7th IFAC Conference on Manufacturing Modelling, Management, and Control
International Federation of Automatic Control
June 19-21, 2013. Saint Petersburg, Russia

Cyclic Scheduling of Flexible Mixed Model Assembly Lines

C. Öztürk*. S. Tunalı**
B. Hnich*. M.A. Örnek******

*Department of Industrial Systems Engineering, İzmir University of Economics,
İzmir, Turkey, (e-mail: cemalettin.ozturk@ieu.edu.tr)
**Department of Business Administration, İzmir University of Economics,
İzmir, Turkey, (e-mail: semra.tunali@ieu.edu.tr)
***Department of Computer Engineering, İzmir University of Economics,
İzmir, Turkey, (e-mail: brahim.hnich@ieu.edu.tr)
****Department of Industrial Systems Engineering, İzmir University of Economics,
İzmir, Turkey, (e-mail: arslan.ornek@ieu.edu.tr)

Abstract: Mixed model assembly lines are used to produce multiple copies of given minimum part set (MPS). Considering only one copy (cycle) of the MPS while solving the mixed-model balancing and scheduling problem yields suboptimal solutions since blocking and idle times of stations between repeated copies are ignored. Modeling and solving this problem in a cyclic manner can significantly overcome these inefficiencies and improve the throughput of the line. In this paper, after investigating the nature of the problem, we propose exact and heuristic methods for practical applications and evaluate their performances on various size test instances.

Keywords: Mixed Model Assembly Lines, Balancing, Cyclic Scheduling, Flexible Manufacturing, Constraint Programming

1. INTRODUCTION

Assignment of tasks to the stations (i.e., balancing) and determining the best schedule of models and tasks at each station are the main decisions in management of mixed-model assembly lines. Simultaneous consideration of these problems along with employing flexible technologies improves responsiveness of the companies to the changes in the market conditions and in turn, they become more competitive in today's global manufacturing environment (Karabati and Sayın, 2003). The current assembly line balancing and sequencing literature (Boysen et al. 2009) neglects the issue of repeated copies (i.e., replications or cycles) of given Minimum Part Set (MPS, the collection of different models that are to be assembled together) and mainly focuses on minimizing the number of stations, cycle time or smoothing the workload for one cycle. However, idle and blocking times in each station between consecutive cycles have to be taken into account to maximize throughput rate and therefore, balancing and scheduling problems must be considered in a cyclic manner. Due to its potential to increase efficiency cyclic scheduling has started receiving attention of both practitioners and researchers in recent years (Sawik, 2011).

Current studies on cyclic scheduling of flow lines reveal that (1) steady state schedules (i.e., the same cycle time in each cycle) are achieved at the very first replication of the MPS and hence it is not needed to solve the cyclic scheduling problem for all repeating cycles (Karabatı and Kouvelis,1996), (2) all of the literature consider only the sequencing of models in the line and assume that task assignment problem is solved previously however, it is noted that the best line efficiency can be obtained by simultaneous consideration of balancing and scheduling problems (Karabatı and Sayın, 2003), (3) cyclic scheduling problems are hard combinatorial optimization problems and require intelligent methods to model and solve (Sawik, 2011).

In this paper, we consider both task assignment and model sequencing problems together for flexible mixed model assembly lines in a cyclic manner along with scheduling of assembly tasks within the same station. Flexibility is ensured by assuming that tasks can be performed by alternative stations (Öztürk et al., 2012). To the best of our knowledge, this is the first study that integrates all aspects of the problem. Because of the combinatorial nature of the problem we propose a Constraint Programming (CP) formulation, which is a widely used modeling and solution method for similar combinatorial optimization problems in the literature (e.g., Özturk et al. 2013). In addition, using instances given in the literature (Öztürk at al. 2013) we investigate how many cycles are required to repeat in order to achieve steady state cycle times. Furthermore, we propose a stabilization algorithm to determine the completion times of the remaining cycles to be executed after makespan is stabilized. The rest of the paper is structured as follows. In Section 2, we define the problem with an illustrative example. The proposed CP model to solve this problem is presented in Section 3. The proposed algorithm to extend the steady state cyclic schedules to the repeating cycles is given in Section 4. The results of experimental studies to test the performance of the proposed model on various test instances and investigation of minimum number of MPS replications to achieve steady state cycle times are given in Section 5. Finally, the concluding remarks and future research directions are given in Section 6.

