A LAGRANGEAN RELAXATION APPROACH FOR
THE MIXED MODEL FLOW LINE SEQUENCING PROBLEM

Deniz Türsel Eliyi*
Dokuz Eylül University, Department of Statistics
Izmir 35160, Turkey

Melih Özlen
Middle East Technical University, Department of Industrial Engineering
Ankara 06531, Turkey

* Corresponding Author
Mailing Address: Dokuz Eylül University, Kaynaklar Campus, Department of Statistics
Buca 35160, Izmir / Turkey
Phone: +90(232) 4128562
Fax: +90(232) 4534265
E-Mail: deniz.eliiyi@deu.edu.tr, deniz.eliiyi@gmail.com
A LAGRANGEAN RELAXATION APPROACH FOR
THE MIXED MODEL FLOW LINE SEQUENCING PROBLEM

Abstract

In this study, a mixed model flow line sequencing problem is considered. Mixed model flow line (MML) is a special case of production lines where products are transported on a conveyor belt, and different models of the same product are intermixed on the same line. We have focused on product-fixed, rate-synchronous lines with variable launching. Our objective function is minimizing makespan. A heuristic algorithm based on Lagrangean relaxation is developed for the problem, and tested in terms of solution quality and computational efficiency.

Keywords: Mixed Model Flow Lines, Makespan, Lagrangean Relaxation

1. Introduction

In this study, we consider a mixed model flow line sequencing problem (MMS). In our mixed model flow line model, there are \( N \) jobs that should be processed by \( M \) stations in the same processing order. There are a total of \( R \) product models (or job-families), and each job belongs to a particular model. The problem therefore resembles a mixed model sequencing problem where different models of the same product are intermixed on the same line.

The assumptions are that there are no machine breakdowns and that all jobs and stations are available at time zero. The transfers between the stations are synchronous, i.e., the transfers take place at the times when all stations finish their jobs. This is commonly the case in many manufacturing systems where jobs are transferred from machine to machine by some kind of automated material handling system. Cycle time or cycle length is defined as the length of time...
between two transfers. It is assumed that the speed of the material handling mechanism can be adjusted by the end of each transfer.

We let $p_{rm}$ denote the processing time of a job of model $r$ on station $m$, where $r=1, \ldots, R$, and $m=1, \ldots, M$. We let $C_j$ denote the time spent between $(j-1)^{th}$ and $j^{th}$ transfers, i.e. the cycle time of the $j^{th}$ cycle, where $j=1, \ldots, N+M-1$, as there are $N+M-1$ cycles in a schedule. Our objective function is minimizing the makespan. The makespan is important when jobs arrive in batches and are delivered simultaneously. It is closely related to the throughput objective, which is an important performance measure for production facilities. Heuristics that tend to minimize the makespan in a machine environment with a finite number of jobs also tend to maximize the throughput rate, when the flow of jobs is constant over time (Pinedo and Chao [1]). Based on the above definitions, the makespan in our problem is expressed as the sum of all cycle times, i.e. $\sum_{j=1}^{N+M-1} C_j$.

Figure 1 illustrates the concept of cycle times in an example mixed model flow line with 3 jobs and 3 stations. Assume all jobs are of different models in the example.

**INSERT FIGURE 1 HERE**

In the figure, it can be observed that only the first station is busy during the first cycle. As the first cycle ends, the job processed on the first station is transferred to the second station, and a new job is loaded to the first one. Then, job 1 passes to the third station and job 2 passes to the second station, while a new job is loaded to the emptied first station. The cycle times, that are the maximum of the processing time of the jobs that are on the line simultaneously, become $C_1 = p_{11}$, $C_2 = \max\{p_{21}, p_{12}\}$, $C_3 = \max\{p_{31}, p_{22}, p_{13}\}$, $C_4 = \max\{p_{32}, p_{23}\}$, $C_5 = p_{33}$. The makespan is $C_1 + C_2 + C_3 + C_4 + C_5$. 
The system in our study differs from classical mixed model sequencing and flow shop studies in the following aspects: The speed of transfer mechanism is fixed in classical mixed model sequencing, i.e. $C_t = C$ for all $t$, whereas the speed of transfer mechanism is adjusted at the end of each transfer by assuming arbitrary $C_t$ in our study. In a classical flow shop model, the transfers between stations are asynchronous, i.e. the complete job is immediately transferred to the next station, whereas we assume synchronous transfers.

There are a large number of papers on a variety of flow shop models in the literature, and the majority of these studies are on makespan minimization. Johnson [2] provided an optimizing rule for makespan minimization when there are two machines. Garey et al. [3] proved that the problem is NP-hard for three machines. Some other note-worthy studies on flow shop models with makespan minimization include those of Palmer [4], Campbell et al. [5], Gupta [6], Baker [7], Widmer and Hertz [8] and Taillard [9].

Mixed model sequencing problems are studied by a number of researchers. Yano and Rachamadugu [10] provided a mathematical programming formulation of the problem that minimizes total utility work. Bolat et al. [11] developed two heuristic algorithms and a branch and bound procedure for minimizing total setup and utility work costs simultaneously. Tsai [12] studied on minimizing the risk of stopping a conveyor, and minimizing total utility work objectives. Miltenburg [13] considered the balancing problem, i.e. the problem of keeping the quantity of each model as constant as possible. Kubiak and Sethi [14] studied the problem of minimizing the total deviation of actual production from the desired production, where Steiner and Yeomans [15] considered minimizing maximum deviation. Karabati and Tan [16] addressed the stochastic cyclic scheduling problem in synchronous assembly and production lines whose stations have no buffers. Karabati and Sayin [17] study the assembly line balancing problem on

When the demand is stable and the number of models is small in mixed model lines, the periodic scheduling of minimal part sets can also be utilized. A minimal part set is defined as the smallest possible set of parts in the same proportion as the demand-mix of the whole working period (Wittrock [22]). After the minimal part set is ordered, the entire schedule is found by repeating the minimal part set order. Wittrock [22] studied on maximizing throughput and minimizing the WIP in a flexible flow line through periodic scheduling. Bolat [23] considered minimizing the total utility work of minimum part sets in a mixed model line with fixed launching rate. Sarker and Pan [24] investigated the effect of launch interval on the total cost of utility time and idle time in a mixed model line. Periodic scheduling in mixed model lines was also addressed by Miltenburg [13].

