technique to MIP in balancing assembly lines with several task assignment restrictions.

For future studies, the proposed approach can be improved with other ALBP types such as u-lines, parallel lines and two-sided lines. Another area is to consider several other objective functions such as minimization of the number of stations and cost of line balancing. And also, stochastic and dynamic task time problems can be solved using the proposed model.

References

[1] Slim Abdennadher and Michael Marte. University course timetabling using constraint handling rules. *Applied Artificial Intelligence*, 14(4):311–325, 2000.

[2] Kürşad Ağpak and Hadi Gökçen. Assembly line balancing: Two resource constrained cases. *International Journal of Production Economics*, 96(1):129–140, 2005.

[3] Joaquín Bautista and Jordi Pereira. Ant algorithms for assembly line balancing. In *International Workshop on Ant Algorithms*, pages 65–75. Springer, 2002.

[4] Joaquin Bautista and Jordi Pereira. Ant algorithms for a time and space constrained assembly line balancing problem. *European journal of operational research*, 177(3):2016–2032, 2007.

[5] Joaquin Bautista, Raúl Suárez, Manuel Mateo, and Ramón Companys. Local search heuristics for the assembly line balancing problem with incompatibilities between tasks. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 3, pages 2404–2409. IEEE, 2000.

[6] Mokhtar S Bazaraa, John J Jarvis, and Hanif D Sherali. *Linear programming and network flows*. John Wiley & Sons, 2011.

[7] Alexander Bockmayr and Nicolai Pisaruk. Solving assembly line balancing problems by combining ip and cp. In *Proceedings of the 6th Annual Workshop of ERCIM Working Group on Constraints*. ERCIM, 2001.

[8] Edward H Bowman. Assembly-line balancing by linear programming. *Operations Research*, 8(3):385–389, 1960.

[9] Nils Boysen and Malte Fliedner. A versatile algorithm for assembly line balancing. *European Journal of Operational Research*, 184(1):39–56, 2008.

[10] Nils Boysen, Malte Fliedner, and Armin Scholl. A classification of assembly line balancing problems. *European journal of operational research*, 183(2):674–693, 2007.

[11] Sally C Brailsford, Chris N Potts, and Barbara M Smith. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research*, 119(3):557–581, 1999.

[12] Yossi Bukchin and Tal Raviv. Constraint programming for solving various assembly line balancing problems. *Omega*, 2017.

[13] Brian J Carnahan, Bryan A Norman, and Mark S Redfern. Incorporating physical demand criteria into assembly line balancing. *IIE Transactions*, 33(10):875–887, 2001.

[14] Albert Corominas, Laia Ferrer, and Rafael Pastor. Assembly line balancing: general resource-constrained case. *International Journal of Production Research*, 49(12):3527–3542, 2011.

[15] Rina Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[16] Richard F Deckro. Balancing cycle time and workstations. *IIE transactions*, 21(2):106–111, 1989.

[17] Radu Gadidov and Wilbert Wilhelm. A cutting plane approach for the single-product assembly system design problem. *International Journal of Production Research*, 38(8):1731–1754, 2000.

[18] Michel Gourgand, Nathalie Grangeon, and Sylvie Norre. Metaheuristics based on bin packing for the line balancing problem. *RAIRO-Operations Research*, 41(2):193–211, 2007.

[19] Ozcan Kilincci. A petri net-based heuristic for simple assembly line balancing problem of type 2. *The International Journal of Advanced Manufacturing Technology*, 46(1-4):329–338, 2010.

[20] Yeo Keun Kim, Yeongho Kim, and Yong Ju Kim. Two-sided assembly line balancing: a genetic algorithm approach. *Production Planning & Control*, 11(1):44–53, 2000.

[21] Robert Klein and Armin Scholl. Maximizing the production rate in simple assembly line balancing—a branch and bound procedure. *European Journal of Operational Research*, 91(2):367–385, 1996.

[22] SD Lapierre and AB Ruiz. Balancing assembly lines: an industrial case study. *Journal of the Operational Research Society*, 55(6):589–597, 2004.

[23] Sophie D Lapierre, Angel Ruiz, and Patrick Soriano. Balancing assembly lines with tabu search. *European Journal of Operational Research*, 168(3):826–837, 2006.

[24] Tae Ok Lee, Yeongho Kim, and Yeo Keun Kim. Two-sided assembly line balancing to maximize work relatedness and slackness. *Computers & Industrial Engineering*, 40(3):273–292, 2001.

