A Mathematical Model for Scheduling a Batch Processing Machine with Multiple Incompatible Job Families, Non-identical Job dimensions, Non-identical Job sizes, Non-agreeable release times and due dates

To cite this article: M Ramasubramaniam and M Mathirajan 2013 *IOP Conf. Ser.: Mater. Sci. Eng.* **46** 012013

A Mathematical Model for Scheduling a Batch Processing Machine with Multiple Incompatible Job Families, Non-identical Job dimensions, Non-identical Job sizes, Non-agreeable release times and due dates

M. Ramasubramaniam[1], M. Mathirajan[2]

[1]Scool of Maritime Management, Indian Maritime University, Chennai, India.

[2]Principal Research Scientist, Department of Management Studies, Indian Institute of Science, Bangalore, India.

E-mail: rams@liba.edu

Abstract. The paper addresses the problem scheduling a batch processing machine with multiple incompatible job families, non-identical job dimensions, non-identical job sizes and non-agreeable release dates to minimize makespan. The research problem is solved by proposing a mixed integer programming model that appropriately takes into account the parameters considered in the problem. The proposed is validated using a numerical example. The experiment conducted show that the model can pose significant difficulties in solving the large scale instances. The paper concludes by giving the scope for future work and some alternative approaches one can use for solving these class of problems.

1. Introduction

Scheduling of parallel batch processing machines has recently attracted a great deal of interest among the researchers. Parallel batch processing machines, by definition, can simultaneously process a number of jobs as a batch where all the jobs have the same beginning and ending times. One could find these problems classified as bounded batch processing machines (BPM) scheduling problem because the size of batch is limited based on the capacity of the BPM.

This paper tries to address one such problem namely scheduling of a Heat-treatment furnace(HTF) that finds application in steel casting industry. A general characteristic of the steel casting industry is the product variety. Castings (or jobs) of any size, shape, quantities can be produced. Also, the materials used in this process are widely different that are selected based on the type of environment and application of the jobs which makes them incompatible from the point of view of processing. Thus, these job families are called multiple incompatible job families. Steel casting industry is a capital intensive industry and competition in this industry is high. So, there is a pressure on the steel casting industries to reduce the lead time to satisfy the customers in meeting the deadlines. One way of reducing the lead time for production is to schedule the bottleneck machines effectively.

From the viewpoint of throughput and utilization of the important and costly resources in the foundry manufacturing, it was felt that the process-controlled furnace operations for the melting and pouring operations as well as the heat-treatment furnace operations are critical for meeting the overall production schedules. The two furnace operations are batch processes that have distinctive constraints on job-mixes in addition to the usual capacity and technical constraints associated with any industrial processes. The benefits of effective scheduling of these batch processes include higher machine utilization, lower work-in-process (WIP) inventory, shorter cycle time and greater customer satisfaction. Although the melting and pouring operations are of central importance, the scheduling of heat-treatment furnaces is also of considerable importance. This is primarily due to the fact that the processing times of heat treatment furnaces are often longer compared to any other operations in the process. This paper attempts to propose methods to reduce the lead time of this process by addressing this bottleneck machine.

Accordingly, in this paper, we address a new research problem of scheduling a BP with Incompatible Job Families (MIJF), NIJS, NIJD and NARD to minimize makespan which is a surrogate measure for increasing throughput and reducing the lead time.

2. Research Problem

The scheduling problem defined in this paper can be denoted in three-field notation: Ŭ| ɓ | ɔ as 1 | p-batch, multiple incompatible job family, non-identical job sizes, non-identical job-dimensions, non-agreeable release times and due dates | Cmax problem. We propose a mathematical model for the research problem with the following assumptions:

- There are *n* jobs (*n* single castings) to be processed and these jobs are from a single job-family. Each job has a size s_i (in terms of weight) with dimensions l_i - length, w_i ï width and h_i - height
- All jobs must pass through the BP
- There is a single BP, which has capacity limit in terms of size S (in terms of weight) and dimensions *L* - Length, *W* - Width, and *H* - Height
- The BP is available continuously
- Orientation of jobs is not allowed
- All data of *S, L, W, H*, s_i , l_i, w_i, and h_i are deterministic and known a priori
- It is assumed that $s_i \leq S$; $l_i \leq L$; $w_i \leq W$; and $h_i \leq H$ for all jobs *i*
- Jobs are assumed to have non-agreeable release times and due dates. i.e .r_i Ò r_j does not imply d_i Ò d_j
- Processing time of job family is assumed to be independent of number of jobs in the batch and constant
- Preparation time, if any, is included in the processing time, and there is no setup time in switching from one batch to another
- Once processing of a batch is initiated, the BP cannot be interrupted and other jobs cannot be introduced into the BP until the current processing is completed.
- Machine breakdowns are not considered

The paper is organized as follows: In section 3, we review the related work. In section 4, we propose a mathematical model. In section 5, we validate the mathematical model with a numerical example. We conclude the paper with a summary and possible future research direction in Section 6.

