Minimizing Makespan for a Flexible Flow Shop Scheduling Problem in a Paint Company

Sandipan Karmakar and Biswajit Mahanty
Department of Industrial Engineering and Management
Indian Institute of Technology
Kharagpur - 721 302, India

Abstract

This paper focuses on a Flexible Flow Shop (FFS) problem in a paint company in an attempt to minimize production makespan. The FFS problem is characterized by multiple products being produced in machines in stages with sequence dependent set up times and infinite intermediate storages. As the FFS problem is computationally complex, two heuristic methods are employed to solve the Mixed Integer Linear Programming (MILP) problem formulated. The first heuristic is based on the Theory of Constraints and the second heuristic is based on a Genetic Algorithm. The Genetic Algorithm approach has resulted in a better production makespan.

Keywords
Flexible Flow Shop, Mixed Integer Linear Programming, Theory of Constraints, Genetic Algorithm.

1. Introduction
A flexible flow shop (FFS) consists of a flow line with several parallel machines on some or all production stages. Multiple products are produced in such a flow line. While all the products follow the same linear path through the system, all of them may not visit all the stages of production. On each of the stages, one of the parallel machines has to be selected for the production of a given product. The production of a product consists of multiple operations, one for each production stage. When an operation is started on a machine, it must be finished without interruption.

The present FFS problem is taken up in a large paint company in India having a variety of products. Every paint product consists of two ingredients called base and tinters. Tinters are color producing agents, which when mixed in some pre-specified amount with colorless bases produce different paint products with different shades. Tinters are strategically important to the company and their production is important to the company's overall productivity. The company is found to be having production shortfall in plants. One of the primary reasons of this shortfall lies in improper scheduling of products to the machines. It is therefore important to find out the best production plan or schedule in which the different activities are scheduled to the existing resources in order to minimize the make span of the production lines.

The production process of the tinters consists of a total of nine products in a product line (named Product 1 to Product 9) in a flexible flow shop with sequence dependent set up times, dedicated machines at few stages, and infinite intermediate storage. Certain assumptions that are made include a single product group (color producing agent), a single product line, no waste generation due to change over from product to product, and zero machine breakdown. Figure 1 gives the process sequence and stage wise machine details.

2. Literature Review
Flexible flow shop scheduling problems are usually solved by branch and bound algorithms (Salvador (1973), Brah and Hunsucker (1991), Carlier and Neron (2000)) and by integer programming (Kurz and Askin (2004), Sawik (2002)). A major disadvantage of the integer programming approaches to scheduling is the need for solving large mixed-integer programs to obtain meaningful optimal solutions (Jiang and Hsiao, 1994). FFS problems thus solved use infinite buffers and sequence-dependent setup times. Some recent works has also considered blocking processor and limited buffers (Sawik 2002). Kurz and Askin, 2004 have emphasized on the need for heuristic or metaheuristic algorithms for solving large and complex FFS problems as they are NP hard for all traditional optimality criteria. Heuristics methods, the most well known methods of solving NP hard FFS problems, are broadly categorized into

two types called holistic approaches and decomposition approaches. In the holistic approaches of heuristic methods, the scheduling of jobs to the processors is done in an integrated way. Whenever a processor goes empty, the jobs are assigned to it using the popular dispatching rules. Agnetis et al. (1997) considered such a procedure ('pull rule') in the automotive industry, comparing the results with a so-called 'push rule'. Negemann (2001) compared several tabu search heuristics with a simulated annealing procedure. Tavakkoli-Moghaddam et al. (2006) proposed a queen-bee-based genetic algorithm to schedule flexible flow lines while considering the blocking processor.

