Mathematical Formulation to Minimize Makespan in a Job Shop with a Batch Processing Machine

Purushothaman Damodaran, Ph.D.
Department of Industrial and Systems Engineering
Northern Illinois University, DeKalb, IL 60115, USA, pdamodaran@niu.edu

Miguel Rojas-Santiago, Ph.D.
Department of Industrial Engineering
Universidad del Norte, Barranquilla, Atlántico, Colombia, miguelrojas@uninorte.edu.co

Abstract

We consider a scheduling problem commonly observed in the metal working industry. We analyze a job shop which is equipped with one batch processing machine (BPM), and several unit-capacity machines. Given a set of jobs, their process routes, processing requirements, and size, the objective is to schedule the jobs such that the makespan is minimized. The BPM can process a batch of jobs as long as the total batch size does not exceed the machine capacity. The batch processing time is equal to the longest processing job in the batch. The problem under study can be represented as *Jm|batch|Cmax*, using the three field notation. If no batches were to be formed, the scheduling problem under study reduces to *Jm||Cmax*, which is known to be NP-hard. A network representation of the problem using disjunctive and conjunctive arcs, and a Mixed-Integer Linear Programming (MILP) are proposed to solve the problem. An experimental study was conducted to evaluate the proposed mathematical formulation.

Keywords: Scheduling, Job shop, Batch processing, MILP, Makespan.

1.Introduction.

Most of the research on job shop systems has been focused on the classical *Jm||Cmax*. However, in this paper, we propose a similar scheduling problem by combining *Jm||Cmax* with the problem of a single batch-processing machine (BPM), $1|\ r_j, batch|Cmax$. This combination, namely *Jm|batch|Cmax*, is a more realistic representation of some practical scheduling problems.

This research is motivated by a practical application observed at the real world, whereas many fabrication facilities have not only unit-capacity machines, but also batching machines. This environment is proper for semiconductor industries, metal working facilities, electronics companies, and so on. For example, in metal working companies, the fabrication of boilers, pressure vessels, heat exchangers, super heaters, and economizers requires machines like press brake or bending machines, cutting equipments, or a furnace to reheat the finished items. In the classical job shop problem, this equipment can be modeled only as unit-capacity machines.

2. Problem description.

Formally, the problem can be described as follows: We are given the set N of jobs; the set M of unit-capacity machines and batching machines, with the subset L of batching processing machines. Each job $j \in N$ is described by its processing time p_{ij} on the machine $i \in M$, or by p_{kj} and s_j when the job $j \in N$ requires to be processed on the machine $k \in L$. In this case, s_j represents the size of the job j, which is known too. However, for this problem, it is required to find first a set B of batches for the set of jobs ($j \in N$) that need to be processed by the batch processing machine $k \in L$, given that the processing time of batch $b \in B$ is defined as $p_{kb}=max\{p_{kj} \mid j \in b\}$. Moreover, the BPM can process at most D jobs simultaneously, and can process a batch only if the total size of the jobs in the batch does not exceed its capacity (S). In summary, the objective of this research is to find a set B of batches for every machine $k \in L$, and to schedule every machine $i \in M$ such that the makespan is minimized.

An instance of the problem discussed above is shown in Table 1 (Pinedo, 2002), which presents the data of a three-job and four-machine problem; with machine 1 as BPM, D=3, and S=10. For this instance, for example, a feasible batch formation is to assign jobs1 and 2 to batch1, and assign job3 to batch2. Figure 1 exhibits a graphical representation of this problem, including the BPM and the two batches described before. For simplicity, the disjunctives arcs (Balas, 1969) only show the relationship among the batch formation and the rest of jobs.