3. Review of Related Works:

Semiconductor industries have attracted the interest of researchers the most [10] whereas, there are only a few studies close to the research problem under consideration. This could be primarily due to the economic & technological importance associated with semiconductor industries.

[5] addressed the problem of scheduling a BPM with incompatible job families with the objectives of minimizing makespan, total completion time and total weighed completion time. The authors propose a mathematical model and a set of heuristic algorithms and a meta heuristic for the different objectives.

Other than this study, other studies focus on meta-heuristic approaches for addressing the scheduling of BPMs [4], [11], [2].

Our studies differ from other studies in that we address an important aspect namely non-identical job dimension which may not be relevant in the semiconductor industry but important in steel casting industry. Also, our study considers non-agreeable release times which are not addressed in the literature. Finally, unlike studies which propose meta-heuristics, our study attempts to develop a mathematical model which would yield optimal solution for the research problem.

Apart from the semiconductor industry, there are only a few studies which focus on addressing the problem of scheduling heat treatment furnaces. Thus we only focus on these few studies which are very close to the research problem.

[8] examine the problem of scheduling multiple heat-treatment furnaces (HTF), of steel casting industry with non-identical job size and assuming identical job dimensions to maximize the utilization of the batch processors under dynamic job arrivals with agreeable release times and due dates. They take a three-stage approach to schedule the heat-treatment furnaces namely BPM selection, job family selection. They conduct a series of computational experiments in which the solution quality of heuristic algorithms is compared with estimated optimal solutions for each problem instances. They conclude that heuristic algorithms based on job family selection criteria which use: (a) the simple average priority of job, (b) the simple average size of job and (c) the weighted average size of job perform better with lesser computational effort.

[9] study the scheduling problem addressed earlier [8] with the objective of minimizing total weighted tardiness. They proposed four greedy heuristic algorithms which differ based on the job family selection. The detailed computational experiments conducted by them reveal that on an average, the heuristic algorithms which use job family selection criteria based on (a) cumulative due date and (b) weighted cumulative due date with job size as weight, perform better when compared with statistical estimation procedure.

Our study differs from these studies in that we consider non-identical job dimensions which take our study closer to reality.

4. Development of mathematical model for scheduling a BP with MIJF, NIJS and NIJD

The problem of scheduling a BPM with MIJF, NIJD, NIJS and NARD could be viewed as an extension of the three-dimensional bin-packing problem.

There are different approaches for solving this type of research problem like mathematical models, heuristic methods or metaheuristic methods. Since out of all this three, mathematical models provide optimal solutions for the problem this paper approaches the problem by proposing a mathematical model.

Mathematical models proposed by various researchers ([3], [12], [1]) for bin-packing problem consider NIJD characteristics explicitly and there is no restriction on the capacity of the bins in terms of job sizes considered in these studies.

Scheduling a BPM, however, is more complex. First, there are multiple incompatible job families which mean jobs from one family cannot be combined with jobs from another family. Also, even within a single job family, job (casting) sizes vary widely (from 10 g to 1000 kg), which puts a tremendous restriction on the number of jobs that can be accommodated in a batch to satisfy the capacity restrictions of BP in terms of size, in addition to dimensions of job. Accordingly, among the mathematical models available in the literature for bin-packing problem, the MILP model proposed by [1] is appropriately modified to address the problem of scheduling a BP with incompatibility limitation and capacity restrictions on both job dimensions and sizes. The proposed MILP model for scheduling a BP with a MIJF, NIJD, NIJS and NARD is as follows:

Indices

i, k	-	$1, 2, 3, \ldots, n$ Jobs
j	-	$1, 2, 3, \ldots, m$ Batches
p	-	$1, 2, 3, \ldots, q$ Families

Parameters

M	-	A large arbitrary number
(l_i, w_i, h_i)	-	Parameters indicating length , width and height of job i
s_i	-	Size of job i
(L, W, H)	-	Length, Width and Height of BP
S	-	Size capacity of BP
t_i	-	Processing time of job i (= processing time of job family)
r_i	-	Release time of job i