2. PROBLEM DEFINITION

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. 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 results in the reduction of or cycle time by increasing the number of eligible stations to perform any assembly task as in surface mount technology lines (Öztürk et al., 2012, 2013). Furthermore, each station on the assembly line has a limited working space area and each assembly task uses a portion of this available working space. Each product requires a subset of the assembly tasks. These products are also referred to as mixed models which generally have similar physical properties, e.g., TV sets of the same brand with different options. Throughout this study, each task *t* of any product *p* is referred as a job <*t,p*> or simply job *j*. Different products may require a different number of jobs. These jobs are classical operations in assembly line literature. The jobs of a product are performed according to the precedence relations defined by the assembly plan for that product. The precedence graphs can be any directed acyclic (or network) graph and not restricted to chains. Note that in chain type precedence graphs, the order of processing the tasks of a product is technologically restricted. However, in network graphs, some tasks can be processed in any order which ensures flexibility of the line. But, this increases complexity of the problem (Öztürk et al., 2013). Each job must be performed on exactly one station. Each job has a processing time and an earliest completion time. Assembly time of a task may change depending on which station it is assigned to. Since the assembly line is a unidirectional workflow, the jobs of any product are not allowed to revisit earlier stations or stages. 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 unless the next station becomes available and it is ready to accept a new product. In other words, blocking of upstream stations is possible. Because of limited buffers, products visit each station in the same order, i.e., product permutation scheduling. We assume that the unloading/loading times of products from/to conveyor belt and transfer times between stations are assumed to be negligible. The illustrative problem involves serially connected 3 stations with 5 products, and 10 common assembly tasks. For example, while the product 1 entails the tasks 1, 2, 3, 4, 6, and 8, product 4 entails tasks 1, 3, 5, 6, 7, 8, 9, and 10. These two products may be two TV sets with different options. For product 1, while the first task must be processed before task 2, by allowing network type precedence relations, tasks 2 and 3 can be processed in any order. The total number of jobs to be assigned and scheduled is 38. As shown in Figures 1, 2 and 3, optimal schedule results in 47 minutes of stable cycle time (i.e., time between completion time of products in consecutive cycles). Note that in these three figures the third dimension in the tuples (<t,p> tuples) shows the cycle number that the job is repeated. Due to the limited space, the full details of the problem parameters are not presented (see Öztürk et al. 2013 for details). As seen in Figure 1, this assembly line is flexible as it allows the processing of the same task at different stations. For instance, while the task 5 is processed at station 1 for products 3, 4 and 5 (see the jobs "<5,3>,1"," <5,4>,1" and "<5,5>,1" in Figure 1), the same task is processed at station 2 for product 2 (see the job "<5,2>,1" in Figure 2). The permutation schedule is found as 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 in the first cycle, the product 1 awaits at the first station for availability of the second station until the 44th minute. In this case station 1 is used as a temporary storage space (buffer) for product 1 between 42.nd and 44.th minutes. Similarly, second replication of product 2 in the second station starts at time 53 (Figure 2) and hence, product 2 waits in the first station in between [50, 53] time interval (Figure 1). In other words, station 1remains blocked for three minutes (shown as ▨ in Figures 1, 2 and 3). However, idle times (shown as ▤) are observed in the third station (i.e., in time interval [23,24]) until the third product arrives at the station). Figures 1, 2 and 3 also indicate that the second station is the bottleneck of the assembly line.

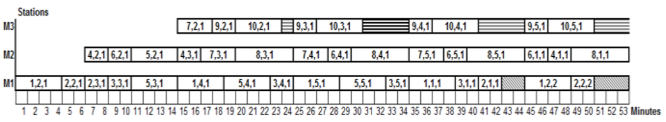

Fig. 1. Schedule of the first cycle.

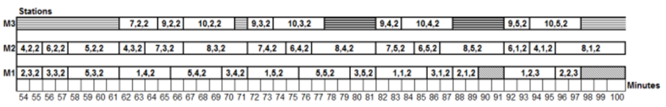

Fig. 2. Schedule of the second cycle.

