AN ITEGER PROGRAMMING MODEL FOR OPEN SHOP SCHEDULING PROBLEM

Jen-Shiang Chen (陳禎祥)
Department of Industrial Engineering and Management
Far East College

ABSTRACT

This paper deals with multi-operation models, which are different from the flow shop and job shop models. In the flow shop or job shop models, the routes of all jobs are known in advance. In practice, the route of the job is often immaterial and up to the scheduler to decide. When the routes of the jobs are open, the model is referred to as an open shop. This paper describes the development of integer programming (IP) technique for the open shop scheduling problem with the objective of minimizing makespan.

Key words: scheduling, open shop, integer programming

1. Introduction

Scheduling concerns the allocation of limited resources to tasks over time. It is a decision-making process that has as a goal the optimization of one or more objectives [7]. Scheduling began to be taken seriously in manufacturing at the beginning of last century with the work of Henry Gantt and other pioneers. In the current competitive environment, effective scheduling has become a necessity for survival in the marketplace.

Scheduling problems may be classified according to various schemes. Day and Hottenstein [2] classified scheduling problems with respect to the nature of the job arrivals at the shop as static or dynamic. Static scheduling is the case where there are fixed number of jobs, all having the same starting time, in a shop with a fixed number of machines. Dynamic scheduling is the case where jobs are continuously arriving at the shop. Another classification of scheduling problems is deterministic vs. stochastic. All parameters in a deterministic problem are known with certainty. A problem is called stochastic if at least one parameter is probabilistic.

In defining a scheduling problem, both the technological constraints on jobs and the scheduling objectives must be specified [4]. Technological constraints are determined principally by the flow pattern of the jobs on machines. Maccarthy and Liu [4] further classified the machine shop environments as job shop, flow shop, open shop, permutation flow shop, single machine shop, parallel machines, and job shop with duplicate machines. Within any of these environments, scheduling may be attempted with respect to various objectives. Baker [1] classified the single objective as makespan, mean completion time,
mean flow time, mean waiting time, mean tardiness, maximum tardiness, and the number of tardy jobs.

Any processing order of jobs on the machines is allowed in a job shop. However, the operations of a job must be processed in a given order on the machines, and this order may be different among jobs. In a flow shop, every job must be processed on all machines in the order of $M_1, M_2, \ldots, M_m$. In the flow shop scheduling problem, if the job ordering is the same on every machine, we say that no passing is allowed since any job is not allowed to pass any former job. The flow shop scheduling problem where no passing is allowed is called the permutation flow shop problem. In the open shop, there are $m$ machines and each job has to be processed again on each one of the $m$ machines. However, some of these processing times may be zero. There are no restrictions with regard to the routing of each job through the machine environment. The scheduler is allowed to determine the route for each job, and different jobs may have different routes.

The assumptions made for the open shop scheduling problem are summarized here. Each job may visit certain machines at most once. The processing times are independent of the sequence. There is no randomness; all the data are known and fixed. All jobs are ready for processing at time zero at which the machines are idle and immediately available for work. No pre-emption is allowed, i.e., once an operation is started, it must be completed before another operation can be started on that machine. Machines never breakdown and are available throughout the scheduling period. There is only one of each type of machine. In-process inventory is allowed.

It is also known in the scheduling literature that the open shop scheduling problem is NP-hard [3, 7, 8] and thus computationally intractable. However, with the advances of powerful computer capacity and efficient integer programming (IP) software, mathematical programming-based scheduling research is beginning to receive more and more attention from researchers [6]. Although it is not an efficient solution method, mathematical programming formulation is a natural way to attack machine scheduling problems [8]. Why then do we study IP models at all? Morton and Pentico [5] explained it with the following two reasons. First, there will always be certain special structure cases that are solvable. If we understand the general approaches, we may recognize these. Second, there are often various partial relaxations of the equations that can be solved and may be useful.

Most IP problems arising in scheduling involve mixed BIP, i.e., some variables are binary and some are continuous. The new development on mixed BIP techniques, together with the substantial progress in computer capacity, has a great impact on IP scheduling models.

The problem of minimizing makespan in an open shop is theoretically challenging. In fact, it is NP-hard in the strong sense [3, 7, 8]. The purpose of this paper, therefore, is to concentrate on mixed binary integer programming (BIP) formulations of scheduling operations in open shops with the objective of minimizing makespan.

