Constraints
DOI 10.1007/s10601-013-9142-6

APPLICATION

Balancing and scheduling of flexible mixed model assembly lines

Cemalettin Öztürk · Semra Tunalı · Brahim Hnich · M. Arslan Örnek

Abstract Mixed model assembly line literature involves two problems: balancing and model sequencing. The general tendency in current studies is to deal with these problems in different time frames. However, in today's competitive market, the mixed model assembly line balancing problem has been turned into an operational problem. In this paper, we propose mixed integer programming (MIP) and constraint programming (CP) models which consider both balancing and model sequencing within the same formulation along with the optimal schedule of tasks at a station. Furthermore, we also compare the proposed exact models with decomposition schemes developed for solving different instances of varying sizes. This is the first paper in the literature which takes into account the network type precedence diagrams and limited buffer capacities between stations. Besides, it is the first study that CP method is applied to balancing and scheduling of mixed model assembly lines. Our empirical study shows that the CP approach outperforms the MIP approach as well as the decomposition schemes.

Keywords Mixed model assembly lines · Balancing · Sequencing · Scheduling · Mixed integer programming · Constraint programming · Decomposition

C. Öztürk · M. A. Örnek
Department of Industrial Systems Engineering, İzmir University of Economics, İzmir, Turkey

S. Tunalı
Department of Business Administration, İzmir University of Economics, İzmir, Turkey

B. Hnich (✉)
Department of Computer Engineering, İzmir University of Economics, İzmir, Turkey
e-mail: hnich.brahim@gmail.com

Published online: 19 February 2013

1 Introduction

Assembly lines are flow-line production systems and consist of serially connected stations where single or mixed model products are manufactured in large amounts. A material handling system like conveyor belt is used to transfer the products from upstream to the downstream stations.

In general, managing a mixed model assembly line involves two dimensions; assignment of tasks to stations and the determination of the production order of models at each station. These two problems have been tackled so far in a sequential manner under the assumption of stable customer demands. Once the tasks are assigned to stations, it remains in effect for the upcoming mid-term (i.e., following weeks or months) and the model sequence only is revised just due to fluctuations changes in short term customer demands. The manufacturers nowadays are forced to respond very quickly to changes in the market conditions. Thus, adopting flexible production systems may provide many advantages for better competition. Hence, there is a tendency in assembly line literature to simultaneously consider both problems in the same time frame by exploiting the advantages of flexible production systems.

In this study, first, a mixed integer programming (MIP) model is developed for a flexible mixed model assembly line environment which considers task assignment and model sequencing problems within the same formulation along with task scheduling at stations. Because of the weakness of MIP approach for solving large scale problems, next, we formulate the same problem as a constraint programming (CP) model which is another exact method with well known success in combinatorial optimization problems by utilizing power of artificial intelligence methods. Moreover, on a wide range of problems including small, medium and large size with varying difficulty, we compare the performances of the proposed MIP and CP models to those found by using new and existing decomposition schemes.

The contributions of this study may be summarized as follows:

1. Unlike the current literature that considers chain-type precedence relations and assumes unlimited buffer capacities between the stations, we consider general type precedence relations and assume limited buffer capacities between the stations while exploiting the permutation schedule nature of the problem (i.e. processing of products at each station in the same order).
2. Besides extending the problem to a more realistic one, we propose a MIP model, a constraint programming (CP) model and also new decomposition schemes to solve this extended problem.
3. We provide an extensive survey of the current literature on simultaneous balancing and scheduling of flexible mixed model assembly lines.

The rest of the paper is structured as follows. In Section 2, we define the problem with an illustrative example. Section 3, summarizes the related literature about simultaneous approaches on balancing and scheduling of assembly lines. The MIP model, the CP model and the new decomposition schemes proposed to solve the SBSFMMAL problem are presented in Sections 4, 5 and 6, respectively. Robustness of the developed models and decomposition schemes are discussed in Section 7. The results of our experimental studies comparing the performance of the proposed models and decomposition schemes are given in Section 8. Finally, the concluding remarks and future research directions are provided in Section 9.

2 Problem definition

We define the problem environment with an illustrative example. Figure 1 shows the layout of a serial assembly line with three stations connected via a conveyor belt.

Each station on the assembly line is capable of performing certain assembly tasks. Each assembly task must be assigned to at least one station where alternative assignments are possible. Hence, there may be more than one station which can perform the same task. This property allows the assembly line to be flexible, which is a desired feature that results in the reduction of production (or cycle) time by increasing the number of eligible stations to perform any assembly task. Furthermore, each station on the assembly line has a limited working space area (as shown in Table 1) and each assembly task uses a portion of this available working space. Table 2 lists the working space requirements for ten assembly tasks on three stations shown in Fig. 1. Note that the entry such as "–" in Table 2 indicates that station cannot perform a given task.

Each product requires a subset of the assembly tasks. These products are also referred to as mixed models which are products that generally have similar physical properties, e.g., TV sets of the same brand with different options. For instance, given in Table 3 are 5 products to be manufactured on the assembly line of Fig. 1 that require a subset of these 10 tasks. For example, product 1 requires tasks 1, 2, 3, 4, 6, and 8 to be produced whereas product 4 requires tasks 1, 3, 5, 6, 7, 8, 9, and 10 instead. These two products may be two TV sets with different options.

Throughout this study, we refer to each task t of any product p as a job $< t,p >$ or simply job j. These jobs are classical operations in assembly line literature. Since we also focus on the scheduling of assembly operations of mixed models in stations, we introduce the following job scheduling constraints.

- Each product involves different number of jobs. For instance, product 1 involves 6 jobs whereas product 4 involves 8 jobs.
- Each job must be performed on exactly one station.
- The jobs of any product are not allowed to revisit earlier stations since the workflow is unidirectional. In other words, backtracking is disallowed.
- Jobs are transferred from one station to another following the precedence relations given in Table 4. The precedence graphs can be any directed acyclic graph and not restricted to chains as in Sawik [52, 53].

Note that the illustrative example shown here is mainly based on Sawik [52, 53] with a modified precedence diagram removing the restriction to chains.

In addition, the processing of any product in any given station cannot finish unless all jobs of the product currently being processed at that station are completed. Since we assume limited buffers between stations, a product cannot leave its current station

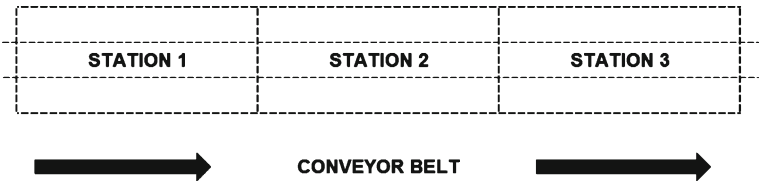

Fig. 1 Example of an assembly line with three stations

Table 1 Total available working space of each station m (in m^2) (b_m)

Stations (m)	1	2	3
b_m	8	10	9

Table 2 Working space required (a_{mt}) for each task t on each station m (in m^2)

Stations (m)			
Tasks (t)	1	2	3
1	1	–	1
2	2	–	2
3	3	–	3
4	1	1	–
5	2	2	–
6	3	3	–
7	–	1	1
8	–	2	2
9	–	3	3
10	–	5	5

Table 3 Product-task requirement matrix

Products (p)					
Tasks (t)	1	2	3	4	5
1	.	.		.	.
2	.	.	.		
3	.		.	.	.
4	.	.	.		
5		.	.	.	.
6	.	.		.	.
7		.	.	.	.
8	.		.	.	.
9		.	.	.	.
10		.	.	.	.

Table 4 Job precedence relations

Products (p)	Precedence relations
1	< 1, 1 >→<2,1>; <1,1>→<3,1>; <2,1>→<4,1>; <3,1>→<6,1>; <4,1>→<8,1>; and <6,1>→<8,1>
2	< 1, 2 >→<2,2>; <1,2>→<4,2>; <2,2>→<5,2>; <4,2>→<6,2>; <5,2>→<7,2>; <6,2>→<9,2>; <7,2>→<9,2>; and <9,2>→<10,2>;
3	< 2, 3 >→<3,3>; <2,3>→<4,3>; <3,3>→<5,3>; <4,3>→<7,3>; <5,3>→<8,3>; <7,3>→<8,3>; <8,3>→<9,3>; and <9,3>→<10,3>;
4	< 1, 4 >→<3,4>; <1,4>→<5,4>; <3,4>→<6,4>; <5,4>→<7,4>; <6,4>→<8,4>; <7,4>→<8,4>; <8,4>→<9,4>; and <9,4>→<10,4>;
5	< 1, 5 >→<3,5>; <1,5>→<5,5>; <3,5>→<6,5>; <5,5>→<7,5>; <6,5>→<8,5>; <7,5>→<8,5>; <8,5>→<9,5>; and <9,5>→<10,5>;

Table 5 Assembly time on station m for each task t in minutes

Tasks (t)	Stations (m)		
	1	2	3
1	4	–	4
2	2	–	2
3	2	–	2
4	2	2	–
5	4	4	–
6	2	2	–
7	–	3	3
8	–	5	5
9	–	2	2
10	–	4	4

unless the next station becomes available and it is ready to accept a new product. In other words, blocking of upstream stations is possible. As stated in Pinedo [48], any assembly line with positive (but finite) intermediate storages between stages can be modelled as an assembly line with zero intermediate storage. This follows from the fact that a storage space capable of containing one product may be regarded as a station at which the processing times of all jobs are equal to zero. Because of limiting the number of products in a station at any given time to at most one, the maximum number of products on the line is equal to the number of stations. Because of limited buffers, products visit each station in the same order, i.e., product permutation scheduling. Products move between the stations via an asynchronous conveyor belt. Once a product is transferred to a station, it can be picked by the operator in that station or the assembly operations can be performed on the conveyor belt. Upon the completion of the jobs at that station, the product is loaded to conveyor belt (if it was unloaded) and is transferred to the next station. Since the conveyor belt is assumed to be asynchronous, and mixed models generally have similar physical properties, the unloading/loading times of products from/to conveyor belt and transfer times between stations are assumed to be negligible. Finally, each job has a processing time and an earliest completion time as shown in Tables 5 and 6 respectively for our illustrative example. Note that the processing time of a job on a particular station is the same as the assembly time of its corresponding task on the same station. As in previous tables, the entry "–" in Table 5 denote incapability of a station in processing any job requiring the corresponding task. Furthermore, it is also possible that a certain task will have different assembly times on different stations (it is not the case in our illustrative example though).

Table 6 Earliest completion time of each job j in minutes (e_j) (see Section 4 for calculations)

Job (j)	e_j	Job (j)	e_j	Job (j)	e_j	Job (j)	e_j	Job (j)	e_j
1,1	4	4,2	6	4,3	4	5,4	8	5,5	8
2,1	6	5,2	10	5,3	8	6,4	8	6,5	8
3,1	6	6,2	8	7,3	7	7,4	11	7,5	11
4,1	8	7,2	13	8,3	13	8,4	16	8,5	16
6,1	8	9,2	15	9,3	15	9,4	18	9,5	18
8,1	13	10,2	19	10,3	19	10,4	22	10,5	22
1,2	4	2,3	2	1,4	4	1,5	4		
2,2	6	3,3	4	3,4	6	3,5	6		

The decisions to be made to solve this problem are threefold: (1) Job assignment and scheduling: assign each job of every product to exactly one station and schedule it, (2) Task assignment: assign each task to at least one station, (3) Product permutation scheduling: find a permutation of products and schedule each product on every station. The following constraints are taken into consideration in making these decisions:

- Each station can perform at most one task at any given time;
- The total tooling space required for the tasks assigned to each station must not exceed the station's finite work space available;
- If a job that involves task t is assigned to a station m, then task t must be assigned to station m;
- The processing of a product in a given station cannot finish unless all jobs of that product currently being processed at that station are completed;
- Since there are limited buffers between stations, a product cannot leave its current station unless the next station becomes available hence blocking is possible;
- It is possible for a product to visit a station without having any assigned jobs in that station. In that case, that station is used as a temporary buffer by that product to await for the next station. But of course, that station becomes unavailable for other products during this awaiting period;
- The revisiting of stations (or backward movement of the conveyor belt) is not allowed;
- The objective function is to minimize the latest completion time of the last job in the assembly line.

