Proceedings of the 1992 IEEE
International Conference on Robotics and Automation
Nice, France - May 1992

SCHEDULING FLEXIBLE FLOW SHOPS WITH NO SETUP EFFECTS*

Shi–Chung Chang, Da–Yin Liao

Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan, 10764

ABSTRACT

This paper presents an efficient, optimization model–based approach for scheduling the production of discrete–part, make–to–order type of flexible flow shops, where setup effects are negligible. A nominal scheduling algorithm based on Lagrangian relaxation and minimum cost linear network flow is first developed for scheduling under nominal conditions. Fast rescheduling algorithms that exploit the economic interpretation of Lagrange multipliers and the network structure of production flows are then proposed for timely adjusting the nominal schedule to cope with disturbances. Numerical results on realistic examples demonstrate that our methodology is quite effective; it generates near–optimal schedules, provides relatively smooth adjustments (i.e., no drastic change from the nominal schedule) and is computationally efficient.

1. INTRODUCTION

Although there have been many new develop–ments in manufacturing automation, effective (optimal and computationally efficient) scheduling methodologies for advanced manufacturing systems remain to be developed so that the productivity and flexibility of an automated factory can be fully exploited [1]. In spite of numerous research results on production scheduling in the literature [2], there still exists quite a gap between scheduling theory and practice. Due to the complexity of a realistic production system, finding the optimal schedule efficiently is often beyond the reach of the existing theories even for a small scale system. In practice, production schedules are usually generated either by simple heuristics or by time consuming computer simulation. Among the various aspects of scheduling a real factory, handling uncertainties in manu–facturing, such as machine failures, material shortage and demand variation, is especially a challenging issue in both theory and practice [3].

In this paper, we propose an optimization model–based and computationally efficient methodology for production scheduling. As depicted in Figure 1.1, the methodology consists of three parts: (1) an efficient algorithm for finding a near–optimal schedule under nominal conditions, i.e., a nominal scheduling algorithm, (2) optimization model–based, fast rescheduling algorithms for timely adjusting the nominal schedule to cope with disturbances and (3) periodic rescheduling using the nominal algorithm based on the latest system status. Part (3) applies the open–loop feedback philosophy of stochastic optimal control [4] and aims at responding to disturbances in a longer time scale. For small disturbances, our fast rescheduling algorithm in (2) applies a neighborhood search to the nominal schedule [5], exploits the structure of the nominal scheduling algorithm and aims at a quick but reasonably graceful response.

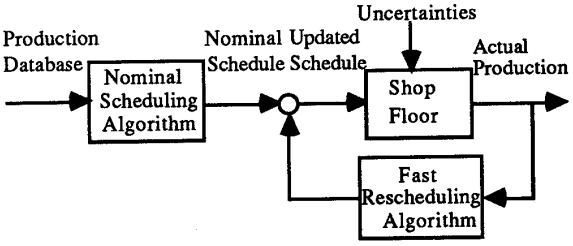

Figure 1.1

We realize the above methodology on the production scheduling of a flexible flow shop (FFS) where setup effects are negligible. A FFS is an extension of the traditional flow shops. It consists of several machining groups, each of them having a

* This work was supported in part by the National Science Council of the Republic of China under Grants NSC–78–0422–E002–09, NSC–79–0422–E002 –05 and NSC–80–0422–E002–06.
* The authors would like to thank Prof.Peter B. Luh and Dr. Debra Hoitomt of the University of Connecticut, Prof. Hong–Mo Yeh of Fu–Jeng University and engineers of Fu–Sheng Industrial Co. for their valuable suggestions and comments.

0-8186-2720-4/92 $3.00 ©1992 IEEE

number of identical machines and a production operation may be processed by any of the identical parallel machines. Though a FFS is more structured than a general flexible manufacturing system, effective scheduling of a FFS is still very challenging and difficult.

We develop algorithms for realizing the methodology and conduct numerical experimentations based on realistic data of a FFS to examine the feasibility, optimality and computational efficiency of these algorithms. Results indicate that our nominal algorithm obtains near–optimal schedules. Our fast rescheduling algorithms are computationally very efficient and usually result in a near–optimal and smooth adjustments for small disturbances.