[25] Haitao Li and Keith Womer. Optimizing the supply chain configuration for make-to-order manufacturing. *European Journal of Operational Research*, 221(1):118–128, 2012.

[26] Chiun-Ming Liu and Chung-Hwa Chen. Multi-section electronic assembly line balancing problems: A case study. *Production Planning & Control*, 13(5):451–461, 2002.

[27] SB Liu, HL Ong, and HC Huang. A bidirectional heuristic for stochastic assembly line balancing type ii problem. *The International Journal of Advanced Manufacturing Technology*, 25(1-2):71–77, 2005.

[28] Irvin Lustig and Jean-Francois Puget. Constraint programming. In S.I. Gass and M.C. Fu, editors, *Encyclopedia of Operations Research and Management Science*, pages 267–273. Springer, 2013.

[29] C. Miralles. Solving procedures for the assembly line worker assignment and balancing problem: Application to sheltered work centres for disabled. In *Proceedings of the XI Escuela Latinoamericana de Verano en Investigacionde Operaciones*, Villa de Leyva, Colombia, 2005.

[30] Koichi Nakade, Akiyasu Ito, and Syed Mithun Ali. U-shaped assembly line balancing with temporary workers. *International Journal of Industrial Engineering*, 21(6):134–146, 2015.

[31] Andreas C Nearchou. Balancing large assembly lines by a new heuristic based on differential evolution method. *The International Journal of Advanced Manufacturing Technology*, 34(9-10):1016–1029, 2007.

[32] Andreas C Nearchou. Maximizing production rate and workload smoothing in assembly lines using particle swarm optimization. *International Journal of Production Economics*, 129(2):242–250, 2011.

[33] Cemalettin Ozturk and M Arslan Ornek. Optimisation and constraint based heuristic methods for advanced planning and scheduling systems. *International Journal of Industrial Engineering*, 23(1):26–48, 2016.

[34] Rafael Pastor, C Andres, A Duran, and M Perez. Tabu search algorithms for an industrial multi-product and multi-objective assembly line balancing problem, with reduction of the task dispersion. *Journal of the Operational Research Society*, 53(12):1317–1323, 2002.

[35] Rafael Pastor and Albert Corominas. Assembly line balancing with incompatibilities and bounded workstation loads. *Ricerca Operativa*, 30:23–45, 2000.

[36] Rafael Pastor, Laia Ferrer, and Alberto García. Evaluating optimization models to solve salbp. In *International Conference on Computational Science and Its Applications*, pages 791–803. Springer, 2007.

[37] Michael L Pinedo. *Scheduling: theory, algorithms, and systems*. Springer, 2012.

[38] Jean-Francois Puget and Irvin Lustig. Constraint programming and maths programming. *The Knowledge Engineering Review*, 16(01):5–23, 2001.

[39] Hindriyanto Dwi Purnomo, Hui-Ming Wee, and Hsin Rau. Two-sided assembly lines balancing with assignment restrictions. *Mathematical and Computer Modelling*, 57(1):189–199, 2013.