2. An integer programming model

In an open shop, there are no restrictions with regard to the routing of each job through the machine environment. The scheduler is allowed to determine the route for each job, and different jobs may have different routes. The notations used in this model are as follows:
$M$ = a very large positive number;
$m$ = number of machines in the shop;
$n$ = number of jobs for processing at time zero;
$J_i$ = job number $i$;
$M_k$ = machine number $k$;
$r_{ik}$ = 1 if $J_i$ requires $M_k$; 0 otherwise;
$N_i$ = number of operations of $J_i$ this is, $N_i = \sum x_{ijk}$;
$O_{ijk}$ = operation number $j$ of $J_i$ on $M_k$;
$s_{ijk}$ = the starting time of $O_{ijk}$;
$p_{ik}$ = processing time of $O_{ijk}$ on $M_k$;
$C_{max}$ = maximum completion time or makespan; $C_{max} = \max_{i=1}^{m} C_i$;
$x_{ijk} = 1$ if $J_i$ is scheduled in the $j$th position for processing on $M_k$; 0 otherwise.
$z_{ijk}^s = 1$ if $O_{ijk}$ precedes $O_{ij'k}$ (not necessarily immediately); 0 otherwise;

The formulations of the optimization model are stated as follows.

Min $C_{max}$

Subject to

$x_{ijk} \leq r_{ik} \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, N_i; \quad k = 1, 2, \ldots, m \quad (2)$

$\sum_{j=1}^{N_i} x_{ijk} = r_{ik} \quad i = 1, 2, \ldots, n; k = 1, 2, \ldots, m \quad (3)$

$\sum_{i=1}^{n} x_{ijk} = 1 \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, N_i \quad (4)$

$s_{ijk} \leq M \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, N_i; \quad k = 1, 2, \ldots, m \quad (5)$

$s_{ijk} + p_{ik} \leq s_{ij'k} + M (2 - x_{ijk} - x_{ij'k}) \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, N_i - 1; \quad k, k' = 1, 2, \ldots, m \quad (6)$

$s_{ijk} - s_{ij'k} \geq p_{ik} - M (2 - x_{ijk} - x_{ij'k}) - M (1 - z_{ijk}^s) \quad 1 \leq i < i' \leq n; j = 1, 2, \ldots, N_i; j' = 1, 2, \ldots, N_i; \quad k = 1, 2, \ldots, m \quad (7)$

$s_{ij'k} - s_{ijk} \geq p_{ik} - M (2 - x_{ijk} - x_{ij'k}) - M z_{ijk}^s \quad 1 \leq i < i' \leq n; j = 1, 2, \ldots, N_i; j' = 1, 2, \ldots, N_i; \quad k = 1, 2, \ldots, m \quad (8)$

$s_{ijk} + p_{ik} \leq C_{max} \quad i = 1, 2, \ldots, n; k = 1, 2, \ldots, m \quad (9)$

$s_{ijk} \geq 0 \quad i = 1, 2, \ldots, n; j = 1, 2, \ldots, N_i; \quad k = 1, 2, \ldots, m \quad (10)$

In the above model, constraint sets (2)-(4) describe the feasible value of $x_{ijk}$. Constraint set (5) enforces $s_{ijk} = 0$ when $x_{ijk} = 0$. Constraint set (6) guarantees that the processing of $O_{ijk}$ can be started only after $O_{ijk}$ is finished. Constraint sets (7) and (8) together enforce the requirement that only one operation may be processed on a machine at any time. That is, either $s_{ijk} - s_{ij'k} \geq p_{ik} - M (2 - x_{ijk} - x_{ij'k})$ or $s_{ij'k} - s_{ijk} \geq p_{ik} - M (2 - x_{ijk} - x_{ij'k})$. Constraint set (9) gives the definition of $C_{max}$ which is to be minimized in the objective function (1). The non-negativity for $C_{max}$ and $s_{ijk}$ and binary restrictions for $x_{ijk}$ and $z_{ijk}^s$ are specified in (10).

We summarize the model sizes, in terms of the number binary variables, constraints and continuous variables, of the IP model in Table 1.

### 3. An illustrative example

The machines required and the processing times are given in the Table 2. We summarize the model sizes, in terms of the number binary variables, constraints and continuous variables, of the IP model in Table 3. This example was solved with Industrial LINGO/PC [10] on a PC PIII/600 with 128M DRAM. And then the computational results, in terms of the number of simplex iterations, branches and solution time, were also reported in Table 3. The optimal makespan is 18. Figure 1 shows the Gantt chart of the
optimal schedule.

4. Conclusions

In this paper we develop a new formulation for general \(n\)-job, \(m\)-machine open shop problems with makespan as the criterion. Computational results of the illustrative example are reported using the proposed IP model to solve the open shop problem. A similar model with other criteria such as mean flow time can be formulated using same techniques with slight modification. Future research may be conducted to improve further the mixed BIP formulations.