The optimal schedule to our illustrative example is shown in Fig. 2 with 53 minutes of latest completion time. As seen in Fig. 2, this assembly line is flexible as it allows the processing of the same task at different stations. For instance, while task 5 is processed at station 1 for products 3, 4 and 5 (see the jobs <5,3>, <5,4> and <5.5> in Fig. 2), the same task is processed at station 2 for product 2 (see the job <5,2> in Fig. 2). The permutation schedule suggests processing the products in the sequence of 2, 3, 4, 5, and 1 at each station. Also note that, although the jobs of product 1 are completed at the first station in 42 minutes, the product 1 waits at the first station for availability of the second station until the 44th minute. In this case station 1 is used as a buffer for product 1 between 42.nd and 44.th minutes. Note that if a station is used only as a buffer, it is introduced as a regular station which is not eligible to process any job.

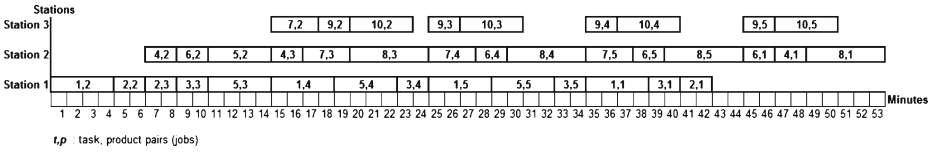

Fig. 2 Optimal schedule for the illustrative example

3 Related works

Balancing and scheduling are among the most important short-term planning issues in mixed model assembly lines (MMAL). While the balancing problem deals with how to allocate assembly tasks for a mix of products among the assembly stations with limited work space and/or limited time so as to balance the station workloads, the scheduling problem involves determining the detailed sequencing and timing of all assembly tasks and products or models at each station, so as to maximize the performance of the line.

In almost all studies on MMAL, balancing and scheduling problems have been considered separately (see [10–12] for comprehensive surveys for balancing and scheduling of MMAL literature). The common justification for this decomposition is the different time frames of balancing and scheduling problems [3]. The balancing decision has a typical planning horizon of several months, and naturally the frequently changing model mix is not known within this time frame. Hence, Scholl [55] and Merengo et al. [40] suggest anticipating the sequencing decisions at the higher planning level as a part of a hierarchical planning approach. However, various uncertainties often make readjustments to the predetermined line balance inevitable. For instance, unforeseeable changes in demand or the availability of new technologies might continuously necessitate a rebalancing of the line [11]. These issues become even more important in the design and scheduling of mixed model assembly lines which involve different models that require different tasks and the same task may have different processing times for different models. Whenever the demand structure shifts to another model-mix, it is required to reconsider the balancing and sequencing decisions. Karabat*ı* and Sayin [30] show that the total assembly time in each station (or station loads) is a function of model sequence for mixed model assembly lines and hence, the best task assignments (i.e, line balance) can be obtained by incorporating model sequence decisions into the line balancing problem. However, Karabat*ı* and Say*ı*n [30] find the allocation of tasks to stations which is the best compatible with given model sequence. On the other hand, authors note that the best approach to the problem is considering balancing and scheduling problems together. That is why some authors have proposed a simultaneous consideration of these two problems within the same time frame [29, 32–34, 43, 52–54].

It should be noted that even if the line is a single model assembly line, how to sequence the tasks within a station so that workstation time is reduced is an important issue in designing assembly lines [2, 5, 39, 45, 46, 56, 60]. The common approach to this issue is to deal with task sequencing problem after the line balancing problem is solved. The task sequencing problem at each workstation can be solved as a Traveling Salesman Problem (TSP). But solving the problem in two separate steps may yield only suboptimal solutions. Hence, an optimal task assignment plan should also consider the sequencing of tasks simultaneously. When mixed models are in case, it is possible to reduce the problem into a single model one (i.e., the assembly time of each task is equal to the weighted sum of the assembly times over the models in the minimum part set) and applying the same procedure as explained. However, Karabat*ı* and Sayin (30) show that this simplification -which is widely used in MMAL literature- provides only suboptimal solutions for the original problem. Therefore, the best schedule of the

operations of mixed models can be obtained with considering model operations as distinct jobs.

Table 7 chronologically lists the current literature on simultaneous consideration of balancing sequencing of assembly lines. As shown in Table 7, while the most widely studied area is balancing and sequencing of mixed model assembly lines with straight line layout, the task sequencing problem in mixed model assembly lines has received less attention.

The further insight gained as a result of surveying the current relevant literature can be summarized as follows:

- Flexible mixed model assembly lines that allow the assignment of tasks to more than single dedicated station simultaneously are rarely studied in the literature.
- Previous works on the task sequencing problem have concentrated on single model lines.
- Only one study [2] has been noted to study task sequencing with comb shape precedence graph in mixed model assembly lines.

Hence, to fill the perceived research gap in these areas, as indicated in the last line of Table 7, this study focuses particularly on the design and scheduling of flexible mixed model assembly lines with the following characteristics:

- Flexible and straight serial lines;
- Mixed models;
- General precedence diagrams;
- Limited buffers;
- Simultaneous consideration of the balancing, model sequencing, and task scheduling problems.

Simultaneous approach to assembly line balancing and scheduling problems brings new computational challenges, especially as the problem instance grows. Besides, using decomposition schemes may result in suboptimal solutions. Therefore, there is a need for a scalable approach to seek optimality. As an exact method, constraint programming (CP) has shown its success in combinatorial optimization problems including scheduling problems [7, 8]. However, there are only few studies which apply CP to assembly line planning issues.

Bockmayr and Pisaruk [9] make the first attempt on solving simple assembly line balancing problem (SALBP) with a CP approach. The authors present a combined integer and CP model for single model lines. In a recent study, Pastor et al. [47] carry out an extensive computational experiment to compare the performance of CP and integer programming formulations to solve SALBP. Results of those experiments show that the CP model formulation performs better and solves problems faster than the MIP formulation even for large size instances. Valle et al. [59] present the application of CP to the problem of selecting and sequencing of assembly tasks so as to minimize assembly time (makespan) of a single product. They use the CP approach to model And/Or alternative assembly plans for a single product and show the applicability of the method to real-life problems. In the case of mixed models, model sequencing becomes a critical issue. The car sequencing problem which is a sub problem of mixed model sequencing problem has been studied using CP approach in the literature [15]. However, the car sequencing problem does not require the assignment of tasks to stations. As a result, to the best of our knowledge, there

Table 7 Overview of current literature

Paper	Line layout			Model		Precedence diagram		Problem type			Solution methodology		
	Straight	U-shape	Flexible line	Single	Mixed	Special	General	Balancing	Sequencing		Exact	Heuristic	Meta heuristic
									Models	Tasks			
Agnetis and Arbib [2]	•			•		•		•		•	•		
Wilhelm [60]	•			•			•	•		•	•		
Kim et al. [32]	•				•		•	•	•				•
Kim et al. [33]		•			•		•	•	•				•
Sawik [52]	•		•		•	•		•	•		•		
Sawik [53]	•		•		•	•		•	•		•		
Miltenburg [43]		•			•		•	•	•				•
Sawik [54]	•		•		•	•		•	•		•		
Özdemir et al. [46]	•			•			•	•		•		•	
Kim et al. [34]		•			•		•	•	•				•
Kara [29]		•			•		•	•	•				•
Scholl et al. [56]	•			•			•	•		•	•		
Andres et al. [5]	•			•			•	•		•			•
Özcan and Toklu [45]	•			•			•	•		•		•	
Martino and Pastor [39]	•			•			•	•		•		•	
Scholl et al. [57]	•			•			•	•		•		•	
Our paper	•		•		•		•	•	•	•	•	•	

is no study in the literature which that simultaneously considers balancing, model sequencing and task scheduling using a CP approach.

4 MIP model development: SBSFMMAL-MIP

In this section, we propose a MIP formulation for our problem. Recall that our problem consists of three sub problems; (1) the assignment and scheduling of each job of every product to exactly one station (job assignment and scheduling); (2) the assignment of each task to at least one station (task assignment); and (3) scheduling of each product at every station (product permutation scheduling).

In what follows, a MIP model is proposed to tackle each sub problem separately and then a complete MIP model is proposed by tying these three models through channeling constraints. The overall objective of the combined model is to minimize the latest completion time of jobs.

Before we embark on modeling our problem we introduce the following:

Sets and indices:

- $i, m \in Stations$,$i, m \in Stations = \{1, ..., |Stations|\}$;
- t: Assembly tasks, $t \in Tasks = \{1, ..., |Tasks|\}$,
- p, q, v: Products (models), $p, q, v \in Products = \{1, ..., |Products|\}$,
- j, r: Designed (*task, product*) pairs or jobs indicate which product requires which task, $j, r \in Jobs \subseteq Tasks$ x $Products$ where $j.task$,$r.task$ and $j.product$,$r.product$ refer to the corresponding task and product of job j and r respectively,
- $Precedence$: The set of immediate predecessor-successor pairs of jobs (j, j') indicates that job j must be performed before job j', $(j, j') \in$ Precedence $\subseteq Jobs$ X $Jobs$,
- $Pred_j$: The set of all predecessors of job j,
- $Stations_j$: The set of stations capable of performing job j,
- $Stations_t$: The set of stations capable of performing task t. Note that although $Stations_j$ and $Stations_t$ can be thought as the same sets, in some circumstances, a job may not be assigned to a station while its referring task can be assigned and performed for other products.
- $Tasks_m$: The set of assignable tasks to station m.

