Mixed-Integer Linear Programming Models for Managing Hybrid Flow Shops with Uniform, Non-Identical Multiple Processors

Santos D.L.* and Ivan Roa

Department of Systems Science and Industrial Engineering, T.J. Watson School of Engineering and Applied Science Binghamton University
Industrial and Systems Engineering Department, Monterrey Technical University, Mexico

Accepted date: February, 2007
Available online

Abstract

Shop floor management is one of the most difficult aspects, mathematically speaking, in most manufacturing environments and costs related to the scheduling of shop floor operations can have a significant impact on an organization’s bottom line. Flow shops with multiple processors at one or more stages are known by a variety of names – FSMPs, hybrid flow shops (HFS), and flexible flow lines, among others. They are important to study because they appear in many environments – semiconductor fabrication, printed circuit board assembly, and automotive assembly, to name a few. With very few exceptions, most of the extant HFS literature involves an often unrealistic assumption – that the multiple processors at a stage are identical in their processing speeds. In actual practice, it is rarely the case that multiple copies of a machine at one stage of processing will be exactly identical in speed.

This work presents mixed-integer linear programming (MILP) models for a variety of performance measures, all of which can impact costs in a HFS environment. A generalized model is presented for minimizing the time to complete all jobs (a.k.a. the makespan), followed by modifications of the model for other common performance measures.

Keywords: Scheduling; Hybrid flow shop; FSMP; Non-identical processors; MILP

1. Introduction

Hybrid flow shop (HFS) environments, also known as flow shops with multiple processors (FSMPs), flexible flow lines, and flexible flow shops, are so called as they are a hybrid between the pure flow shop scheduling problem (wherein all jobs follow the same processing sequence through the shop) and the parallel processor problem (where, at one or more stages in the flow shop, multiple copies of a machine are available for processing at those stages). Since most of the FSMP literature did not begin to appear until the late 1980s and early 1990s (see Wittrock, 1985; Brah, 1988; Gupta, 1988; Shah, 1989; Brah and Hunsucker, 1991; Brah et al., 1991; Deal and Hunsucker, 1991; and Santos, 1993, for examples), study of FSMPs is still relatively young. Furthermore, with few exceptions (Roa and Santos, 1999; Roa, 2000; and Santos and Roa, 2006, for examples), most of the extant literature in FSMP scheduling uses an assumption that is often (but not always) unrealistic. The assumption used in most FSMP scheduling articles (including our own) is that, for the stages that have multiple processors, the multiple processors are identical in their processing speeds (see Lee and Kim, 2004; and Choi et al., 2005, in addition to aforementioned references for examples).

While the extant literature provides sound research, the identical processors assumption is often not realistic for the following reasons (Santos and Roa, 2006): the newly added equipment may be of a newer (or older) generation and be faster (or slower) in its processing; the newly added equipment may be from a different manufacturer than the machine(s) already in place (while the multiple machines may have the same capabilities, they will likely be of differing speed); and even “identical” machines (same manufacturer, same generation) will have some natural variation between them causing them to not be equal in their processing speeds. This work presents mixed-integer linear programming (MILP) models for hybrid flow shops wherein the multiple processors at a stage that has multiple processors are not identical in speed. Model development is based upon the work by Brah et al. (1991) for the FSMP with identical processors. The non-identical processors under study in this work are uniform in speed. The primary model covered in this work is an MILP for makespan minimization. Subsequently, modifications of the model to handle other scheduling criteria (completion-time based and due-date based) will be discussed.

2. Non-Identical Parallel Processors

For the uniform case of parallel processors, their
processing speeds are different, but the time taken for a job to be processed on one machine is proportional to the time it would take to be processed on another machine. A pre-specified speed factor \( (s_k) \) is the proportionality between them (Friesen and Langston, 1983; and Dessouky et al., 1990). In determining the processing time of a job on a machine, the following transformation is used, \( p_{jk} = p_j / s_k \).

In the above,
- \( p_{jk} \) is the processing time of job \( j \) on machine \( k \) with speed factor;
- \( p_j \) is the processing time of job \( j \) without speed factor (a.k.a. baseline processing time); and
- \( s_k \) is the speed factor for machine \( k \); in this work, the slowest machine is where \( k = 1 \) and \( s_1 = 1.0 \) (later in this study, another subscript \( j \) will be added to denote the stage of processing wherein the multiple machines reside).