To the best of our knowledge, the unique study on the MMS problem is due to Soylu [25]. They propose an optimizing Branch and Bound algorithm for moderate-sized problem instances. They carry on experimentation with 20 to 35 jobs, and 2 and 4 machines, and reported that the algorithm can solve to optimality up to a quite limited number of 35 jobs of 3 models.
with 3 stations in 90 minutes of CPU time. The results also indicated that algorithm efficiency is very sensitive to the number of models and stations.

In this study, we develop a heuristic algorithm for the problem of minimizing makespan on mixed model lines based on Lagrangean relaxation approach. To solve the Lagrangean dual problem, we apply the well-known subgradient optimization procedure (Fisher [26]) that iterates between upper and lower bounding procedures, and iteratively updates the Lagrangean multipliers.

In the next section, we define our problem, and present its mathematical programming formulation and its complexity status. In Section 3, we present our heuristic procedure. In Section 4, we provide the results of our computational experiments. Finally, we conclude in Section 5.

2. Problem Definition

There are $M$ stations, $N$ jobs, and $R$ job models in the flow shop. Each job belongs to one model, and $D_i$ denotes the demand (or order) for model $r$. Obviously, there will be $N$ positions in the schedule to be occupied by $N$ jobs. Jobs of the same model have the same processing time on each station.

Processing of a job must be completed on the current machine before it is started on the succeeding machine. This means that initially all jobs are available and that each machine is restricted to processing only one job at any particular time. Since the first machine in the facility arrangement is the first to be visited by each job, the other machines are idle and other jobs are queued. The stations are indexed in the order of being visited by the jobs, i.e. all jobs first enter station 1, then station 2, and so on. Transfer times of the jobs between stations are negligible. Job
splitting and preemption are not allowed; a job once started on a station is processed until it finishes its operation on that station. The setup times are independent of the job sequence and are therefore included in the processing times of jobs.

Below, we formulate the problem as a mixed integer programming model.

Decision variable: \(X_{rn} = \begin{cases} 1 & \text{a job of model } r \text{ occupies the } n^{th} \text{ position in the schedule} \\ 0 & \text{otherwise} \end{cases}\)

Parameters: 
\(p_{rm}\): processing time of a job of model \(r\) on station \(m\)
\(D_r\): demand for model \(r\)

\[
z = \text{Min} \sum_{j=1}^{N+M-1} C_j \tag{0} \]

subject to:
\[
\sum_{r=1}^{R} X_{rn} = 1 \quad n = 1, ..., N \tag{1}
\]
\[
\sum_{n=1}^{N} X_{rn} = D_r \quad r = 1, ..., R \tag{2}
\]
\[
C_j \geq \sum_{r=1}^{R} p_{rk} X_{r(j,k+1)} \quad j, k \in JK = \begin{cases} j = 1, ..., M - 1 & k = 1, ..., j \\ j = M, ..., N & k = 1, ..., M \\ j = N + 1, ..., N + M - 1 & k = j - N + 1, ..., M \end{cases} \tag{3}
\]
\[
X_{rn} \in \{0,1\} \quad r = 1, ..., R \quad n = 1, ..., N \tag{4}
\]

The objective function expressed in (0) minimizes makespan. Constraint set (1) guarantees that every position in the sequence is occupied by exactly one job. Constraint set (2) implies that the demand for each model is fully satisfied. Constraint set (3) identifies the cycle time. Note that there are \(M*N\) constraints in constraint set (3).
The classical flow shop problem is NP-hard in the strong sense for \( m \geq 3 \) (Garey et al. [3]). However, the complexity of MMS, which has the additional constraint of synchronous transfers, is still open.

### 3. Lagrangean Relaxation

In this section, we present a heuristic algorithm based on Lagrangean relaxation for the MMS problem. Our Lagrangean Relaxation (LR) procedure is designed with the intention of tackling large-sized problem instances in small computational times. In Lagrangean relaxation, the aim is to relax some constraints of the original problem so that the resulting (relaxed) model is an easy-to-solve one. There is a trade-off in making relaxations: Either the relaxation is made so that the relaxed model is solved easily at an expense of generating looser bounds, or some tighter bounds can be found by spending more effort on the solution of the relaxed problem.