Parameters:

- a_{mt}: Working space requirement of task t on station m, in m^2
- b_m: Total working space of station m, in m^2
- d_{mj}: Assembly processing duration for job j on station m,in minutes
- e_j: Earliest completion time for job j which is calculated iteratively as follows, $e_j = \max_{r \in Pred_j} \{e_r\} + \min_{m \in Stations_j} \{d_{mj}\}$
- $M = \sum_{j \in Jobs} \max_{m \in Stations_j} \{d_{mj}\}$

4.1 Job assignment and scheduling problem

To model the job assignment and scheduling problem, the following decision variables are introduced:

$$X_{mj} = \begin{cases} 1 \, if \;\; job \;\; j \;\; is \, assigned \, to \, station \, m \\ 0 \, otherwise \end{cases}$$

C_{mj}: Completion time of job j on station m

$$Z_{mjr} = \begin{cases} 1\, if\, on\, station\, m\ job\ \ j\, precedes\ job\ r \\ 0\, otherwise \end{cases}$$

C_{max}: Makespan of the schedule

The model of the job assignment and scheduling problem is as follows:

$$Minimize\, C_{\max} \tag{1}$$

subject to:

$$\sum_{m \in Stations_j} X_{mj} = 1\ \forall j \in Jobs \tag{2}$$

$$C_{mj} \geq e_j X_{mj}\ \forall j \in Jobs, \forall m \in Stations_j \tag{3}$$

$$C_{mj} \leq M X_{mj}\ \forall j \in Jobs, \forall m \in Stations_j \tag{4}$$

$$C_{mj} + d_{mr} X_{mr} \leq C_{mr} + M\left(1 - Z_{mjr}\right) \forall j, r \in Jobs, \forall m \in Stations_j \cap Stations_r | j.product < r.product \tag{5}$$

$$X_{mj} + X_{mr} - 2\left(Z_{mjr} + Z_{mrj}\right) \geq 0 \forall j, r \in Jobs, \forall m \in Stations_j \cap Stations_r | j.product < r.product \tag{6}$$

$$X_{mj} + X_{mr} \leq Z_{mjr} + Z_{mrj} + 1 \forall j, r \in Jobs, \forall m \in Stations_j \cap Stations_r | j.product < r.product \tag{7}$$

$$C_{mj} + d_{mr} X_{mr} \leq C_{mr} + M\left(1 - Z_{mjr}\right) \forall j, r \in Jobs, \forall m \in Stations_j \cap Stations_r | (j.product = r.product) \wedge (j.task \neq r.task) \tag{8}$$

$$X_{mj} + X_{mr} - 2\left(Z_{mjr} + Z_{mrj}\right) \geq 0 \forall j, r \in Jobs, \forall m \in Stations_j \cap Stations_r | (j.product = r.product) \wedge (j.task \neq r.task) \tag{9}$$

$$X_{mj} + X_{mr} \leq Z_{mjr} + Z_{mrj} + 1 \forall j, r \in Jobs, \forall m \in Stations_j \cap Stations_r | (j.product = r.product) \wedge (j.task \neq r.task) \tag{10}$$

$$C_{mj} \leq C_{\max}\ \forall j \in Jobs, \forall m \in Stations_j \tag{11}$$

$$C_{mj} \leq 0,\ X_{mj} \leq 0\ \forall j \in Jobs, \forall m \notin Stations_j \tag{12}$$

$$C_{mj'} \geq C_{ij} + d_{mj'} - M\left(1 - X_{mj'}\right) \forall \left(j, j'\right) \in Precedence, \forall i \in Stations_j, \forall m \in Stations_{j'} \tag{13}$$

$$\sum_{i \in Stations_j} iX_{ij} \leq \sum_{m \in Stations_{j'}} mX_{mj'}\ \forall \left(j, j'\right) \in Precedence \tag{14}$$

$$C_{mj}, C_{\max} \geq 0,\ X_{mj} \in \{0, 1\}\ \forall j \in Jobs, \forall m \in Stations_j \tag{15}$$

$$Z_{mjr} \in \{0, 1\}\ \forall j, r \in Jobs, \forall m \in Stations \tag{16}$$

Objective function (1) minimizes the latest completion time. Constraints (2) guarantee that each job is assigned to exactly one station. Constraints (3) give lower bounds of job completion times. Constraints (4) state if a job is not assigned to one of its eligible stations, then that job cannot be processed on that station. Constraints (5) are the disjunctive constraints for the jobs of the different products. Due to constraints (5) two distinct jobs of different products cannot be processed simultaneously at the same station. Constraints (6) and (7) enforce that the disjunctive variable Z_{mjr} or Z_{mrj}) takes value 1 if and only if both jobs s and r are assigned to the same station m. Constraints (5)–(7) are formulated for jobs of different products and this differentiation is ensured by quantifier $j.product < r.product$. Once jobs of different products are scheduled according to their corresponding product indexes in increasing order ($j.product < r.product$), since the schedule will be the same, formulating the same constraints for ($j.product < r.product$) will be redundant. As stated in problem definition, precedence diagram of a product may be general -or network- type, which implies the possibility of processing some jobs of a product having no precedence relations on the same station. Since stations are disjunctive resources, a disjunctive constraint must prevent simultaneous processing of such jobs on disjunctive stations. Hence, constraints (8), (9), and (10) extend the MIP model given in Sawik [53] to general precedence graphs. Note that constraints (8)–(10) are similar to (5)–(7), but they, instead, enforce disjunctive constraints among the jobs of the same product (i.e., corresponding product of distinct jobs j and r is the same, $j.product < r.product$). As an example, while constraints (5)–(7) handle the scheduling of jobs, <5,3> and <1,4> in the first station, constraints (8)–(10) deal with scheduling all jobs of product 4 in the first station (i.e., jobs <1,4>, <5,4>, <3,4>) as shown in the Fig. 2 for the illustrative example explained at Section 2. Constraints (11) give the makespan of the schedule. Constraints (12) tighten the bounds of the job completion (C's) and job assignment (X's) variables for non-eligible stations. Constraints (13) maintain the precedence restriction for each product among its jobs. Due to unidirectional flow, constraints (14) avoid the revisiting of a station for each product. Finally, constraints (15) and (16) give domains of variables.

4.2 Task assignment problem

Since the task assignment problem involves assigning each task to at least one eligible station, the following binary variables Y_{mt} are introduced.

$$Y_{mt} = \begin{cases} 1\, if\ task\ t\ is\ assigned\ to\ station\ m \\ 0\, otherwise \end{cases}$$

The task assignment problem is then modeled as follows:

$$\sum_{m \in Stations_t} Y_{mt} \geq 1\ \forall t \in Tasks \tag{17}$$

$$\sum_{t \in Tasks_m} a_{mt} Y_{mt} \leq b_m\ \forall m \in Stations \tag{18}$$

$$Y_{mt} = 0\ \forall t \in Tasks,\ \forall m \notin Stations_t \tag{19}$$

$$Y_{mt} \in \{0, 1\}\ \forall m \in Stations,\ \forall t \in Tasks \tag{20}$$

Constraints (17) ensure that each task is assigned to at least one station. Note that these constraints make assembly line flexible as they allow alternative assignments of tasks to stations. In other words, a task can be performed in different alternative stations for different products as explained in Section 2. Constraints (18) ensure that the working space capacity of each station is not exceeded. Constraints (19) forbid assignment of tasks to noneligible stations. Finally, constraints (20) give domains of the task assignment variables.

4.3 Product permutation scheduling problem

To model the product scheduling problem, we define the following decision variables.

A_{mp}: Arrival time of product p to station m,

D_{mp}: Departure time of product p from station m

$$U_{vp} = \left\{ \begin{array}{l} 1\, if\ product\ p\ is\ the\ v^{th}\ product\ processed \\ 0\, otherwise \end{array} \right\}$$

The product scheduling problem is formulated as follows:

$$\sum_{p \in Products} U_{vp} = 1\ \forall v \in Products \tag{21}$$

$$\sum_{v \in Products} U_{vp} = 1\ \forall p \in Products \tag{22}$$

$$D_{mp} - A_{mq} \leq M\left(2 - U_{v,p} - U_{v+1,q}\right) \forall p, q, v \in Products, \forall m \in Stations | p \neq q, v = 1 \tag{23}$$

$$D_{mp} - A_{mq} \leq M\left(2 - U_{v-1,p} - U_{v,q}\right) \forall p, q, v \in Products, \forall m \in Stations | p \neq q, v > 1 \quad (24)$$

$$A_{m+1,p} = D_{m,p} \ \forall p \in Products, \forall m \in Stations | m < |Stations| \quad (25)$$

$$A_{m,p} = D_{m-1,p} \ \forall p \in Products, \forall m \in Stations | m = |Stations| \quad (26)$$

$$U_{vp} \in \{0, 1\} \ \forall v, \ p \in Products \quad (27)$$

$$A_{mp}, \ D_{mp} \geq 0 \ \forall p \in Products, \forall m \in Stations \quad (28)$$

Constraints (21)–(24) express the permutation schedule constraints which imply that the sequence of products is the same for all stations. Constraints (21) and (22) ensure that each product is assigned to exactly one position in the sequence and vice versa. For any two adjacent products on any station, constraints (23) and (24) guarantee that the arrival time of the next product is greater than or equal to the departure time of the previous product. In other words, due to disjunctive nature of the stations, the departure and arrival of two products do not overlap. Constraints (25) and (26) ensure that the arrival time of a product to a station is equal to the departure time from the previous station due to disallowing unlimited intermediate buffers between stations. Note that, if a buffer of unlimited capacity is considered between the stations, then the sign "=" in constraints (25) and (26) is changed to "≥". Finally, constraints (27) and (28) give variable domains.

4.4 The complete MIP model

The channeling constraints combining the job assignment and scheduling and the task assignment problems are as follows:

$$X_{mj} \leq Y_{m,j.task} \ \forall j \in Jobs, \forall m \in Stations_j \quad (29)$$

$$Y_{mt} \leq \sum_{j \in Jobs | \, j.task = t} X_{mj} \ \forall m \in Stations \ \forall t \in Tasks \quad (30)$$

Constraints (29) ensure that jobs are assigned to the stations where the required tasks are performed and logically equivalent to: $X_j = m \Rightarrow Y_{m,j.task} = 1 \, \forall j \in Jobs, \forall m \in Stations_j$ Constraints (30) give a valid upper bound for task assignment variables and are formulated to reduce unnecessary alternative solutions. If (30) is not formulated, a task t would be assigned to a station although none of the jobs that include the task t were not assigned to that station. In other words, a task can be assigned to a station if and only if at least one of the jobs that require that task is assigned to that station.

The channeling constraints combining the job assignment and scheduling and the product permutation scheduling problems are as follows:

$$A_{m,j.product} \leq C_{mj} - d_{mj} X_{mj} + M\left(1 - X_{mj}\right) \ \forall j \in Jobs, \forall m \in Stations_j \quad (31)$$

$$D_{m,\,j.product} \geq C_{mj}\ \forall j \in Jobs, \forall m \in Stations_j \tag{32}$$

$$D_{mp} \geq A_{mp} + \sum_{j \in Jobs \mid j.product = p} d_{mj} X_{mj}\ \forall p \in Products, \forall m \in Stations \tag{33}$$

$$D_{mp} \leq C_{\max}\ \forall p \in Products, \forall m \in Stations \tag{34}$$

Since a product can not leave its current station before its jobs are completed on that station, the station arrival and departure times of products are made consistent with the corresponding job activities' starting and completion times through constraints (31) and (32). To start assembly jobs for any product in any station, that product must arrive to that station first, hence constraints (31) enforce that the arrival time of a product to a station is earlier than the starting times of all its jobs on that station. A product cannot leave any station before all jobs of the product are completed on that station, hence constraints (32) ensure that the departure time for each product from any station is greater than or equal to the completion time of all its jobs on that station. Recall that *j.product* refers to the corresponding product of job *j*. Constraints (33) channel product arrival and departure times by enforcing that departure time of any product in any station is later than arrival time of that product to that station and the total assembly time on that station. Constraints (33) are not formulated as equality since unlimited buffers are disallowed; a product may stay in its current station and may wait the availability of the next station. The makespan of the schedule is enforced to be greater than or equal to all departure times of products by constraints (34).

The complete model that involves the constraints of each sub-problem as well as all channeling constraints is shown in Fig. 3.