Continuous Decision Variables

(a_i, b_i, c_i)	-	Coordinates of the front left bottom corner of job i. It is assumed that the origin is fixed at Front Left Bottom corner of the BP
(a_k, b_k, c_k)	-	Coordinates of the front left bottom corner of job k. It is assumed that the origin is fixed at Front Left Bottom corner of the BP
C_{max}	-	Completion time of last batch
c_i	-	Completion time of job i
R^j	-	Release time of batch j
P^j	-	Processing time of batch j
C^j	-	Completion time of batch j

Binary Decision Variables

$$x_{ipj} - \begin{cases} 1 \text{ if job } i \text{ of family } p \text{ is assigned to batch } j \\ 0 \text{ otherwise} \end{cases}$$

$$y_{pj} - \begin{cases} 1 \text{ if family } p \text{ is assigned to batch } j \\ 0\text{otherwise} \end{cases}$$

$$left_{ik} = \begin{cases} 1 \text{ if job } i \text{ is on left side of job } k \ \forall\ i < k; \\ 0 \text{ otherwise} \end{cases}$$

$$right_{ik} = \begin{cases} 1 \text{ if job } i \text{ is on right side of job } k \ \forall\ i < k; \\ 0 \text{ otherwise} \end{cases}$$

$$behind_{ik} = \begin{cases} 1 \text{ if job } i \text{ is behind job } k \ \forall\ i < k; \\ 0 \text{ otherwise} \end{cases}$$

$$front_{ik} = \begin{cases} 1 \text{ if job } i \text{ is in front of job } k \ \forall\ i < k; \\ 0 \text{ otherwise} \end{cases}$$

$$below_{ik} = \begin{cases} 1 \text{ if job } i \text{ is below job } k \ \forall\ i < k; \\ 0 \text{ otherwise} \end{cases}$$

$$above_{ik} = \begin{cases} 1 \text{ if job } i \text{ is above job } k \ \forall\ i < k; \\ 0 \text{ otherwise} \end{cases}$$

Min C_{max}

Subject to:

$$\sum_{j=1}^{m}\sum_{p=1}^{q} x_{ipj} = 1 \qquad \forall i \qquad (1)$$

$$\sum_{p=1}^{q} y_{pj} \le 1 \qquad \forall j \qquad (2)$$

$$a_i + l_i \le L \qquad \forall i \qquad (3)$$

$$b_i + w_i \le W \qquad \forall i \qquad (4)$$

$$c_i + h_i \le H \qquad \forall i \qquad (5)$$

$$x_{ipj} \le y_{pj} \qquad \forall i, p, j \qquad (6)$$

$$\sum_{i=1}^{n} s_i x_{ipj} \le S \qquad \forall p, j \qquad (7)$$

$$a_i + l_i \le a_k + (1 - left_{ik})M \qquad \forall i,k,i<k \qquad (8)$$

$$a_k + l_k \le a_i + (1 - right_{ik})M \qquad \forall i,k,i<k \qquad (9)$$

$$b_i + w_i \le b_k + (1 - behind_{ik})M \qquad \forall i,k,i<k \qquad (10)$$

$$b_k + w_k \le b_i + (1 - front_{ik})M \qquad \forall i,k,i<k \qquad (11)$$

$$c_i + h_i \le c_k + (1 - below_{ik})M \qquad \forall i,k,i<k \qquad (12)$$

$$c_k + h_k \le c_i + (1 - above_{ik})M \qquad \forall i,k,i<k \qquad (13)$$

$$left_{ik} + right_{ik} + behind_{ik} + front_{ik} +$$
$$below_{ik} + above_{ik} \ge x_{ipj} + x_{kpj} - 1 \qquad \forall i, k, p, j,\ i<k \qquad (14)$$

$$left_{ik} + right_{ik} + behind_{ik} + front_{ik} + below_{ik} + above_{ik} \le 1 + x_{ipj} - x_{kpj} \quad \forall i,k,p,j,\ i<k \qquad (15)$$

$$P^j \ge t_i x_{ipj} \quad \forall i,p,j \qquad (16)$$

$$R^j \ge r_i x_{ipj} \quad \forall i,p,j \qquad (17)$$

$$C^j \ge R^j + P^j \quad \forall j \qquad (18)$$

$$R^{j+1} \ge C^j \quad \forall j \text{ and } j<n \qquad (19)$$

$$c_i \ge C^j - M(1 - x_{ipj}) \quad \forall i,p,j \qquad (20)$$

$$C_{max} = C^n \qquad (21)$$

Binary restriction:
$left_{ik}, right_{ik}, behind_{ik}, front_{ik}, below_{ik}, above_{ik}, x_{ipj}, y_{ipj} = 0 \text{ or } 1$

