General models for hybrid flow shop sequencing problems with sequence dependent setup times

M. Zandieh
Department of Industrial Engineering,Amirkabir University of Technology
z7725953@aut.ac.ir
S. M. T. Fatemi Ghomi
Department of Industrial Engineering,Amirkabir University of Technology
fatemi@aut.ac.ir
S. M. Moattar Husseini
Department of Industrial Engineering,Amirkabir University of Technology
(تاریخ دریافت ۸۴/۴/۲۶، تاریخ تصویب ۸۴/۷/۱۶)

Abstract

This study presents two mixed integer programming models for scheduling hybrid flow shops with sequence dependent setup times.In the first model , we assumed the machines at each stage are identical,but in the second model the machines at each stage are different.These models may be used to determine optimal solutions for hybrid flow shop problems of moderate size, and these problems could then be used as benchmarks for testing new sequencing heuristics.

Keywords: Short-term scheduling; Hybrid flow shop; Sequence dependent setup times; Mathematical programming

Introduction

The purpose of this paper is twofold: (1) to present two mixed integer programming models for optimizing a family of hybrid flow shop sequencing problems with sequence-dependent setup times to minimize makespan, and (2) to compare these models on the basis of problem size, ease of implementation, and the flexibility of extension to variants.

Section 2 briefly reviews the related literature. In section 3 problem definition is presented. Sections 4 and 5 contain the general models. In section 6, the models are compared. Section 7 concludes and presents directions for future research.

Literature Review

Several flow patterns can be encountered, depending on the number of operations required to process a job and on the number of available machines per operation. When a job requires only one operation for its completion, we characterize it as single-operation; otherwise, we call it multi-operation. In the latter case, the concept of routing may be introduced based on machines, we have single machine shop, flow shop, permutation flow shop, job shop, and open shop scheduling problems. When processing stages are considered instead of machines, we have parallel machine shop, hybrid flow shop, job shop with duplicate machines scheduling problems.The diagram in Fig.1 illustrates schematically the relationships between the different machine environments (Zandieh and Fatemi, 2003).

A hybrid flow shop model allows us to represent most of the production systems. The process industry such as chemical, pharmaceutical, oil, food, tobacco, textile, paper, and metallurgical industry can be modelled as a hybrid flow shop. In the literature, the notion of hybrid flow shop has emerged in the 70s (Arthanary and Ramaswamy, 1971).

A hybrid flow shop consists of a series of production stages or workshops, each of which has several facilities in parallel (Elmaghraby and Karnoub, 1995).

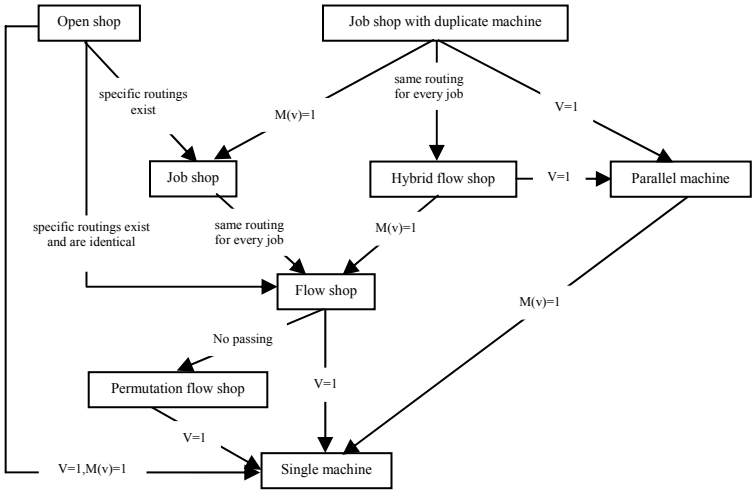

Fig. 1: A classification for scheduling problems based on resource environments.

Some stages may have only one facility, but for the plant to be qualified as a hybrid flow shop, at least one stage must have several facilities. The flow of products in the plant is unidirectional. Each product is processed at only one facility in each stage and at one or more stages before it exits the plant. Each stage may have multiple parallel identical machines. Each job is processed by at most one machine at each stage.