Table 1. Model size of the IP model

<table>
<thead>
<tr>
<th>Number of Binary Variables</th>
<th>Number of Constraints</th>
<th>Number of Continuous Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>(m \sum_{i=1}^{n} N_i N_i) + (m \sum_{i=1}^{n} N_i)</td>
<td>(2m \sum_{i=1}^{n} N_i N_i + m^2 \sum_{i=1}^{n} N_i) + (m \sum_{i=1}^{n} N_i) + (m \sum_{i=1}^{n} N_i)</td>
<td>(m \sum_{i=1}^{n} N_i) + 1</td>
</tr>
</tbody>
</table>

Table 2. The machine required and the processing times for the illustrative example

<table>
<thead>
<tr>
<th>Jobs</th>
<th>Machine required</th>
<th>Processing times</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(M_1, M_2, M_3)</td>
<td>(p_{11} = 5, p_{12} = 6, p_{13} = 7)</td>
</tr>
<tr>
<td>2</td>
<td>(M_1, M_2)</td>
<td>(p_{21} = 3, p_{22} = 2)</td>
</tr>
<tr>
<td>3</td>
<td>(M_1, M_2, M_3)</td>
<td>(p_{31} = 4, p_{32} = 3, p_{33} = 5)</td>
</tr>
<tr>
<td>4</td>
<td>(M_2, M_3)</td>
<td>(p_{42} = 2, p_{43} = 5)</td>
</tr>
</tbody>
</table>

Remarks: \(N_1 = 3, N_2 = 2, N_3 = 3, \) and \(N_4 = 2\).

Table 3. Computational results for the illustrative example

<table>
<thead>
<tr>
<th>Number of binary variables</th>
<th>Number of constraints</th>
<th>Number of simplex iterations</th>
<th>Number of branches</th>
<th>CPU time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>141</td>
<td>353</td>
<td>20149</td>
<td>757</td>
<td>7</td>
</tr>
</tbody>
</table>

Figure 1. Optimal schedule for the illustrative example
Appendix A

A.1. LINGO’s modeling code for the open shop model of the illustrative example

Models:

! The LINGO code for the open shop model with $C_{max}$ as the criterion;

SETS:

JOB @FILE('C:\InData.ldt') / Ni;
MACHINE @FILE('C:\InData.ldt');
POSITION @FILE('C:\InData.ldt');
TIME(JOB, MACHINE): Pik, Rik;
JOBPOSMAC(JOB, POSITION, MACHINE):
Xijk, Sijk;
ZPRECEDES(MACHINE, JOB, POSITION, JOB, POSITION)| &2 #LT# &4: Zkij;
ENDSETS

DATA:

bM=9999;
Ni= @FILE('C:\InData.ldt');
Pik= @FILE('C:\InData.ldt');
Rik= @FILE('C:\InData.ldt');
ENDDATA

! The objective;

[OBJ] MIN = Cmax;

@FOR(JOB(I):
    @FOR(POSITION(J)| J #LE# Ni(I):
        @FOR(MACHINE(K):
            Xijk(I, J, K)<= Rik(I, K);
        );
    );

@FOR(JOB(I):
    @FOR(MACHINE(K):
        @SUM(POSITION(J)| J #LE# Ni(I):
            Xijk(I, J, K))= Rik(I, K);
    );

@FOR(JOB(I):
    @FOR(POSITION(J)| J #LE# Ni(I):
        @SUM(MACHINE(K): Xijk(I, J, K))= 1;
    );

@FOR(JOB(I):
    @FOR(POSITION(J)| J #LE# Ni(I):
        @FOR(MACHINE(K):
            Sijk(I, J, K)<= bM* Xijk(I, J, K);
        );
    );

@FOR(JOB(I):
    @FOR(POSITION(J)| J #LE# Ni(I)-1:
        @FOR(MACHINE(K):
            @FOR(MACHINE(KP)| KP #NE# K:
                Sijk(I, J, K)+ Pik(I, K)<= Sijk(I, J+1, KP)
                + bM* (2- Xijk(I, J, K)- Xijk(I, J+1, KP));
            );
        );

@FOR(JOB(I):
    @FOR(MACHINE(KP)| KP #NE# K:
        @FOR(MACHINE(IP)| IP #GT# I:
            @FOR(POSITION(J)| J #LE# Ni(IP):
                @FOR(POSITION(JP)| JP #LE# Ni(IP):
                    @FOR(MACHINE(K):
                    );
                );
            );

@FOR(JOB(I):
    @FOR(MACHINE(KP)| KP #NE# K:
        @FOR(MACHINE(IP)| IP #GT# I:
            @FOR(MACHINE(IP)| IP #GT# I:
                @FOR(POSITION(J)| J #LE# Ni(IP):
                    @FOR(POSITION(JP)| JP #LE# Ni(IP):
                        @FOR(MACHINE(K):
                        );
                    );
                );
            );
        );
    );
)
A.2. An input data file for using in the appendix

A.1.

! This is an input data file, called InData.ldt, and placed in the directory “C:\”;
! Define sets of JOB, MACHINE, and POSITION;
J1 J2 J3 J4~
M1 M2 M3~
P1 P2 P3~

References