Non-negativity constraints:
$c_i, P^j, R^j, C^j \ge 0$

The objective of the model is to minimize the completion time of the last batch. Constraint set (1) ensures that each job of a job family is assigned only once to the batch. Constraint set (2) stipulates that at most only one family can be assigned to a batch. Constraints (3)- (5) ensure that all the jobs assigned to a batch fit within the physical dimensions of the BP.Constraint (6) ensures that a job from a job family is assigned to a batch if and only if that job family is assigned to that batch.Constraint (7) ensures that the weight capacity of BP is not exceeded. Constraints (8ï 13) ensure that jobs do not overlap with one another. This check for overlap is necessary only if a pair of jobs is placed in the same BP. This is taken care of by Constraints (14ï 15).

Constraint (16) defines the processing time of a batch as greater than or equal to the processing time of each job (= processing time of the job family) in that batch. Constraint (17) ensures that release time of a batch is the maximum of the release times of all jobs in the batch. Calculation of Completion time of the jobs is taken care by (18). Ensuring those jobs have their release time atleast equal to the completion of time of previous batch is by (19). Constraint (20) ensures that completion time of job is atleast equal to the completion time of batch, if and only if the job is in that batch. Constraint (21) is completion time of the last batch.

The number of binary variables, in the proposed MILP model is $nm + fm + 3n(n-1)$ and the model has $3n(n-1)+2n(n-1)m+4nfm+5n+3m+fm+1$ number of constraints, where f is the number of job families, n number of jobs, m number of constraints.

5. Validation of the Proposed MILP model

To validate the proposed MILP model, we develop a numerical example assuming one BP with two job families having n = 10 jobs with the processing time of the first job family as 15 h and second as 12 h. The Capacity of the BP in terms of size of the BP = 1000 kg; and in terms of dimensions: L = 1500 mm; W = 950 mm; H = 900 mm. In addition, the size, dimensions, and processing time of the 10 jobs are given in Table 1.

The optimal solution for the proposed MILP model is obtained using LINGO. The appropriate LINGO set code to generate is given in Appendix 1. Using this set code, an ILP model is generated for the numerical example presented in Table 1 and solved.

Table 1: A numerical example for research problem

Parameter	Job = *j* =									
	1	2	3	4	5	6	7	8	9	10
Size$_j$(kg*)	275	115	189	250	133	160	157	222	298	291
length$_j$(mm$^\#$)	466	223	321	164	165	251	359	474	253	155
width$_j$(mm)	105	188	142	258	292	149	301	190	202	197
height$_j$(mm)	245	256	204	213	103	213	281	150	236	290
Release time$_j$(h)	3	3	4	6	10	13	15	15	19	19
Due date$_j$(h)	20	35	34	50	36	61	47	45	56	70
Job Family	1	1	1	1	1	2	2	2	2	2

* Kilogram, $^\#$ Millimetre

Table 2: Optimal solution obtained using the proposed MILP model

Batch	Optimal Solution				
	Jobs in the batch	Family	Release time of the batch (h)	Processing time of the batch (h)	Completion time of the Batch (h)
1	1,2,3,4,5	1	10	15	25
2	6, 7, 8	2	25	12	37
3	9, 10	2	37	12	49
$\boldsymbol{C_{max}}$ **= 49 h**; Computational Time = 00: 19:01 in P4 with 2.40 GHz, 1 GB of RAM; Number of Iterations = 2161072					

The optimal solution obtained for the numerical example is presented in Table 2 which gives the batch wise job details, job family details, release time, processing time and the completion time of each batch.

6. Summary and Future Work

Based on our observation from steel casting industry, we addressed an important research problem of scheduling a HTF in steel casting industry. This problem has not been given due attention in the literature. This led us to development of a mathematical model for scheduling the HTF with MIJF, NIJD, NIJS and NARD with the objective of minimizing makespan. The proposed mathematical model is validated using a numerical example by generating and solving the model in LINGO.