The remainder of this paper is organized as follows. In Section 2, we first formulate the nominal scheduling problem mathematically. Section 3 states the development of a nominal scheduling algorithm by modifying our earlier work [6]. Section 4 describes our fast rescheduling algorithms for handling production uncertainties. Numerical results of these algorithms are given in Section 5. Section 6 then concludes this paper.

2. PROBLEM FORMULATION

Consider a make–to–order FFS which manufactures medium–volume and medium–variety of discrete parts with each type of parts having its own due date and demanded quantity. For simplicity of presentation without limiting much generality of results, we assume that production flow of all types of parts go through the M machine groups in the same sequence. The M machine groups can be organized as a line of production as shown in Figure 2.1, where buffer m locates before machine group m and buffer M+1 represent the stock of finished parts respectively. Both the input and stock buffers are infinite in size. We also assume that all demands are released at the beginning of the scheduling horizon.

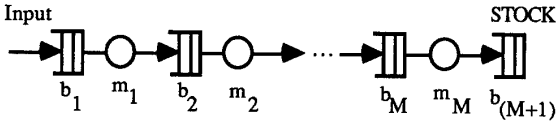

Figure 2.1

Let us define some notations for such a FFS.

Notations:

I : total number of part types;
i : part type index, $i = 1, \cdots, I$;
D_i : demand for type i parts;
d_i : due date of type i parts;
M : total number of machine groups;
m : machine group index, $m = 1, \cdots, M$;
T : total number of time periods within the scheduling horizon;
t : time period index, $t = 1, \cdots, T$;
C_{mt} : capacity of machine group m at period t;
P_{im} : processing time of part type i on machine groups m;
b : buffer index;
S_b : capacity of buffer b;
h_{ib} : holding cost per type i part in buffer b;
X_{ibt} : number of type i parts in buffer b at the beginning of time period t;
u_{imt} : number of type i parts loaded onto machine group m for processing at period t;
Z_{it} : number of type i parts arriving at the stock during period t, where $Z_{it} = u_{iM(t-P_{iM})}$;
O_{mt} : overtime capacity of machine group m during period t, where $0 \le O_{mt} \le C^o_{mt}$, C^o_{mt} is the maximum overtime capacity;
l_{mt} : overtime cost per unit overtime capacity of machine group m during period t.

In the flow line, the amount of type i parts loaded onto machine (m–1) for processing at period $t-P_{i(m-1)}$ go into buffer m after $P_{i(m-1)}$ periods of processing. The first buffer of the line serves as a source with an initial level of D_i units of type i parts while the last buffer serves as a sink and accumulates finished parts. Flows of parts must satisfy the following flow balance equations:

Flow Balance Equations

$$X_{i11} = D_i \text{ and } X_{im1} \text{ given, } m=2,\cdots,M; \quad (2.1.a)$$

$$X_{i1(t+1)} = X_{i1t} - u_{i1t}; \quad (2.1.b)$$

$$X_{im(t+1)} = X_{imt} - u_{imt} + u_{i(m-1)(t-P_{i(m-1)})}, \quad m=2,\cdots,M; \quad (2.1.c)$$

$$X_{i(M+1)(t+1)} = X_{i(M+1)t} + u_{iM(t-P_{iM})}; \quad (2.1.d)$$

for $i=1,\cdots,I$ and $t=1,\cdots,T-1$.

Since a batch of u_{imt} parts loaded onto machine group m needs P_{im} periods to complete the processing, the total number of parts which are being processed by machine group m during time period t must not exceed its processing capacity, i.e.,

Machine Capacity Constraints

$$\sum_{i=1}^{I} \sum_{\tau=t-P_{im}+1}^{t} u_{im\tau} \le C_{mt} + O_{mt}, \quad \forall\, m, \forall\, t. \quad (2.2)$$

Similarly, the total number of parts lodging in buffer b at period t should not exceed its capacity,

Buffer Capacity Constraints

$$\sum_{i=1}^{I} X_{ibt} \le S_b, \qquad \forall\, b=2,\cdots,M, \forall\, t \quad (2.3)$$