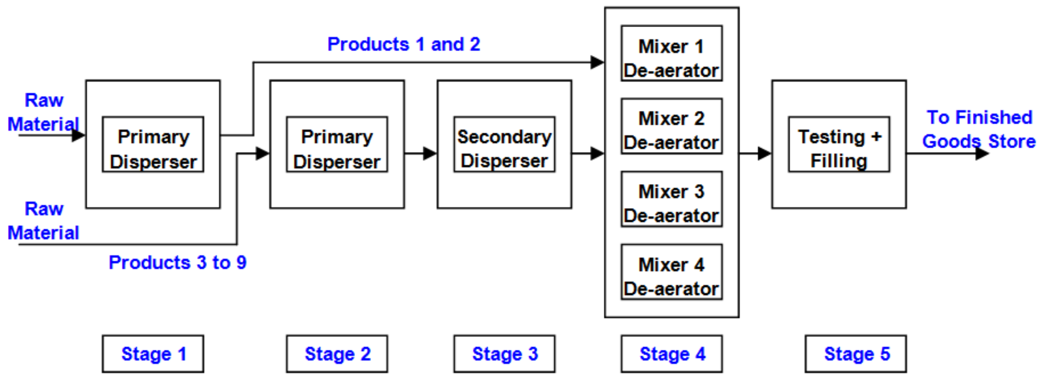

Figure 1: Process Sequence and Stage-wise Machine Details

Decomposition approaches divide the overall scheduling problem into segments that are considered consecutively neglecting the interdependencies of the different segments requiring strategies to effectively handle those. There are stage-oriented, job-oriented, and problem-oriented decomposition approaches. Three variants of flow line adaptations were developed by Ding and Kittichartphayak (1994) for make span minimization. Lee and Vairaktarakis (1994) presented a similar algorithm for two stages, which they extended to an unlimited number of stages by successively concatenating schedules from two-stage sub-problems. A comparative study for make span and mean flow time minimization has been conducted by Brah and Loo (1999). Jin et al. (2002) considered a genetic algorithm to generate a good job sequence by minimizing the make span. Kurz and Askin (2004) developed a genetic algorithm minimizing the make span that included sequence-dependent setup times for jobs that did not have to visit all production stages. When more than two stages were considered, the genetic algorithm outperformed the procedures presented by Kurz and Askin (2003). Serifoglu and Ulusoy (2004) solved a problem where processing an operation may simultaneously require several machines of a stage for minimizing make span.

3. Formulation of the Mixed Integer Linear Programming (MILP) Problem

The FFS scheduling problem considered in the paper is formulated as an MILP problem based on the traveling salesman problem (TSP). The basic information required to describe the jobs (products) are the following:

(i) Processing time (p_j): the amount of time required in processing of the job j
(ii) Ready time (r_j): the point of time at which the job j is available
(iii) Due date (d_j): the point of time at which the job j is due to complete
(iv) Completion time (C_j): the point of time at which the processing of job j is completed
(v) Flow time (F_j): also called as Turnaround time, is the time a job j spends in a system given by ($C_j - r_j$)
(vi) Lateness (L_j): the time by which the completion time of job j exceeds its due date, given by ($C_j - d_j$)
(vii) Tardiness (T_j): Lateness of job j if it fails to meet its due date or zero otherwise and is given by max(0, L_j)

The MILP formulation has a number of assumptions. The formulation uses deterministic time data. All the jobs are available at zero time and they are not produced in batches. The set up times are sequence dependent. One machine can process one job at a time without preemption and without error. Machines are available at all the times with no unwanted breakdowns or no prescheduled maintenance. There is no travel between the stages and intermediate buffer storages between stages are infinite. The objective of the MILP formulation is to minimize the make span of one full batch of nine products. Makespan is defined as the completion time of the last job in a production sequence.

The Symbols used in formulation are as following

g : No. of stages (index t))	n : No. of jobs to be scheduled (index i, j)
m_t : No. of machines at stage t	g_j : Last stage visited by job j
p_{ti} : Processing time of job i at stage t	s_{tij}: Set up time of job j immediately preceded by job i at stage t
S_i : Set of stages visited by job i	S_t : Set of jobs that visit stage t
C_{ti} : Completion time of job i at stage t	x_{tij}: 1, if job i is immediately preceded by job $j$at stage t, 0 otherwise

The decision variables are: $x_{tij} \in \{0, 1\}$ and $C_{tj} \geq 0$

The objective function can now be defined as:

Minimize the makespan ($C_{\max}$) = max $\{C_{gj}\}$ where C_{gj}: Completion time of job j at the last stage g_j (1)

Subject to the following constraints:

No machine should sit idle at any stage at any time: $\sum_{j=1}^{n} x_{t0j} = m_t, \forall t = 1......g$ (2)

One and only one machine is scheduled for a single job at each stage.

$$\sum_{j \in \{S_t, n+1\}} x_{tij} = 1, \forall i; t \in S_i \qquad \sum_{i \in \{0, S_t\}} x_{tij} = 1, \forall j; t \in S_i \qquad \text{(3) and (4)}$$

Non-preemption at machines - i.e. once a job has started processing in a machine must not leave the machine before its processing finishes.