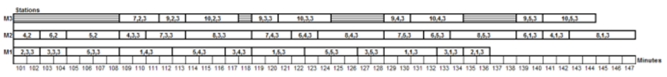

Fig. 3. Schedule of the third cycle.

Since the permutation schedule is 2, 3, 4, 5, and 1, the product 1 is always the last product in each cycle and defines the completion time of that replication (or cycle). According to Figures 1, 2 and 3, the first cycle is completed at 53th minute which is the completion time of the last operation for product 1 in station 2. Similarly, the product 1 is completed at 100th minute and at 147th minute in the second and the third cycles, respectively. Karabatı and Kouvelis (1996) define the cycle time as the time length between starting time of models in consecutive replication of MPS. Therefore, while cycle time of the first replication is 53 minutes, it stabilizes to 47 minutes in the second and the third replications, so the system reaches the steady state (Karabatı and Kouvelis, 1996) in the second replication. As can be seen in Figures 1, 2 and 3, all task assignments and permutation schedules are the same for the consecutive cycles, so consistency between the cycles is also satisfied.

3. CONSTRAINT PROGRAMMING MODEL

In this section we propose a CP formulation for cyclic scheduling of flexible mixed model assembly lines. This 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) permutation scheduling of products. Before we embark on modelling each sub problem we introduce the following:

Sets and indices:

c : *cycles*, $c \in Cycles = \{1,...,|Cycles|\}$,

$i, m \in Stations = \{1,...,|Stations|\}$;

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, $j,r \in Jobs \subseteq Tasks \text{ x } Products$ 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 $\subseteq Jobs \text{ X } Jobs$,

$Pred_j$: 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 we defined the eligible sets of stations for jobs and tasks separately just to emphasize that the capability of a station to perform a given task can change from product to product.

$Tasks_m$: 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, $e_j = \max_{r \in Pred_j} \{e_r\} + \min_{m \in Stations_j} \{d_{mj}\}$

$$M = \sum_{j \in Jobs} \max_{m \in Stations_j} \{d_{mj}\}$$

Global Constraints

We introduce the global constraints that we use in our CP formulation:

disjunctive(α) : All the activities of the collection *α* should not overlap,

element(I,Table,V) : V is equal to the Ith item of Table.

alldifferent(x_1, . . . , x_n) : values assigned to the variables x_1, . . . , x_n must be pairwise distinct.

3.1 Job Assignment and Scheduling Problem

Note that the *Job* which is earlier defined in section 2 is called activity (Öztürk et al., 2013) and each activity, $\delta_{j,c},\ j \in Jobs, c \in Cycles$ is associated with three variables *start*($\delta_{j,c}$), *end*($\delta_{j,c}$) and *duration*($\delta_{j,c}$) ranging in {0,…, |*Cycles*|*M*}. These three variables represent the start time, the end time and the duration of each activity $\delta_{j,c}$ in each cycle, respectively. Each activity $\delta_{j,c}$ has to be processed on a station $m \in Stations_j$. Stations are disjunctive resources which can process at most one job at any given time. It should be noted that since jobs are non-preemptive, $duration(\delta_{j,c})$ 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 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. Note that, since each job j is performed in the same station for all cycles, there is no need to add a cycle dimension to this variable. The second decision variable is the *makespan$_c$* of each cycle and is defined as a non-preemptive activity in which *duration*(*makespan$_c$*) is set to 0. The model of the job assignment and scheduling problem is as follows:

$$\text{Minimize } end(makespan_{|Cycles|}) \tag{1}$$

subject to:

$$duration(\delta_{j,c}) = p_{X_j,j} \quad \forall j \in Jobs, \forall c \in Cycles \tag{2}$$

$$end(\delta_{j,c}) \geq e_j \quad \forall j \in Jobs, \forall c \in Cycles \tag{3}$$

$$disjunctive\left(\delta_{j,c}, \forall j \in Jobs \,|\, X_j = m\right) \quad \forall m \in Stations \ \forall c \in Cycles \tag{4}$$

$$end(\delta_{j,c}) \leq start(makespan_c) \ \forall j \in Jobs, \forall c \in Cycles \tag{5}$$