It could be noted that even for a small scale instance such as the one used for solving the mathematical model in this paper, the computational time is reasonably high. In general, it is widely accepted that integer programming models are difficult to solve to optimality, especially for large instances. Thus, it is possible that larger instances can pose significant difficulties in solving the model proposed in this paper. Considering these things in mind, one could first test the model extensively on large scale instances to estimate the computational time for this class of models. Secondly, one could think of reformulating this model to make it solvable in reasonable time limit.

As an alternative, the problem can be approached by using heuristic methods that could result in near optimal solutions very quickly. Also, meta-heuristic methods like genetic algorithms, tabu search etc. could be developed for solving the problem which can provide better solutions than a simple heuristic method at the expense of more computational effort.

7. References

[1] Chen C S Lee s and Shen Q S 1995 An Analytical Model for the Container Loading Problem *European Journal of Operational Research* 80(1) 68-76
[2] Damodaran P and Velez-Gallego M.C 2012 A simulated annealing algorithm to minimize makespan of parallel batch processing machines with unequal job ready times *Expert Systems with Applications* 39 1451-1458
[3] Fasano G 2008 MIP based heuristic for non standard 3D-bin packing problems *4OR: A Quarterly Journal of Operations Research* 6(3) 291ï 310
[4] KashanA Karimi B 2007Scheduling a single batch-processing machine with arbitrary job sizes and incompatible job families: an ant colony framework*Journal of Operational Research Society* 59 1269-1280
[5] Koh S G Koo P H Kim D C and Hur W S 2005 Scheduling a single batch processing machine with arbitrary job sizes and incompatible job families *International Journal Of Production Economics*98(1) 81-96
[6] Martello S Pisinger D and Vigo D2000 The Three dimensional Bin Packing Problem *Operations Research*48(2) 256-267
[7] Mathirajan M (2002) Heuristic scheduling algorithms for parallel heterogenous batch processors *PhD Thesis* Department of Management Studies Indian Institute of Science
[8] Mathirajan M Sivakumar A I and Chandru V (2004a) Scheduling algorithms for heterogeneous batch processors with incompatible job-families *Journal Of Intelligent Manufacturing* 15(6) 787-803
[9] Mathirajan M and Sivakumar A I (2006a) Minimizing total weighted tardiness on heterogeneous batch processing machines with incompatible job families *International Journal Of Advanced Manufacturing Technology*28(9) 1038-1047
[10] Mathirajan M and Sivakumar A I (2006b) A literature review classification and simple meta-analysis on scheduling of batch processors in semiconductor *International Journal Of Advanced Manufacturing Technology*29(9-10) 990-1001
[11] Nong Q Ng C T and Cheng T C E (2008) The bounded single-machine parallel-batching scheduling problem with family jobs and release dates to minimize makespan *Operations*

*Research Letters*36(1) 61-66

[12] Padberg M (2000) Packing small boxes in a big box *Mathematical Methods of Operations Research* Vol 52 pp1ï 21

[13] Pinedo M L (1995) Scheduling: Theory Algorithms and Systems Prentice Hall

[14] Ramasubramaniam M (2008) óBatch Processor Scheduling ï A Class of Problems in Steel Casting Industriesô *PhD thesis* Department of Management Studies Indian Institute of Science Bangalore India