Several mathematical formulations based on a continuous representation of the time domain have been published. Pinto and Grossmann (1995) proposed a continuous time MILP formulation for short-term scheduling of multistage batch plants. Since there was not a variable explicitly addressing consecutive orders, their formulation had difficulties handling order-sequence-dependent constraints. Pinto and Grossmann (1996) presented an alternate model that explicitly incorporated pre-ordering constraints into a representation of time slots for units. Their model was suitable for only pre-ordered orders and still did not address the sequence dependencies. Cerda, Henning and Grossmann (1996) proposed an MILP model for the short-term scheduling of a single-stage multi-product batch plants with parallel lines. They applied a set of tri-index variables (order, order, unit) to handle order sequence-dependencies explicitly. Ierapetritou and Floudas (1998) presented a continuous time formulation for the short-term scheduling of batch processes. The application of their formulation was restricted to pre-ordered problems. Chi-Wai and Gupta (2000) proposed continuous time MILP model for short term scheduling of multistage batch plants with sequence dependent consideration. They proposed a novel formulation that replaces the tetra-index binary variables without losing the model's generality. Moon and Hrymak (1999) addresses the short-term scheduling problem for the batch anealing process in the heat treatment of steel coils with sequence dependent consideration. They proposed a novel MILP model for scheduling and a novel algorithm for solving the problem.

Stafford (1988) focused on the standard flowshop (with no parallel machines and no sequence-dependent setup times). Srikar and Ghosh (1986) considered a permutation flowshop with sequence-dependent setup times in their MILP

model, which used many fewer variables than the previous models. Srikar and Ghosh (1986) used decision variables that focused on whether a job is scheduled anytime before another job. However, Stafford and Tseng (1990) discovered several problems with Srikar and Ghosh (1986), corrected these and extended this modeling concept to non-sequence-dependent setup time flowshops, no-intermediate-queue flowshops and sequence-dependent setup time, no-intermediate- queue flowshops. Rios-Mercado and Bard (1998) also consider the sequence-dependent setup time flowshop and develop several valid inequalities for Models based on the traveling salesman problem and the Srikar-Ghosh model.

No literature has been discovered addressing models for the hybrid flow shop with sequence- dependent setup times. The key features that are missing are the multiple machines per stage and the ability of jobs to skip some stages.

Problem definition

Let *V* be the number of workshops in series. Let *N* be the number of jobs to be processed and *M(v)* be the number of machines in parallel at each stage *v*. We assume that machines are initially setup for a nominal job 0 at every stage. Job *N*+1 exists at every stage only to indicate the end of the process, if needed. We have the following definitions.

N=number of jobs to be processed
V =number of serial workshoptages
v_i =last stage visited by job i
M(v) = the number of machines in parallel at each stage v
p(i,v) =processing time for job i at stage v
s(i,j,v) =sequence dependent setup time from job i to job j at stage v
W(i) =set of workshop stages visited by job i
J(v) =set of jobs that visit workshop stage v
J(v,0) =J(v) plus job 0
J(v,N+1) =J(v) plus job N+1

The processing time of job 0 is set at 0. The setup time from job 0 indicates the time to move from the nominal set point state. We assume that all jobs currently in the system must be completed at each stage before the jobs under consideration may begin setup. The completion times of job 0 at each stage are set to the earliest setup time may begin at that stage. The setup time for job *N*+1 is set at 0; this job only exists to indicate the end of the schedule. We also include the restriction that every stage must be visited by at least as many jobs as there are machines in that stage.