$$c_{tj} - c_{ti} + M_t(1 - x_{tij}) \geq s_{tij} + p_{tj}, i = 0......n, j = 1......n, t \in S_i$$
$$M_1 = \sum_{i=1}^{n} (p_{1i} + \max(s_{1ji})) \text{ and } M_t = M_{t-1} + \sum_{i=1}^{n} (p_{ti} + \max(s_{tji})) \qquad (5)$$

M_t : the upper bound of completion time of processing at stage t

Once a job has not completed processing at the previous stage can not start processing at the next stage.

$$c_{tj} - c_{(t-1)j} + M_{tj}(1 - x_{tij}) \geq s_{tij} + p_{tj}, i = 0......n, j = 1......n, t \in S_i \qquad (6)$$
$$M_{tj} = p_{tj} + \max(s_{tij}) \qquad \text{with } i, j \in \{0, n+1\}, t = 1, 2......g$$

Jobs are not assigned to the stages which they don't visit: $x_{tij} \leq p_{tj};\ x_{tji} \leq p_{tj}$ (7)

Completion time of a job at stage t, which does not visit stage t, is set to the job's completion time at stage t–1

$$c_{1j} - c_{10} \geq M_t p_{1j};\ c_{tj} - c_{(t-1)j} \geq M_t p_{tj} \text{ where } j = 1......n, t = 1......g \qquad (8)$$
$$c_{tj} \geq c_{t0} \qquad (9)$$

Make span of the schedule must be always equal or greater than the completion time of last product at last stage

$$z \geq c_{gjj} \qquad (10)$$

The FFS problem for the paint company, considered in this paper, is found to have 546 constraints and 304 decision variables. The large size of the problem necessitated solution of the problem by heuristic methods.

4. Heuristic based on Theory of Constraints

Theory of Constraints (TOC) is an easy-to-understand manufacturing philosophy already implemented in various manufacturing environments. TOC states that a local optimum is not an optimum at all, and that the overall system performance is governed by the bottleneck resource (Goldratt, 1997). The idea is thus to find the bottleneck stage and to optimize the whole system performance by exploiting it. The basic approach of the TOC-based heuristic presented in this paper consists of three steps as explained below:

Step 1: Identifying the Bottleneck Stage

(i) Compute the modified processing times by adding the minimum setup times with the processing times.
(ii) Compute the 'Flow Ratio' for every stage by summing job-wise the ratio of modified processing times to that of the number of machines in a stage
(iii) Select the stage with maximum Flow ratio as the bottleneck stage
(iv) Compute the release times for all jobs as the sum of modified processing times prior to the bottleneck stage
(v) Compute the estimated flow as the sum of flow ratios of all the stages
(vi) Compute bottleneck trails for all the jobs as the difference between Estimated flow and sum of modified processing time prior to the bottleneck stage

Step 2: Sequencing the Bottleneck Stage

(i) Schedule jobs in non-decreasing order of release times for all the jobs
(ii) If tie exists in release times of any jobs then rank them by non-decreasing values of bottleneck trails else prioritize them by their importance
(iii) Schedule jobs on the machines according to the precedent ranking. If there is more than one machine available at time *T*, then assign the next job to the machine with less workload until time *T*.
(iv) Compute starting and completion times for each job on the bottleneck stage.

Step 3: Sequencing on Non-Bottleneck Stages

(i) Stages after the bottleneck station: proceed in a similar way as for bottleneck stage but a given job may be scheduled as soon as it is completed by the precedent stage
(ii) Stages before the bottleneck station: in order to respect the delivery time to the bottleneck station, jobs are scheduled according to release times and bottleneck trails and processing times criteria explained in step 2

Results of applications of TOC based heuristic

The TOC based heuristic presented above is applied for the FFS problem considered in this paper. In this problem, there are nine products being processed in eight machines in five stages. The process sequence and stage wise machine details are given in Figure 1. Table 1 below gives the necessary computations for this problem. After flow ratios are computed for every stage by making use of the modified processing times for the nine products, stage 3 is identified as the bottleneck stage as it is having the highest flow ratio. Afterwards, the release times and bottleneck trails are calculated. The sequence of products in bottleneck stage will be: [1]=Product 4, [2] =Product 7, [3] =Product 5, [4] =Product 9, [5] =Product 6, [6] =Product 8, [7] =Product 3.