In our LR procedure, we choose to relax constraint set (3) with Lagrange multipliers, \( \lambda_{jk} \). Note that this constraint set is the complicating one, as it forces synchronous transfers. The relaxed model can be expressed as follows:

\[
LR(z) = \min \sum_{j=1}^{N+M-1} C_j - \sum_{j,k \in JK} \lambda_{jk} C_j + \sum_{j,k \in JK} \sum_{r=1}^{R} \lambda_{jk} p_{rk} X_{r(j \rightarrow k+1)}
\]

subject to: (1), (2), (4), and

\[
\lambda_{jk} \geq 0, j,k \in JK.
\]

When \( \lambda_{jk} \) values are known, the relaxed model decomposes into two subproblems, one in assignment and one in cycle time variables. The first subproblem includes only the assignment variables, and is given below in explicit form:
Subproblem 1:

\[ LR_1(z) = \text{Min} \sum_{j,k \in JK} \lambda_{j,k} \sum_{r=1}^{R} p_{r,k} x_{r(j-k+1)} \]

subject to:

\[ \sum_{r=1}^{R} x_{r,n} = 1 \quad n = 1, \ldots, N \quad (1) \]

\[ \sum_{n=1}^{N} x_{r,n} = D_r \quad r = 1, \ldots, R \quad (2) \]

\[ x_{r,n} \leq 1 \quad r = 1, \ldots, R \quad n = 1, \ldots, N \quad (4') \]

Note that the subproblem becomes a transportation problem with fixed \( \lambda_{jk} \), hence, constraint set (4) of the original formulation can be replaced with (4'). This subproblem can be solved by transportation simplex method.

Subproblem 2 below is a simple linear programming model with only upper and lower bounds on variables.

Subproblem 2:

\[ LR_2(z) = \text{Min} \sum_{j=1}^{M+N+1} C_j - \sum_{j,k \in JK} \lambda_{j,k} C_j \]

subject to:

\[ C_j \leq \text{Max} \left\{ \text{Max} \left\{ p_{r,k} \right\} \right\} \quad r = 1, \ldots, R \quad j,k \in JK = \begin{cases} j = 1, \ldots, M - 1 & k = 1, \ldots, j \\ j = M, \ldots, N & k = 1, \ldots, M \\ j = N + 1, \ldots, N + M - 1 & k = j - N + 1, \ldots, M \end{cases} \quad (5) \]

\[ C_j \geq \text{Max} \left\{ \text{Min} \left\{ p_{r,k} \right\} \right\} \quad r = 1, \ldots, R \quad j,k \in JK = \begin{cases} j = 1, \ldots, M - 1 & k = 1, \ldots, j \\ j = M, \ldots, N & k = 1, \ldots, M \\ j = N + 1, \ldots, N + M - 1 & k = j - N + 1, \ldots, M \end{cases} \quad (6) \]
We add constraint sets (5) and (6) as sets of valid inequalities for Subproblem 2 to strengthen our formulation. We know that any cycle time is dependent on the busy machines during that cycle. Hence, each cycle time is computed as the maximum of the processing times of the jobs that are processed in that cycle. Since any job can be scheduled during a cycle, the maximum of the processing times of the jobs that can be scheduled on the busy machines in a specific cycle is taken as an upper bound for that cycle. This is done by constraint set (5). Similarly, constraint set (6) provides a lower bound for the cycle time as the maximum (among machines) of the minimum processing times that can be scheduled on any busy machine. These bounds are utilized in the Lagrangean heuristic lower bound computation.

Subproblem 2 is a simple unconstrained linear programming model with upper and lower bounds on variables, whose optimal solution is readily available. We simply set a variable $C_j$ equal to its lower bound, established by constraint set (6), if its objective function coefficient is nonnegative. If not, we set $C_j$ equal to its upper bound, which is established by constraint set (5).

Note that the optimal objective function value of the relaxed problem is the sum of the optimal objective function values of the two subproblems, i.e., $LR(z) = LR_1(z) + LR_2(z)$. $LR(z)$ is a lower bound for the original problem, MMS.

An upper bound, i.e., a feasible solution for the problem is also necessary in the course of our procedure. For this purpose, we convert the above lower bound solution into a feasible solution by employing the following Lagrangean heuristic procedure.

Lagrangean Heuristic:

S1. Compute $C_j$ values for each cycle using constraint set (3) and the optimal solution of Subproblem 1 $X_{r,n}^*$.
S2. Calculate the new objective function value, using computed $C_j$ values. Note that $LR'(z)$ is an upper bound for the original problem, as it belongs to a feasible solution.

The time complexity of this simple heuristic procedure is $O(M*N)$, as $M*N$ constraints are considered for finding the $C_j$ values.

We utilize subgradient optimization procedure by Fisher [26] for solving the Lagrangean dual problem, which can be defined as $LD(z) = \max_{\lambda \geq 0} \{LR(z)\}$. Subgradient optimization is an iterative procedure that generates multipliers starting from an initial set of Lagrange multipliers. It can be viewed as a procedure that attempts to maximize the lower bound value obtained by solving the relaxed model for a specific choice of multipliers.

We present below the stepwise description of the LR procedure for our problem.

**Lagrangean Relaxation (LR) Procedure:**

S0. Set $t = 1$. Initialize $\lambda_{jk}^t = (1-0.01)/M$, and $\alpha' = 0.5$.

S1. Solve Subproblems 1 and 2 with Lagrange multipliers $\lambda_{jk}^t$. Let the optimal variables be $X_{rn}^t$ and $C_j^t$. Set $LB^t = LR_1^t(z) + LR_2^t(z)$.

S2. Find an upper bound, $UB^t$, using the Lagrangean heuristic explained above, with $X_{rn}^t$ and constraint set (3). Set $UB = \min\{UB, UB^t\}$.

If $\lceil LB \rceil = UB$, then the solution is optimal, stop.