4.5 Novel features of the MIP model

The MIP model proposed here to solve the SBSFMMAL problem contributes to the current relevant literature in the following ways:

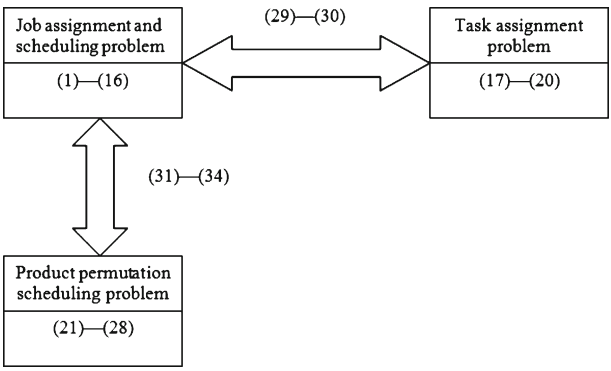

Fig. 3 The complete MIP model

1. In contrast to the literature, our model can be used for any type of precedence relations and it determines the schedule of assembly operations at each station due to constraints (5)–(10).
2. Unlike most of the existing literature that assume unlimited intermediate buffers between the stations, in this study, a buffer of limited capacity is considered between the stations. Note that blocking of stations by completed products never happens with the assumption of unlimited intermediate buffers. However, this is not realistic and/or common in industry. Depending on the type of material transfer systems used in the assembly line, the maximum number of products to store temporarily between the stations changes. If an accumulated conveyor is used, there can be products between the downstream and the upstream stations and the number of products between these stations depends on the length of the line. But if the conveyor is non-accumulated, any product awaits in its current station for the availability of the downstream station. If unlimited or considerably large buffer space is available in the facility, after completion of the all assembly processes, the products may be moved from the assembly line and the station becomes available for the product in the upstream station. When the downstream station becomes ready to accept the new products, products in the unlimited buffer can be loaded back to the transfer line. As explained in constraints (25) and (26), our model can be used tackle this problem both under the assumption of limited and also unlimited buffer capacities.
3. Sawik [53] formulates $|Products| \times (|Products| - 1) \times |Tasks|^2 \times |Stations|$ constraints in the worst case to handle the non-overlapping of products in each station and omits the permutation scheduling nature of the problem. In fact, product non-overlapping can be ensured with less number of constraints by exploiting the permutation scheduling constraints. Indeed, in our model, the total number of constraints (21)–(24), (31)–(33) that enforce the non-overlapping of products in each station is equal to $2 \times |Products| + |Stations| \times \left[|Products|^2 \times (|Products| - 1) + 2 \times |Jobs| + |Products|\right]$ in the worst case. For the illustrative example given in Section 2 where $|Products| = 5$, $|Stations| = 3$, $|Tasks| = 10$ and in the worst case, it can be assumed that all products require all tasks hence $|Jobs| = |Tasks| \times |Products| = 10 \times 5 = 50$,(i.e., regarding the illustrative example it is equal to 38.). Therefore, while the required number of constraints to model the product non-overlapping constraints is 6000 in Sawik [53], it is only 625 in our formulation. This result shows that our formulation is more compact and it may be used to solve larger size problem instances.

5 CP model formulation: SBSFMMAL-CP

Although mathematical programming techniques are successful to solve and find good solutions for some basic scheduling problems, an increase in the instance size and adding side constraints prevent using these techniques for larger instances [7]. CP, on the other hand, has been proven to be a powerful technique in effectively modeling and efficiently solving scheduling problems (see [31, 35, 62]) for recent applications of CP in scheduling domain). Existing tools such as CHIP [24] with its "cumulative constraint" [1] or ILOG Solver [27] with its scheduling extension ILOG Scheduler [27], embed several constraint propagation algorithms for non-preemptive

scheduling. For a more general comparison between MIP and CP, the reader may refer to Hentenryck [22], Brailsfard et al. [13], Puget and Lustig [50], Milano and Wallace [41], and Focacci et al. [17] for a detailed comparison.

CP models can be represented either using the constructs of the software used for coding [19, 28] or they can be formalized mathematically [7, 18, 42]. Due to clarity of the model, the latter approach has been adapted in this paper.

Before giving the details of the CP formulation, we introduce the global constraints that we use in our model. A global constraint is a constraint that states relations between an unfixed numbers of variables. Although it is possible to formulate the same relations with using traditional constraints, global constraints help in two ways: expressing the same relations between decision variables more declaratively and increasing the efficiency of the solution process by exploiting the structure of the problem to implement specialized filtering algorithms (e.g., [26, 58]).

We use the following global constraints in our CP formulation:

- *disjunctive*(α): All the activities of the collection α should not overlap with each other, [6].
- *element*(*I,Table,V*): V is equal to the Ith item of *Table* or in short we use the notation $\text{Table}_I = \text{V}$ [25].
- *alldifferent*(x_1,..., x_n): values assigned to the variables x_1,..., x_n must be pairwise distinct, [51].

In the following, each sub problem and channeling constraints are formulated separately as in the MIP formulation.

5.1 Job assignment and scheduling problem

The jobs are defined as activities and each activity, δ_j, $j \in Jobs$ is associated with three variables *start* (δ_j), *end* (δ_j) and *duration* (δ_j) ranging in $\{0, \ldots, M\}$. These three variables represent the start time, the end time and the duration of each activity δ_j, respectively. Each activity δ_j has to be processed on a station $m \in Stations_j$. Stations are disjunctive resources which can process at most one job at a time. It should be noted that since jobs are non-preemptive, *duration* (δ) is also equal to d_{mj} for assigned station $m \in Stations_j$.

In addition to the activity variables, two types of decision variables are introduced. The first set of variables is used to model the job station assignment. That is, for each job j, a variable X_j is defined whose domain is the set of stations capable of performing job j ($Stations_j$), i.e., $X_j = m$ if and only if job j is assigned to the station m. X_j variables are a reformulation of binary variables X_{mj} introduced in Section 4. The declarative efficiency of CP approach is obvious since it prevents to use binary decision variables and constraints (2) in MIP model. The second decision variable is the *makespan* of the schedule and it is defined as a non-preemptive activity with variables *start* (*makespan*) and *end* (*makespan*). Since *start*(*makespan*) = *end*(*makespan*), *duration* (*makespan*) is set to 0. The model of the job assignment and scheduling problem is as follows:

$$Minimize\ end\ (makespan) \tag{35}$$

subject to:

$$duration\ (\delta_j) = d_{X_j, j}\ \forall j \in Jobs \tag{36}$$

$$end\left(\delta_j\right) \geq e_j \; \forall j \in Jobs \tag{37}$$

$$disjunctive\left(\delta_j, \forall j \in Jobs \,\middle|\, X_j = m\right) \; \forall m \in Stations \tag{38}$$

$$end\left(\delta_j\right) \leq start\ (makespan) \; \forall j \in Jobs \tag{39}$$

$$end\left(\delta_j\right) \leq start\left(\delta_{j'}\right) \forall \left(j, j'\right) \in Precedence \tag{40}$$

$$X_j \leq X_{j'} \; \forall \left(j, j'\right) \in Precedence \tag{41}$$

The objective function (35) minimizes the makespan as (1) in the MIP model. Constraints (36) ensure that duration of each job is equal to the processing time of that job on its assigned station and logically equivalent to $X_j = m \Rightarrow duration\left(\delta_j\right) = d_{m,j} \forall j \in Jobs, \forall m \in Stations_j$. However, through the use of the global *element* constraint [25], the above constraints are expressed in CP by using variable indexing in a more compact way and achieving more effective propagation. Constraints (37) guarantee that each activity's end time is larger than its earliest completion time as (3) in the MIP model. Since stations of the assembly line are disjunctive resources, jobs assigned to the same station cannot be processed simultaneously (38). We can effectively and efficiently enforce these constraints by employing the *disjunctive* global constraint which employs the edge finding algorithm [6]. As Smith [58] mentions, using *disjunctive* global constraint avoids to formulate constraints (5)–(10) in the MIP model. Constraints (39)–(41) are CP formulation of the MIP model constraints (11), (13) and (14) in a more compact and efficient way. Constraints (39) enforce that each job is completed before the makespan activity. Constraints (40) enforce the precedence relations among the jobs of each product. Finally, due to the unidirectional flow (see Fig. 1), constraints (41) ensure that a product does not revisit any station by forcing to assign a successor job (j') to the same or a later station than its predecessor (j).

5.2 Task assignment problem

Constraints (17)–(20) formulated in the CP model are the same as in the MIP.

5.3 Product permutation scheduling problem

Since products are also associated with stations through their corresponding jobs on these stations, we introduce an activity for every product on each station. We declare a two dimensional array of activities for each product-station pairs as β_{pm},$\forall p \in Products, \forall m \in Stations$. $start(\beta_{pm})$, $end(\beta_{pm})$ and $duration(\beta_{pm})$ ranging in $\{0, \ldots, M\}$ to represent the start time, the end time and the duration of each product on each station, respectively. Defined as disjunctive resources, stations can process at most one product at any given time. In other words, products occupy the stations for two reasons, to await for the completion of the corresponding jobs on the same station

and/or to await for the availability of the next station. During this time, any other product cannot use the occupied station. Therefore, the occupation of stations by products is modeled with disjunctive global constraints.

The last set of variables formulates the product sequence. For each product p and for each position v, $U_v = p$ if and only if product p is the v^{th} product processed. These variables are reformulation of binary variables U_{vp} in the MIP model.

The product permutation scheduling model formulation is given below:

$$alldifferent\left(U_1, U_2, ..., U_{|Products|}\right) \tag{42}$$

$$end\left(\beta_{U_v,m}\right) \leq start\left(\beta_{U_{v+1},m}\right)\ \forall v \in Products, \forall m \in Stations | v = 1 \tag{43}$$

$$end\left(\beta_{U_{v-1},m}\right) \leq start\left(\beta_{U_v,m}\right)\ \forall v \in Products, \forall m \in Stations | v > 1 \tag{44}$$

$$end\ \left(\beta_{p,m}\right) = start\ \left(\beta_{p,m+1}\right)\ \forall p \in Products, \forall m \in Stations | m < |Stations| \tag{45}$$

$$end\ \left(\beta_{p,m-1}\right) = start\ \left(\beta_{p,m}\right)\ \forall p \in Products, \forall m \in Stations | m = |Stations| \tag{46}$$

$$disjunctive\left(\beta_{pm}, \forall p \in Products\right)\ \forall m \in Stations \tag{47}$$

Constraints (42)–(46) are the CP reformulation of the MIP constraints (21)–(26). The permutation schedule requires a unique position in the product sequence for each product on the assembly line. Hence, in constraints (42), we use the *alldifferent* global constraint [51] to effectively (with less number of constraints) and efficiently (faster than other consistency techniques) model the permutation of products on stations. Constraints (42) are the reformulation of the linear assignment type constraints (21)–(22) in the MIP model and enforce that products are assigned to different positions in the product sequence. For any two adjacent products in the product sequence (U_v, U_{v+1}) on any station, constraints (43) and (44) guarantee that the arrival time of the next product (U_{v+1}) is greater than or equal to the departure time of the previous product (U_v) and are equivalent to the MIP constraints (23) and (24). Note that we also employ the element global constraint for variable indexing in constraints (43) and (44). Due to the assumption of limited buffer space between the stations, constraints (45) and (46) ensure that each product awaits at the current station until the next station becomes available as in the MIP constraints (25) and (26). Finally, due to disjunctive nature of the stations, constraints (47) ensure that any two products cannot exist on the same station at the same time and a product is launched to a station after the previous one departs. Since our model has constraints (42)–(46) that are equivalent to the MIP constraints (21)–(26), constraints (47) can be considered as redundant. However, they help to reduce the search effort by exploiting the disjunctive nature of the problem.