First model

In the TSP based model, following model A from Rios-Mercado and Bard (1998), the following additional variables are used.

r(v)=time at which stage *v* can begin processing, v=1,..., V
C(i,v)=completion time for job *i* at stage *v*, v=1,..., V, i=1,..., N
X(i,j,v) =1 if job *i* is processed immediately before job *j* at stage *v,* and 0 otherwise
$i \in J(v,0), j \in J(v,N+1), i \neq j, v = 1,...,V$

Several observations can be made which guide the formulation of this model. Each stage *v* has a time at which it can begin processing, which is denoted *r(v)* . We consider the time each stage can begin processing to be the completion time of job 0 at that stage, and we further assume that it is the same for all machines at a stage. Every stage exists independently except that a job is not available to begin setup in stage *v* until it has completed processing on stage *v*-1. Each stage can be considered as a traveling salesman problem with multiple salesmen, equal to the number of machines in that stage. We assign jobs to machines implicitly; by interpreting a group of variables, such as X(0,3,1)=1 , X(4,2,1)=1,and X(2,0,1)=1, we see that jobs 0, 3, 4 and 2 are assigned to a machine in stage 1 in that order. Equations (1) to (11) comprise model 1.

Model 1:

$$\min \quad z$$

$$s.t:$$

(1)

$$\sum_{j\in J(v)} X(0,j,v) = M(v); \quad \forall v \tag{2}$$

$$\sum_{j\in J(v,N+1)-\{i\}} X(i,j,v) = 1; \quad \forall v, \quad i \in J(v,0) \tag{3}$$

$$\sum_{i\in J(v,0)-\{j\}} X(i,j,v) = 1; \quad \forall v, \quad j \in J(v,N+1) \tag{4}$$

$$C(0,v) = r(v); \quad \forall v \tag{5}$$

$$C(j,v) - C(i,v) + \left[1 - X(i,j,v)\right] B(v) \geq S(i,j,v) + p(j,v); \quad \forall v, \quad i \in J(v,0), \quad j \in J(v,N+1), \; i \neq j \tag{6}$$

$$C(j,v) - C(i,v-1) + \left[1 - X(i,j,v)\right] B(j,v) \geq S(i,j,v) + p(j,v); \quad \forall v, \quad i \in J(v,0), \quad j \in J(v,N+1), i \neq j \tag{7}$$

$$C(i,v) \geq C(0,v); \quad \forall v \quad i \in J(v) \tag{8}$$

$$C(i,v) \geq C(i,v-1); \quad v = 2,..,V, \quad i \in J(v) \tag{9}$$

$$z \geq C(i,v_i); \quad i = 1,..,N \tag{10}$$

$$X(i,j,v) \in \{0,1\}; \quad \forall i,j,v \tag{11}$$

The objective is to minimize makespan, which is denoted by *z* and is incorporated into later constraints. Job 0 must be scheduled on every machine in every stage. This is expressed by constraint set (2). At every stage, every job must have exactly one predecessor and exactly one successor. This is expressed by the constraint sets (3) and (4). The completion time of job 0 on each stage is set to be the time each stage is available in constraint set (5). The completion time of two jobs in the same stage depend on whether those jobs are scheduled together or not. If two jobs, namely *i* and *j*, are scheduled in that order with no jobs between them, job *j*'s completion time is at least the sum of job *i*'s completion time, the setup time from *i* to *j*, and job *j*'s processing time. If *i* and *j* are scheduled in any other way, the difference between the two completion times is larger than a very negative number B(v). B(v) is an upper bound on the time stage v completes processing, similar to the upper bound *A* in Rios-Mercado and Bard (1998).

$$B(1) = \sum_{i=1}^{n}\left[p(i,1) + \max_{j\in\{0,1,..,N\}}\{S(j,i,1)\}\right], \text{ and}$$

$$B(v) = B(v-1) + \sum_{i=1}^{n}\left[p(i,v) + \max_{j\in\{0,1,..,N\}}\{S(j,i,v)\}\right].$$