S3. Determine step size: $s^t = \frac{\alpha' \cdot (UB - LB^t)}{\sum_{j,k \in JK} \left[ \sum_{r=1}^{g} p_{rk} X_{r(j+k+1)}^t - C_j^t \right]^2}$

Update multipliers: $\lambda_{jk}^{t+1} = \max\{0, \lambda_{jk}^t + s^t \left( \sum_{r=1}^{g} p_{rk} X_{r(j+k+1)}^t - C_j^t \right)\}$
If LB does not improve for a certain number of iterations \((\text{NumIt})\), then

\[
\alpha^{t+1} = \alpha^t \times \text{multiplier}.
\]

If total number of iterations \(\geq\) \(\text{Iteration-limit}\), stop,
Else, go to S1.

In S2, \(\lceil \text{LB} \rceil\) denotes the smallest integer greater than the lower bound value.

We provide a small but complete numerical example to clarify the steps of our LR procedure. Consider the following instance of the problem with three jobs and two machines: \(N = 3\), \(M = 2\), \(R = 2\), \(D_1 = 1\), \(D_2 = 2\), \(p_{11} = 4\), \(p_{12} = 2\), \(p_{21} = 2\), \(p_{22} = 3\). When the model is written for this small instance, one can see that the cycle times for the first and the last cycles are trivial, i.e. \(C_1 = p_{11}.X_{11} + p_{21}.X_{21}\) and \(C_4 = p_{12}.X_{13} + p_{22}.X_{23}\) in the optimal solution. Then the model can be reduced to the following:

\[
z = \text{Min} \ C_2 + C_3 + 4X_{11} + 2X_{21} + 2X_{13} + 3X_{23}
\]

subject to:

Constraint set (1): \(X_{11} + X_{21} = 1\)
\(X_{12} + X_{22} = 1\)
\(X_{13} + X_{23} = 1\)

Constraint set (2): \(X_{11} + X_{12} + X_{13} = 1\)
\(X_{21} + X_{22} + X_{23} = 2\)

Constraint set (3): \(C_2 \geq 4X_{12} + 2X_{22}\)
\(C_2 \geq 2X_{11} + 3X_{21}\)
\(C_3 \geq 4X_{13} + 2X_{23}\)
\(C_3 \geq 2X_{12} + 3X_{22}\)
Constraint set (4): \( X_{r,n} \in \{0,1\}, \ r = 1, 2 \) and \( n = 1, 2, 3 \)

Constraint set (5): \( C_j \geq 0, \ j = 1,\ldots, 4 \)

Relaxation of constraint set (3) yields the following two subproblems.

**Subproblem 1:**

\[
LR_1(z) = \text{Min} \ (4+2\lambda_{22})X_{11} + (4\lambda_{21}+2\lambda_{32})X_{12} + (2+4\lambda_{31})X_{13} + (2+3\lambda_{22})X_{21} + (2\lambda_{21}+3\lambda_{32})X_{22} + (3+2\lambda_{31})X_{23}
\]

subject to constraint sets (1), (2), and
\( X_{r,n} \leq 1, \quad r = 1, 2 \) and \( n = 1, 2, 3 \)

**Subproblem 2:**

\[
LR_2(z) = \text{Min} \ (1-\lambda_{21}-\lambda_{22})C_2 + (1-\lambda_{31}-\lambda_{32})C_3
\]

subject to
\( 2 \leq C_j \leq 4 \quad j = 2, 3. \)

For demonstration purposes, we let \( \lambda_{1,jk} = 0.1 \) for all \( j \) and \( k \), and \( \alpha^1 = 0.5 \). In the first iteration, optimal solution of the first subproblem is \( X_{13}^1 = X_{21}^1 = X_{22}^1 = 1 \) with \( LR_1^1(z) = 5.2 \). The second subproblem yields \( C_{1,2}^1 = C_{1,3}^1 = 2 \) with \( LR_2^1(z) = 3.2 \). Then, \( LB^1 = 8.4 \). The upper bound value is found using the optimal assignment values from \( LR_1^1(z) \) and constraint set (3) as \( UB^1 = Max\{2,3\} + Max\{4,3\} + 2 + 2 = 11 \). Step size and updated multiplier values then become: \( s^1 = 0.2166, \lambda_{21}^2 = 0.1, \lambda_{22}^2 = 0.3166, \lambda_{31}^2 = 0.5332, \lambda_{32}^2 = 0.3166 \).

Using the updated multiplier values in the second iteration, optimal solution of the first subproblem becomes \( X_{12}^2 = X_{21}^2 = X_{23}^2 = 1 \) with \( LR_1^2(z) = 8.05 \). The second subproblem yields
\( C^2_2 = C^2_3 = 2 \) with \( LR^2_2(z) = 1.47 \). Then, \( LB^2 = 9.52 \), and \( UB = 11 \). Updated step size and multipliers are: \( s^2 = 0.1483, \lambda^3_{21} = 0.3966, \lambda^3_{22} = 0.4649, \lambda^3_{31} = 0.5332, \lambda^3_{32} = 0.3166 \).

In the third iteration, first subproblem yields \( X^3_{313} = X^3_{321} = X^3_{322} = 1 \) with \( LR^3_{11}(z) = 9.27 \). Subproblem 2 yields \( C^3_2 = C^3_3 = 2 \) with \( LR^3_2(z) = 0.58 \). Then, \( LB^3 = 9.85 \), and \( UB = 11 \). Step size and multipliers are updated as: \( s^3 = 0.0960, \lambda^4_{21} = 0.3966, \lambda^4_{22} = 0.5609, \lambda^4_{31} = 0.7253, \lambda^4_{32} = 0.4126 \).

