OPTIMIZATION OF FLEXIBLE FLOW SHOP SCHEDULING WITH
SEQUENCE DEPENDENT SETUP TIME AND LOT SPLITTING

Vinit Saluja1 and Ajai Jain2

1Assistant Professor, Geeta Engineering College, Naultha,
Email- saluja_vinit123@rediffmail.com
2Professor, National Institute of Technology,Kurukshetra,
Email- ajaijain12@gmail.com

Abstract:
This paper presents optimization of makespan for ‘m’ stages ‘n’ jobs flexible flow shop scheduling problem with sequence dependent setup time using genetic algorithm (GA) approach. A restart scheme as suggested by Ruiz et al., 2006 has also been applied to prevent the premature convergence. The paper also assesses the affect of lot splitting size on makespan. Three case studies are taken into consideration. Five simulation runs for each lot size are taken and minimum value among them is taken as optimal makespan. Result shows that as the lot splitting size increases the makespan decreases, reaches minimum value and then increases with further increase in lot splitting size. Thus there is an optimal lot splitting size which results in the minimum makespan. For considered case studies, the lot splitting size is three.

Keywords:Flexible Flow shop, Genetic algorithm, Makespan, Lot splitting.

1. Introduction
Scheduling is a process of assigning a set of tasks to resources over a period of time (Pinedo2001). Effective scheduling plays an important role in today’s competitive manufacturing world. Performance criteria such as machine utilization, manufacturing lead times, inventory costs, meeting due dates, customer satisfaction, and quality of products are all dependent on how efficiently the jobs are scheduled in the system. A flow shop is characterized by a unidirectional flow of work, i.e. all jobs have same processing order through all the machines. In simple flow shop, each machine centre includes just one machine and in flexible flow shop for at least one machine centre or stage, there exists more than one machine available for processing (Yaurima Victor et al. 2009). The flow shop problem consists of ‘m’ machines and ‘n’ jobs and the scheduler objective is to find an optimal ordering of ‘n’ jobs (Baker et al. 2009). Flow shop scheduling has been widely used in many industrial applications such as manufacturing of variety of printers in computer industry, tyre manufacturing industry and printed circuit board manufacturing industry.

In actual practice, many problems are encountered in which some time is to be spent in bringing a given facility to a desired state of processing the job. The time spent is called setup time. If the time required to setup machine for the next job is independent of the job that was the immediate predecessor on the machine then setup time is called sequence independent and if the setup time is dependent on the job that was the immediate predecessor on the machine it is called sequence dependent. In a survey of industrial schedulers, 70% of the schedulers reported that they had to deal with sequence dependent setups (Luh et al. 1998).

A lot consists of several discrete and identical items that are to be processed on several machine configured as a flow shop. Instead of transferring the entire lot after all of its item have been processed on a machine, the transferring of items of the lot in smaller batches is called subloting. This technique of splitting a lot into sublots and processing different sublots simultaneously over different machine, albeit still maintaining their movement over the machines in accordance with their flow shop configuration, is called Lot splitting (Marimuthu et al. 2008).

2 Literature Review
Several researchers have addressed the problem of flow shop scheduling. Some important contributions are discussed below.

Yoon Suk Hun et al. (2002) proposed a hybrid genetic algorithm (HGA) for n job m machine lot streaming flow shop scheduling problem with equal size sub lot and infinite capacity buffer between successive machines. The proposed HGA overcome the premature convergence and maintain search power of genetic algorithm (GA) by using Non Adjacent Pairwise Interchange (NAPI) method. The performance of HGA approach was compared with that of NAPI method and the result of computational experiment shows HGA works well.

Marimuthu et al. (2005) evaluated heuristic search algorithm with lot streaming for a two machine flow shop problem with multiple jobs to minimize makespan. Three heuristic search
algorithm are evaluated i.e. Bakers Algorithm (BA), Genetic Algorithm (GA) and Simulated Annealing (SA) algorithm. Result indicates that GA and SA outperform Bakers Algorithm. Among GA and SA, GA outperforms SA in almost all problems.