$$end(\delta_{j,c}) \leq start(\delta_{j',c}) \quad \forall (j,j') \in Precedence \ \forall c \in Cycles \tag{6}$$

$$X_j \leq X_{j'} \quad \forall (j,j') \in Precedence \tag{7}$$

$$\left(end(\delta_{j,c}) \leq start(\delta_{r,c})\right) \Rightarrow \left(end(\delta_{j,c+1}) \leq start(\delta_{r,c+1})\right)$$
$$\forall j,r \in Jobs, \forall c \in Cycles$$
$$|(j.product = r.product) \wedge (j \neq r) \wedge (|Cycles| > 1) \wedge (c = 1) \tag{8}$$

$$\left(end(\delta_{j,c-1}) \leq start(\delta_{r,c-1})\right) \Rightarrow \left(end(\delta_{j,c}) \leq start(\delta_{r,c})\right)$$
$$\forall j,r \in Jobs, \forall c \in Cycles$$
$$|(j.product = r.product) \wedge (j \neq r) \wedge (|Cycles| > 1) \wedge (c > 1) \tag{9}$$

The objective function (1) minimizes the makespan of the last cycle which also ensures the minimum cycle time. Constraints (2) ensure that duration of each job is equal to the processing time of that job on its assigned station and is logically equivalent to $X_j = m \Rightarrow duration(\delta_{j,c}) = p_{m,j}$. However, through the use of the global *element* constraint, the above constraints are expressed in CP by using variable indexing in a more compact way and achieving more effective propagation. Constraints (3) guarantee that each activity's end time is larger than its earliest completion time for each cycle. Since stations of the assembly line are disjunctive resources, jobs assigned to the same station cannot be processed simultaneously in each cycle (4). We can effectively and efficiently enforce these constraints by employing the *disjunctive* global constraint which employs the edge finding algorithm. Constraints (5) enforce that in each cycle, each job is completed before the makespan activity of that cycle. Constraints (6) enforce the precedence relations among the jobs of each product for each cycle. Finally, since we assume unidirectional flow, constraints (7) avoid revisiting of an assembly station for each product by forcing to assign a successor job (*j'*) to the same or a later station than its predecessor (*j*). Constraints (8) and (9) ensure that jobs of products are processed in the same order in each cycle.

3.2 Task Assignment Problem

To model the task assignment problem, we introduce binary variables Y_{mt} where

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

To ensure consistency in each cycle, each task must be assigned and performed in the same station in consecutive cycles. Hence, there is no cycle dimension in the task assignment variables. Based on this principle, the problem is then modelled as follows:

$$\sum_{m \in Stations_t} Y_{mt} \geq 1 \qquad \forall t \in Tasks \qquad (10)$$

$$\sum_{t \in Tasks_m} a_{mt} Y_{mt} \leq b_m \qquad \forall m \in Stations \qquad (11)$$

$$Y_{mt} = 0 \qquad \forall t \in Tasks, \forall m \notin Stations_t \qquad (12)$$

Constraints (10) 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. Constraints (11) ensure that the working space capacity of each station is not exceeded. Constraints (12) forbid assignment of tasks to noneligible stations.

3.3 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 three dimensional array of activities for each product-station pairs in each cycle as β_{pmc}, $\forall p \in Products, \forall m \in Stations, \forall c \in Cycles$. *start*($\beta_{pmc}$), *end*($\beta_{pmc}$) and *duration*($\beta_{pmc}$) ranging in {0,…, |*Cyckes*|*M*} to represent the start time, the end time and the duration of each product on each station in each cycle, 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. Note that since the permutation schedule is the same for each cycle, there is no need to define these variables with a cycle dimension.