Table 1: Data for a Three-Job and Four-Machine Problem

Jobs	Machine Sequence	Processing Time	s_j
1	1→2→3	$\boldsymbol{p_{11}}$= 9, $\boldsymbol{p_{21}}$= 8, $\boldsymbol{p_{31}}$= 4	3
2	1→2→4	$\boldsymbol{p_{12}}$= 5, $\boldsymbol{p_{22}}$= 6, $\boldsymbol{p_{42}}$= 3	7
3	3→1→2	$\boldsymbol{P_{33}}$= 10, $\boldsymbol{p_{13}}$= 4, $\boldsymbol{p_{23}}$= 9	5

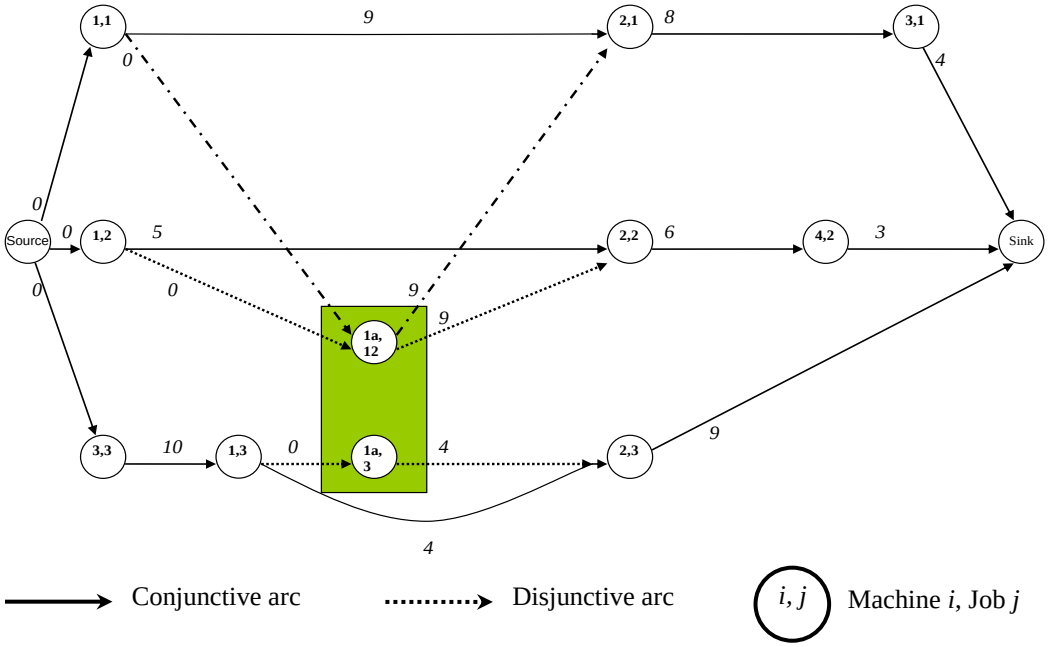

Figure 1: A Three-Job and Four-Machine Problem Representation

3. Previous related works.

Most research on job shop scheduling to date has focused on unit-capacity machines, but job shop with batch processing machines have begun to be studied, until now. Unit-capacity machines can process one job at a time,

while batch-processing machines can process a number of jobs simultaneously as a batch, with all jobs in a batch starting and ending processing simultaneously.

Some operations research analysts and engineers call those systems as complex job shop (Mason and Fowler, 2000), because they are characterized by different types of workcenters; some consisting of multiple identical machines with one or more batch processing machines. An example of a complex job shop is a wafer fabrication facility, where integrated circuits are fabricated on silicon wafers using a variety of chemical and thermal processes (Mason and et al., 2002).

The majority of research of the complex job shop scheduling approach has been carried out in the semiconductor industry, and has been classified by Gupta and Sivakumar (2006) into four categories: dispatching rules, analytical methods, heuristics, and artificial intelligence techniques. More specifically, decomposition methods based on the shifting bottleneck heuristic (Ovacik and Uzsoy, 1997) and lagrangian relaxation with dynamic programming are the most popular techniques applied to this kind of manufacturing environment.

4. The mixed-integer linear programming

The *Jm|batch|Cmax* problem under study can be formulated as a mixed integer problem with the following notation:

Sets:

B	Set of batches
I	Set of machines
J	Set of jobs
O_j	Set of all operations arranged in sequential order that are required by job j.

Parameters:

n	Total number of jobs
m	Total number of machines
BPM	Batch processing machine number
M	A big number.
p_{ij}	Processing time of job j on machine i
s_j	Size of job j
S	Capacity of the BPM
D	Maximum number of jobs allowed in any batch.