Mirimuthu et al. (2008) proposed two evolutionary algorithms namely genetic algorithm (GA) and hybrid evolutionary algorithm (HEA) for scheduling ‘n’ jobs ‘m’ machine flow shops under lot sizing environment. The objective is to minimise the makespan/flow time. The performance of proposed GA and HEA are compared with Baker’s algorithm (BA) and Simulated Annealing (SA) algorithm. Computational result shows that the proposed algorithms GA and HEA are capable of producing optimal solution.

Yaurima Victor et al. (2008) proposed a modified Genetic Algorithm to minimise makespan of hybrid flow shop with unrelated machines, sequence dependent time and availability constraint. Authors found that the algorithm gives 6.27% better value of the objective function in average.

Jarboui Bassem et al. (2011) proposed a hybrid genetic algorithm for solving no wait flow shop scheduling problem to minimise makespan and the total flow time. The variable neighborhood search (VNS) is used as an improvement procedure in the last step of the genetic algorithm (GA). VNS helps to prevent the algorithm from being stuck into local optima.

Pang King Wah (2012) presented a Genetic Algorithm based heuristic approach to solve the problem of two machine no wait flow shop scheduling with setup time on machine is class dependent and the objective is to minimize the maximum lateness of the job processed. Computational experiments show that significant improvement is attainable using the proposed method when compared with earliest due date (EDD) and merge heuristic algorithm.

Wang Shijin et al. (2013) proposed a genetic algorithm for two stages no wait hybrid flow shop problem with a single machine on the first stage and multiple parallel identical machines on the second stage to minimise makespan. Results showed that the genetic algorithm with linear order crossover (LOX) as crossover operator and swap as mutation operator performs better in terms of mean percentage deviation of the solution. Authors stated that future effort can be made by embedding and testing other crossover and mutation operators.

Mirabi Mohammad et al. (2014) proposed a novel hybrid genetic algorithm (HGA) with three genetic operators. Proposed HGA applies a modified approach to generate a pool of initial solutions, and also uses an improved heuristic called the iterated swap procedure to improve the initial solutions. Computational experimental results show that the proposed HGA performs very competitively with respect to accuracy and efficiency of solution.

Li Dongniet al. (2014) evaluated heuristic-search genetic algorithm (HSGA) for scheduling problem of a multi-stage hybrid flow shop (HFS) with single processing machines and batch processing machines. Computational results indicate that as compared with meta-heuristics, the HSGA has a significant advantage with respect to the computational efficiency.

3 Problem Formulation

Literature review reveals that flexible flow shop scheduling problem with consideration of sequence dependent setup times and lot splitting has not been attempted by the researchers and this is the first attempt in this direction. In the present work, an attempt is made to optimize flexible flow shop scheduling problem with sequence dependent setup time and lot splitting for makespan as performance measure. Thus, the problem considered in the present work is described below:

“There is a order of ‘n’ jobs/part types to be processed on a set of ‘m’ stages in flexible flow shop. Each stage has m number of identical parallel machines and each parttype/job has n number of identical parts. The processing time of every operation of each job on machines is known in advance. The setup time of every operation on machines for each job is sequence dependent and is also known in advance. The objective is to find the optimal schedule for makespan as performance measure”

In accordance with three field notation (α/β/γ), the problem specified can be represented as:(Allahverdiet al. 2008)

\[ FF / ST_{sd} prec / C_{max} \]

4 Adopted Methodology

In order to find the optimal schedule for a flexible flow shop with sequence dependent setup times, a genetic algorithm based methodology is adopted in the present work. This section presents the details of the adopted methodology.

4.1 Representation

Encoding is the first step of GA. Each feasible solution is encoded as a chromosome (string or individual) also called a genotype (encoded solution). Various encoding schemes such as binary encoding, permutation encoding, value encoding etc. are available for flow shop scheduling. The selected encoding scheme should be able to represent the desired information. The present work utilizes permutation encoding for representation.

In
In this method, strings (chromosome) are coded as a sequence of numbers (genes) with each gene representing one of the operations of job involved. The specific operation represented by the genes was interpreted according to the order of the gene in the chromosome. The length of chromosome depends upon the number of jobs/parttypes and the number of stages available in flow shop on which jobs are to be processed. Figure 1 represents the chromosome structure for a production order consisting of 4 parttypes to be processed in flow shop having 2 stages.