In the fourth iteration, the first subproblem gives \( X^4_{312} = X^4_{321} = X^4_{323} = 1 \) with \( LR^4_{11}(z) = 10.54 \). Subproblem 2 yields \( C^3_2 = 2, C^3_3 = 4 \) with \( LR^3_2(z) = -0.4666 \). Then, \( LB^4 = 10.07 \), and \( UB = 11 \). Here we stop saying that this solution is optimal, since the smallest integer value greater than the lower bound is 11, which is equal to the upper bound. Hence, the jobs of model 2 (or jobs belonging to family 2) are scheduled as the first and third jobs, and the job of model 1 is scheduled as the second job in the optimal solution, which yields a makespan value of 11. The optimal schedule and cycle times are shown in Figure 2.

4. Computational Experience

In this section, we describe an experiment to test the efficiency of our LR procedure. The experiment is conducted on a PC with Intel Pentium 4 2.8 Ghz processor and 1 GB of RAM running under Linux, specifically Fedora Core 5, operating system. The LR procedure and Heuristic 1 are implemented in C and compiled with GCC 4.1.0. Borland C++BuilderX 1.0 is utilized as the development environment. CPLEX 8.1.1 is used for solving Subproblem 1 with transportation simplex and finding the optimal solutions.
We generate random test problems for different problem sizes of \( N = 10, 15, 20, 30, 60 \) and 120 jobs, \( R = \lceil N/4 \rceil, \lceil N/2 \rceil \), and \( N \) job models, and \( M = 3, 5 \) and 8 stations. Note that when \( R = N \), the problem becomes a flow shop with synchronous transfers. We use discrete uniform distributions between 1 and 20 to generate processing times. The demands of the models are randomly generated.

4.1 Design of Experiments

We perform pilot runs with relatively small sized problem instances to fine-tune the parameter values of the LR procedure.

Initial values of the Lagrange multipliers \( \lambda_{jk} \) do not significantly affect the performance of the subgradient optimization procedure (Fisher [26]). Hence, in our LR procedure, we set them to values which guarantee that coefficients of \( C_j \) in Subproblem 2 are small but positive, i.e., we set \( \lambda_{jk} = (1-0.01)/M \), \( \forall j, k \). The limit on the number of iterations for updating \( \alpha \) value (\( \text{Numit} \)), the multiplier used for the update (\( \text{multiplier} \)) and limit on the total number of iterations (\( \text{Iteration-Limit} \)) play critical roles on the performance of the proposed procedure. Two levels are proposed for values of each of these parameters, and a full-factorial design is used to check the significance. In many of the studies, the multiplier value is set as 0.5, but the value of 0.75 is also used to evaluate the effect of slow convergence on the performance of the algorithm. Three levels, 400, 800 and 1200 are used for evaluating the effect of the total number of iterations. The parameter \( \text{Numit} \) indicates the limit on the number of iterations without improvement, and the levels for this parameter are set as 20, 30 and 40. As a result of the experimentation, the total number of iterations to be used is set as 400 for small and medium size instances (up to 30 jobs), 800 for 60-job instances, and 1200 for 120-job instances. Similarly, the value of \( \text{Numit} \) is set as
20 for instances with up to 30 jobs, while it is increased to 30 and 40 for 60-job and 120-job instances, respectively. The parameter *multiplier* and is set as 0.5 for all instances.

Different sizes of problems that are used are given in Table 1. 10 random instances are generated for each combination of $N$, $M$ and $R$; hence we generate and solve a total of 540 instances.

**INSERT TABLE 1 HERE**

As it can be observed from the table, 3, 5 and 10 models are used for experimentation with 10 jobs, and the number of models increases with the number of jobs.

A simple heuristic (Heuristic 1) is also included in the experimentation to better assess the usefulness of our algorithm. The heuristic is a trivial one that schedules the models in decreasing order of their demands. Jobs of the same model are scheduled as a batch in this heuristic.

Consider the small example in Section 3, with $N=3$, $M=2$, $R=2$, $D_1=1$, $D_2=2$, $p_{11}=4$, $p_{12}=2$, $p_{21}=2$, $p_{22}=3$. When Heuristic 1 is executed on this instance, it schedules the jobs of model 2 to the first and second positions in the sequence, since the demand for model 2 is higher than that for model 1. The last position of the sequence is therefore filled with the remaining job of model 1. The makespan becomes 11. Note that Heuristic 1 yields an alternative optimal solution for this small instance, where $X_{21} = X_{22} = X_{13} = 1$.

**4.2 Discussion of Results**

For small instances with 10, 15 and 20 jobs, we use the gap between the optimal solution and our LR solution as a performance measure.
The performance of our Lagrangean Relaxation (LR) procedure for these instances is investigated through Table 2. The table presents, for all parameter combinations, the average and maximum CPU times (in seconds) of the LR procedure, and the CPU times for CPLEX to find optimal solutions. The table also presents, for both the LR procedure and Heuristic 1, the average and maximum upper bound deviations as a percentage of the optimal solutions. The gap between the solutions of Heuristic 1 and our LR procedure is taken as another performance criterion. The last column in the table presents the number of times our LR procedure finds the optimal solution (out of 10 instances).

**INSERT TABLE 2 HERE**

The table reveals that our LR procedure provides a very powerful upper bound for small instances. The procedure yields average deviations of no more than about 5.7% from the optimal solution. It can be observed from the table that, almost all of the instances with 10 jobs, and the majority of the instances having small number of job models are solved to optimality by our algorithm. In total, about 34% of the total number of instances are solved to optimality. The average deviations for $N = 10$, 15 and 20 do not exceed 0.4%, 1.7% and 2.8%, respectively, which are very satisfactory. The performance of Heuristic 1 for these instances is far worse than our algorithm. The average deviation of Heuristic 1 from the optimum solution reaches up to 35% for the settings with $N = 20$ jobs. The average gap between the solutions of Heuristic 1 and our algorithm is about 18%.