5.4 The complete CP model

The channeling constraints between CP formulation of the job assignment and scheduling problem and the task assignment problem are as follows:

$$Y_{X_j,\,j.task} = 1 \;\forall j \in Jobs \tag{48}$$

$$Y_{mt} \leq \sum_{j\,in\,Jobs|\,j.task=t} (X_j = m)\;\forall m \in Stations,\; \forall t \in Tasks \tag{49}$$

Constraints (48) and (49) are equivalent to the MIP constraints (29) and (30), respectively. In constraints (48) we use the variable indexing feature of CP to express that jobs are assigned to the stations where the required tasks are performed. Constraints (49) can be expressed logically as: $X_j = m \Rightarrow Y_{m,\,j.task} = 1\;\forall j \in Jobs, \forall m \in Stations_j$.

The channeling constraints between the CP formulation of the job assignment and scheduling problem and the product permutation scheduling problem are as follows:

$$(X_j = m) \Rightarrow (start\,(\beta_{j.product,m}) \leq start\,(\delta_j))\;\forall j \in Jobs, \forall m \in Stations \tag{50}$$

$$(X_j = m) \Rightarrow (end\,(\beta_{j.product,m}) \geq end\,(\delta_j))\;\forall j \in Jobs, \forall m \in Stations \tag{51}$$

$$duration(\beta_{pm}) \geq \sum_{j\in Jobs|\,j.product=p} (X_j = m) \times duration(\delta_j)\;\forall p \in Products, \forall m \in Stations \tag{52}$$

$$end\;(\beta_{pm}) \leq start\;(makespan)\;\forall p \in Products, \forall m \in Stations \tag{53}$$

Constraints (50)–(53) are equivalent of the MIP constraints (31)–(34), respectively. The start and end times of product activities are made consistent with the start and end times of their corresponding job activities' start and end times in constraints (50) and (51) as (31) and (32) in the MIP model. Constraints (51) restrict the start times of the product activities and ensure that the product must launch to the station before its job activities are started. Constraints (52) ensure that on each station, ending time of a product activity is greater than or equal to the ending time of each corresponding job assigned to that station, if any. Otherwise, an upper bound for the completion time of the product activity is expressed in (53). Finally, the time spent by any product activity at any station includes the processing time and the waiting time for availability of the next station. Hence, the duration of each product activity is greater than or equal to the sum of the durations of the corresponding job activities on that station as expressed in constraints (52). The complete CP model is shown in Fig. 4.

6 Decomposition methods

Decomposition methods are widely used to solve large size instances of the SBSFM-MAL type MIP models in the literature [53] because of their more computational

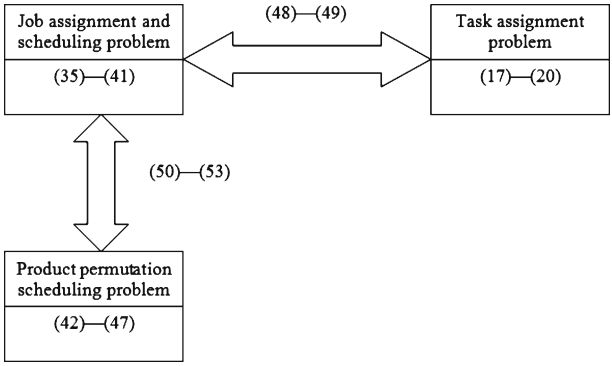

Fig. 4 The complete CP model

tractability. While decomposition methods are myopic approaches (i.e., does not seek optimality), simultaneous models aim to find optimal solutions. Therefore, we call simultaneous models and decomposition methods as exact and inexact methods respectively. Although decomposition approaches are capable of finding near optimal solutions in much shorter time than the exact MIP models [53] there is a need to test the performance of the proposed new exact models with inexact decomposition methods in terms of solution time and quality.

In this section, we propose decomposition schemes and use them to carry out comparative performance evaluation of proposed SBSFMMAL-MIP and SBSFMMAL -CP models.

Decomposition methods used in this paper are based on a combination of solving the three sub problems; assignment of tasks and jobs to stations (A), product permutation scheduling (P) and scheduling of jobs (J).

Classification of the decomposition schemes used in this paper is presented in Fig. 4. There are three main schemes:

- In the first scheme, each sub problem is solved independently. Sub problem A is solved using the MIP model and is denoted by A(MIP). Sub problem P is solved using a dispatching rules based method and is denoted by P(D). Finally, sub problem J is solved using either a CP or MIP model and is denoted by J(MOD). Thus, we denote the whole class as A(MIP)+P(D)+J(MOD). As dispatching rule based methods we use either the shortest processing time (SPT) method or the longest processing time (LPT). In this class we have four possible decomposition schemes. For instance, A(MIP)+P(SPT)+J(CP) denotes the decomposition scheme in which A is solved using MIP, P is solved suing the SPT dispatching rule and J is solved using CP.
- In the second scheme, A(MIP)+P&J(MOD), sub problems P and J and solved together in a single optimization model. Since P&J can either be solved using CP or MIP, we end up with two possible decomposition methods in this class.
- Finally, in the third decomposition scheme P(D)+A&J(MOD), sub problems A&J are instead combined in a single model. The resulting number of decomposition methods in this class is four. For instance, P(LPT)+A&J(MIP) solve P using the dispatching rule LPT and the sub problems A and J using MIP.

Note that only the reasoning of method A(MIP)+P&J(MOD) is taken from the literature. All other decomposition methods are developed during this study (Fig. 5).

6.1 Decomposition method A(MIP)+*P*(D)+J(MOD)

We start with the most naive and myopic decomposition method A(MIP)+P(D)+J(MOD). This method is a three phase decomposition scheme which separates the problem into three interconnected sub problems. First, a MIP model (A) assigns tasks and jobs to the stations so as to minimize the total workload on the bottleneck station. Next, priority based dispatched rules (D) are used to determine the permutation schedule of products (P). Finally in the third step, using predetermined assignment and permutation scheduling decisions as inputs, jobs are scheduled to find the best possible makespan under given conditions using an optimization model (J). In the following subsections, we give details of each step.

6.1.1 A(MIP) model

In the first step, the tasks and jobs are assigned to the stations so as to minimize the total workload (W) on the bottleneck station to achieve a balanced workload between stations. The assignment problem is formulated as a MIP model (see Appendix 1) due to its success in similar type problem domain.

6.1.2 Procedure P(D)

As stated in Pinedo [48], the dispatching rules (D) are useful when one attempts to find a reasonably good schedules with regard to a single objective such as the makespan. In this section, we use the permutation schedule nature of our model formulation and propose a dispatching rule based procedure. We employ well the

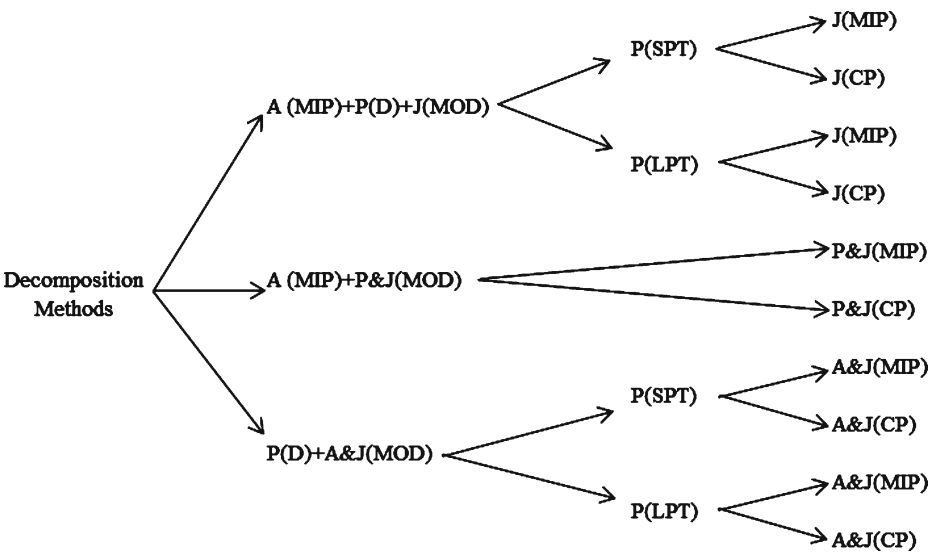

Fig. 5 Classification of decomposition methods used in this study

known Longest Processing Time first (LPT) and Shortest Processing Time first (SPT) dispatching rules in our procedure. Recall that we formulate the permutation schedule in constraints (21)–(22) in the MIP model and (42) in the CP model by using U_{vp} and U_v variables respectively. We expect to improve the solution effort by removing these variables and constraints from the model.

In classical flowshop scheduling problems where jobs are not related and alternative machines for jobs are not considered, it is easy to define the processing times of jobs and the ordering the jobs according to the their processing times. However, this reasoning cannot be implemented directly to our problem because: (1) the permutation schedule is defined for products; and (2) since which jobs are assigned to which station is not known before solving the problem, the processing times of the products are also not known in advance. Fortunately, However, the earliest completion times of products are known and they give good lower bounds globally for the processing time of products on the line. Earliest completion time of each product (EC_p) is equal to the completion time of its last job and formulized as $EC_p = \max_{\forall j \in Jobs | j.product = p} \{e_j\}, \forall p \in Products$.

In summary, since the permutation schedule enforces that the processing order of products are the same for each station in the line, we apply priority based scheduling rules. Furthermore, we assume that the processing time of each product is equal to its EC_p and we order the products according to SPT and LPT rules. Details of the procedure are explained are as follows.

Procedure P(D)

Step 1: $EC_p = \max_{\forall j \in Jobs | j.product = p} \{e_j\} \; \forall p \in Products$

Step 2: Sort the products according to their EC_pvalues in increasing order (SPT) or in decreasing order (LPT) and determine the permutation schedule as $U_v = v^{th}\, product\, in\, the\, sorted\, product\, list \; \forall v \in Products$.
Note that if the succeeding scheduling model is a MIP model the following additional step is used to convert U_v information in Step 2 to $U_{v,p}$

Step 3: $U_{v,p} = \left\{ \begin{array}{l} 1\, if\, U_v = p \\ 0\, otherwise \end{array} \right\}$

6.1.3 Model J(MOD)

In this section, we propose two alternative optimization models J(MIP) and J(CP) to schedule jobs with respect to task and job assignment decisions and product permutation scheduling decision predetermined in A(MIP) and P(D) steps.

J(MIP) model Using the total workload (W) as a lower bound on the makespan and considering job assignments $(X_{mj}, \forall m \in Stations, \forall j \in Jobs)$ and product permutation scheduling $(U_{vp}, \forall v \in Products, \forall p \in Products)$ decisions, the scheduling model determines the best schedule of jobs and products to minimize the makespan as formulated in Appendix 2.

Model J(CP) In an intermediate pre-processing phase, we translate the job assignment decisions $(X_{mj}, \forall m \in Stations, \forall j \in Jobs)$ in A(MIP) to $X_j, \forall j \in Jobs$ variables of the CP model by using logical expression, $X_{mj} = 1 \Rightarrow X_j = m \forall m \in Stations, \forall j \in Jobs$. Next, as in J(MIP), we use the total workload on the bottleneck station (W) as a lower bound on the makespan; the scheduling model determines the

best schedule of jobs and products to minimize the makespan. The scheduling model is presented in Appendix 3.

6.2 Decomposition method A(MIP)+P&J(MOD)

The main reasoning of this decomposition scheme is to combine P(D) and J(MOD) and to decide the schedule of the jobs and the products together via a single optimization model [53]. This approach is less myopic than method A(MIP)+P(D)+J(MOD) because it allows changing the product permutation schedule to find better job schedules but it also makes the problem more difficult to solve. While A(MIP) model presented in 6.1.1 is used to assign tasks and jobs to stations, MIP and CP alternatives of the P&J(MOD) are as follows.