Decision Variables:

$Cmax$	Makespan
PB_b	Processing time of batch b
s_{ij}	Starting time of job j on machine i
SB_b	Starting time of batch b.

$$X_{jb} = \begin{cases} 1, \text{ if job } j \text{ is assigned to batch } b \\ 0, \text{ otherwise} \end{cases}$$

$$Z_{ili} = \begin{cases} 1, \text{ if job } j \text{ and job } l \text{ are assigned to machine } i \\ 0, \text{ otherwise.} \end{cases}$$

The proposed model to solve the problem under study is:

Minimize $Cmax$ (4.1)

Subject to:

$$\sum_{b \in B} X_{jb} = 1, \quad \forall j \in J \tag{4.2}$$

$$\sum_{j \in J} s_j * X_{jb} \leq S, \quad \forall b \in B \tag{4.3}$$

$$\sum_{j \in J} X_{jb} \leq D, \quad \forall b \in B \tag{4.4}$$

$$PB_b \geq p_{ij} * X_{jb}, \quad \forall b \in B, i = BPM, \forall j \in J \tag{4.5}$$

$$SB_b \geq SB_{b-1} + PB_{b-1}, \quad \forall b \in B \wedge b > 1 \tag{4.6}$$

$$s_{ij} \leq SB_b + M*(1 - X_{jb}), \quad i = BPM, \forall j \in J, \forall b \in B \tag{4.7}$$

$$SB_b \geq s_{ij} + p_{ij} - M*(1 - X_{jb}), \quad \forall j \in J, \forall b \in B, \forall i \rightarrow k \in Oj \wedge k = BPM \tag{4.8}$$

$$s_{kj} \geq SB_b + PB_b - M*(1 - X_{jb}), \quad \forall i \rightarrow k \in Oj \wedge i = BPM, \forall j \in J, \forall b \in B \tag{4.9}$$

$$\sum_{j \in J} s_{ij} = \sum_{b \in B} SB_b, \quad i = BPM \tag{4.10}$$

$$s_{kj} \geq p_{ij} + s_{ij}, \quad \forall i \rightarrow k \in Oj, \forall j \in J \tag{4.11}$$

$$s_{ij} \leq s_{il} + M*(1 - Z_{jli}), \quad i = BPM, \forall j \wedge \forall l \in J \wedge j \neq l \tag{4.12}$$

$$s_{ij} + p_{ij} \leq s_{il} + M*(1 - Z_{jli}), \quad \forall i \in I \wedge i \neq BPM, \forall j \wedge \forall l \in J \wedge j \neq l \tag{4.13}$$

$$Z_{jli} + Z_{lji} = 1, \quad \forall j \wedge \forall l \in J \wedge j \neq l, \forall i \in I \tag{4.14}$$

$$s_{ij} + p_{ij} \leq Cmax, \quad \forall i \in I, \forall j \in J \tag{4.15}$$

$$SB_b + PB_b \leq Cmax, \quad \forall b \in B \tag{4.16}$$

$$Cmax \geq 0 \tag{4.17}$$

$$PB_b \geq 0, \quad \forall b \in B \tag{4.18}$$

$$s_{ij} \geq 0, \quad \forall i \in I, \forall j \in J \tag{4.19}$$

$$SB_b \geq 0, \quad \forall b \in B \tag{4.20}$$

$$X_{jb} \in \{0,1\}, \quad \forall j \in J, \forall b \in B \tag{4.21}$$

$$Z_{jli} \in \{0,1\}, \quad \forall i \in I, \forall j \wedge \forall l \in J \wedge j \neq l \tag{4.22}$$