**Figure 1 Structure of Chromosome**

<table>
<thead>
<tr>
<th>Parttype</th>
<th>1st gn</th>
<th>2nd gn</th>
<th>3rd gn</th>
<th>4th gn</th>
<th>5th gn</th>
<th>6th gn</th>
<th>7th gn</th>
<th>8th gn</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st op</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

Legend: gn = gene; op = operation

**4.2 Initialization**

In this step, initial population is generated having a fixed number of chromosomes and it is called population size. Initial population contains suitable number of solutions for the problem. Generally, initial population is generated randomly (Deb 2006). The present work considers a population size equal to 10 and it was generated randomly.

**4.3 Evaluation of fitness function**

Each chromosome gives a measure of fitness via a fitness function (evaluation or objective). It is performance evaluation of chromosomes. GA is naturally suitable for solving maximization problems (Deb 2006). Since objective functions in the present work was minimization of makespan \([f(x)]\), this minimization problem is transformed into maximization problem by following transformation rule (Deb 2006)

\[
F(x) = 1 / [1 + f(x)]
\]

where \(f(x)\) = value of makespan, 
\(F(x)\) = fitness function

**4.4 Selection**

Selection operator determines which chromosomes undergo for crossover and mutation. This decision is based on fitness of the chromosomes. During selection, the fitness of chromosomes is compared in order to choose the better chromosomes to drive search in good region of search space. Various selection methods such as Roulette Wheel Selection, Tournament Selection and Rank Selection can be used for selection in Genetic Algorithm. In the present study, Tournament selection was used with a selection pressure of 2.

**4.5 Crossover**

Crossover is used as the main Genetic operator and the performance of a GA is heavily dependent on it. A crossover operator is used to recombine two strings to get a better string. In the crossover, new strings are created by exchanging information among strings of the mating pool. A crossover operator is mainly responsible for the search of new strings. In this study, two point crossover with a crossover probability of 0.85 was used. Due to above crossover methodology, some illegal offsprings may generate. Then repairing was done to resolve the illegitimacy of off spring after mutation.

**4.6 Mutation**

Mutation is regarded as an integral part of a GA. Mutation generates an offspring solution by randomly modifying the parent’s feature. It helps to preserve a reasonable level of population diversity, and provides a mechanism to escape from local optima. In the present work, swap mutation with mutation probability of 0.15 was used.

**4.7 Repairing**

As discussed earlier, some illegal offsprings may generate during crossover. For this, repairing is needed to resolve the illegitimacy of off springs after mutation. A repairing procedure was utilized for this purpose. It checks the string from left to right. If at any point, some genes repeats more than required and some genes are missing, then excess genes at any place are replaced by missing genes randomly.

**4.8 Elitism**

After generating offspring’s, the parent strings of previous generation may get completely replaced. The best individuals can be lost in two cases (i) if, they are not selected to reproduce and/or (ii) they are destroyed by crossover or mutation (Mitchell 2002). Thus, elitism strategy is used in order to force GA to retain some number of the best individuals at each generation. In the present work, elitism transfers a good individual from previous population to population of next generation with the elitism rate of 0.9 and it means that 10% best population is carried on into the next generation. For example, if, population size is 10, then the total number of best individuals from previous generation to be carried into next generation is equal to one.
4.9 Termination Criterion
It refers to the stopping criterion for further exploration in the search space. In the present work, if the fitness value did not change for 150 iterations, GA terminates and algorithm reaches to the optimum value of makespan.

4.10 Restart Scheme
The population evolves as the GA proceeds. Sometimes, the population has a low diversity for the process to avoid becoming trapped in a local optimum. In order to avoid premature convergence, a restart scheme as suggested by Ruiz et al.(2006) was utilized. If the best seen fitness value is not promoted for more than a pre specified number of generations (no change), the restart phase commences to regenerate the population. In the present work, restart scheme was applied if there is no improvement in the fitness value for 10 successive iterations.

The above methodology was coded in MATLAB® and executed on Windows platform on Intel (R) Core 2 Duo CPU T6400@ 2.00 GHz, 4 GB RAM.