[40] Brahim Rekiek, Pierre De Lit, and Alain Delchambre. Hybrid assembly line design and user's preferences. *International Journal of Production Research*, 40(5):1095–1111, 2002.
[41] Brahim Rekiek, Pierre De Lit, Fabrice Pellichero, Thomas L'eglise, Patrick Fouda, Emanuel Falkenauer, and Alain Delchambre. A multiple objective grouping genetic algorithm for assembly line design. *Journal of intelligent manufacturing*, 12(5-6):467–485, 2001.
[42] ILOG S.A. *ILOG OPL 12.6 Language Manual*. IBM, France, 2013a.
[43] ILOG S.A. *CP Optimizer 12.6 User's Manual*. IBM, France, 2013b.
[44] M.E. Salveson. The assembly line balancing problem. *Journal of Industrial Engineering*, 6:18–25, 1955.
[45] Tadeusz Sawik. Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line. *European Journal of Operational Research*, 143(1):115–124, 2002.
[46] Pierre Schaus, Yves Deville, et al. A global constraint for bin-packing with precedences: Application to the assembly line balancing problem. In *AAAI*, pages 369–374, 2008.
[47] Armin Scholl. *Balancing and sequencing of assembly lines*. Physica-Verlag Heidelberg, 1999.
[48] Armin Scholl, Malte Fliedner, and Nils Boysen. Absalom: Balancing assembly lines with assignment restrictions. *European Journal of Operational Research*, 200(3):688–701, 2010.
[49] Mohamed Siala, Emmanuel Hebrard, and Marie-José Huguet. A study of constraint programming heuristics for the car-sequencing problem. *Engineering Applications of Artificial Intelligence*, 38:34–44, 2015.
[50] Christine Solnon. *Ant colony optimization and constraint programming*. John Wiley & Sons, 2013.
[51] El-Ghazali Talbi. *Metaheuristics: from design to implementation*. John Wiley & Sons, 2009.
[52] Daria Terekhov, Mustafa K Doğru, Ulaş Özen, and J Christopher Beck. Solving two-machine assembly scheduling problems with inventory constraints. *Computers & Industrial Engineering*, 63(1):120–134, 2012.
[53] Seyda Topaloglu, Latif Salum, and Aliye Ayca Supciller. Rule-based modeling and constraint programming based solution of the assembly line balancing problem. *Expert Systems with Applications*, 39(3):3484–3493, 2012.
[54] Mariem Trojet, Fehmi H'Mida, and Pierre Lopez. Project scheduling under resource constraints: Application of the cumulative global constraint in a decision support framework. *Computers & Industrial Engineering*, 61(2):357–363, 2011.
[55] Gonca Tuncel and Seyda Topaloglu. Assembly line balancing with positional constraints, task assignment restrictions and station paralleling: A case in an electronics company. *Computers & Industrial Engineering*, 64(2):602–609, 2013.
[56] H Fatih Uğurdağ, Ram Rachamadugu, and Christos A Papachristou. Designing paced assembly lines with fixed number of stations. *European Journal of Operational Research*, 102(3):488–501, 1997.
[57] Pedro M Vilarinho and Ana Sofia 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.
[58] Pedro M Vilarinho and Ana Sofia Simaria. Antbal: an ant colony optimization algorithm for balancing mixed-model assembly lines with parallel workstations. *International journal of production research*, 44(2):291–303, 2006.
[59] Wilbert E Wilhelm and Radu Gadidov. A branch-and-cut approach for a generic multiple-product, assembly-system design problem. *INFORMS Journal on Computing*, 16(1):39–55, 2004.
[60] LJ Zeballos. A constraint programming approach to tool allocation and production scheduling in flexible manufacturing systems. *Robotics and Computer-Integrated Manufacturing*, 26(6):725–743, 2010.
[61] Qiaoxian Zheng, Ming Li, Yuanxiang Li, and Qiuhua Tang. Station ant colony optimization for the type 2 assembly line balancing problem. *The International Journal of Advanced Manufacturing Technology*, 66(9-12):1859–1870, 2013.

==========Reviewers' comments==========

Reviewer #1: Compared to previously submitted version, the paper has been improved, in particular the description of the CP model is now more clear.

But I think the main issues I highlighted in my original review are still not completely fixed.

1/ Experimental study

First, the conclusions from the experimental study that compares the MIP and the CP model are weak, mostly because the results are based on a relatively small number of instances given that the behavior of both models are in fact quite close to each other (no model clearly dominates the other). Even if the problem instances on are not on the same objective function, I think they could be easily re-used and adapted (not necessarily to compare with their results) so that the experiments are performed on a larger set of instances and up to larger problem sizes (there are problems up to 1000 tasks there).

While we agree with the above comment, we would like state that our aim in this paper is two-fold: First, solve the AR-ALBP efficiently, and secondly observe the effect of additional constraints on the problem solution and computational time. Hence, we constructed the experimental design to survey the restrictions as well as their types. We also would like to note that the instances are as large as 148 tasks and 27 stations (Scholl, 1999). These instances are noted as medium to large problems in the literature. We think that the complexity is abundant.

Some statements are presented as the result of the experiments whereas they can be proved without doing any experiment, for instance on p18, "[for Case 3] it can be conjectured that the line efficiency will decrease and cycle time will increase [compared to Case 1 and 2]". Well this is not a conjecture, this is clearly a property given that Case 3 is more constrained than both Case 1 and Case 2.

We totally agree with the above comment. However, since the scientific proof is not provided (or sought in that perspective), and they are the result of numerical experiments, we have just noted this observation as conjecture. The sentence has been modified as a statement now.

The comparison based on the number of constraints in both models is not convincing at all ("However, CP models have more number of constraints than MIP models which is an advantage for CP to converge quickly"). First, the number of constraints is not that different between the two models (usually less than a factor 2) and, most importantly, what matters is not the number of constraints but the nature of these constraints and how they are capable of restricting the variables domain and thus, reduce the search space. For instance I suggest below some alternative formulation of constraints in the MIP model, that would largely inflate the number of constraints in the MIP model (and maybe improve its performance) to a point that exceeds the number of constraints in the CP model and in this case your argument would not hold anymore.