Appendix 1

```
SETS :
  FAMILIES:STARTIME;
  JOBS:P,Q,HT,WEIGHT,X,Y,Z,RELTIME,DUEDATE,PROCTIME,COMPTIMEJOB;

FNCS:CTIME,FNCHEIGHT,FNCCAPACITY,FNCLENGTH,FNCWIDTH,PROCTIMEBAT,RELTIMEBAT,
COMPTIMEBAT;
ORIENTATION(JOBS,JOBS)|&1 #LT# &2:LEFT,RIGHT,BEHIND,INFRONT,BELOW,ABOVE;
FAMTOFNCS(FAMILIES,FNCS):FTOB;
JOBTOFAMTOFNC(JOBS,FAMILIES,FNCS):JTOFTOB;


 ENDSETS
DATA :

 FAMILIES = F1,F2  ;

JOBS,FNCS,P,Q,HT,WEIGHT,FNCLENGTH,FNCWIDTH,PROCTIME,RELTIME,DUEDATE,FNCHEIGHT
,FNCCAPACITY =  @OLE ('C:\\3DINCOMPCH2\\12jobs.XLS');
 ENDDATA

 BIGM=1500;

 MAXJOBS =  @SIZE (FNCS);

!OBJECTIVE TO MINIMIZE MAKESPAN;
MIN  = CMAX;
!SUBJECT TO CONSTRAINTS:
!ASSIGN EACH JOB TO ONE BATCH;
 @FOR (JOBS(I):
   @SUM (FAMILIES(FAM):
 @SUM (FNCS(B): JTOFTOB(I,FAM,B)))=1);

!AT MOST ONE JOB FAMILY CAN BE ASSINGED TO A BATCH;
 @FOR (FNCS(B):
  @SUM (FAMILIES(FAM): FTOB(FAM,B))<=1);

 ! A JOB FROM JOB FAMILY IS ASSINGED TO A BTATCH ONLY IF
 THE JOB FAMILY IS ASSIGNED TO A BATCH;
 @FOR (JOBS(I):
   @FOR (FAMILIES(FAM):
    @FOR (FNCS(B): JTOFTOB(I,FAM,B) <= FTOB(FAM,B))));

!SIZE CONSTRAINT;
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(J):
 @SUM (JOBS(I): WEIGHT(I)*JTOFTOB(I,FAM,J)) <= 1000));


!ORIENTATION OF JOBS CONSTRAINT;
 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: X(I)+ P(I) <= X(K)+(1-LEFT(I,K))*BIGM) );


 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: X(K)+ P(K) <= X(I)+(1-RIGHT(I,K))*BIGM ));
 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Y(I)+ Q(I)<= Y(K)+(1-BEHIND(I,K))*BIGM ));

  @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Y(K)+Q(K) <= Y(I)+(1-INFRONT(I,K))*BIGM ));

 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Z(I)+ HT(I) <= Z(K)+(1-BELOW(I,K))*BIGM ));

 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Z(K)+ HT(K) <= Z(I)+(1-ABOVE(I,K))*BIGM ));

 @FOR (FAMILIES(FAM):
 @FOR (FNCS(J):
 @FOR (JOBS(K):
   @FOR (JOBS(I)|I #LT# K:
LEFT(I,K)+RIGHT(I,K)+BEHIND(I,K)+INFRONT(I,K)+BELOW(I,K)+ABOVE(I,K)
>= JTOFTOB(I,FAM,J)+JTOFTOB(K,FAM,J)-1))));
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(J):
 @FOR (JOBS(K):
   @FOR (JOBS(I)|I #LT# K:
LEFT(I,K)+RIGHT(I,K)+BEHIND(I,K)+INFRONT(I,K)+BELOW(I,K)+ABOVE(I,K)
<= 1+JTOFTOB(I,FAM,J)-JTOFTOB(K,FAM,J)))));

!LENGTH CONSTRAINT;
 @FOR (JOBS(I): X(I)+ P(I)<= 1500);

!WIDTH CONSTRAINT;
 @FOR (JOBS(I): Y(I)+ Q(I)<= 950);

!HEIGHT CONSTRAINT;
 @FOR (JOBS(I): Z(I)+ HT(I)<= 900);

!CALCULATE PROCESSING TIME OF A BATCH;
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(B):
  @FOR (JOBS(J):   PROCTIMEBAT(B) >= PROCTIME(J)*  JTOFTOB(J,FAM,B))));


!RELEASE TIME OF A BATCH IS MAXIMUM OF ALL RELEASE TIMES OF JOBS IN A BATCH;
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(B):