6.2.1 Model P&J(MIP)

Model P&J(MIP) is formulated by adding product permutation scheduling constraints (21) and (22) to the model presented in J(MIP) with their domain definitions (27). The resulting P&J(MIP) model is given in Appendix 4.

6.2.2 Model P&J(CP)

As in MIP version, model P&J(CP) is formulated by adding the product permutation scheduling constraint (42) to the model presented in J(CP). The resulting P&J(MIP) model is presented in Appendix 5.

6.3 Decomposition method P(D)+A&J(MOD)

The last combination of the decomposition approach is P(D)+A&J(MOD) which solves the product permutation scheduling problem with the dispatching rule based methods SPT or LPT, then solves the assignment of tasks and jobs problem with the scheduling of jobs using a MIP or CP model. This approach is also less myopic than method A(MIP)+P(D)+J(MOD) because sub problem A&J(MOD) allows to change assignment of tasks and jobs to find better job schedules. But this flexibility also makes it computationally more costly.

Since procedure P(D) is the same as previously presented methods, we give details of the A&J(MOD) only.

6.3.1 Model A&J(MIP)

Model A&J(MIP) is formulated by removing the product permutation scheduling consraints (21), (22) and (27) from the SBSFMMAL-MIP model as shown in Appendix 6.

6.3.2 Model A&J(CP)

Model is A&J(CP) formulated by just removing the *alldifferent* global constraint from the SBSFMMAL-CP model as shown in Appendix 7.

7 Robustness of the exact models and decomposition methods

In practice, many unexpected circumstances may arise in manufacturing systems like machine failures, change in customer demand (i.e., product mix changes and/or changes in the product structure like new tasks). These uncertain situations cause deviations in the production schedules. Including estimation of uncertain events (i.e., pro-active) and/or repairing a disturbed schedule quickly with minimum changes (i.e., reactive) are two techniques to maintain stability in scheduling problems. Robustness of a schedule is measured with its ability to absorb uncertain and/or unpredictable events [49].

Davenport et al. [14] discuss slack-based pro-active methods which are based on providing extra time (i.e., slack) to each activity to absorb uncertainty and to achieve robust schedules. Authors show that the most robust schedules are obtained with temporal protection method [20] which suggests estimating the probability of a machine failure during execution of an activity and extending the actual activity duration as the expected machine downtime. The authors' simulation studies reveal that the temporal protection method results in more robust schedules than not taking uncertainty into account but produces worse quality solutions because of adding the expected machine stoppage time to each activity before the scheduling decision is made. Hence, they claim that the quality of a pro-active solution can be increased by including the decision on the amount of slack to the scheduling method. For this reason, Davenport et al. [14] propose two new methods; time window slack and focused time window slack. In the time window slack approach, based on the statistics on machine failures and downtimes, the minimum amount of slack is set for each activity. However, the position of the activities in the schedule may affect the possibility of having an uncertain event during the execution. For example, assume that the machine is recently fixed at the beginning of the schedule. In that case, if there were no machine failures until the last activity in the schedule, the probability of such a failure is more likely to occur during the execution of the last activity. Therefore, the focused time window method first calculates the probability that a breakdown event will occur at or before time t in the schedule. Then, it gives the lower bound for the amount of slack for each activity which executes at a particular time point $t$to absorb machine downtimes.

The pro-active methods mentioned above assume the availability of statistical information and predictability of uncertain events. However, there may be many unpredictable circumstances in the manufacturing system like adding an urgent order, cancelling some of the scheduled activities and/or changes in the product structure because of changes in customer requirements. In this case, the reactive scheduling methods [49] become important to maintain robustness.

Note that when the exact and decomposition methods proposed in this study are used in practical applications, they can be easily integrated with the pro-active methods for robust schedules. Once the statistics on machine failures (i.e., mean time between failures and mean downtime) are collected, new sets of constraints can be formulated into the model to apply proactive robust scheduling methods [14]. In addition, the models and decomposition methods proposed in this study enforces the robustness of the generated schedules against machine failures by allowing flexible task assignment. In case unpredictable events occur, these models and methods can quickly react and produce high quality schedules. As stated in

Boysen et al. [11], building resequencing buffers between stations can be considered to empower reactive schedules. Resequencing buffers can help to produce robust schedules by preventing stoppages/delays caused from the cancellation/addition of customer orders and/or the changes in the product structure.

8 Experiments

In this section, the performance of the proposed SBSFMMAL-MIP and SBSFMMAL-CP models are compared to those found by using the proposed decomposition over various test instances of different sizes.

The number of jobs in a scheduling problem is the most important factor which affects the size of the problem [48]. In our problem, as explained in Section 2, the jobs refer to (task, product) pairs. Since for any mixed model assembly lines $|Tasks| > |Products|$, the most important factor which determines the number of jobs and the tractability of the problem instance is the number of tasks. Hence, the number of products and stations are secondary important characteristics.

Therefore, we generate our test instances by using three factors with three levels; 10, 20 and 30 tasks, 5, 6 and 7 products and 3, 4 and 5 stations. Hence, the total number of test instances is $3 \times 3 \times 3 = 27$. To clarify the performance of each method when the problem size increases, we classify them as small (10 tasks), medium (20 tasks) and large (30 tasks). All test instances are available from http://homes.ieu.edu.tr/~cozturk/SBSFMMAL.rar.

The test instances are run on a personal computer with AMD Phenom II X4 955 3.21 GHz Processor, 4 GB RAM and Microsoft Windows 7 operating system. OPL Studio 3.7 (2003) which includes ILOG CPLEX 9.0, ILOG Solver 6.0 and ILOG Scheduler 6.0 libraries is used to model and solve the models and decomposition methods. The runtime of each method is limited to 3600 seconds.

As stated in Van Hentenryck et al. [23], one of the original and most important features of constraint programming is the ability to program search procedures. A well developed search procedure may dramatically affect the quality and performance of the solution process. Hence, we develop a search strategy for our CP models. Our search strategy proceeds by first generating a product sequence by branching on U variables and assigning products sequentially to the most constrained variable that has the smallest domain. This is basically known as a fail-first principle [21]. We then select a station for each job activity from their alternative resource pool and assign sequentially the earliest start times to these jobs. As we solve the permutation scheduling problem by using SPT and LPT dispatching rules, we omit the first step of the search strategy –generating values for U- in J(CP) and A&J(CP) models of the decomposition methods.

Table 8 presents the solutions found to each problem instance by using the proposed methods and decomposition schemes. The symbol "*" in this table is used to denote that the solution found is optimal, i.e., solver terminates before time limit is reached. Table 8 also shows the total number of jobs and the best method as bold for each instance.

Results of the experimental study are analyzed in Table 9. In this table "–" means that the average percentage gap with the best solution and solution time is not calculated since some of the instances cannot be solved with the corresponding method. S,

Table 8 Computational results for exact models and decomposition methods

Instance characteristics					Exact methods				Decomposition methods							
Instance size	|*Tasks*|	|*Products*|	|*Stations*|	|*Jobs*|	SBSFMMAL-MIP		SBSFMMAL-CP		A(MIP)+P(D)+J(MOD)							
									P(SPT)				P(LPT)			
									J(MIP)		J(CP)		J(MIP)		J(CP)	
					Makespan (min.)	Solution time (sec.)	Makespan	Solution time (sec.)	Makespan (min.)	Solution time (sec.)	Makespan (min.)	Solution time (sec.)	Makespan (min.)	Solution time (sec.)	Makespan (min.)	Solution time (sec.)
Small	10	5	3	38	**53***	3,88	**53***	0,67	62	0,64	62	0,67	57	0,78	57	0,98
	10	5	4	38	**44***	56,25	**44***	0,89	58	0,67	58	0,67	49	0,71	49	0,65
	10	5	5	38	**41***	374,03	**41***	1,81	52	0,74	52	0,73	44	0,74	44	0,71
	10	6	3	44	**62***	33,97	**62***	1,03	71	0,68	71	0,73	66	0,7	66	0,67
	10	6	4	44	**49***	194,47	**49***	3,92	66	0,71	66	0,68	53	0,73	53	0,68
	10	6	5	44	**47**	3600	**47***	4,97	69	0,92	69	0,89	61	0,95	61	0,85
	10	7	3	52	**72***	3443,39	**72***	3,13	81	0,7	81	0,7	76	0,71	76	0,67
	10	7	4	52	**56**	3600	**56***	19,78	74	0,76	74	0,68	61	0,77	61	0,75
	10	7	5	52	53	3600	**51***	71,77	66	1,06	66	0,98	56	1,06	56	0,99
Medium	20	5	3	76	**108***	1132,59	**108***	3,25	125	0,88	125	0,85	114	1,17	114	1,09
	20	5	4	76	**109***	1712,47	**109***	1,39	121	0,98	121	0,84	116	0,96	116	0,84
	20	5	5	76	93	3600	**92**	307,83	114	1,74	114	1,56	100	1,73	100	1,59
	20	6	3	88	123	3600	**119***	59,2	137	0,9	137	0,82	125	0,99	125	0,81
	20	6	4	88	**115**	3600	**115***	15,36	129	1,03	129	0,93	125	1,06	125	0,89
	20	6	5	88	99	3600	**97**	158,44	111	1,59	111	1,4	106	1,6	106	1,37
	20	7	3	104	144	3600	**143***	63,24	161	0,99	161	0,84	149	1,01	149	0,89
	20	7	4	104	139	3600	**135***	243,58	151	1,15	151	0,95	146	1,17	146	0,9
	20	7	5	104	143	3600	127	3382,45	152	5,07	152	4,83	134	5,08	134	4,83
Large	30	5	3	114	172	3600	**160***	1345,16	172	1,21	172	1,2	164	1,34	164	1,06
	30	5	4	114	N/A	3600	**147**	2161,2	158	8,7	158	8,48	157	8,73	157	8,47
	30	5	5	114	N/A	3600	**107**	2663	165	241,09	165	241,14	137	241,11	137	241,12
	30	6	3	132	196	3600	**176**	1298,33	192	1,37	192	1,07	186	1,46	186	1,13
	30	6	4	132	N/A	3600	157	345,88	175	24,33	175	24,05	175	24,35	175	24,03
	30	6	5	132	N/A	3600	133	1529,05	145	608,83	145	606,93	153	608,8	153	607,02
	30	7	3	156	N/A	3600	**201**	3042,5	219	1,6	219	1,2	213	1,59	213	1,2
	30	7	4	156	N/A	3600	**169**	3132,91	206	53,68	206	53,19	192	53,68	192	53,19
	30	7	5	156	N/A	3600	150	292,48	185	45,28	185	44,69	168	45,33	168	44,7

Table 8 (continued)