We can easily observe that the most dominant factor that affects the performance of both algorithms is the number of models. However, this effect is quite dramatic for Heuristic 1, which is an expected result. As for our algorithm, the performance steadily decreases as the number of models increase, but this decrease seems not to be a linear one, when the increase in the number
of job models is considered. The number of stations does not seem to affect the quality of the solutions found by the algorithms.

The results show that as the problem gets larger, the optimal solution time grows exponentially, reaching up to 20 hours for an instance with 20 jobs, 8 machines and 20 job models. On the other hand, our LR procedure is computationally very efficient, and provides a solution for any problem instance in less than 4 seconds. This is mainly due to the simple upper and lower bounding mechanisms that are incorporated in the procedure.

For instances with 30, 60 and 120 jobs, the optimal solutions are not reachable in manageable CPU times. Therefore, for these instances we compare our LR procedure with Heuristic 1. The performance is investigated through Table 3. The table presents, for all parameter combinations, the average and maximum CPU times (in seconds) of the LR procedure. The table also presents the gap between the solutions of Heuristic 1 and our LR procedure. We do not include the CPU times of Heuristic 1 in the table since the heuristic is very simple and the computation times are insignificant, i.e at most one second.

**INSERT TABLE 3 HERE**

The table reveals that our LR procedure outperforms Heuristic 1 for the large instances, as well. We can observe from the table that the percentage gap between solutions of the two algorithms seems stable as the number of jobs increases. The average gaps are about 18%. However, the gaps get larger as the number of models increases. The gaps for \( R = \lceil N/4 \rceil, \lceil N/2 \rceil, \) and \( N \) models become approximately 14%, 17% and 22%, respectively, averaging for all \( N \) values. These gaps are also expected to intensify if the variation in the processing times increases. The number of stations does not seem to affect the gap between the solutions.
Our LR procedure is computationally very efficient for the large instances, as well. Even the solution of the largest instance with 120 jobs, 120 job models and 8 machines takes no more than 5.3 minutes on the average.

5. Conclusions

In this study, we consider the mixed model flow line sequencing problem (MMS) with makespan objective. We formulate the problem as a mixed integer mathematical model. A Lagrangean Relaxation-based heuristic algorithm is developed for the problem.

We test our algorithm using computational experimentation. For small size instances, the gap between the optimal solution and our LR solution is used as a performance measure. The algorithm has an outstanding performance for these instances, resulting in no more than 5.5% deviation from the optimal solution. One conclusion from the experiment is that the most significant parameter affecting the problem difficulty is the number of models. For large instances of the problem, we compare the solution quality of our LR algorithm using the results of a straightforward heuristic. We observe that our algorithm outperforms the heuristic for both small and large instances when the quality of solutions is considered.

The developed LR procedure generates quality solutions in very small computation times; therefore it may be of great use when one is more interested in nice and quick solutions rather than a guarantee of optimality. Future research may point the development of branch and bound approaches that use our lagrangean-relaxation based upper and lower bounds.
References


Vitae

Deniz Türsel Eliyi received her B.S., M.S. and Ph.D. degrees in Industrial Engineering from the Middle East Technical University, Ankara, Turkey. She has worked as a research assistant in the same department between years 1998 and 2005. She currently works as an assistant professor at the Operations Research Division of Statistics Department in Dokuz Eylul University, Izmir, Turkey. Her research interests are in production planning, scheduling and combinatorial optimization.

Melih Özlen received his B.S. and M.S. degrees in Industrial Engineering from the Middle East Technical University, Ankara, Turkey. He is a research assistant and a Ph.D. student in the Industrial Engineering Department of Middle East Technical University. His research interests are in optimization and scheduling.
Figure 1. Cycle Times in a Mixed Model Flow Line.
Figure 2. The optimal solution and the cycle times for the example problem.
Table 1. Problem settings used in experimentation.

<table>
<thead>
<tr>
<th># of Positions (N)</th>
<th># of Models (R)</th>
<th># of Machines (M)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3, 5, 10</td>
<td>3, 5, 8</td>
</tr>
<tr>
<td>15</td>
<td>4, 8, 15</td>
<td>3, 5, 8</td>
</tr>
<tr>
<td>20</td>
<td>5, 10, 20</td>
<td>3, 5, 8</td>
</tr>
<tr>
<td>30</td>
<td>8, 15, 30</td>
<td>3, 5, 8</td>
</tr>
<tr>
<td>60</td>
<td>15, 30, 60</td>
<td>3, 5, 8</td>
</tr>
<tr>
<td>120</td>
<td>30, 60, 120</td>
<td>3, 5, 8</td>
</tr>
</tbody>
</table>
Table 2. Results of the experiment for $N = 10$, 15 and 20.