### 3. Assumptions for the Scheduling Environment

The assumptions used in this work are consistent with general studies in scheduling (French, 1982) and investigations in other FSMP identical-processor studies (see Brah and Loo, 1999; and Santos et al., 1995a, 1995b, 1996, 2001, among others) and are as follows:

1. A job may not be processed on more than one stage at a time;
2. The number of jobs, \( n \), is known and fixed;
3. All jobs arrive at time zero;
4. The processing times of the jobs at each of the stages are known and are constant (but modifiable according to the speed factors with which this work relates);
5. Set-up is independent of the job sequence (and can be included in transfer time variables incorporated into the model);
6. All jobs follow the same sequence;
7. A job may only visit one machine at a stage that has multiple processors;
8. No operation may be pre-empted (once started it must be processed to completion);
9. The flow shop has \( m \geq 2 \) stages in series;
10. Each stage has \( M_j \geq 1 \) machines \( (j=1 \ldots m) \) with the inequality holding for at least one stage;
11. All machines are available at time zero and are fully functional during the scheduling period;
12. No machine may process more than one job at a time;
13. Machines may be idle; and
14. In-process inventory is allowed.

### 4. Mixed-Integer Linear Programming Model

Prior to presentation of the general model, a list of variables and constants to be used in the MILP to minimize the makespan in a hybrid flow shop with non-identical multiple processors will be presented, some of which have been described in the assumptions, above.

#### Shop Floor Configuration Counters and Constants

- \( n \) = number of jobs (constant)
- \( i \) = index for the jobs \( (i = 1, 2, \ldots, n) \)
- \( r \) = another index for the jobs, starting at 2 \( (r = 2, 3, \ldots, n) \), used in the noninterference constraints
- \( m \) = number of stages (constant)
- \( j \) = index for the stages \( (j = 1, 2, \ldots, m) \)
- \( M_j \) = number of machines at stage \( j \) (constants)
- \( M_j \geq 1 \) for all \( j \), with inequality holding for at least one stage
- \( k \) = index for the machines at stage \( j \) \( (k = 1, 2, \ldots, M_j) \)

#### Completion Time and Process Time Variables and Constants

- \( C_{max} \) = makespan
- \( C_{ij} \) = completion time of job \( i \) at stage \( j \)
- \( C_{ij0} \) = ready (release) time of job \( i \) (constants), not to be confused with the use of the job counter \( r \) in the noninterference constraints
- \( t_{ij} \) = travel time from stage \( j\)-1 to stage \( j \) (can be used for non-sequence-dependent setup time problems; constants)
- \( p_{ij} \) = baseline processing time of the job on the slowest machine at that stage of job \( i \) on stage \( j \) (constants)
- \( s_{kj} \) = speed factor of machine \( k \) at stage \( j \) (constants)

#### Variables and Constants for Precedence Relationships

- \( X_{ij} = 1 \), if job \( i \) precedes job \( j \) on stage \( j \) at machine \( k \); or 0, otherwise
- \( Y_{ij} = 1 \), if job \( i \) on stage \( j \) is assigned to machine \( k \); or 0, otherwise
- \( Q \) = a very large number utilized for either-or (noninterference) relationships

The notation for an instance of a hybrid flow shop problem with minimum makespan can be expressed as: \( n/FFm/C_{max} \). For completeness, the instance would further have to supply all of the \( M_j, t_{ij}, s_{kj}, r, p_{ij} \) and baseline processing time (\( p_{ij} \) values).

#### 4.1. Generalized n/FFm/C_{max} MILP Model for Uniform Processors

Based upon the above definitions, the complete MILP is now presented.

Minimize \( C_{max} \)

\[
\begin{align*}
\text{s.t. } \\
C_{max} & \geq C_{im} \quad \text{for all } i
\end{align*}
\]
\[
\sum_{i=1}^{M} Y_{ik} = 1 \quad \text{for all } i \text{ and } j \\
C_i^j - C_{ij}^j \geq \sum_{k=1}^{M} Y_{ik} \frac{p_{ij}}{s_{ij}} + t_j \quad \text{for all } i \text{ and } j \\
Q(2 - Y_{ik}^j - Y_{ik}^j + X_{ij}) + C_i^j - C_{ij}^j \geq \frac{p_{ij}}{s_{ij}} \\
Q(3 - Y_{ik}^j - Y_{ik}^j - X_{ij}) + C_i^j - C_{ij}^j \geq \frac{p_{ij}}{s_{ij}} \\
\text{for all } i, r, j,
\]

and \( k \) where \( i < r \)

\[
Y_{ik} = \{0, 1\} \quad \text{for all } i, j, \text{ and } k
\]

\[
X_{ij} = \{0, 1\} \quad \text{for all } i, j-1, \text{ and } k \text{ where } i < r
\]

\[
C_i^j \geq r_i \quad \text{for all } i
\]

\[
C_i^0 = r_i \quad \text{for all } i
\]