The use of overtime can not exceed the overtime capacity, i.e.,

Overtime Capacity Constraints

$$0 \le O_{mt} \le C^{o}_{mt}, \qquad \forall\, m, \forall\, t. \quad (2.4)$$

Moreover, the following integrality constraints should also be satisfied:

X_{ibt}, u_{imt} and O_{mt} are nonnegative integers,

$$\forall\, i, \forall\, b, \forall\, m, \forall\, t. \quad (2.5)$$

The objective of production scheduling has three folds: (i) to deliver products just in time to avoid the overdue penalty and inventory cost of finished products, (ii) to reduce the holding cost of in–process inventories and (iii) to reduce the overtime cost. Define the earliness/tardiness penalty coefficient for per unit type i part completed at time t as

$$\psi_{it} = \begin{cases} A_i(d_i - t), & t > d_i \\ B_i(t - d_i), & t \le d_i \end{cases}, \quad A_i, B_i \in R^1.$$

We then formulate the scheduling problem as

$$(P) \quad \min_{\mathbf{u},O} \sum_{i=1}^{I} \sum_{t=1}^{T} \Big(\psi_{it} Z_{it} + \sum_{b=2}^{M} h_{ib} X_{ibt}\Big) + \sum_{m=1}^{M} \sum_{t=1}^{T} l_{mt} O_{mt}$$

subject to constraints (2.1–5).

3. NOMINAL ALGORITHM DEVELOPMENT

The scheduling problem (P) formulated in Section 2 for a FFS under nominal conditions is an integer programming problem of NP–hard [7]. Instead of looking for the optimal solution, we now develop for it a near–optimal and computationally efficient solution algorithm.

3.1 Lagrangian Relaxation and Decomposition

Observing that the coupling among production flows of different part types is caused by their competition for processing and storage resources, we apply Lagrangian relaxation to machine and buffer capacity constraints (2.2) and (2.3) and form the dual problem of (P) as

$$(D) \quad \max_{\boldsymbol{\lambda}\ge 0,\, \boldsymbol{\pi}\ge 0} \Phi(\boldsymbol{\lambda},\boldsymbol{\pi}),$$

subject to (2.1), (2.4) and (2.5),

where λ_{mt} and π_{bt} are Lagrange multipliers,

$$\Phi(\boldsymbol{\lambda},\boldsymbol{\pi}) \equiv \sum_{i=1}^{I} \min_{\mathbf{u}_i} PL_i(\mathbf{u}_i,\boldsymbol{\lambda},\boldsymbol{\pi}) + \sum_{m=1}^{M} \sum_{t=1}^{T} \min_{O_{mt}} OL_{mt}(O_{mt},\lambda_{mt}) - \sum_{m=1}^{M} \sum_{t=1}^{T} \lambda_{mt} C_{mt} - \sum_{b=2}^{M} \sum_{t=1}^{T} \pi_{bt} S_b, \quad (3.1)$$

$$PL_i(\mathbf{u}_i,\boldsymbol{\lambda},\boldsymbol{\pi}) \equiv \sum_{t=1}^{T} \Big\{\psi_{it} Z_{it} + \sum_{m=1}^{M} \lambda_{mt} \sum_{\tau=t-P_{im}+1}^{t} u_{im\tau} + \sum_{b=2}^{M} (h_{ib}+\pi_{bt}) X_{ibt}\Big\}, \quad (3.2)$$

and

$$OL_{mt}(O_{mt},\lambda_{mt}) \equiv (l_{mt} - \lambda_{mt}) O_{mt}. \quad (3.3)$$

Note that for a given set of Lagrange multipliers $\boldsymbol{\lambda}$ and $\boldsymbol{\pi}$, there are two classes of independent subproblems in (3.1):

(1) Production Scheduling Subproblem:

$$(PS\text{–}i) \quad \min_{\mathbf{u}_i} PL_i(\mathbf{u}_i,\boldsymbol{\lambda},\boldsymbol{\pi})$$

subject to (2.1) and (2.5); and

(2) Overtime Capacity Allocation Subproblem:

$$(OCA\text{–}mt) \quad \min_{O_{mt}} OL_{mt}(O_{mt},\lambda_{mt})$$

subject to (2.4) and (2.5)

3.2 Solution Algorithms for Subproblems

A. Network Flow Algorithm for (PS–i)

The set of flow balance equations (2.1) of (PS–i) render themselves naturally to a network representation. Each node of the network corre–sponds to one flow balance equation. The arcs represent either part processing paths with u_{imt}'s as flows on them or parts carried over in buffers

between two time periods with X_{imt}'s as their flows. Buffer 1 corresponds to the source node while buffer M+1 to the sink node. Machine and buffer capacity constraints impose flow bounds on the arcs.