Constraint set (6) enforces the restriction that the machine must be idle before setup can begin. The completion time of a job in one stage also depends on its completion time in the previous stage.We define C(i,0)=0, iϵ(0,1,..,N). We incorporate this idea by ensuring that the completion time of job *j* in stage *v* is at least as large as the completion time in stage *v*-1 plus the setup from the immediately previous job in stage *v*, plus the processing time of job j. Constraint set (7) enforces the concept that the job must be available before setup can begin. The value B(j,v) is set to $B(j,v) = p(j,v) + \max_i\{S(i,j,v)\}$. Since job 0 is some sort of surrogate for machine availability, we require every job to be processed after job 0 at every stage. Allowing that some jobs do not actually require processing in every stage, we include constraint set (8). Since jobs must be processed in a linear fashion through the stages, we include the restriction that completion times must be nondecreasing as the stage number increases. This is enforced through constraint set (9). The makespan is defined to be the latest time a job completes processing. Because the flowshop is hybrid, each job j's last stage is v_j. The makespan is the largest of the completion times of all the jobs, denoted by $C(j,v_j)$ for each job. Since jobs 0 and N+1 exist to begin and end the schedule, we do not include these jobs for consideration in defining the makespan in constraint set (10).

Second model

In the second model, jobs are explicitly placed into positions on machines, following Wagner (1959) and Stafford (1988). If there were one machine at every stage and the jobs were required to be processed in the same order at every stage, this would result in a permutation schedule. However, we will be making the investment in extra decision variables so that permutation schedules will not be required. Moreover, with possibly a different number of machines at every stage and jobs that may skip stages, it is not clear how to define a permutation schedule in this context. A specific job is called job *i*. The job in the *j*th position will be called the *j*th job. The following decision variables are used.

X(i,j,v,k)=1 if job i is scheduled in position j on machine k in stage v and 0 otherwise
e(j,v,k)=setup beginning time for jth job on machine k in stage v
d(j,v,k)=completion time for jth job on machine k in stage v
w(j,v,k)=idle time after jth job completes and before start of setup of (j+1)st job on machine k in stage v
T(j,v,k)=processing time of the jth job on machine k in stage v
S(j,v,k)=setup time from the jth to (j+1)st job on machine k in stage v

Position 0 on every machine at every stage will be occupied by job 0. Note that a different job 0 can be created for each machine and each stage in this problem, so that the completion of job 0 can represent the end of an in-progress schedule through the appropriate assignment of values to variables *d(0,v,k)*. The largest position number required for each machine in stage *v* is $|J(v)|$, because job 0 is not counted as one of the jobs in each stage. Equations (12) - (26) comprise model 2.

The objective is to minimize makespan, which is denoted by *z* and is incorporated into later constraints. Every job that visits stage *v* must be assigned to exactly one position on one machine in that stage. We will later explicitly assign job 0 to the 0th position, so we need only consider the other jobs that visit stage *v*. This is expressed by constraint set (13). We explicitly assign job 0 to the 0th position on all machines at all stages in constraint set (14). Every position of every machine in every stage has at most one job assigned to it, expressed by constraint set (15). Constraint set (15) only requires that at most one job is assigned to each possible position. In no way does it requires that jobs be scheduled in consecutive positions. For example, up to this point, a valid assignment may have jobs in positions 0, 1, 3, and 4 on a machine in a particular stage. To prevent this, we require all positions after an empty position to empty as well. This can be enforced by constraint set (16). A couple of "helper" variables are defined to simplify the presentation of the model. These variables are *T(j,v,k)* and *S(j,v,k)*, which associate processing and setup times with jobs by position instead of names. For example, if job 3 is the 4th job on stage 2, machine 3, the variable T(4,2,3) takes on the value p(3,2). The processing times are associated with the position numbers by using the constraint set (17).

Model 2:

$$\min \quad z$$

$$s.t:$$

$$\sum_{j=1}^{|J(v)|}\sum_{k=1}^{M(v)} X(i,j,v,k) = 1; \quad \forall v, i \in J(v) \tag{12}$$

$$X(0,0,v,k) = 1; \quad \forall v, \forall k \tag{13}$$

$$\sum_{i \in J(v)} X(i,j,v,k) \le 1; \quad \forall v, \forall k, j = 1,..,|J(v)| \tag{14}$$

$$X(h,j+1,v,k) \le \sum_{i \in J(v,0)-\{h\}} X(i,j,v,k); \quad \forall v, \forall k, h \in J(v), j = 1,..,|J(v)|-1 \tag{15}$$