The product permutation scheduling model formulation is given below:

$$alldifferent\left(U_1, U_2, \ldots, U_{|Products|}\right) \qquad (13)$$

$$end\left(\beta_{U_v,m,c}\right) \leq start\left(\beta_{U_{v+1},m,c}\right)$$
$$\forall v \in Products, \forall m \in Stations, \forall c \in Cycles \mid v = 1 \qquad (14)$$

$$end\left(\beta_{U_{v-1},m,c}\right) \leq start\left(\beta_{U_v,m,c}\right)$$
$$\forall v \in Products, \forall m \in Stations, \forall c \in Cycles \mid v > 1 \qquad (15)$$

$$end(\beta_{p,m,c}) = start(\beta_{p,m+1,c})$$
$$\forall p \in Products, \forall m \in Stations, \forall c \in Cycles \mid m < |Stations| \qquad (16)$$

$$end(\beta_{p,m-1,c}) = start(\beta_{p,m,c})$$
$$\forall p \in Products, \forall m \in Stations, \forall c \in Cycles \mid m = |Stations| \qquad (17)$$

$$disjunctive\left(\beta_{pmc}, \forall p \in Products\right)$$
$$\forall m \in Stations, \forall c \in Cycles \qquad (18)$$

$$start\left(\beta_{U_1,m,c}\right) \geq start\left(\beta_{U_{|Products|},m,c-1}\right)$$
$$\forall m \in Stations, \forall c \in Cycles \mid \left(|Cycles| > 1\right) \wedge \left(c > 1\right) \qquad (19)$$

$$start\left(\beta_{U_1,m,c+1}\right) \geq start\left(\beta_{U_{|Products|},m,c}\right)$$
$$\forall m \in Stations, \forall c \in Cycles \mid \left(|Cycles| > 1\right) \wedge \left(c = 1\right) \qquad (20)$$

The permutation schedule requires a unique position in the product sequence for each product on the assembly line. Hence, in constraints (13), we use the *alldifferent* global constraint to effectively (with less number of constraints) and efficiently (faster than other consistency techniques) model the permutation of products on stations. Constraints (13) enforce that products are assigned to different positions for given product sequence. For any two adjacent products in the product sequence (U_v, U_{v+1}) on any station, constraints (14) and (15) 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) in each cycle. Note that we also employ the *element* global constraint for variable indexing in constraints (14) and (15). Due to the assumption of limited buffer space between the stations, constraints (16) and (17) ensure that each product awaits at the current station until the next station becomes available in each cycle. Due to disjunctive nature of the stations, constraints (18) 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 in each cycle. Since our model has constraints (14)—(15), constraints (18) can be considered as redundant. However, they help to reduce the search effort by exploiting the disjunctive nature of the problem. Finally, the start and end time of products in consecutive cycles are made consistent with constraints (19) and (20). Due to these constraints; the first product in the permutation schedule in each station has to wait the completion of the last product in the permutation schedule in the previous cycle on the same station.

3.4 The Complete CP Model

The channelling constraints between the job assignment and scheduling problem and the task assignment problem are as follows:

$$Y_{X_j, j.task} = 1 \qquad \forall j \in Jobs \qquad (21)$$

$$Y_{mt} \leq \sum_{j \ in \ Jobs | j.task = t} \left(X_j = m \right) \ \forall m \in Stations \ , \forall t \in Tasks \qquad (22)$$

Constraints (21) which use the variable indexing feature of CP express that jobs are assigned to the stations where the required tasks are performed. Constraints (21) are logically equivalent to: $X_j = m \Rightarrow Y_{m,j.task} = 1 \ \forall j \in Jobs, \forall m \in Stations_j$. Constraints (22) give a valid upper bound for the task assignment variables and are formulated to reduce unnecessary alternative solutions. If (22) is not formulated, a task *t* would be assigned to a station although none of the jobs which include task *t* is 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. Constraints (22) are also symmetry breaking constraints which help to improve solution process by pruning the search tree.

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

$$\left(X_j = m \right) \Rightarrow \left(start\left(\beta_{j.product,m,c} \right) \leq start\left(\delta_{j,c} \right) \right)$$
$$\forall j \in Jobs, \forall m \in Stations, \forall c \in Cycles \qquad (23)$$

$$\left(X_j = m \right) \Rightarrow \left(end\left(\beta_{j.product,m,c} \right) \geq end\left(\delta_{j,c} \right) \right)$$
$$\forall j \in Jobs, \forall m \in Stations, \forall c \in Cycles \qquad (24)$$

$$duration\left(\beta_{pmc} \right) \geq \sum_{j \in Jobs | j.product = p} \left(X_j = m \right) \times duration\left(\delta_{jc} \right)$$
$$\forall j \in Jobs, \forall m \in Stations, \forall c \in Cycles \qquad (25)$$

$$end(\beta_{pm}) \leq start(makespan_c)$$
$$\forall j \in Jobs, \forall m \in Stations, \forall c \in Cycles \qquad (26)$$