  @FOR (JOBS(J):   RELTIMEBAT(B) >= RELTIME(J)*  JTOFTOB(J,FAM,B))));


!CALCULATE COMPLETION TIME OF A BATCH;
 @FOR (FNCS(B):  COMPTIMEBAT(B) = RELTIMEBAT(B)+PROCTIMEBAT(B)    );


!RELEASE TIME OF A BATCH SHOULD BE LESS THAN THE COMPLETION TIME OF
 THE PREVIOUS BATCH;
 @FOR (FNCS(B)|B #LT#  @SIZE (FNCS):  RELTIMEBAT(B+1) >= COMPTIMEBAT(B) );

!CALCULATE COMPLETION TIME OF A JOB;
```

```
  @FOR (FAMILIES(FAM):
  @FOR (FNCS(B):
   @FOR (JOBS(J):   COMPTIMEJOB(J) >= COMPTIMEBAT(B)-1000*(1-JTOFTOB(J,FAM,B)))));

!COMPLETION TIME OF A JOB IS LESS THAN ITS DUE DATE;
  !@FOR(JOBS(J): COMPTIMEJOB(J) <= DUEDATE(J));

!CALCULATE CMAX;
  @FOR (FNCS(B)|B #EQ#  @SIZE (FNCS): CMAX = COMPTIMEBAT(B)   );


!DECISION VARIABLES;
  @FOR (JOBS(I)|I  #GE# 1 #AND# I #LE# 5:
     @FOR (FAMILIES(FAM)|FAM #EQ# 1:
  @for (FNCS(B):  @bin (JTOFTOB(I,FAM,B)))));

  @FOR (JOBS(I)|I  #GE# 6 :
     @FOR (FAMILIES(FAM)|FAM #EQ# SETS :
   FAMILIES:STARTIME;
   JOBS:P,Q,HT,WEIGHT,X,Y,Z,RELTIME,DUEDATE,PROCTIME,COMPTIMEJOB;

FNCS:CTIME,FNCHEIGHT,FNCCAPACITY,FNCLENGTH,FNCWIDTH,PROCTIMEBAT,RELTIMEBAT,
COMPTIMEBAT;
ORIENTATION(JOBS,JOBS)|&1 #LT# &2:LEFT,RIGHT,BEHIND,INFRONT,BELOW,ABOVE;
FAMTOFNCS(FAMILIES,FNCS):FTOB;
JOBTOFAMTOFNC(JOBS,FAMILIES,FNCS):JTOFTOB;

  ENDSETS
DATA :

 FAMILIES = F1,F2  ;


JOBS,FNCS,P,Q,HT,WEIGHT,FNCLENGTH,FNCWIDTH,PROCTIME,RELTIME,DUEDATE,FNCHEIGHT
,FNCCAPACITY =  @OLE ('C:\\3DINCOMPCH2\\12jobs.XLS');
  ENDDATA

 BIGM=1500;

 MAXJOBS =  @SIZE (FNCS);

!OBJECTIVE TO MINIMIZE MAKESPAN;
MIN  = CMAX;
!SUBJECT TO CONSTRAINTS:
!ASSIGN EACH JOB TO ONE BATCH;
  @FOR (JOBS(I):
     @SUM (FAMILIES(FAM):
  @SUM (FNCS(B): JTOFTOB(I,FAM,B)))=1);

!AT MOST ONE JOB FAMILY CAN BE ASSINGED TO A BATCH;
  @FOR (FNCS(B):
    @SUM (FAMILIES(FAM): FTOB(FAM,B))<=1);

  ! A JOB FROM JOB FAMILY IS ASSINGED TO A BTATCH ONLY IF
 THE JOB FAMILY IS ASSIGNED TO A BATCH;
  @FOR (JOBS(I):
   @FOR (FAMILIES(FAM):
    @FOR (FNCS(B): JTOFTOB(I,FAM,B) <= FTOB(FAM,B))));

!SIZE CONSTRAINT;
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(J):
 @SUM (JOBS(I): WEIGHT(I)*JTOFTOB(I,FAM,J)) <= 1000));


!ORIENTATION OF JOBS CONSTRAINT;
 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: X(I)+ P(I) <= X(K)+(1-LEFT(I,K))*BIGM) );


 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: X(K)+ P(K) <= X(I)+(1-RIGHT(I,K))*BIGM ));

 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Y(I)+ Q(I)<= Y(K)+(1-BEHIND(I,K))*BIGM ));


 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Y(K)+Q(K) <= Y(I)+(1-INFRONT(I,K))*BIGM ));

 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Z(I)+ HT(I) <= Z(K)+(1-BELOW(I,K))*BIGM ));

 @FOR (JOBS(K):
  @FOR (JOBS(I)|I #LT# K: Z(K)+ HT(K) <= Z(I)+(1-ABOVE(I,K))*BIGM ));


 @FOR (FAMILIES(FAM):
 @FOR (FNCS(J):