The objective function (4.1) is to minimize the makespan. Constraint (4.2) ensures that each job *j* is assigned to exactly one batch. Constraint (4.3) establishes that the total size of the jobs in a batch *b* does not exceed the capacity (*S*) of the batch processing machine. Constraint (4.4) takes into account that the total jobs in a batch *b* do not exceed the maximum *D* allowed for any batch. Constraint (4.5) determines the processing time for batch *b* (i.e., the longest processing time of all the jobs that belong to that batch). Constraint (4.6) ensures that the starting time of batch *b* is at least equal to the completion time of batch *b-1*. Constraint (4.7) determines that the starting time of batch *b* with job *j* begins later or at the same starting time of job *j* in the BPM. Constraint (4.8) ensures that the starting time of batch *b* with job *j* begins when the previous operations in machine *i* of job *j* are completed. Constraint (4.9) establishes that once the batch *b* with the job *j* is finished, the next operation of job *j* can be begun. In constraint (4.10), the summation of the starting time of all the jobs in the BPM is equal to the summation of the starting time of all the batches in the same machine. Constraint (4.11) assures the precedence: job *j* goes to the next operation in machine *k* when it is done on machine *i*. Constraints (4.12) and (4.13) force the order or sequence that each pair of jobs, *j* and *l*, goes on the BPM and on the rest of machines. In constraint (4.14), either sequence, (*job* ***j***, *job* ***l***) or (*job* ***l***, *job* ***j***), is scheduled on machine *i*. Constraints (4.15) and (4.16) determine that the *Cmax* is greater than or equal to the completion time of each job and of each batch. Constraints (4.17), (4.18), (4.19) and (4.20) set that the *Cmax*, the processing time of the batch *b*, the starting time of job *j* on

machine *i*, and the starting time of batch *b* are are non-negative, respectively. Constraints (4.21) and (4.22) restrict X_{jb} and Z_{jli} to be binary.

5. Computational experiments

To test the analytical formulation, the batch processing machine capacities (*S* and D) were assumed to be equal to 10 and 3, respectively. The job sizes were sampled from a discrete uniform (DU) random variable, so that $s_j \sim DU[1, S]$; the machine number and processing time for each step of a job were obtained from a collection of benchmark job shop instances for OR studies (Beasley, 1990). The instances were renamed as ***n***j***m***mc_***c***, where ***n*** is number of jobs, ***m*** is number of machines, and ***c*** is a consecutive number to determine what type of instance it is being referred to (see Table 2). For example, 6j06mc_1 means that this is the first problem (*c*=1) with 6 jobs and 6 machines.

All the experiments were conducted on a Core Duo PC, clocked at 1.86 GHz with 1 GB of RAM. Each problem instance was run three times based on which machine (machine 1 or 2 or 3) was considered as a BPM. The mathematical model for the *Jm|batch|Cmax* problem was coded in AMPL, and all the instances were solved with a commercial solver, CPLEX version 11.0. Given that the branch and bound approach used by CPLEX may take prohibitively long computational time to report the optimum solution, the run time was restricted to 1800 seconds for each instance. For smaller instances (i.e., ft06 to la23), CPLEX reported a feasible solution in 1800 seconds or less, but for larger instances (i.e., with more than 150 operations [15 jobs*10 machines]), it was necessary to modify the specified run time. In other words, for larger instances, CPLEX was run several times, increasing the run time in a step size of 1800 seconds, up to obtaining a feasible solution. The maximum allowed run time was equal to 28800 seconds (8 hours).

The results of the experiments are summarized in Table 2. Column (1) is the original name of the instance; columns (2) and (3) present the run code and the batch processing machine specified for each run. Column (4) shows the makespan. Columns (5) and (6) present the values of the absolute mixed-integer optimality gap tolerance (*absmipgap*), and the relative mixed-integer optimality gap tolerance (*relmipgap*) from CPLEX. AMPL/CPLEX (ILOG S. A., 2006) defines *absmipgap* as the absolute value of the difference between the current best integer solution found so far, and the optimal value of the LP relaxation or best bound deduced from all the node subproblems solved so far; it defines *relmipgap* as the ratio between *absmipgap* and (1+abs[*best bound*]). Column (7) shows the run time in seconds.

Table 2: CPLEX Results

Original Instance	Renamed Instance	BPM	Cmax	abs mipgap	rel mipgap	Run Time
ft06	6j6mc_1	1	**55***	0	0%	1.42
		2	**53***	0	0%	2.41
		3	**55***	0	0%	1.73
la01	10j05mc_1	1	**666***	0	0%	92.08
		2	**666***	0	0%	45.98
		3	**666***	0	0%	72.86
la02	10j05mc_2	1	**655***	0	0%	395.77
		2	**655***	0	0%	430.88
		3	**655***	0	0%	62.73
la03	10j05mc_3	1	**588***	0	0%	158.14
		2	586	1	0.17%	1800.16
		3	**597***	0.0345	0.01%	177.55