$$T(j,v,k) = \sum_{i \in J(v)} p(i,v)X(i,j,v,k);$$
$$\forall v, \forall k, j = 0,..,|J(v)| \quad (17)$$

$$S(j+1,v,k) = \sum\sum Y(i,j,h,v,k)S(i,h,v)$$
$$\forall v, \forall k, j = 0,..,|J(v)|-1 \quad (18)$$

$$Y(i,j,h,v,k) \le X(i,j,v,k);$$
$$\forall v, \forall k, i \in J(v), h \in J(v) - \{i\}, j = 1,..,|J(v)|-1 \quad (19)$$

$$Y(0,0,h,v,k) \le X(h,l,v,k);$$
$$\forall v, \forall k, h \in J(v) \quad (20)$$

$$Y(i,j,h,v,k) \le X(h,j+1,v,k);$$
$$\forall v, \forall k, i \in J(v), h \in J(v) - \{i\}, j = 1,..,|J(v)|-1 \quad (21)$$

$$Y(0,0,h,v,k) \ge X(0,0,v,k) + X(h,l,v,k) - 1;$$
$$\forall v, \forall k, h \in J(v) \quad (22)$$

$$Y(i,j,h,v,k) \ge X(i,j,v,k) + X(h,j+1,v,k) - 1;$$
$$\forall v, \forall k, i \in J(v), h \in J(v) - \{i\}, j = 1,..,|J(v)|-1 \quad (23)$$

$$d(0,v,k) = r(v,k);$$
$$\forall v, \forall k \quad (24)$$

$$e(j+1,v,k) = d(j,v,k) + w(j,v,k);$$
$$\forall v, \forall k, j = 0,..,|J(v)|-1 \quad (25)$$

$$d(j,v,k) = e(j,v,k) + s(j,v,k) + T(j,v,k);$$
$$\forall v, \forall k, j = 1,..,|J(v)| \quad (26)$$

$$d(j,v,k) \le e(i,f,v,k) + [1 - X(i,f,u,l)]$$
$$B + [1 - X(i,j,v,k)]B;$$
$$\forall v, \forall k, v \prec u \prec V, \forall l, i \in [J(v) \cap J(u)],$$
$$j = 1,..,|J(v)|, f = 1,..,|J(u)| \quad (27)$$

$$z \ge d(j,v,k); \quad (28)$$

$$X(i,j,v,k) \in \{0.1\}; \quad \forall i,j,v,k \quad (29)$$

The setup time from job *i* to job *h*, if job *i* is assigned to position *j* and job *h* is assigned to position *j*+1 on machine *k* in stage v will be called the setup to the (*j*+1)st job, *S(j+1,v,k)* , and can be found by the following equation

$$S(j+1,v,k) = \sum_{i \in J(v)-\{h\}} \sum_{h \in J(v)} X(i,j,v,k)S(i,h,v)X(h,j+1,v,k);$$
$$\forall v, \forall k, j = 0,..,|S(v)|-1$$

This is non-linear. We introduce a new binary variable Y(I,j,h,v,k) which is 1 when job *i* is in position *j* and job *h* is in position *j*+1 on machine *k* in stage v, and 0 otherwise. Constraint set (27) is replaced by the constraint sets (18), (19), (20) , (21) , (22) and (23). At every machine on every stage, job 0's completion time is that machine's ready time. At this time, we compute the machine's ready time in constraint set (24) as the sum of the processing of job 0 up to and including the current stage, or $r(v,k) = \sum_{u=1}^{v} T(0,u,k)$, though this may vary by application. The (*j*+1)st job in each stage on each machine can begin processing after the *j*th job completes processing and some waiting time passes. The waiting time can account for the fact that the (*j*+1)st job is not done with its processing on the previous stage it visits and machine *k* on stage v is idle while waiting for the job to arrive. Since the objective is makespan, which is defined by only one job's completion time, the waiting time variable may be positive even when the job is available for setup. This set of scheduling variables allows machines to be idle when jobs are available and is shown in constraint set (25). The *j*th job's completion time is the sum of its available-to-setup time, the time to setup from the (*j*-1)st job to the *j*th job, and its processing time, as shown in constraint set (26). Because we require jobs to finish processing at stage v before setup can begin at stage v+1, we need the available-to-setup time for jobs in stage v to be at least as large as the completion time of the jobs in stage v-1. However, our available-to-setup and completion time variables are in terms of the position each job has, not the jobs themselves. For this reason, we need to translate these times so

Table 1: Size complexity of the SDST models.