 @FOR (JOBS(K):
   @FOR (JOBS(I)|I #LT# K:
LEFT(I,K)+RIGHT(I,K)+BEHIND(I,K)+INFRONT(I,K)+BELOW(I,K)+ABOVE(I,K)
>= JTOFTOB(I,FAM,J)+JTOFTOB(K,FAM,J)-1))));
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(J):
 @FOR (JOBS(K):
   @FOR (JOBS(I)|I #LT# K:
LEFT(I,K)+RIGHT(I,K)+BEHIND(I,K)+INFRONT(I,K)+BELOW(I,K)+ABOVE(I,K)
<= 1+JTOFTOB(I,FAM,J)-JTOFTOB(K,FAM,J)))));

!LENGTH CONSTRAINT;
 @FOR (JOBS(I): X(I)+ P(I)<= 1500);

!WIDTH CONSTRAINT;
 @FOR (JOBS(I): Y(I)+ Q(I)<= 950);

!HEIGHT CONSTRAINT;
 @FOR (JOBS(I): Z(I)+ HT(I)<= 900);


!CALCULATE PROCESSING TIME OF A BATCH;
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(B):
  @FOR (JOBS(J):   PROCTIMEBAT(B) >= PROCTIME(J)*  JTOFTOB(J,FAM,B))));


!RELEASE TIME OF A BATCH IS MAXIMUM OF ALL RELEASE TIMES OF JOBS IN A BATCH;
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(B):
  @FOR (JOBS(J):   RELTIMEBAT(B) >= RELTIME(J)*  JTOFTOB(J,FAM,B))));


!CALCULATE COMPLETION TIME OF A BATCH;
 @FOR (FNCS(B):  COMPTIMEBAT(B) = RELTIMEBAT(B)+PROCTIMEBAT(B)   );


!RELEASE TIME OF A BATCH SHOULD BE LESS THAN THE COMPLETION TIME OF
 THE PREVIOUS BATCH;
 @FOR (FNCS(B)|B #LT#  @SIZE (FNCS):  RELTIMEBAT(B+1) >= COMPTIMEBAT(B) );

!CALCULATE COMPLETION TIME OF A JOB;
 @FOR (FAMILIES(FAM):
 @FOR (FNCS(B):
  @FOR (JOBS(J):   COMPTIMEJOB(J) >= COMPTIMEBAT(B)-1000*(1-JTOFTOB(J,FAM,B)))));

!COMPLETION TIME OF A JOB IS LESS THAN ITS DUE DATE;
 @FOR (JOBS(J): COMPTIMEJOB(J) <= DUEDATE(J));

!CALCULATE CMAX;
 @FOR (FNCS(B)|B #EQ#  @SIZE (FNCS): CMAX = COMPTIMEBAT(B)   );


!DECISION VARIABLES;
 @FOR (JOBS(I)|I  #GE# 1 #AND# I #LE# 5:
   @FOR (FAMILIES(FAM)|FAM #EQ# 1:
 @for (FNCS(B):  @bin (JTOFTOB(I,FAM,B)))));

 @FOR (JOBS(I)|I  #GE# 6 :
   @FOR (FAMILIES(FAM)|FAM #EQ# 1:
 @for (FNCS(B): JTOFTOB(I,FAM,B)=0)));

 @FOR (JOBS(I)|I  #GE# 6 :
   @FOR (FAMILIES(FAM)|FAM #EQ# 2:
 @for (FNCS(B):  @bin (JTOFTOB(I,FAM,B)))));

 @FOR (JOBS(I)|I  #LE# 5 :
   @FOR (FAMILIES(FAM)|FAM #EQ# 2:
 @for (FNCS(B): JTOFTOB(I,FAM,B)=0)));

 @FOR (FAMTOFNCS: @BIN (FTOB));
 @FOR (ORIENTATION: @BIN (LEFT));
 @FOR (ORIENTATION: @BIN (RIGHT));
 @FOR (ORIENTATION: @BIN (BEHIND));
 @FOR (ORIENTATION: @BIN (INFRONT));
 @FOR (ORIENTATION: @BIN (BELOW));
 @FOR (ORIENTATION: @BIN (ABOVE));

# 1:
```

```
@for (FNCS(B): JTOFTOB(I,FAM,B)=0)));

@FOR (JOBS(I)|I  #GE# 6 :
  @FOR (FAMILIES(FAM)|FAM #EQ# 2:
@for (FNCS(B):  @bin (JTOFTOB(I,FAM,B)))));

@FOR (JOBS(I)|I  #LE# 5 :
  @FOR (FAMILIES(FAM)|FAM #EQ# 2:
@for (FNCS(B): JTOFTOB(I,FAM,B)=0)));

@FOR (FAMTOFNCS: @BIN (FTOB));
@FOR (ORIENTATION: @BIN (LEFT));
@FOR (ORIENTATION: @BIN (RIGHT));
@FOR (ORIENTATION: @BIN (BEHIND));
@FOR (ORIENTATION: @BIN (INFRONT));
@FOR (ORIENTATION: @BIN (BELOW));
@FOR (ORIENTATION: @BIN (ABOVE));
```