As the cost function of (PS–i) is linear in arc flows, a subproblem (PS–i) is essentially a minimum cost linear network flow (MCLNF) problem, which has an integer optimal solution [7]. We adopt the RELAX code developed by Bertsekas and Tseng [8] for solving each (PS–i).

B. Algorithm for (OCA–mt)

Each subproblem (OCA–mt) is a simply constrained, static, linear optimization problem. Given λ_{mt} and $\{u_i, \forall\ i\}$, we determine the solution O_{mt} to (OCA–mt) according to the complementary slackness conditions [7] as follows:

1) if $(l_{mt} - \lambda_{mt}) > 0$ then $O_{mt} = 0$;

2) if $(l_{mt} - \lambda_{mt}) < 0$ then $O_{mt} = C^{o}_{mt}$;

3) if $(l_{mt} - \lambda_{mt}) = 0$ then $O_{mt} = \text{Min}\{\ \text{Max}[\ 0,\ \sum_{i=1}^{I} \sum_{\tau=t-P_{im}+1}^{t} u_{im\tau} - C_{mt}\ ],\ C^{o}_{mt}\}$.

3.3 Subgradient Algorithm for the Dual

After solving all the subproblems for a given set of Lagrange multipliers (λ,π), we update (λ,π) according to the subgradient method of [9]. Interested readers may refer to [9] for more details.

3.4 Construction of a Good Feasible Schedule

The primal problem (P) is not a convex optimization problem because of integer decision variables. So solution to the dual generally results in an infeasible schedule, i.e., some of the capacity constraints ((2.2) or (2.3)) may be violated. We now develop an iterative algorithm that exploits the network structure of material flows and the marginal cost interpretation of Lagrange multipliers to adjust the dual solution to a near–optimal feasible schedule. Major steps of the algorithm are as followed.

CGFSA

Step 0 Initialize with the schedule obtained from solving the dual problem.

Do for *t* from *1* to *T*

Do for *m* from *1* to *M*

Step 1 Check if the capacity constraint of machine group *m* or the associated buffer are violated at time period *t*.

Step 2 If so, determine, according to the descending order of priority factors (PF) among different types as defined in [6], the type(s) of parts and quantity that should be removed from facility *m* (a machine group or a buffer) at time *t* to eliminate the excessive production flow.

Step 3 Remove the excessive production flow(s) from the production schedule. Since the type(s) of parts to remove is identified in *Step 2*, consider again the material flow network of the type and focus on the upstream and downstream subnets of the arc with the excessive flow. Pull the excessive amount of flow out of both subnets in a way that results in a minimum production cost change, i.e., by solving a MCLNF problem for each subnet. Then update arc flows of the material flow network according to the removal.

Step 4 Reschedule the removed production flow. This is done by first modifying the arc capacities of the flow network where the capacity of an arc belonging to the interval $[0,t-1]$ is set to the residual capacity of the corresponding facility while that of an arc in $[t,T]$ is set to the difference between the corresponding facility capacity and the flow of this arc from *Step 3*. The removed flow is then rerouted through the modified network by solving again a MCLNF problem.

Enddo

Enddo

4. FAST RESCHEDULING ALGORITHMS

Uncertainties in machine availability, material supply, demand quantity and demand types are frequently encountered in a factory and have significant impacts on production scheduling. When an unexpected event of these kinds occurs and makes the original schedule infeasible, rescheduling is needed to maintain schedule feasibility in a timely, economical and smooth way, which essentially requires a very similar function to the *CGFSA*. Ideas and steps of the *CGFSA* therefore constitute the backbone of our fast rescheduling algorithm developments.