The statement is just an observation from Table 12. We agree with the above contention. In future studies, research on different formulations of both MIP and CP can be conducted.

In general the arguments for explaining why in some cases the CP model outperforms the MIP one are not enough justified. I know it is very difficult to explain the different performance of two completely different formulations and resolution approaches. I think that each element of explanation should be carefully assessed based on a large number of instances, otherwise it is better not to mention it. I think it is better to say that one cannot really explain why a given approach is better than another one (given the context, this is acceptable), rather than giving vague and ungrounded explanations.

Actually, our claim in the study is neither approach is superior, but rather they are equivalent. We argue that while MIP solutions are widely applied to AR-ALBP, CP solution methods provide

comparable results and can be used as an alternative efficient solution method.

Compared to the original submission, some explanation is provided on how the additional restrictions (in Table 3 and 4) have been selected. But there is something I do not really understand. You say that the restrictions are added in such a way that the optimal solution to the baseline problem (without restriction) satisfies them so that it is still an optimal solution for the more restricted problems. But then how do you explain that the optimal cycle times are not the same on Tables 5,6 and 7. Shouldn't they all be equal to the optimal cycle time of the baseline problem? Also, is it not the case that there is some bias with this way to generate restriction? The generated instances will all be such that if they do not increase the optimal value of the objective. Is it realistic? Also, the number of restrictions added to the different instances on Tables 3 and 4 seem rather small compared to the instance size.

We think that there is a misdepiction about the addition of restrictions. We added additional restrictions in a way that it does not change the known optimal solution, deliberately. For example, -in an extreme illustration- adding some restrictions may violate the precedence relations and result in infeasible solution. Also, one can purposely add restrictions to change the optimal. We added restrictions in this perspective. However, it may have changed the optimal solution unintentionally (This is expected and led to property/conjecture above, but not done calculatedly).

2/ Description of how CP works

The concern I expressed on the description of how CP in general (and CP Optimizer in particular) works has not been really addressed in this new version. In section 3, p7, you should clearly separate the general notions used in CP (problem definition, constraint propagation, tree search) from the description of how they are exploited in a particular solver (CP Optimizer). In particular:

The description of the general notions used in CP is difficult to understand. For instance:

- Title of section 3.1 (Constraint programming methodology): I don't think CP is a methodology. It is more an "approach", or a "technique". The methodology of CP is quite similar to the one of MIP for instance.

Related title has been changed as “approach”.

- "Furthermore, CP solvers have two fundamentals concepts in general: search tree, propagation and domain reduction.". When reading, I parse 3 concepts in this sentence (search tree + propagation + domain reduction)

The statement has been changed. These two fundamentals concepts are search tree and propagation-domain reduction.

- "An algorithm which solves CSP consists of two main situations: variable selection and value selection, when all the constraints are satisfied.". I don't understand.

The statement has been explained clearly.

- "Hence, we obtain a consistent assignment at every iteration of the backtracking". I don't understand. When you backtrack because the current partial assignment is detected inconsistent by constraint propagation you do not have a consistent assignment ...

The statement has been changed as “… at every iteration of the domain reduction”.

- Algorithm 1 is really difficult to understand and misleading:

Algorithm 1 is updated according to Pinedo (2012).

* "Generate the set of constraints": what does it mean? In general CP search does not generate any constraint after a decision has been taken to assign a value to a variable ...

The statement is corrected.

* "If there is not task for assignent then backtrack": what does it mean ? That there is no more tasks for assignment because all tasks have been assigned and we have found a feasible solution? That some variables have an empty domain (but in this case the backtrack occurred before, during constraint propagation)?

The statement is removed and the related algorithm step is corrected.

* Step 3 (Feasibility): "If any feasible assignment does not satisfy al least one of the constraints then backtrack". I don't think CP solvers work this way. It is in propagation that infeasibility is detected. There is no Step 3.

The statement is corrected and the related algorithm step is corrected.

* "Adjust variables and objective value." You never mentioned objective functions and COP before. I think that you should directly present COP instead of CSP in Definition 1.

The statement is corrected and the statement "objective value" is removed.

- "CP supports only discrete variables, and continuous variables are not supported": Even if CP Optimizer does not support them, some CP framework support continuous variables!