<table>
<thead>
<tr>
<th>$M$</th>
<th>$N$</th>
<th>$R$</th>
<th>CPU time (LR) Avg. (sec.)</th>
<th>Max. (sec.)</th>
<th>CPU time (CPLEX) Avg. (sec.)</th>
<th>Max. (sec.)</th>
<th>$LR - OPT$ Avg. (%)</th>
<th>Max. (%)</th>
<th>$H1 - OPT$ Avg. (%)</th>
<th>Max. (%)</th>
<th>$H1 - LR$ Avg. (%)</th>
<th>Max. (%)</th>
<th># of opt.*</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>10</td>
<td>3</td>
<td>0.3</td>
<td>0.7</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>6.62</td>
<td>14.46</td>
<td>6.62</td>
<td>14.46</td>
<td>10</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>5</td>
<td>0.3</td>
<td>0.3</td>
<td>0.1</td>
<td>1.0</td>
<td>0.07</td>
<td>0.68</td>
<td>11.06</td>
<td>21.29</td>
<td>10.98</td>
<td>21.29</td>
<td>9</td>
</tr>
<tr>
<td>3</td>
<td>10</td>
<td>10</td>
<td>0.5</td>
<td>0.7</td>
<td>0.5</td>
<td>2.0</td>
<td>0.53</td>
<td>2.21</td>
<td>28.58</td>
<td>36.15</td>
<td>27.92</td>
<td>36.15</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>4</td>
<td>0.4</td>
<td>0.5</td>
<td>0.8</td>
<td>3.0</td>
<td>0.62</td>
<td>2.99</td>
<td>12.00</td>
<td>25.81</td>
<td>11.31</td>
<td>25.81</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>8</td>
<td>0.7</td>
<td>0.8</td>
<td>13.5</td>
<td>78.0</td>
<td>0.97</td>
<td>2.20</td>
<td>19.51</td>
<td>39.25</td>
<td>18.33</td>
<td>37.04</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>15</td>
<td>1.3</td>
<td>1.6</td>
<td>3042.1</td>
<td>25233.0</td>
<td>2.58</td>
<td>4.28</td>
<td>28.36</td>
<td>40.96</td>
<td>25.13</td>
<td>38.02</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>5</td>
<td>1.2</td>
<td>1.3</td>
<td>96.1</td>
<td>313.0</td>
<td>0.71</td>
<td>2.33</td>
<td>15.58</td>
<td>27.84</td>
<td>14.77</td>
<td>26.96</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>10</td>
<td>1.9</td>
<td>2.1</td>
<td>12653.2</td>
<td>43813.0</td>
<td>2.72</td>
<td>5.33</td>
<td>25.15</td>
<td>33.96</td>
<td>21.82</td>
<td>30.07</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>20</td>
<td>20</td>
<td>3.4</td>
<td>3.9</td>
<td>29859.9</td>
<td>67791.0</td>
<td>5.66</td>
<td>8.81</td>
<td>35.00</td>
<td>43.91</td>
<td>27.73</td>
<td>36.21</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>3</td>
<td>0.2</td>
<td>0.3</td>
<td>0.0</td>
<td>0.0</td>
<td>0.00</td>
<td>0.00</td>
<td>15.20</td>
<td>35.03</td>
<td>15.20</td>
<td>35.03</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>5</td>
<td>0.3</td>
<td>0.4</td>
<td>0.2</td>
<td>1.0</td>
<td>0.05</td>
<td>0.50</td>
<td>13.53</td>
<td>29.73</td>
<td>13.47</td>
<td>29.73</td>
<td>9</td>
</tr>
<tr>
<td>5</td>
<td>10</td>
<td>10</td>
<td>0.6</td>
<td>0.9</td>
<td>1.0</td>
<td>2.0</td>
<td>1.27</td>
<td>2.91</td>
<td>27.41</td>
<td>34.68</td>
<td>25.82</td>
<td>33.14</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>4</td>
<td>0.4</td>
<td>0.5</td>
<td>2.0</td>
<td>13.0</td>
<td>0.72</td>
<td>1.85</td>
<td>13.72</td>
<td>26.89</td>
<td>12.90</td>
<td>25.83</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>8</td>
<td>1.0</td>
<td>1.4</td>
<td>41.8</td>
<td>274.0</td>
<td>2.33</td>
<td>5.56</td>
<td>22.87</td>
<td>29.96</td>
<td>20.04</td>
<td>25.40</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>15</td>
<td>15</td>
<td>1.8</td>
<td>2.1</td>
<td>2320.8</td>
<td>8358.0</td>
<td>2.82</td>
<td>4.35</td>
<td>28.73</td>
<td>41.44</td>
<td>25.25</td>
<td>38.94</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>5</td>
<td>1.1</td>
<td>1.4</td>
<td>92.1</td>
<td>305.0</td>
<td>1.41</td>
<td>2.25</td>
<td>15.27</td>
<td>27.46</td>
<td>13.63</td>
<td>24.83</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>10</td>
<td>1.6</td>
<td>1.8</td>
<td>9177.0</td>
<td>29605.0</td>
<td>2.94</td>
<td>4.68</td>
<td>22.80</td>
<td>29.15</td>
<td>19.30</td>
<td>26.18</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>20</td>
<td>20</td>
<td>3.1</td>
<td>3.6</td>
<td>38868.1</td>
<td>61435.0</td>
<td>3.54</td>
<td>6.08</td>
<td>30.75</td>
<td>40.70</td>
<td>26.27</td>
<td>35.93</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>3</td>
<td>0.7</td>
<td>0.8</td>
<td>0.1</td>
<td>1.0</td>
<td>0.00</td>
<td>0.00</td>
<td>8.21</td>
<td>20.10</td>
<td>8.21</td>
<td>20.10</td>
<td>10</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>5</td>
<td>0.7</td>
<td>0.9</td>
<td>0.1</td>
<td>1.0</td>
<td>0.55</td>
<td>2.34</td>
<td>11.73</td>
<td>24.22</td>
<td>11.13</td>
<td>23.40</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
<td>10</td>
<td>0.5</td>
<td>0.6</td>
<td>3.7</td>
<td>13.0</td>
<td>0.91</td>
<td>4.55</td>
<td>18.57</td>
<td>24.55</td>
<td>17.50</td>
<td>22.91</td>
<td>4</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>4</td>
<td>0.8</td>
<td>1.0</td>
<td>2.4</td>
<td>16.0</td>
<td>0.66</td>
<td>1.99</td>
<td>11.97</td>
<td>21.59</td>
<td>11.23</td>
<td>20.91</td>
<td>5</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>8</td>
<td>1.2</td>
<td>1.4</td>
<td>68.9</td>
<td>195.0</td>
<td>1.49</td>
<td>2.59</td>
<td>19.88</td>
<td>27.39</td>
<td>18.12</td>
<td>24.52</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>15</td>
<td>15</td>
<td>1.6</td>
<td>1.8</td>
<td>11957.9</td>
<td>62067.0</td>
<td>3.15</td>
<td>5.17</td>
<td>28.37</td>
<td>32.11</td>
<td>24.45</td>
<td>27.83</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>5</td>
<td>1.2</td>
<td>1.5</td>
<td>290.3</td>
<td>1784.0</td>
<td>1.93</td>
<td>4.03</td>
<td>14.87</td>
<td>28.53</td>
<td>12.74</td>
<td>28.20</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>10</td>
<td>1.7</td>
<td>1.9</td>
<td>14726.9</td>
<td>42236.0</td>
<td>3.35</td>
<td>6.56</td>
<td>21.49</td>
<td>30.17</td>
<td>17.60</td>
<td>25.27</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>20</td>
<td>20</td>
<td>2.9</td>
<td>3.9</td>
<td>50690.7</td>
<td>74450.0</td>
<td>2.84</td>
<td>7.03</td>
<td>24.01</td>
<td>32.00</td>
<td>20.61</td>
<td>28.33</td>
<td>0</td>
</tr>
</tbody>
</table>