4.1 Machine Availability

Suppose that some machines unexpectedly become unavailable at time period *t* and will last till time period $t+d$. Assume that there exists an emergent buffer space to temporarily store parts at the unavailable machines. To adjust the production schedule, we first update the machine capacities during $[t,t+d]$ and apply steps 1 and 2 of *CGFSA* to

determine the excessive production flows of the nominal production schedule due to machine capacity shortage. Since the production schedule over $[0,t-1]$ has been implemented, we need only construct material flow networks over the duration $[t,T]$ for rescheduling. Arc costs of these networks are calculated by using the stored nominal Lagrange multipliers. We then remove excessive flows from and reroute them into their respective flow networks by using steps 3 and 4 of the *CGFSA*.

4.2 Material Shortage

Consider a case where type—i parts are in short of materials and can not be processed as originally scheduled by machine group m at time period t and the required materials will not be available until time period $t+d$. We first use *Step 3* of *CGFSA* to remove from the production schedule the production flows of type—i parts scheduled at machine group m during period $[t,t+d]$. Then we reroute these flows into the production schedule. The rerouting procedure is basically the same as *Step 4* of the *CGFSA* except that the capacities of machine group m during $[t,t+d]$ are set to zero for the production network of part type i and are set to the residual values after removing type—i flows for production networks of other part types.

4.3 Demand Variation

Demanded quantities may often be changed during production. When demand for type i parts increases unexpectedly by q units at time t, we first use *Step 4* of *CGFSA* to push these q units of increase into the material network of type i parts over $[t,T]$. As this insertion may cause capacity violations in $[t,T]$, we then perform the *CGFSA* to check the feasibility of the capacity constraints and construct a good feasible schedule over $[t,T]$. As for a demand decrease, we can apply *Step 3* of the *CGFSA* to pull the decreased amount out of the production schedule over $[t,T]$.

4.4 Producing New Types of Parts

In a flexible production system, it is important to handle unexpected orders of small amounts but very high priority (i.e., hot orders) after a production schedule has been developed. As the overall characteristics of the scheduled system may not change much after adding a new order of low volume, multipliers associated with the original schedule should be a good starting point to re—run the nominal scheduling algorithm and should be efficient in computation. Hoitomt et. al. [10] has exploited this idea for job shop scheduling. We adopt the same idea here and our results in Section 5 also confirm its effectiveness for our flow shop problem.

5. NUMERICAL STUDY

Numerical experimentations are conducted in this section to demonstrate the feasibility and effectiveness of our scheduling method. All of our experimentations are performed on a SUN/SPARC—IPC workstation.

5.1. A Realistic Example

The flow shop mainly produces four different *brands* of products every season. There are eight machine groups in the shop. Data about this shop is listed in Tables 5.1 and 5.2. Assume that this shop receives ten orders, each requesting parts of one of the four basic brands. We treat the parts of one order as one *type*. The corresponding production demands and due dates are listed in Table 5.3.

Table 5.1 Data of Four basic Brands of Parts

Brand	Processing Time								Demand
	1	2	3	4	5	6	7	8	
A	3	4	1	4	4	X*	2	2	90
B	3	5	1	3	X	5	2	2	60
C	3	4	3	X	4	X	2	2	30
D	3	X	1	3	X	5	2	2	30

* X denotes that the part need not be processed on this machine group.

Table 5.2 System Capacity

Machine Group Index	1	2	3	4	5	6	7	8
Machine Capacity	20	45	45	30	10	25	15	15
Buffer Capacity	500	100	80	110	90	60	75	80
Overtime Capacity	10	22	22	15	5	12	7	7

Table 5.3 Data of Parts

Type	1	2	3	4	5	6	7	8	9	10
Brand	A	B	C	D	A	B	A	B	A	B
Due Date	100	80	35	70	50	60	87	90	20	45
Quantity	30	15	30	30	25	15	20	15	15	15

In applying our nominal algorithm to this example, we initialize all the multipliers as zero and the scheduling time horizon is set to 110 periods. As a result it takes 3748.23 CPU seconds to obtain the

feasible schedule with cost of 520.000. The coressponding dual cost is 515.998 and the resultant relative duality gap is 0.77%. We consider such a solution as near–optimal.