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 (23) and (24) Constraints (23) restrict the start times of the product activities and ensure that the product must arrive to the station before its job activities are started. Constraints (24) 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 (26). Finally, the time spent by any product activity at any station includes the processing time and the waiting time until the next station becomes available. 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 (25). The complete CP model is formulated as follows:

Minimize (1)

Subject to (2)—(26)

Once the CP formulation (1)—(26) is solved for given |*Cycles*|, the optimal cycle time is equal to the difference between makespan of the last two cycles as, *cycle time* = $makespan_{|Cycles|}$-$makespan_{|Cycles|-1}$.

4. STABILIZATION ALGORITHM

Although the given CP model is able to find the optimal cycle times, it could be computationally intractable or time consuming to run the model by including all cycles for large-size applications. To deal with computational difficulties for large-size problems, we propose the following scheme which can be used to approximate the steady state cycle times. Note that the algorithm is based on the proposed CP model which is the novel future of this study. The algorithm propagates the resulting schedules in CP model to the repeating cycles.

```
Algorithm: STABILIZATION

1. Solve the CP model given in (1)—(26) problem for a number of cycles $n$, $n<|Cycles|$

2. Calculate actual processing time of each product on each station

3. Update station ready times for the next cycle, $n+1$

4. Generate starting and completion time of for all products on each station for cycles $n+1,...,|Cycles|$

    4.1 Calculate starting and completion times of the first product in the permutation schedule for all stations.

    4.2 Schedule rest of the products in the permutation schedule.

    4.3 Update station ready times for the next cycle by using the completion time of the last product on each station

5. Calculate steady state cycle time as the difference between the completion time of the cycles |Cycles|-1 and |Cycles|.
```

5. EXPERIMENTS

In this section, using the instances given in Öztürk (2013) we first show the required number of replications of a given

MPS to reach steady state cycle times. All test instances are available at http://homes.ieu.edu.tr/~cozturk/SBSFMMAL.rar . Note that for these instances, optimal makespan for cycle 1 is known. Table 1 shows cycle times for each replication. In this table, while the column "CP with |Cycles|=3" presents the results of the proposed CP model minimizing cycle time of the last replication, column "CP with |Cycles|=1" shows the results of the model minimizing cycle time of the first replication. Note that the completion times of the second and the third cycles in "CP model with |Cycles|=1" are calculated using the STABILIZATION algorithm given in the previous section. These instances were 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 Solver 6.0 and ILOG Scheduler 6.0 libraries is used to code and solve the proposed CP models. All solutions for the |Cycles|=3 case are optimal except for the last instance which the optimality is not proven in 1 hour. Results in Table 1 reveal that the stable cycle times are achieved in the second replication of the MPS. Furthermore, as shown in the second, the sixth and the eighth instances, minimum cycle time in the first cycle may not be optimal when all the repeated cycles are considered.

Table 1. Comparison of cycle times

			CP with |*Cycles*|=3			CP with |*Cycles*|=1 and STABILIZATION		
			Cycle Time			Cycle Time		
|*Tasks*|	|*Products*|	|*Stations*|	First Cycle	Second Cycle	Third Cycle	First Cycle	Second Cycle	Third Cycle
10	5	3	53	47	47	53	47	47
10	5	4	44	34	34	44	34	34
10	5	5	41	30	30	41	30	30
10	6	3	62	56	56	62	56	56
10	6	4	49	40	40	49	40	40
10	6	5	47	38	38	47	38	38
10	7	3	72	66	66	72	66	66
10	7	4	56	46	46	56	46	46
10	7	5	51	42	42	51	42	42
20	5	3	**111**	**93**	**93**	**108**	**97**	**97**
20	5	4	109	88	88	109	88	88
20	6	3	**120**	**108**	**108**	**119**	**109**	**109**
20	6	4	115	97	97	115	97	97
20	7	3	**144**	**132**	**132**	**143**	**133**	**133**
20	7	4	135	117	117	135	117	117
30	5	3	160	131	131	160	131	131

To generalize our findings, we run all instances given by Öztürk et al. (2013) with |Cycles|=2. Results of the experiments in Table 2 show scalability of the proposed CP model formulation where instances with (*) are optimal.