IP model Number of	model I	model II
Binary Variables	$\sum_{v=1}^{V}\left(\lvert J(v)\rvert\right)\left(\lvert J(v)\rvert+1\right)$	$\sum_{v=1}^{V} M(v).\left(\lvert J(v)\rvert^3-\lvert J(v)\rvert^2+\lvert J(v)\rvert+1\right)$
Other Variables	(N+1)V	$5\sum_{v=1}^{V}\left(\lvert J(v)\rvert+1\right)M(v)$
Constraints	$2V+N-\lvert J(v)\rvert+2\sum_{v=1}^{V}\left(\lvert J(v)\rvert.(2+\lvert J(v)\rvert)\right)$	$\sum_{v=1}^{V}\left[3\lvert J(v)\rvert+4M(v)+M(v).\lvert J(v)\rvert\left(3\lvert J(v)\rvert^2-2\lvert J(v)\rvert+5\right)\right]$

Table 2 : Empirical model sizes and solution information (computer time given are in second).

Problem	Problem Size	model I		model II	
1	V=2, $\mathbf{M(v)}=\mathbf{1},\lvert J(v)\rvert=6,\forall v$	Binary Var.	64	Binary Var.	386
		Constraint	196	Constraint	1292
		Time	251	Time	321
2	V=2, $\mathbf{M(v)}=\mathbf{2},\lvert J(v)\rvert=6,\forall v$	Binary Var.	84	Binary Var.	772
		Constraint	196	Constraint	3004
		Time	31	Time	3100,gap21%
3	V=2, $\mathbf{M(v)}=\mathbf{2},\forall v, J(1)=\{1,4,6\}, J(2)=\{2,3,4,5,6\}$	Binary Var.	42	Binary Var.	272
		Constraint	116	Constraint	876
		Time	0.31	Time	3.2

that we can make the required connections between these times in terms of the jobs themselves. Constraint set (27) ensures that the time setup can begin for job *i* on all stages $u \succ v$ must be no earlier than the time job *i* ends on stage v. The makespan is defined to be the latest time a job completes processing in constraint set (28).

Comparing models 1 and 2

Now that both models have been described, they can be compared in terms of the number of variables and constraints. Table 1 contains the number of binary and non-binary variables and the number of constraints in each model. The number of constraints and binary variables are much higher in the second model. In order to better demonstrate the magnitude of the difference, Table 2 shows the number of binary variables and constraints for several different sized problems. Each cell shows the number of binary variables and constraints in that order. These problems have been solved in LINDO on a Pentium 4, running time at 1400MHz. The additional information in each cell shows the solving time.

Clearly, the number of binary variables and constraints is much larger for model 2. In these few small examples, the solving time required in the solution of model 2 IPs is much higher than in the model 1 IPs. In problem 2, model 2 was not able to solve the problem optimally in the 1 hour computational time limit imposed, while model 1 solved the problem optimally in less than a minute. Problem 2 differs from Problem 1 structurally only in that the stages have two machines each. Model 2 cannot tell the difference between these two machines at each stage, which certainly causes the solver to spend time considering alternative solutions that are identical. Problem 3 is easily solved by both models, even though it has two machines at each stage. This problem is easier because only two jobs visit both stages, so the interaction between the stages is less when compared to problems 1 and 2.

Despite the above performance, model 2 has significant advantages over model 1 due to the adaptations that model 2 can

accommodate. Model 2 can accommodate machines at stages with different ready times, non-identical processing times and differing capabilities. Model 2 can also accommodate inserted idle time in schedules, as well as times during which machines may not be available, such as during planned maintenance. For this reason, we believe that the basic structure of model 2 has an inherent value.