Table 1: Computation of Flow Ratios and identification of Bottleneck stage

Modified processing times	**Stage 1**	**Stage 2**	**Stage 3**	**Stage 4**	**Stage 5**	**Release Times**	**Bottleneck Trails**
	P_{tj}	P_{tj}	P_{tj}	P_{tj}	P_{tj}	R^B_j	D^B_j
Product 1	172			351	135	0	0
Product 2	178			361	136	0	0
Product 3		250	167	241	142	250	4663.8
Product 4		111	194	197	162	111	4802.8
Product 5		165	211	317	141	165	4748.8
Product 6		180	301	212	137	180	4733.8
Product 7		120	293	241	136	120	4793.8
Product 8		202	138	181	136	202	4711.8
Product 9		169	222	198	141	169	4744.8
Flow Ratio	350	1197	**1526**	574.75	1266		

The production makespan is computed by making use of the TOC based heuristic presented earlier. The makespan comes out to be 2040 min. At the last stage, the sequence of the products will be P3-P1-P2-P7-P5-P9-P6-P8-P3.

5. Heuristic based on Genetic Algorithm (GA)

Genetic algorithms are implemented as a computer simulation in which a population of abstract representations (called chromosomes or genotypes) of candidate solutions (called individuals, creatures, or phenotypes) to an optimization problem evolves toward better solutions. Traditionally, solutions are represented in binary as strings of

0s and 1s, but integer or literal permutation encoding are best used in combinatorial optimization problems. Their uses are found in scheduling, transportation, airline crew scheduling, knapsack problems and so on. Once the genetic representation and the fitness function are defined, GA proceeds to initialize a population of solutions randomly, and then improve it through repetitive application of mutation, crossover and selection operators. As the present problem is a flexible flow shop problem with more than one identical parallel machine in at least one stage, so strings should be capable of capturing the information on job sequence as well as machine allocation in the stages where there are multiple identical parallel machines. So encoded solutions have been made of two parts, in first part there is a simple job sequence and in second part the machine numbers to which the jobs are allocated at the fourth stage which are having four mixers. These four machines at stage 4 are numbered as 4,5,6,7. Part 1 is encoded as per permutation encoding. Figure 2 shows the encoding of the chromosome:

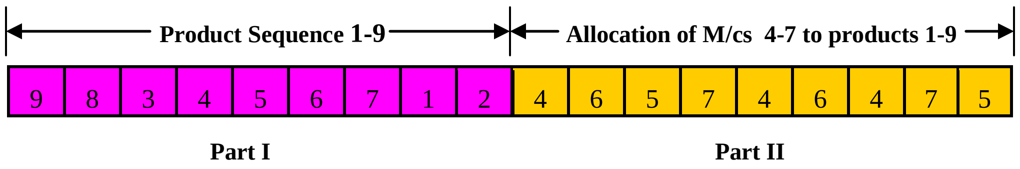

Figure 2: Encoding of the Chromosome

The fitness function for the problem is expressed as:

$$\text{Fitness Maximize} \quad f = \; = -\text{Cmax} \; = -\max\ \{C_{gj}\} \tag{11}$$

where C_{gj}: Completion time of job j at the last stage g_j.

The make span of a schedule is computed in an iterative way, as the completion time of a job at any stage depends upon the completion time of that job at its previous stage and that of the job which is preceding it. The completion time of a job j is computed in the following way when at stage k job j is preceded by job i:

$$C_{k,\,j} = \max[C_{k,\,i}; C_{k-1,\,j}] + S_{k,\,i,\,j} + p_{k,\,j} \tag{12}$$

Where $S_{k,i,j}$ is the set up time of job j at stage k when it is immediately preceded by job i and p_{kj} is the processing time of job j at stage k.

Implementation of GA

Tournament selection with replacement is chosen for the present case. In tournament selection randomly a number of chromosomes are selected from the initial population (called tournament size) then a single chromosome with best fitness value is kept in the mating pool and rest are kept back in initial population. Crossover function is applied separately for the two parts of the chromosomes. For part I the crossover is used as Partially Matched crossover (PMX) operator. But for part II a single point crossover function is applied. The mutation is carried out simply by interchanging two part 1 locations and also by interchanging two part II locations. Figure 3 shows the Final Makespan values for different crossover and mutation probability. From GA runs the minimum make span is found to be 1925 minutes corresponding to crossover probability (p_c) =1 and mutation probability (p_m) = 0.2. The corresponding product sequence is: P2-P4-P8-P3-P5-P7-P1-P6-P9. The allocations of 4 machines at stage 4 are: P2: 4, P4: 7, P8: 4, P3: 6, P5: 5, P7: 7, P1: 4, P6: 5, and P9: 6. The make span of 1925minutes ≈ 32.08h contributed to a marked improvement in the production to the company (increase by over 30%).