* Number of times the LR procedure gives the optimal solution (out of 10 instances).
Table 3. Results of the experiment for $N = 30, 60$ and $120$.

<table>
<thead>
<tr>
<th>$M$</th>
<th>$N$</th>
<th>$R$</th>
<th>Avg. time (sec.)</th>
<th>Max. time (sec.)</th>
<th>Avg. (%)</th>
<th>Max. (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>30</td>
<td>8</td>
<td>2.3</td>
<td>2.7</td>
<td>12.51</td>
<td>19.91</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>15</td>
<td>4.0</td>
<td>4.6</td>
<td>20.01</td>
<td>29.71</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>30</td>
<td>6.3</td>
<td>7.0</td>
<td>24.76</td>
<td>32.09</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>15</td>
<td>11.1</td>
<td>12.5</td>
<td>15.10</td>
<td>21.50</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>30</td>
<td>12.2</td>
<td>14.1</td>
<td>17.80</td>
<td>25.19</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>60</td>
<td>36.9</td>
<td>41.3</td>
<td>20.85</td>
<td>25.17</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>30</td>
<td>37.6</td>
<td>46.2</td>
<td>15.23</td>
<td>20.85</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>60</td>
<td>102.1</td>
<td>120.8</td>
<td>21.88</td>
<td>26.96</td>
</tr>
<tr>
<td>3</td>
<td>120</td>
<td>120</td>
<td>266.7</td>
<td>307.9</td>
<td>22.20</td>
<td>26.01</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>8</td>
<td>2.4</td>
<td>2.9</td>
<td>15.06</td>
<td>21.15</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>15</td>
<td>3.5</td>
<td>4.4</td>
<td>15.27</td>
<td>22.20</td>
</tr>
<tr>
<td>5</td>
<td>30</td>
<td>30</td>
<td>6.8</td>
<td>7.8</td>
<td>23.01</td>
<td>31.26</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>15</td>
<td>10.9</td>
<td>12.7</td>
<td>15.65</td>
<td>20.20</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>30</td>
<td>15.5</td>
<td>16.9</td>
<td>18.77</td>
<td>22.71</td>
</tr>
<tr>
<td>5</td>
<td>60</td>
<td>60</td>
<td>38.6</td>
<td>42.4</td>
<td>24.00</td>
<td>28.84</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>30</td>
<td>38.7</td>
<td>50.1</td>
<td>14.24</td>
<td>18.54</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>60</td>
<td>77.8</td>
<td>126.1</td>
<td>13.30</td>
<td>22.91</td>
</tr>
<tr>
<td>5</td>
<td>120</td>
<td>120</td>
<td>297.2</td>
<td>335.9</td>
<td>24.56</td>
<td>27.83</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>8</td>
<td>2.4</td>
<td>3.3</td>
<td>11.73</td>
<td>24.62</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>15</td>
<td>3.8</td>
<td>4.4</td>
<td>15.13</td>
<td>22.41</td>
</tr>
<tr>
<td>8</td>
<td>30</td>
<td>30</td>
<td>5.9</td>
<td>6.8</td>
<td>17.83</td>
<td>23.28</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>15</td>
<td>11.3</td>
<td>13.9</td>
<td>10.34</td>
<td>12.87</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>30</td>
<td>14.9</td>
<td>16.8</td>
<td>14.78</td>
<td>18.68</td>
</tr>
<tr>
<td>8</td>
<td>60</td>
<td>60</td>
<td>40.4</td>
<td>44.6</td>
<td>19.37</td>
<td>20.61</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>30</td>
<td>40.6</td>
<td>53.3</td>
<td>10.98</td>
<td>14.95</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>60</td>
<td>99.8</td>
<td>131.0</td>
<td>13.71</td>
<td>19.31</td>
</tr>
<tr>
<td>8</td>
<td>120</td>
<td>120</td>
<td>321.3</td>
<td>339.2</td>
<td>19.49</td>
<td>21.73</td>
</tr>
</tbody>
</table>