Conclusions

Two models have been presented for scheduling the hybrid flow shops with sequence-dependent setup times. These models differ in their modeling perspective. Though the second one is much larger, it is more adaptable to variants of the problem. Several strategies will be pursued in improving model 2. The first strategy will be to add constraints to differentiate between machines in each stage, in an effort to break the symmetry that exists in the stages, following work done by Sherali, sith, and Lee (2000) in the design of optical networks. The second strategy will focus on redefining the variables, since all the information in the *X* variables is replicated in the *Y* variables. The effectiveness of these strategies will be evaluated by generating and solving instances of this problem.

References

1 – Arthanary, T. S. and Ramaswamy, K. G. (1971). "An extension of two machine sequencing problems." *Opsearch*, Vol. 8, PP. 10-22.

2 - Cerda, J., Henning, P., & Grossmann, I. E. (1996). "A mixed integer linear programming model for short-term scheduling of single stage multi-product batch plants with parallel lines." *Industrial Engineering & Chemical Research,* Vol. 36, PP. 1695-1707.

3 - Chi-Wai Hui, and Avaneesh, Gupta. (2000). "A novel MILP formulation for short-term scheduling of multistage multi-product batch plants." *Computers & Chemical Engineering,* Vol. 24, PP. 1611-1617.

4 – Elmaghraby, S. E. and Karnoub, R. E. (1995). *Production control in flexible flowshops: an example from textile manufacturing.* OR Report N°305 OR & IE Dept. North Carolina State University. USA.

5 - Ierapetritou, M. G. and Floudas, C. A. (1998). "Effective continuous time formulation for short-term scheduling. 1. Multipurpose batch processes." *Industrial Engineering & Chemical Research,* Vol. 37, PP. 4341-4359.

6 - Pinto, J. M. and Grossmann, I. E. (1995). "A continuous time mixed integer linear programming model for short-term scheduling of multistage batch plants." *Industrial Engineering & Chemical Research,* Vol. 34, PP. 3037-3051.

7 - Pinto, J. M. and Grossmann, I. E. (1996). "A continuous time MILP model for short-term scheduling of batch plants with preordering constraints." *Computers & Chemical Engineering,* Vol. 20, PP. S1197-S1202.

8 - Rios-Mercado, R. Z. and Bard, J. F. (1998). "Computational experience with a branch-and-cut algorithm for flowshop scheduling with setups." *Computers and Operations Research,* Vol. 25, No. 5, PP. 351-366.

9 - Sherali, H. D., Smith, J. C. and Lee, Y. (2000). "Enhanced model representations for an intra-ring synchronous optical network design problem allowing demand splitting." *INFORMS Journal on Computing,* Vol. 12, No. 4, PP. 284-298.

10 - Srikar, B. N. and Ghosh, S. (1986). "A MILP model for the *N*-job, *M*-stage flowshop with sequence dependent set-up times." *International Journal of Production Research,* Vol. 24, No. 6, PP. 1459-1474.

11 - Stafford, E. F. (1988). "On the development of a mixed-integer linear programming model for the flowshop sequencing problem." *Journal of the Operational Research Society,* Vol. 39, No. 12, PP. 1163-1174.

12 - Stafford, E. F. and Tseng, F.T. (1990). "On the Srikar-Ghosh MILP model for the *N*x*M* SDST flowshop Problem." *International Journal of Production Research,* Vol. 28, No. 10, PP. 1817-1830.

13 – Sungdeuk, Moon. And Hrymak, Andrew N. (1999). "Scheduling of the batch annealing process-deterministic case." *Computers & Chemical Engineering,* Vol. 23, PP. 1193-1208.

14 - Wagner, H. M. (1959). "An integer linear-programming model for machine scheduling." *Naval ResearchLogistics Quarterly*, Vol. 6, PP. 131-140.

15 - Zandieh, M. and Fatemi,S. M. T. (2003). "A Framework and A Classification Scheme for modelling production systems." *Proceedings of the Second National Industrial Engineering Conference,* Yazd University,Yazd, Iran , PP. 308-315.