The objective function (1) implies that we want to minimize the makespan. Constraint set (2) establishes that the makespan can be no smaller than the completion time of a job after the last stage of processing. Constraint set (3) establishes that any job \( i \) can only be assigned to one machine \( k \) at stage \( j \). Set (4) establishes that the completion time of job \( i \) at stage \( j \) is at least as large as its completion time on stage \( j-1 \) plus its processing time on stage \( j-1 \).

Constraint sets (5) and (6) establish the noninterference of jobs \( i \) and \( r \) being processed on the same machine \( k \) at stage \( j \). If job \( i \) precedes job \( r \) on machine \( k \) at stage \( j \), then constraint set (5) takes the following, inactive form: \( Q + C_i^j - C_{ij}^j \geq p_{ij} / s_{ij} \). This is a desired effect as we would not want, nor would it be possible for job \( i \) to complete after job \( r \) at this stage if \( i \) precedes \( r \) on the same machine. Further, if job \( i \) precedes job \( r \) on machine \( k \) at stage \( j \), then constraint set (6) takes the following, active form (and desired effect): \( C_i^j - C_{ij}^j \geq p_{ij} / s_{ij} \). Note that these constraints, (5) and (6), will appropriately switch roles if job \( r \) precedes job \( i \) on machine \( k \) at stage \( j \). This set of constraints, (5) and (6), need only be developed for the index \( i \) being less than \( r \) (\( i < r \)). If either job \( i \) or job \( r \) (or both) do not get assigned to machine \( k \) at stage \( j \), then either \( Q \) or \( 2Q \) is added to the left side of (5) and either \( 2Q \) or \( 3Q \) is added to the left side of (6) rendering both (5) and (6) to be inactive. Variables in constraint sets (7) through (10) have been defined in the notation section.

### 4.2 Example Problem for MILP Demonstration

We now present a small 4/FF2/C\(_{max}\) MILP formulation. In this instance, we will assume that travel times and sequence-independent setup times are negligible (so, \( t_{ij} = 0 \) for all \( i \) and \( j \)). Further, we will assume that all jobs are available to be processed at the same time (so, \( r_i \) can be set to 0 for all \( i \)). The baseline processing times and their corresponding times for the faster machines are shown in Table 1. The other known values that complete the problem are provided in the following list:

- \( M_1 = 3 \)
- \( s_{11} = 1.00, s_{21} = 1.20, s_{31} = 1.40 \)
- \( M_2 = 2 \)
- \( s_{12} = 1.00, s_{22} = 1.15 \)
- \( Q = 1000.0 \)

Even though this example problem is small, the MILP formulation is lengthy. Therefore, due to article-size limitations, the MILP formulation for the example is not included in this work. The interested reader should find, in an attempt to generate the MILP for this problem, the following numbers of constraints: 4 for Set (2); 8 for Set (3); 8 for Set (4); 60 for Sets (5) and (6), combined; 20 binary integer variable declarations for Set (7); 12 binary integer variable declarations for Set (8); Set (9), non-negativity, generally needs not be explicitly stated for most solver packages; and 4 for Set (10). The interested reader is also encouraged to contact the authors for a copy of the MILP formulation for this example.