Original Instance	Renamed Instance	BPM	Cmax	abs mipgap	rel mipgap	Run Time
abz5	10j10mc_1	1	1238	40	3%	1800.22
		2	1238	19	2%	1800.12
		3	1223	121	10%	1800.66
ft10	10j10mc_2	1	940	97	10%	1800.41
		2	897	103	11%	1800.47
		3	935	133	14%	1800.42
la16	10j10mc_3	1	956	79	8%	1800.14
		2	**945***	0	0%	166.50
		3	873	84	10%	1800.58
la06	15j05mc_1	1	815	130	16%	1800.19
		2	926	355	38%	1800.19
		3	926	357	39%	1800.33
la07	15j05mc_2	1	829	189	23%	1800.36
		2	890	358	40%	1800.41
		3	890	402	45%	1800.30
la08	15j05mc_3	1	863	311	36%	1800.28
		2	863	332	38%	1800.20
		3	863	385	45%	1800.22
la21	15j10mc_1	1	1148	360	31%	1800.16
		2	1105	287	26%	1800.22
		3	1075	260	24%	1800.38
la22	15j10mc_2	1	970	205	21%	1800.38
		2	963	209	22%	1800.31
		3	953	198.6	21%	1800.41
la23	15j10mc_3	1	1032	290	28%	1800.39
		2	1032	289	28%	1800.50
		3	1032	310	30%	1800.48
la36	15j15mc_1	1	1369	337	25%	3600.36
		2	1292	248	19%	3600.39
		3	1340	272	20%	3600.62
la37	15j15mc_2	1	1449	377	26%	3600.27
		2	1470	363	25%	3600.28
		3	1421	332	23%	3600.30
la38	15j15mc_3	1	1234	239	19%	3600.45
		2	1287	311	24%	3600.42
		3	1199	208	17%	3600.30
ft20	20j05mc_1	1	1165	620	53%	9600.69
		2	1194	696	58%	14400.70
		3	1225	710	58%	9000.70
la11	20j05mc_2	1	1098	466	42%	3600.52
		2	1222	677	55%	3600.45
		3	1222	681	56%	3600.66
la12	20j05mc_3	1	1039	556	54%	3600.30
		2	1027	473	46%	3600.38
		3	1039	530	51%	3600.27

Original Instance	Renamed Instance	BPM	Cmax	abs mipgap	rel mipgap	Run Time
la26	20j10mc_1	1	1223	407	33%	14390.30
		2	1444	638	44%	10767.60
		3	1584	773	49%	9912.47
la27	20j10mc_2	1	1418	648	46%	10500.60
		2	1497	742	50%	11529.10
		3	1357	575	42%	7200.34
swv01	20j10mc_3	1	1542	731	47%	7200.25
		2	1636	843	52%	14400.60
		3	1584	802	51%	9100.55
abz7	20j15mc_1	1	795	337	42%	12628.20
		2	714	252	35%	10800.60
		3	862	402	47%	10800.50
abz8	20j15mc_2	1	768	292	38%	8149.39
		2	868	384	44%	11000.60
		3	828	340	41%	11350.50
swv06	20j15mc_3	1	2198	1,100	50%	7500.50
		2	2004	931	46%	9690.52
		3	2536	1,432	56%	9450.55
yn1	20j20mc_1	1	985	264	27%	9309.53
		2	960	217	23%	14400.60
		3	1131	395	35%	11000.50
yn2	20j20mc_2	1	979	249	25%	14400.50
		2	995	275	28%	6750.50
		3	1014	284	28%	12600.60
yn3	20j20mc_3	1	1364	632	46%	14400.50
		2	1126	404	36%	10900.50
		3	1383	653	47%	7875.53
la31	30j10mc_1	1	1892	1,117	59%	14347.60
		2	1941	1,147	59%	14389.20
		3	2102	1,331	63%	14385.60
la32	30j10mc_2	1	1998	1,165	58%	14395.00
		2	1959	1,129	58%	28190.00
		3	2180	1,350	62%	13969.50
la33	30j10mc_3	1	1893	1,133	60%	14102.60
		2	1913	1,130	59%	14400.60
		3	1769	1,046	59%	14073.10
swv11	50j10mc_1	1	**0****	0	0%	68932.00
		2	4426	3,622	82%	28775.40
		3	4661	3,864	83%	28766.60
swv12	50j10mc_2	1	4463	3,664	82%	28744.80
		2	5387	4,574	85%	28779.40
		3	**0****	0	0%	89933.40
swv13	50j10mc_3	1	4602	3,771	82%	28212.00
		2	5250	4,419	84%	27479.20
		3	5070	4,254	84%	28771.00