Instance characteristics					Decomposition methods											
Instance size	|*Tasks*|	|*Products*|	|*Stations*|	|*Jobs*|	A (MIP)+P & J (MOD)				P (D)+ A & J (MOD)							
					P & J(MIP)		P & J(CP)		P (SPT)				P (LPT)			
					Makespan (min.)	Solution time (sec.)	Makespan (min.)	Solution time (sec.)	A & J (MIP)		A & J (CP)		A & J(MIP)		A & J (CP)	
									Makespan (min.)	Solution time (sec.)	Makespan (min.)	Solution time (sec.)	Makespan (min.)	Solution time (sec.)	Makespan (min.)	Solution time (sec.)
Small	10	5	3	38	55	1,125	55	1,032	61	0,68	61	0,562	57	0,84	57	0,546
	10	5	4	38	47	1172	47	1,109	56	1,4	56	0,75	46	1,1	46	0,625
	10	5	5	38	42	1,421	42	1,171	50	5,66	50	0,703	44	10,39	44	0,766
	10	6	3	44	64	1,437	64	1,063	70	0,9	70	0,641	66	1,13	66	0,625
	10	6	4	44	51	1,64	51	1,218	64	1,77	64	0,578	51	1,38	51	0,563
	10	6	5	44	57	1,938	57	1,797	58	12,46	58	0,906	48	13,32	48	1,109
	10	7	3	52	74	5,609	74	1,61	80	1,06	80	0,672	76	1,18	76	0,704
	10	7	4	52	59	5,719	59	2,625	72	2,44	72	0,703	59	2,73	59	0,734
	10	7	5	52	52	6,422	52	2,062	64	29,09	64	0,797	55	32,76	55	0,844
Medium	20	5	3	76	114	1,343	114	1,361	120	8,93	120	0,75	**108**	21,49	**108**	0,75
	20	5	4	76	113	1,453	113	1,281	111	33,8	111	1,109	116	24,44	116	0,813
	20	5	5	76	97	2,173	97	2,031	**92**	481,92	**92**	3600	99	239,43	**99**	3600
	20	6	3	88	124	1,954	124	1,656	132	18,48	132	0,907	**119**	23,58	119	0,891
	20	6	4	88	121	2,046	121	1,75	121	83,89	121	24,625	125	47,33	125	1,187
	20	6	5	88	98	2,453	98	2,078	105	1302,25	105	3600	99	950,49	99	3600
	20	7	3	104	148	6,874	148	5	156	27,45	156	0,906	**143**	30,49	**143**	0,828
	20	7	4	104	142	7,297	142	4,438	142	155,45	142	202,172	146	169,38	146	1,813
	20	7	5	104	123	11,703	123	9,297	**110**	3600	134	3600	116	3600	119	3600
Large	30	5	3	114	162	1,797	162	1,516	165	159,3	165	39,766	163	684,58	163	47,75
	30	5	4	114	149	8,781	149	8,407	156	2349,86	156	178,625	N/A	3600	**147**	208,078
	30	5	5	114	119	223,875	119	223,812	N/A	3600	**107**	3600	N/A	3600	156	3600
	30	6	3	132	181	3,282	181	1,608	188	791,87	188	32,141	184	1425,75	184	24,937
	30	6	4	132	163	24,187	163	23,813	N/A	3600	162	667,125	N/A	3600	**152**	570,485
	30	6	5	132	140	571,91	140	567,781	N/A	3600	166	3600	N/A	3600	**121**	3600
	30	7	3	156	209	11,86	209	8,344	223	3600	212	42,109	N/A	3600	210	24,093
	30	7	4	156	181	60,219	181	58,672	N/A	3600	184	362,625	N/A	3600	172	149,532
	30	7	5	156	164	52,391	164	51,234	N/A	3600	**146**	587,89	N/A	3600	148	166,828

Table 9 Analysis of computational results

Instance characteristics	Exact methods						Decomposition methods											
	SBSFMMAL-MIP			SBSFMMAL-CP			A(MIP)+P(D)+J(MOD)											
							P(SPT)						P(LPT)					
							J (MIP)			J (CP)			J (MIP)			J (CP)		
	Instance size class			Instance size			Instance size			Instance size			Instance size			Instance size		
	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L
Number of solutions found	9	9	2	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
Number of optimal solutions	8	3	0	9	6	1	0	0	0	0	0	0	0	0	0	0	0	0
Number of optimality proven solutions	6	2	0	9	6	1	0	0	0	0	0	0	0	0	0	0	0	0
Number of solutions better than other methods	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
% gap with the best solution	0.44	4.446	–	0.00	1.72	1.77	27.30	17.22	19.05	27.30	17.22	19.05	10.55	8.65	13.25	10.55	8.65	13.25
General average	–			1.16			21.19			21.19			10.82					
Average solution time (sec.)	1656	3116	–	11.99	1333	3556	0.76	1.59	109.6	0.75	1.45	109.1	0.79	1.64	109.6	0.77	1.47	109.1
General average	–			1639.81			37.31			37.1			37.34			37.11		

Table 9 (continued)

Instance characteristics	Decomposition methods																	
	A (MIP)+P & J(MOD)						P (D)+ A & J(MOD)											
							P(SPT)						P(LPT)					
	P & J (MIP)			P & J (CP)			A & J (MIP)			A & J (CP)			A & J (MIP)			A & J (CP)		
	Instance size			Instance size			Instance size			Instance size			Instance size			Instance size		
	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L	S	M	L
Number of solutions found	9	9	9	9	9	9	9	9	4	9	9	9	9	9	2	9	9	9
Number of optimal solutions	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	3	0
Number of optimality proven solutions	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Number of solutions better than other methods	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	2
% gap with the best solution	5.75	5.07	7.00	5.75	5.07	7.00	21.82	5.73	–	21.82	8.16	8.24	5.65	4.27	–	5.65	4.57	6.65
General average	5.94			5.94			–			12.74			–			5.62		
Average solution time (sec.)	2.94	4.14	106.5	1.52	3.21	105	6.16	634.7	–	0.7	1226	1012	7.2	567.4	–	0.72	1201	932.4
General average	37.85			36.58			–			746.19			–			711.28		

M and L refer to small, medium and large size instance classes. Based on these results, we can state that although the SBSFMMAL-MIP model suggested in this study fully considers all the novel aspects that are pertinent to our problem, it does not seem to be computationally efficient to tackle with large-size real-world problems. Moreover, the proposed decomposition schemes are observed to be performing well with respect to solution time, but their solution quality is rather poor in comparison to the proposed SBSFMMAL-CP model in general. Among the MIP based decomposition methods the once using A&J(MIP) model are able to use only a small number of large size instances. However, having the smallest gap, P(LPT)+A&J(CP) is the best decomposition method studied. Results of the experiments also show that LPT based priority rules outperforms SPT one in terms of makespan values with slightly worse solution times. As a result, the proposed SBSFMMAL-CP model outperforms all other approaches over all instances from the small to large-size real-world problems with acceptable and usable solution times. Hence, we can state that the proposed SBSFMMAL-CP model is the best approach to simultaneously solve the flexible mixed model assembly line balancing and scheduling problem in our experiments. Furthermore, success of the SBSFMMAL-CP model in varying size of instances in terms of number of products, stations, tasks and product structure (i.e., number of precedence relations) shows its ability to produce robust schedules.

To generalize the results of this comparative experimental study so that the practitioners can be motivated to use these models to study the real-world problems, lastly a large-scale test instance involving 50 common tasks which are to be scheduled for 5 products on 20 stations is generated. In total, this large-scale test instance involves 190 jobs (task, product pairs) to be assigned and scheduled. At the end of the one hour run time only SBSFMMAL-CP is able to solve this large-scale instance.

9 Conclusions

Today's competitive market requires more flexible production systems that respond rapidly to changes in the market conditions. Therefore, balancing assembly lines, regarded as a tactical level problem, becomes an operational problem. As stated in Karabat*ı* and Say*ı*n [30] workload in a station is affected by the mixed model operations schedule. Hence, for an effective line management, task assignment and operations scheduling problems must be considered simultaneously. As mentioned in the survey of relevant literature, there are a scarce number of papers dealing with this topical problem.

In real-world problems, task sequencing at each station is usually dealt with only after the tasks have been assigned to that station and this problem is solved as a TSP problem in which the setup times correspond to distances between the cities. However, this approach is myopic since it could yield suboptimal solutions for the whole problem. Hence, new procedures and models are needed for simultaneously solving both the task assignment and scheduling problems.

In this paper, we suggest a novel MIP model which includes task assignment and model sequencing along with task scheduling within the same framework to solve balancing and scheduling of flexible mixed model assembly lines problem (SBSFMMAL-MIP). Unlike previous studies, our study considers general type precedence diagrams and limited buffer spaces between the stations. Furthermore, our model considers the permutation scheduling structure of the problem which

allows us to formulate more useful model with less number of constraints. Although all the aspects of the problem studied are fully included within a single MIP model, its applicability to large-scale real world problems is found to be computationally inefficient. Hence, to address this problem, we propose a CP model (SBSFMMAL-CP) in this paper which is shown to be both efficient (in terms of modeling with more compact and flexible formulation) and effective (in terms of solution effort) method for solving simultaneous balancing and scheduling of flexible mixed model assembly lines. Furthermore, to evaluate the performance of the proposed CP model, we compare it with various decomposition methods and also complete MIP model. According to the computational studies, SBSFMMAL-CP outperforms all other approaches over all size of test instances.

In a future research, the proposed approaches can be used to solve the SBSFMMAL problem with some extensions such as parallel stations and/or lines. Proposed methods in this paper assumes fixed number of stations and try to minimize the latest completion time of a given set of mixed models. These methods can be modified and used to determine the minimum number of stations which ensures a given cycle time. Besides minimizing the latest completion time, leveling stations loads or other objective functions could be considered [16, 61].

Although the CP model is observed to perform well for big instances, another future research issue could be to further improve its performance by developing symmetry breaking constraints [58] or by hybridizing with local and/or large neighborhood search algorithms [4, 36, 44]. Furthermore, facilities available in the new solvers like optional interval variables (see [37, 38]) can be tested for better quality and/or faster solutions. Lastly, the SBSFMMAL problem can be studied and analyzed under a cyclic scheduling policy [48].

Acknowledgements The authors thank the referees for their valuable and constructive comments which greatly improved organization of the paper.

Appendices

Appendix 1: A (MIP) model

$$Minimize\ W \tag{54}$$

$$subject\ to: \quad W \geq \sum_{j \in Jobs} d_{mj} X_{mj}\ \forall m \in Stations \tag{55}$$

$$W \geq 0 \quad \text{and } (2),(14),(15),(17),\ldots,(20),(29),(30) \tag{56}$$

The A(MIP) model minimizes the maximum workload in the line (54) which is greater than or equal to the total workload of each station (55).

Appendix 2: J(MIP) model

$$\begin{aligned} &\textit{Minimize } (1) \\ &\textit{subject to}: \\ &C_{\max} \geq W \\ &\text{and } (3),\ldots,(13),(15),(16),(23),\ldots,(26),(28),(31),\ldots,(34) \end{aligned} \tag{57}$$

(57) formulates the lower bound on the makespan.

Appendix 3: model J(CP)

$$\begin{aligned} &\textit{Minimize } (35) \\ &\textit{subject to}: \\ &\textit{end}\,(\textit{makespan}) \geq W \\ &\text{and } (36),\ldots,(40),(43),\ldots,(47),(50),\ldots,(53) \end{aligned} \tag{58}$$

(58) gives a lower bound on the completion time of the makespan activity as in (57).