This model was converted for input to Hyper-LINDO/PC© v.6.01 (henceforth, referred to as LINDO). The solution for this model was obtained by LINDO in under 10 seconds with an Intel Pentium4® processor (3.00 GHz). The optimal makespan is 14.57 and, to be succinct, only the completion times and sequence-independent setup times are negligible (so, \( t_{ij} = 0 \) for all \( i \) and \( j \)). Further, we will assume that all jobs are available to be processed at the same time (so, \( r_i \) can be set to 0 for all \( i \)). The baseline processing times and their corresponding times for the faster machines are shown in Table 1. The other known values that complete the problem are provided in the following list:

- \( M_1 = 3 \)
- \( s_{11} = 1.00, s_{21} = 1.20, s_{31} = 1.40 \)
- \( M_2 = 2 \)
- \( s_{12} = 1.00, s_{22} = 1.15 \)
- \( Q = 1000.0 \)

### Table 1. Processing Times for the Example 4/FF2/C\(_{max}\) Problem

<table>
<thead>
<tr>
<th>Job</th>
<th>Stage 1</th>
<th>Stage 2</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>( p_i / 1.00 )</td>
<td>( p_i / 1.20 )</td>
</tr>
<tr>
<td>J1</td>
<td>5.00</td>
<td>4.17</td>
</tr>
<tr>
<td>J2</td>
<td>7.00</td>
<td>5.83</td>
</tr>
<tr>
<td>J3</td>
<td>2.00</td>
<td>1.67</td>
</tr>
<tr>
<td>J4</td>
<td>6.00</td>
<td>5.00</td>
</tr>
</tbody>
</table>
Figure 1. Optimal Makespan Schedule for Example Problem

- CMAX = 14.57
- C12 = 7.57
- C22 = 14.57
- C32 = 6.67
- C42 = 14.50
- C11 = 3.57
- C10 = 0.00
- C21 = 7.00
- C20 = 0.00
- C31 = 1.67
- C30 = 0.00
- C41 = 6.67
- C40 = 0.00

The Gantt chart to depict this optimal solution is shown in Figure 1. Although optimal, it represents a delay schedule in that machine 2 at stage 2 does not begin processing job 3 until time 2.32, but it can start as early as time 1.67 (the completion time of job 3 at stage 1).

5. Other Performance Measures

Modifications will now be presented for the MILP to handle the following criteria (adding additional variables as appropriate): total completion time and mean completion time; maximum flow time; total flow time and mean flow time; maximum lateness; total lateness; maximum tardiness; and total tardiness. Unless explicitly stated, the changes to follow are all modifications of the general model as shown in Section 4.1. These changes should be treated independently, even though the equation numbering is consecutive. The procedure will typically be to replace the objective function (1) of the general model with a new objective function and to replace constraint set (2) of the general model with either one or two new sets of inequalities; all other constraints of the general model should remain.

5.1 Total Completion Time and Mean Completion Time

To modify the model to handle total completion time, the objective function (1) should be replaced with $C_{tot}$. Constraint set (2) should be changed to the following:

$$C_{tot} \geq \sum_{i=1}^{n} C_{im}$$  \hspace{1cm} (11)

To handle the mean completion time, the objective function (1) should be replaced with $C_{bar}$. Constraint set (2) should be changed to the following:

$$C_{bar} \geq \left( \frac{1}{n} \right) \sum_{i=1}^{n} C_{im}$$  \hspace{1cm} (12)

5.2 Maximum Flow Time

The objective function (1) should be replaced with $F_{max}$ and we should replace constraint set (2) in the general model with the following two sets of constraints:

$$F_i = C_{im} - r_i \quad \text{for all } i$$  \hspace{1cm} (13)

$$F_{max} \geq F_i \quad \text{for all } i$$  \hspace{1cm} (14)

There is a short-cut method (by combining (13) and (14)) but with this method, we can capture all of the flow times; (13) will also be used in some subsequent changes, below.

5.3 Total Flow Time and Mean Flow Time

For total flow time, the objective function (1) should be replaced with $F_{tot}$ and we should replace constraint set (2) of the general model with constraint set (13) as defined in Section 5.2 and to be accompa-
nied with the following:

\[ F_{tot} \geq \sum_{i=1}^{n} F_i \quad (15) \]

For mean flow time, we should replace (1) with \( F_{bar} \) and replace constraint set (2) with (13) defined above and to be accompanied with the following:

\[ F_{bar} \geq \left( \frac{1}{n} \right) \sum_{i=1}^{n} F_i \quad (16) \]

5.4 Maximum Lateness

All of the aforementioned criteria (makespan and those in Sections 5.1 – 5.3) are known as completion-time based performance measures. In order to consider the lateness of a job \( (L_i) \), due-dates \( (d_i) \) have to be assigned for them. As a job can be said to have positive lateness (i.e. it is tardy) or negative lateness (i.e. it is early), then incorporating lateness into an MILP is best done so by substituting the following everywhere in the model in which \( L_i \) is to be used (even though some solvers allow for unrestricted variables):

\[ L_i^+ - L_i^- \quad \text{for all } i \]

In the solution of an LP with such a transformation, at most one of the components (positive or negative) can have a non-zero, positive value. Also, with this transformation, tardiness-based performance measures can utilize the \( L_i^+ \) values (as we will, below) and earliness-based performance measures (not considered in this work) can utilize the \( L_i^- \) values.