6. CONCLUSIONS

Balancing assembly lines, regarded as a tactical level problem, becomes an operational problem and must be considered along with sequencing decisions in Today's competitive market. However, in the relevant literature, there are a scarce number of papers dealing with this topical problem and none of them considers cyclic nature of the problem. Furthermore, studies about the cyclic scheduling of assembly lines neglect the balancing aspect of the problem. Hence, in this paper, we propose a CP model for simultaneous balancing and scheduling of flexible mixed model assembly lines with task scheduling in a cyclic manner. We show that the best schedule may not be obtained in the first cycle and the system reaches the steady state cycle time in the second replication. We also present a STABILIZATION scheme to extend results of the CP model to all cycles of the MPS. The model can be extended to find the minimum number of stations to achieve a given cycle time. Furthermore, sequence dependent setup times could be considered. Since the required number of cycles to reach the steady state cycle time may be different for individual stations, extending the proposed model and STABILIZATION scheme to assembly lines with parallel stations (Öztürk et al., 2012) could be another research direction with stochastic task times.

Table 2. Result of experiments

Instance Size	|*Tasks*|	|*Products*|	|*Stations*|	|*Jobs*|	Stable Cycle Time	Solution Time (sec.)
SMALL	10	5	3	38	47*	3.015
	10	5	4	38	34*	3.094
	10	5	5	38	30*	10.375
	10	6	3	44	56*	29.141
	10	6	4	44	40*	36.281
	10	6	5	44	38*	121.218
	10	7	3	52	66*	237.922
	10	7	4	52	46*	282.688
	10	7	5	52	42*	930.953
MEDIUM	20	5	3	76	93*	19.453
	20	5	4	76	88*	11.266
	20	5	5	76	70	3600
	20	6	3	88	108*	448.297
	20	6	4	88	97*	1207.922
	20	6	5	88	77	3600
	20	7	3	104	132*	1013.594
	20	7	4	104	117*	1281.969
	20	7	5	104	103	3600
LARGE	30	5	3	114	131	3600
	30	5	4	114	113	3600
	30	5	5	114	101	3600
	30	6	3	132	148	3600
	30	6	4	132	119	3600
	30	6	5	132	124	3600
	30	7	3	156	174	3600
	30	7	4	156	137	3600
	30	7	5	156	108	3600

Lastly, to improve performance of the proposed CP model, new search strategies, symmetry breaking constraints and hybridizing with local search methods could be considered.

REFERENCES

Boysen, N., Fliedner, M., and Scholl, A. (2009). Sequencing mixed-model assembly lines: Survey, classification and model critique. *European Journal of Operational Research*, 192, 349-773.

ILOG. (2003). OPL Studio 3.7. Language Manual.

Karabati, S., and Kouvelis, P. (1996). Cyclic scheduling in flow lines: modelling, observations, effective heuristics and a cycle time minimization procedure. *Naval Research Logistics*, 43, 211–231.

Karabatı S. and Sayın S. (2003). Assembly line balancing in a mixed-model sequencing environment with synchronous transfers. *European Journal of Operational Research*, (149), 417–429.

Öztürk C., Tunalı S., Hnich B. and Örnek A. (2013). Balancing and Scheduling of Flexible Mixed Model Assembly Lines. *Constraints*. http://dx.doi.org/10.1007/s10601-013-9142-6

Öztürk C., Tunalı S., Hnich B. and Örnek A. (2012). Balancing and scheduling of flexible mixed model assembly lines with parallel stations, *The International Journal of Advanced Manufacturing Technology*. http://dx.doi.org/10.1007/s00170-012-4675-1

Sawik,T., (2011). Batch versus cyclic scheduling of flexible flow shops by mixed-integer programming. *International Journal of Production Research*, 50 (18), 5017-5034.