5.2 Rescheduling

We now compare the effectiveness of re–scheduling between using the fast rescheduling algorithms and direct rescheduling by the nominal scheduling algorithm. Consider the following five perturbed scenarios of the previous example:

M1) machine group 6 suddenly loses 11 units of its 25–unit capacity from time period 32 to time period 36;

M2) the scheduler is notified at time period 50 that there will be short of type–2 parts at machine group 3 from time period 67 to 72;

M3) the scheduler is notified at time 30 that the demand of type–5 parts is reduced by 5 units;

M4) the scheduler is notified at time 20 that the demand of type–4 parts is increased by 3 units and;

M5) a new order of 5 units of brand D parts is to be rushed through the system starting from time period 53.

Table 5.4 lists the rescheduling results of these five test scenarios. Note that the initial multipliers for direct rescheduling by using the nominal scheduling algorithm are set to the optimal values from scheduling the nominal scenario. From these results we can see that our fast rescheduling algorithms are very effective for handling small perturbations: the computation times required are well within the limitation for real time application (less than 3 seconds except for M5) and the adjusted schedules are often as good as those by direct rescheduling. In M5, the fast and the direct rescheduling algorithms are essentially the same. So their CPU times are the same.

Table 5.4 Test Results of Rescheduling Algorithms

Scenario		M1	M2	M3	M4	M5
Our Fast Rescheduling Algorithms	CPU Time (sec)	1.61	1.11	1.59	2.64	1161.71
	Cost	540.10	521.50	515.50	529.60	588.00
Re-Solving Our Nominal Algorithm	CPU Time (sec)	1582.02	473.52	991.07	1115.92	1161.71
	Cost	540.10	521.50	515.50	525.40	588.00
	Dual Cost	538.10	518.86	513.50	522.90	576.36
	Duality Gap	0.37%	0.51%	0.39%	0.48%	1.98%

6. CONCLUSIONS

We have developed an effective, optimization model–based methodology for scheduling and re–scheduling a class of flexible flow shops. Numerical results demonstrated its effectiveness and potential for real applications. However, the handling of rework and scrap due to defects is a significant issue in pproduction scheduling and is not addressed in this paper. Extension of our methodology to include this aspect will be reported in an upcoming paper.

REFERENCES

[1] P. G. Ranky, *Computer Integrated Manufacturing*, Prentice–Hall, UK, 1986.

[2] J. Blazewicz, G. Finke, R. Haupt and G. Schmidt, "New Trends in Machine Scheduling," *European Journal of Operational Research* 37, pp. 303–317, 1988.

[3] S. F. Smith, P. S. Ow, J.–Y. Potvin, N. Muscettola and D. C. Matthys, "An Integrated Framework for Generating and Revising Factory Schedules,", *J. Opl. Res. Soc.*, Vol. 41, No. 6, 1990.

[4] D. P. Bertsekas, *Dynamic Programming*, Prentice–Hall, Inc., 1987.

[5] B. P. Bona, C. Greco and G. Menga, "Hybrid Hierarchical Scheduling and control Systems in Manufacturing," *IEEE Trans. on Robotics and Automation*, Vol. 6, No. 6, Dec 1990, pp. 673–686.

[6] S.–C. Chang, D.–Y. Liao, and F.–S. Hsieh, "Scheduling Flexible Flow Shops of No Setup Cost by a Lagrangian Relaxation and Network Flow Approach," *Proc. of the 1991 IEEE Intern. Conf. on R & A*, April 1991, pp. 1054–1059.

[7] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*, Prentice–Hall, Inc., 1982.

[8] D. P. Bertsekas and P. Tseng, "Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problems," *Operations Research*, Vol. 36, No. 1, 1988.

[9] M. Held, P. Wolfe and H. Crowder, "Validation of Subgradient Optimization," *Math. Programming*, Vol. 6, pp. 62–88, 1974.

[10] D. J. Hoitomt, P. B. Luh, E. Max and K. R. Pattipati, "Scheduling Jobs with Simple Precedence Constraints on Parallel Machines," *Control Systems Magazine*, Vol. 10, No. 2, 1990.