To minimize the maximum lateness, the objective function (1) can be replaced with \( L_{max}^+ - L_{max}^- \). Constraint set (2) should be replaced with the following:

\[ L_{max}^+ - L_{max}^- \geq C_i - d_i \quad \text{for all } i \quad (17) \]

5.5 Total Lateness and Mean Lateness

To minimize the total lateness in our model, the objective function (1) can be replaced with \( L_{tot}^+ - L_{tot}^- \). Constraint set (2) should be replaced with the following:

\[ L_{tot}^+ - L_{tot}^- \geq \sum_{i=1}^{n} C_i - d_i \quad (18) \]

For mean lateness, we should replace (1) with \( L_{bar}^+ - L_{bar}^- \) and replace constraint set (2) with the following:

\[ L_{bar}^+ - L_{bar}^- \geq \left( \frac{1}{n} \right) \sum_{i=1}^{n} C_i - d_i \quad (19) \]

5.6 Maximum Tardiness

As tardiness is only defined if a job has positive lateness, in modifying the model to handle this crite-

\[ L_i^+ - L_i^- \geq C_i - d_i \quad \text{for all } i \quad (20) \]

5.7 Total Tardiness and Mean Tardiness

For total tardiness, the objective function (1) should be replaced with \( T_{tot} \) and we should replace constraint set (2) of the general model with constraint set (20) as defined in Section 5.6 and to be accompanied with the following:

\[ T_{tot} \geq \sum_{i=1}^{n} L_i^+ \quad (22) \]

For mean tardiness, we should replace (1) with \( T_{bar} \) and replace constraint set (2) with (20) defined in Section 5.6 and to be accompanied with the following:

\[ T_{bar} \geq \left( \frac{1}{n} \right) \sum_{i=1}^{n} L_i^+ \quad (23) \]

5.8 Other Criteria

The above are but a few, yet common completion-time based and due-date based criteria used in scheduling problems. In addition to the aforementioned, there are weighted versions of the performance measures, for which the models can be easily modified. Weighted versions may be especially important to consider because different customers’ jobs may be of different priorities, or cost factors, and these priorities will need to be included into the performance measures (objective functions).

6. Summary and Future Work

Hybrid flow shops are important to study as the environment is manifested in a variety of business sectors including, but not limited to the following (Santos, 1993; Linn and Zhang, 1999): electronics manufacturing, semiconductor processing, the pharmaceutical industry, and food/agriculture production. This paper presents a variety of MILP formulations for use in a scheduling environment that has yet to receive the attention it deserves. As can be argued, MILP and other optimal procedures may not be practical for large-sized problems. Nonetheless, these techniques can be useful for a variety of situations. The obvious situation for its use regards realistic problems of small size that are not uncommon, particularly for certain industries where processing times are large (for a real-world application of an MILP for parallel processor scheduling, see Santos and Heath, 1997).
The MILP herein has also been useful in gauging the quality of a lower bound developed for the makespan criterion for this environment (see Roa and Santos, 1999). MILP models can also be used to assist a practitioner in obtaining a feasible (not necessarily optimal) solution for a scheduling instance—a nontrivial task. This feasible solution can be compared against a lower bound on the objective to determine if the gap suggests significant room for improvement; if so, the solution may be improved via a tailored heuristic, such as the shifting bottleneck method or the exchange heuristic (Yang and Ignizio, 1987; Santos et al., 2001), or via use of metaheuristics such as Simulated Annealing (for an application in the FSMP, see Younes et al., 1998), Tabu Search, and Genetic Algorithms.

Furthermore, with the ever-increasing computational speeds, MILPs may also prove beneficial as the size of the model that the solver packages can handle will keep increasing (though not, of course, at the same rate). Future work for this problem can be focused on other scheduling criteria not covered in this work, like minimizing the number of tardy jobs, other cost-based (particularly, WIP and machine utilization) performance measures, or considering criteria based on earliness.

References