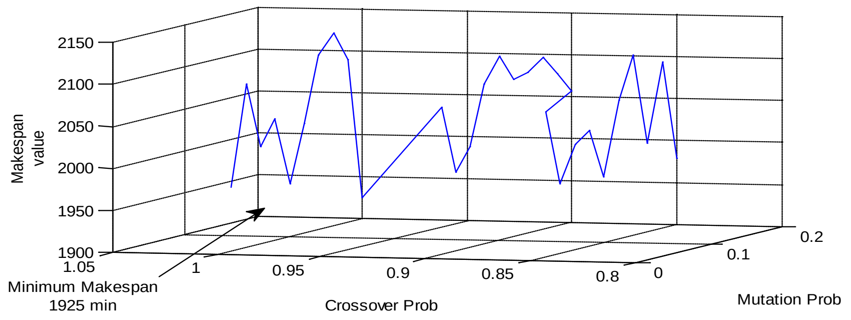

Figure 3: Final Makespan values for Different Crossover and Mutation Probability

6. Conclusions

This paper has considered a Flexible Flow Shop (FFS) problem in a paint company. The problem was formulated with the use of Mixed Integer Linear Programming (MILP) method. The resulting formulated was computationally complex and thus warranted heuristic methods for its solutions. Two heuristic methods were employed. The first method employed was the Theory of Constraints and the second method was based on a Genetic Algorithm. The Genetic Algorithm approach was found to result in a better production makespan.

References

1. Agnetis, A., Pacifici, A., Rossi, F., Lucertini, M., Nicoletti, S., Nicolo, F., Oriolo, G., Pacciarelli, D., Pesaro, E., 1997, “Scheduling of flexible flow lines in an automobile assembly plant,” European Journal of Operational Research, 97, 348–362.
2. Brah, S.A., Hunsucker, J.L., 1991, “Branch and bound algorithm for the flow shop with multiple processors,” European Journal of Operational Research, 5 (1), 88–99.
3. Brah, S.A., Loo, L.L., 1999, “Heuristics for scheduling in a flow shop with multiple processors,” European Journal of Operational Research, 113 (1), 113–122.
4. Carlier, J., Ne´ron, E., 2000, “An exact method for solving the multi-processor flow-shop,” RAIRO Operations Research, 34, 1–25.
5. Ding, F.Y., Kittichartphayak, D., 1994, “Heuristics for scheduling flexible flow lines,” Computers & Industrial Engineering, 26, 27–34.
6. Goldratt, E.M.,1997, “Critical Chain”, North River, Great Barrington, MA.
7. Jiang, J., Hsiao, W., C., 1994, “Mathematical programming for the scheduling problem with alternate process plans in FMS,” Selected papers from the 16th annual conference on Computers and industrial engineering, Pergamon Press, Inc. Elmsford, NY, USA, 15 – 18.
8. Jin, Z.H., Ohno, K., Ito, T., Elmaghraby, S.E., 2002, “Scheduling hybrid flowshops in printed circuit board assembly lines,” Production and Operations Management, 11 (2), 216–230.
9. Kurz, M.E., Askin, R.G., 2003, “Comparing scheduling rules for flexible flow lines,” International Journal of Production Economics, 85 (3), 371–388.
10. Kurz, M.E., Askin, R.G., 2004, “Scheduling flexible flow lines with sequence-dependent setup times,” European Journal of Operational Research, 159 (1), 66–82.
11. Lee, C.Y., Vairaktarakis, G.L., 1994, “Minimizing makespan in hybrid flowshops,” Operations Research Letters, 16, 149–158.
12. Negemann, E.G., 2001, “Local search algorithms for the multi-processor flow shop scheduling problem,” European Journal of Operational Research, 128 (1), 147–158.
13. Salvador, M.S., 1973, “A solution to a special class of flow shop scheduling problems,” In: Elmaghraby, S.E. (Ed.), Symposium on the Theory of Scheduling and Its Applications, Springer, Berlin, 83–91.
14. Sawik, T., 2002, “Balancing and scheduling of surface mount technology lines”, International Journal of Production Research, 40(9); 1973-1991.
15. Serifoglu, F.S., Ulusoy, G., 2004, “Multiprocessor task scheduling in multistage hybrid flow-shops: A genetic algorithm approach,” Journal of the Operational Research Society, 55, 504–512.
16. Tavakkoli, R., M., Gholipour, Y., Safaei, N., 2006, “A hybrid simulated annealing for capacitated vehicle routing problems with the independent route length”, Applied Mathematics and Computation, 176(2), 445-454.