Appendix 4: model P&J(MIP)

$$\begin{aligned} &\textit{Minimize } (1) \\ &\textit{subject to}: \\ &(3),\ldots,(13),(15),(16),(21),\ldots,(28),(31),\ldots,(34),(57) \end{aligned}$$

Appendix 5: model P&J(CP)

$$\begin{aligned} &\textit{Minimize } (35) \\ &\textit{subject to}: \\ &(36),\ldots,(40),(42),\ldots(47),(50),\ldots,(53),(58) \end{aligned}$$

Appendix 6: model A&J(MIP)

$$\begin{aligned} &\textit{Minimize } (1) \\ &\textit{subject to}: \\ &(2),\ldots,(20),(23),\ldots,(26),(28),\ldots,(34) \end{aligned}$$

Appendix 7: model A&J(CP)

Minimize (35)

subject to :

(36),. . . ,(41),(43),. . . ,(53)

References

1. Aggoun, A., & Beldiceanu, N. (1993). *Extending CHIP in order to solve complex scheduling and placement problems (Mathl. Comput. Modelling)* (vol. 17, no. 7, pp. 57–73). Pergamon Press Ltd.
2. Agnetis, A., & Arbib, C. (1997). Concurrent operations assignment and sequencing for particular assembly problems in flow lines. *Annals of Operation Research, 69*, 1–31.
3. Akgündüz, O.S., & Tunalı, S. (2009). An adaptive genetic algorithm approach for the mixed-model assembly line sequencing problem. *International Journal of Production Research, 48*(17), 5157–5179.
4. Appa, G., Mourtos, I., Magos, D. (2002). *Integrating constraint and integers programming for the orthogonal latin squares problem, principles and practice of constraint programming - CP 2002* (pp. 79–90). Springer.
5. Andres, C., Miralles, C., Pastor, R. (2008). Balancing and scheduling tasks in assembly lines with sequenc-dependent setup times. *European Journal of Operational Research, 187*, 1212–1223.
6. Baptiste, P., & Le Pape, C. (1996). Edge-finding constraint propagation algorithms for disjunctive and cumulative scheduling. In *Proceedings of the fifteenth workshop of the UK planning special interest group*. Liverpool.
7. Baptiste, P., Le Pape, C., Nuijten, W. (2001). *Constraint-Based Scheduling*. Kluwer Academic Publishers, Boston.
8. Bartak, R. (2003). *Constraint-based scheduling: An introduction for newcomers. Intelligent manufacturing systems 2003* (pp. 69–75).
9. Bockmayr, A., & Pisaruk, N. (2001). Solving an assembly line balancing problem by combining IP and CP. In *Proceedings of the 6th annual workshop of ERCIM working droup on constraints*. Prague, Czech Republic.
10. Boysen, N., Fliedner, M., Scholl, A. (2008). Assembly line balancing: Which model to use when. *International Journal of Production Economics, 111*, 509–528.
11. Boysen, N., Fliedner, M., Scholl, A. (2009). Sequencing mixed-model assembly lines: Survey, classification and model crituqe. *European Journal of Operational Research, 192*, 349–773.
12. Boysen, N., Fliedner, M., Scholl, A. (2009). Production planning of mixed-model assembly lines: overview and extensions. *Production Planning & Control, 20*(5), 455–471.
13. Brailsfard, S.C., Potts, C.N., Smith, B.M. (1999). Constraint satisfaction problems: algorithms and applications. *European Journal of Operational Research, 119*, 557–581.
14. Davenport, A., Gefflot, C., Beck, J. (2001). Slack-based techniques for robust schedules. In *Proceedings of 6th European conference on planning (ECP-01)*.
15. Dincbas, M., Simonis, H., van Hentenryck, P. (1988). Solving the car-sequencing problem in constraint logic programming. In *Proceedings of the European conference on artificial intelligence (ECAI-88)* (pp. 290–295). München.
16. Drexl A., & Kimms, A. (2001). Sequencing JIT mixed-model assembly lines under station-load and part-usage constraints. *Management Science, 47*(3), 480–491.
17. Focacci, F., Lodi, A., Milano, M. (2002). Mathematical programming techniques in constraint programming: A short overview. *Journal of Heuristics, 8*, 7–17.
18. Focacci, F., Laborie, P., Nuijten, W. (2000). Solving scheduling problems with setup times and alternative resources. *AIPS 2000 Proceedings* (pp. 92–101).
19. Fourer, R., & Gay, D.M. (2002). Extending an algebraic modeling language to support constraint programming. *INFORMS Journal on Computing, 14*(4), 322–344.
20. Gao, H. (1995). *Building robust schedules using temporal protection protection-an empirical study of constraint based scheduling under machine failure uncertainty*. Master's thesis, Department of Industrial Engineering, University of Toronto.

21. Haralick, R.M., & Elliott, G.L. (1980). Increasing tree search efficiency for constraint satisfaction problems, *Artificial Intelligence, 14*, 263–314.
22. Hentenryck, P. (2002). Constraint and integer programming in OPL. *INFORMS Journal on Computing, 14*(4), 345–372.
23. Hentenryck, P., Perron, L., Puget, J.F. (2000). Search and strategies in OPL. *ACM Transactions on Computational Logic, 1*(2), 285–320.
24. Hentenryck, P. (1989). *Constraint satisfaction in logic programming*. MIT Press, Cambridge.
25. Hentenryck, P., Carillon, J.P. (1988). *Generality versus specificity: An experience with AI and OR techniques (AAAI-88, 1988)* (pp. 660–664). Minnesota.
26. Hoeve, W., Katriel, I. (2006). Global Constraints. In Rossi, F., van Beek, P., Walsh, T. (Ed.), *Handbook of constraint programming* (pp. 169–208). Elsevier Science, Amsterdam.
27. ILOG (2003). *OPL Studio 3.7*. Language Manual.
28. Jain, V., & Grossmann, I.E. (2001). Algorithms for Hybrid MILP/CP Models for a Class of Optimization Problems. *INFORMS Journal on Computing, 13*(4), 258–276.
29. Kara, Y. (2008). Line balancing and model sequencing to reduce work overload in mixed-model U-line production environments. *Engineering Optimization, 40*(7), 669–684.
30. Karabatı, S., & Sayın, S. (2003). Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *European Journal of Operational Research, 149*, 417–429.
31. Khayat, G.E., Langevin, A., Riopel, D. (2006). Integrated production and material handling scheduling using mathematical programming and constraint programming. *European Journal of Operations Research, 175*, 1818–1832.
32. Kim, Y.K., Kim, J.Y., Kim, Y. (2000a). A coevolutionary algorithm for balancing and sequencing in mixed model assembly lines. *Applied Intelligence, 13*, 247–258.
33. Kim, Y.K., Kim, S.J., Kim, J.Y. (2000b). Balancing and sequencing mixed-model U-lines with a co-evolutionary algorithm. *Production Planning & Control, 11*, 754–764.
34. Kim, Y.K., Kim, J.Y., Kim, Y. (2006). An endosymbiotic evolutionary algorithm for the integration of balancing and sequencing in mixed-model U-lines. *European Journal of Operational Research, 168*, 838–852.
35. Krogt R., Geraghty J., Salman M.R., Little J. (2010). On supporting Lean methodologies using constraint-based scheduling. *Journal of Scheduling, 13*, 301–314.
36. Laborie, P., & Godard, D. (2007). Self-adapting large neighbourhood search: Application to single-mode scheduling problems. In *Proc. of the 3rd multidisciplinary international conference on scheduling: Theory and applications (MISTA)* (pp. 276–284).
37. Laborie, P., & Rogerie, J. (2008). Reasoning with conditional time-intervals. In *Proc. 21th international FLAIRS conference (FLAIRS 2008)* (pp. 555–560).
38. Laborie, P., Rogerie, J., Shaw, P., Vilím, P. (2009). Reasoning with conditional time-intervals, Part II: An algebraical model for resources. In *Proc. 22th international FLAIRS conference (FLAIRS 2009)* (pp. 201–206).
39. Martino, L. & Pastor, R. (2010). Heuristic procedures for solving the general assembly line balancing problem with setups. *International Journal of Production Research, 48*(6), 1787–1804.
40. Merengo, C., Nava, F., Pozzetti, A. (1999). Balancing and sequencing manual mixed-model assembly lines. *International Journal of Production Research, 37*(12), 2835–2860.
41. Milano, M., & Wallace, M. (2006). Integrating operations research in constraint programming. *OR, 4*, 175–219.
42. Milano, M., Ottosson, G., Refalo, P., Thorsteinsson, E.S. (2002). The role of integer programming techniques in constraint programming's global constraints. *INFORMS Journal on Computing, 14*(4), 387–402.
43. Miltenburg, J. (2002). Balancing and scheduling mixed-model U-shaped production lines. *International Journal of Flexible Manufacturing Systems, 14*, 119–151.
44. Ottosson, G., Thorsteinsson, E.S., Hooker, J.N. (2002). Mixed global constraints and inference in hybrid CLP-IP solvers. *Annals of Mathematics and Artificial Intelligence, 34*(4), 271–290.
45. Özcan, U., & Toklu, B. (2009). Balancing two-sided assembly lines with sequence-dependent setup times. *International Journal of Production Research, 48*(18), 5363–5383.
46. Özdemir, R.G., Ayağ, Z., Çakır, D. (2004). *Hazırlık sürelerinin azaltılması için bir hat dengeleme modeli (YA/EM 2004)*. Kocaeli.
47. Pastor, R., Ferrer, L., Garcia, A. (2007). Evaluating optimization models to solve SALBP. *Lecture Notes in Computer Science, 2007*(4705), 791–803.
48. Pinedo, M.L. (2008). *Scheduling theory, algorithms, and systems, 3rd edn*. Springer, New York.
49. Policella, N., Cesta, A., Oddi, A., Smith, S. (2004). Generating robust schedules through temporal flexibility. In *Proc. ICAPS 04*.

50. Puget, J.F., & Lustig, I. (2001). Constraint programming and maths programming. *Knowledge Engineering Review, 16*(1), 5–23.
51. Regin, J.C. (1994). *A filtering algorithm for constraints of difference in CSPs (AAAI-94)* (pp. 362–367). Washington.
52. Sawik, T. (2000). Simultaneous vs. sequential loading and scheduling of flexible assembly systems. *International Journal of Production Research, 38*, 3267–3282.
53. Sawik, T. (2002). Monolithic vs. hierarchical balancing and scheduling of a flexible assembly line. *European Journal of Operational Research, 143*, 115–124.
54. Sawik, T. (2004). Loading and Scheculing of a flexible assembly system by mixed integer programming. *European Journal of Operational Research, 154*, 1–19.
55. Scholl, A. (1999). *Balancing and sequencing assembly lines, 2nd edn*. Physica, Heidelberg.
56. Scholl, A., Boysen N., Fliedner M. (2008). The sequence-dependent assembly line balancing problem. *OR Spectrum, 30*(3), 579–609.
57. Scholl, A., Boysen, N., Fliedner, M. (2011). The assembly line balancing and scheduling problem with sequence-dependent setup times: problem extension, model formulation and efficient heuristics. *OR Spectrum*. doi:10.1007/s00291–011–0265–0.
58. Smith, B. (2006). Modelling. In Rossi, F., van Beek, P., Walsh, T. (Eds.), *Handbook of constraint programming* (pp. 377–406). Elsevier Science, Amsterdam.
59. Valle, C.D., Marquez, A.A., Gasca, R.M., Toro, M. (2003). On selecting and scheduling assembly plans using constraint programming. *Lecture notes in computer science, 2003*(2774), 1329–1336.
60. Wilhelm, W.E. (1999). A column-generation approach for the assembly system design problem with tool changes. *International Journal of Flexible Manufacturing Systems, 11*, 177–205.
61. Yavuz, M., & Akcali, E. (2007). Production smoothing in just-in-time manufacturing systems: a review of the models and solution approaches. *International Journal of Production Research, 45*(16), 3579–3597.
62. Zeballos, L.J., Quiroga, O.D., Henning, G.P. (2010). A constraint programming model for the scheduling of flexible manufacturing systems with machine and tool limitations. *Engineering Applications of Artificial Intelligence, 23*, 229–248.