* Optimal solution
** Time limit with NO integer solution

In the interval between 1800 seconds and 28800 seconds, most of the instances reported an integer solution, not the optimal. Only instances 50j10mc_1 and 50j10mc_2, with 50 jobs and 10 machines, tested with machine1 and

machine3 as BPMs, respectively, were run with a time greater than 28800. However, CPLEX did not report any solution after running these two instances for 68932 seconds and 89933.40 seconds, respectively. Therefore, only 12 of 102 problems achieved the optimum in less than 1800 seconds. For these problems, notice in Table 2 that the values of *absmipgap* and *relmipgap* are equal to zero. Table 3 presents the average CPU time required for obtaining a feasible solution, and the average *relmipgap* achieved for instance. In conclusion, the experimental study indicates that the commercial solver was only able to solve small problem instances with 10 jobs or less in a reasonable time (avg. *relmipgap*≈0), and it was inadequate to solve moderate-sized problems (with jobs > 10) (avg. *relmipgap*>0) in the specified time limit.

Table 3: CPLEX Results

Total Operations	Instance	Avg. CPU Time*	Avg. relmipgap
36	6j*6mc	**1.85**	**0.00**%
50	10j*5mc	359.57	0.09%
75	15j*5mc	1,800.28	35.56%
100	10j*10mc	1,618.84	8.55%
100	20j*5mc	6,067.19	52.63%
150	15j*10mc	1,800.36	25.68%
200	20j*10mc	10,555.76	45.93%
225	15j*15mc	3,600.38	22.12%
300	20j*15mc	10,152.37	44.40%
300	30j*10mc	15,805.91	59.70%
400	20j*20mc	11,293.20	32.74%
500	50j*10mc	**39,821.53**	**83.11**%

***** seconds

Although the proposed mixed-integer formulation produces superior schedules for problem instances with a small number of operations, the amount of solution time required for larger operations is quite impractical. Therefore, four dispatching rules and three batch forming heuristics were investigated, and later combined for reducing the solution time. The dispatching rules SPT and Critical Ratio (CR) were combined with Modified DELAY (MD) and Modified First Fit Decreasing (MFFD); whereas Most Work Remaining (MWKR) and Most Operations Remaining (MOPNR) were combined with MD, First Fit (FF), and MFFD.

To assess the quality of the different combinations of dispatching rules along with batch forming heuristics, the best solution ($Cmax^{DR}$) of each combination was compared to MILP. Table 4 presents the percentage difference in solution (or gap) between the best $Cmax^{DR}$ from a dispatching rule along with a batch forming heuristic, and the $Cmax^{MILP}$ from the mathematical model, computed as in equation (5.1).

$$\%Gap = \frac{C_{\max}^{DR} - C_{\max}^{MILP}}{C_{\max}^{MILP}} \times 100\% \qquad (5.1)$$

Table 4: Avg. % Gap of Best Dispatching Rule vs. CPLEX per Instance

Total Operations	Instance	Avg. % Gap
36	6j*6mc	15.31%
50	10j*5mc	29.00%
75	15j*5mc	19.57%
100	10j*10mc	29.92%
100	20j*5mc	19.35%
150	15j*10mc	**38.31%**
200	20j*10mc	33.01%
225	15j*15mc	37.25%
300	20j*15mc	19.35%
300	30j*10mc	15.16%
400	20j*20mc	16.52%
500	50j*10mc	**14.05%**

For all the instances, the results of CPLEX are better than the dispatching rules ($\%gap^{DR}>0$). Moreover, the minimum (14.05%) and maximum (38.31%) average gap corresponds to 50jobs*10machines and 15jobs*10machines, respectively. However, for all the test cases, the dispatching rules used less CPU time than CPLEX (see Table 5). The minimum average CPU time was 0.24 seconds, which corresponds to 36 operations (6jobs*6machines); the maximum average CPU time was 1.66 seconds, which corresponds to 500 operations (50jobs*10machines). This contrasts with CPLEX, which, for the same problems, spent 1.85 seconds and 39821.53 seconds, respectively.