The statement is corrected.

- "Although CP generally needs more memory usage [than MIP] since each variable is defined as a domain". This is generally wrong.

The statement is changed as following. Even if CP is generally needs more usage, it can be say that this situation is not valid anymore because the CP solvers have improved significantly in the last few years.

The description of the CP Optimizer specific features does not bring much as it is described in the paper.

- "Many search strategies can be used to guide the backtracking for obtaining an assignment: depth first, multi-point, restart and automatic. In this study, since we concentrated on the modeling of AR-ALBP and achieving a solution, automatic search strategy is preferred as default setting in OPL to improve the search performance and to guide the current solution towards the optimal solution". This paragraph is very strange in the middle of the description of the general principles of CP as (1) this is very CP Optimizer centric, and (2) you even do not mentioned before what OPL is. I would clearly separate (maybe in different subsections) the general principles of CP and the specifics of CP Optimizer.

This statement is removed and added to the numerical results section since this search strategy setting is inherently software tuning.

- After the description of the CP Optimizer model in the end of section 4 you write "The above procedure can generate all possible task assignments for relevant data set and compute the objective functions for a given number of stations.". First, I would not call the approach a "procedure" as CP Optimizer if a model&run approach, so you only need to give the declarative formulation of your model, there is not

any procedure to run. And if you mean the procedure described in Algorithm 1, it is very far from the actual automatic search of CP Optimizer. Furthermore the search will of course not generate *all* possible task assignments but only (hopefully efficiently) explore the search space to find a good or optimal assignment.

The statement "procedure" is changed as "model". We proposed a CP model.

- "Overall, modeling the problem as a constraint programming is simpler due to the practical language of CP script". I would not call the CP language a script as a script suggest a procedural language whereas the language of the CP model (like MIP) is purely declarative.

The statement is corrected as following. "Overall, modelling the problem as a constraint programming is simpler due to the flexibility of CP solver formulation (Bukchin and Raviv, 2017)."

As a final remark on the presentation of CP, I would suggest to make section 3.1 much less detailed. You could focus on the differences between MIP and CP from a modeling point of view (not restricted to linear constraints, availability of global constraints like the pack constraint you use) and explain in a paragraph the main principles of the search. The main contribution of the article is not the presentation of how CP works. And this would leave more place for the presentation of the models and for the experimental section.

We respect the Referee's proficiency in the subject. But we feel the explanations are the fundamentals for the description of CP. For the general audience and for the terms we denoted in the manuscript, these depictions are the least explanation.

3/ Other comments

In the MIP model, for precedence constraints (Eq. 14) and for linked tasks restrictions (Eq 17), did you try some disaggregated formulations? These formulations are known to be stronger in several contexts (see the "disaggregated time-indexed formulation" for RCPSP). These formulations would boil down to something like:

- For precedences: sum_{t=1}{t=k} (x_ik - x_jk) >= 0 for (i,j) in P_r, k in 1..|W|

- For linked tasks restrictions: x_ik = x_jk for (i,j) in LT, k in 1..|W|

Of course, it is possible that different CP or MIP formulation for ALBP can be developed. Furthermore in the literature, various modelling approach is proposed for ALBP. Considering model in this study is generally accepted model by many researchers (Scholl, 1999). We think that your modelling suggestion can be a future research.

In the MIP model, how exactly is constraint (18) linearized?

We use "abs" function in ILOG to linearize this constraint and this situation is explained in the manuscript.

In the CP model in section 4, the solution example "For example an assignment A depended on this CSP ..." does not bring anything as the corresponding problem instance is not given.

We removed the example.

In the models (both MIP and CP), the formatting of the text could be made clearer because the first-line indents of the paragraphs (Objective function, Decision variable, Precedence relations, ...) makes it

hard to differentiate the different items.

This situation results from the journal manuscript style and also we use Latex class belonging to the journal to prepare the manuscript.

Also, in the models it would be easier to use a convention to distinguish variables from constants. For instance lower cases for variables, upper-cases for constants. Here we have variables "StationNumber" and "c".

StationNumber is changed as stationNumber by using lower cases. So that all variables are defined as lower cases.

References:

Armin Scholl. Balancing and sequencing of assembly lines. Physica-Verlag Heidelberg, 1999.

Michael L Pinedo. Scheduling: theory, algorithms, and systems. Springer, 2012.

Yossi Bukchin and Tal Raviv. Constraint programming for solving various assembly line balancing problems. Omega, 2017.