Table 5: Avg. CPU Time of Dispatching Rules vs CPLEX per Instance

Total Operations	Instance	Avg. CPU Time*	
		D.R**	CPLEX
36	6j*6mc	**0.24**	**1.85**
50	10j*5mc	0.33	359.57
75	15j*5mc	0.48	1,800.28
100	10j*10mc	0.33	1,618.84
100	20j*5mc	0.59	6,067.19
150	15j*10mc	0.48	1,800.36
200	20j*10mc	0.60	10,555.76
225	15j*15mc	0.50	3,600.38
300	20j*15mc	0.62	10,152.37
300	30j*10mc	0.92	15,805.91
400	20j*20mc	0.68	11,293.20
500	50j*10mc	**1.66**	**39,821.53**

***** Seconds

****** Dispatching Rules

6. Conclusions and future work

Most of the combinatorial optimization problems are concerned with the efficient allocation of limited resources to meet desired objectives of a company. Generally, these objectives need to be accomplished in a finite time period, during which many problems arise, and require to be solved very quickly. Dispatching rules have been most commonly used in practice for scheduling. Our *Jm|batch|Cmax* problem is strongly NP-hard, and with our mathematical formulation, optimal solution can be obtained only for small problem instances (with operations $\leq$ 50). For larger problem instances, dispatching rules can be used to obtain a solution in shorter computational time.

However, CPLEX can provide better feasible solutions than dispatching rules, when allowed to run for longer time. The time taken by CPLEX ranges from 1800 to 28800 seconds for the above mentioned problem, with a feasible solution far from the optimum.

This research has extended the class of scheduling problems researched in academia by combining the classical *Jm||Cmax* problem with $1|r_j,batch|Cmax$ (Damodaran et al., 2007). A number of interesting directions for future research can emerge from this dissertation that might benefit the practitioners, and might allure other academicians. Mainly, the problem scope and the solution approach should be considered first in future studies. To solve large problems in reasonable CPU times, future works with the characteristic of the problem object of this study should consider exploring other procedures.

References.

Balas, E., (1969) "Machine Sequencing Via Disjunctive Graphs: An Implicit Enumeration Algorithm". *Operations Research*;17:941-957.

Beasley, J. E., (1990). "Set of 82 JSP Test Instances". Retrieved June 30,2009, from *OR-library*: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/files/jobshop1.txt.

Damodaran, P., Srihari, K., Lam, S. S., (2007). "Scheduling a Capacitated Batch-Processing Machine to Minimize Makespan". *Robotics and Computer-Integrated Manufacturing*; 23(2):208-216.

Gupta, A. K., Sivakumar A. I., (2006). "Job Shop Scheduling Techniques in Semiconductor Manufacturing". *The International Journal of Advanced Manufacturing Technology*; 27(11-12):1163-1169.

ILOG S. A., (2006). "ILOG AMPL CPLEX System Version 10.0 User's Guide January 2006". Retrieved June 30, 2009, from AMPL website: http: //www.ampl.com/BOOKLETS/amplcplex100userguide.pdf.

Mason, S. J., Fowler, J. W., (2000). "Maximizing Delivery Performance in Semiconductor Wafer Fabrication Facilities". *Simulation Conference Proceedings, 2000*; 2:1458-1463 o.2.

Mason, S. J., Fowler, J. W., Carlyle, W. M., (2002). "A Modified Shifting Bottleneck Heuristic for Minimizing Total Weighted Tardiness in Complex Job Shops". *Journal of Scheduling*;5(3):247-262.

Ovacik, I. M., Uzsoy, R., (1997). "Decomposition Methods for Complex Factory Scheduling Problems". Boston, Mass: *Kluwer Academic Publishers.*

Pinedo, M., (2002). "Scheduling Theory, Algorithms, and Systems". Second ed. *Upper Saddle River*, New Jersey: Prentice-Hall, Inc.

Authorization and disclaimer

Authors authorize LACCEI to publish the paper in the conference proceedings. Neither LACCEI nor the editors are responsible either for the content or for the implications of what is expressed in the paper.