Following assumptions in line with previous studies (Zandieh et al. 2011, Marimuthu et al. 2005) was made in the present work.
(1) All jobs are available for processing at time zero (2) Machines never breakdown and available throughout the scheduling period (3) No machine may process more than one operation at the same time (4) Infinite buffer exist between stages and before the first and after the last stage (5) Job processing cannot be interrupted (6) A job cannot be processed on more than one machine at the same time (7) No pre-emption is allowed (8) A job follows precedence constraint of the operation.

5 Results and Discussion
Three case studies are taken into consideration in order to assess the affect of lot splitting. The optimization is carried out for each case study with crossover probability of 0.85 and mutation probability of 0.15. Five simulation runs are taken for each case study and minimum makespan among five runs is taken as optimal makespan. The details of these case studies are described below.

5.1 Case study 1
Table 1 provides the detail of flexible flow shop configuration and production order received. For this case study, optimization is carried out using adopted methodology as described in section 4. In order to assess the affect of lot size, it is varied from 1 to 6. Table 2 summarizes the optimal makespan for each lot size. Table 2 clearly indicates that as lot splitting size increases from 1 to 6 for each part type, initially minimum makespan reduces, reaches a minimum value and then increases. Thus there is an optimal lot splitting size. For this case study the optimal lot splitting size is 3. Fig. 2 shows the convergence curve for lot size 3.

<table>
<thead>
<tr>
<th>Lot Size</th>
<th>Optimal Makespan (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>8708</td>
</tr>
<tr>
<td>2</td>
<td>7941</td>
</tr>
<tr>
<td>3*</td>
<td>7570</td>
</tr>
<tr>
<td>4</td>
<td>7614</td>
</tr>
<tr>
<td>5</td>
<td>7693</td>
</tr>
<tr>
<td>6</td>
<td>7847</td>
</tr>
</tbody>
</table>

* indicates the optimal lot size

Figure 2 Convergence Curve for Lot Size 3 (Case study 1)

5.2 Case study 2
Table 3 provides the detail of flexible flow shop configuration and production order received. For this case study, optimization is carried out using adopted methodology as described in section 4. In order to assess the affect of lot size, it is varied from 1 to 6. Table 4 summarizes the optimal makespan for each lot size. Table 4 clearly indicates that as lot splitting size increases from 1 to 6 for each part type, initially minimum makespan reduces, reaches a minimum value and then increases. Thus there is an optimal lot splitting size. For this case study the optimal lot splitting size is 3. Fig. 3 shows the convergence curve for lot size 3.
Table 3Shop Configuration and Production Order Data  
(Case study 1)

<table>
<thead>
<tr>
<th>No. of jobs</th>
<th>09</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of stages</td>
<td>03</td>
</tr>
<tr>
<td>No. of machines per stage</td>
<td>03</td>
</tr>
<tr>
<td>Setup time</td>
<td>U[10-25]</td>
</tr>
<tr>
<td>Processing time</td>
<td>U[05-75]</td>
</tr>
<tr>
<td>Quantity of each job</td>
<td>U[60-90]</td>
</tr>
</tbody>
</table>

Table 4Makespan Corresponding to Lot Sizes (Case study 2)

<table>
<thead>
<tr>
<th>Lot Size</th>
<th>Optimal Makespan (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5276</td>
</tr>
<tr>
<td>2</td>
<td>5125</td>
</tr>
<tr>
<td>3*</td>
<td>5088</td>
</tr>
<tr>
<td>4</td>
<td>5129</td>
</tr>
<tr>
<td>5</td>
<td>5175</td>
</tr>
<tr>
<td>6</td>
<td>5223</td>
</tr>
</tbody>
</table>

* indicates the optimal lot size

Table 5Shop Configuration and Production Order Data  
(Case study 1)

<table>
<thead>
<tr>
<th>No. of jobs</th>
<th>06</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of stages</td>
<td>04</td>
</tr>
<tr>
<td>No. of machines per stage</td>
<td>03</td>
</tr>
<tr>
<td>Setup time</td>
<td>U[60-90]</td>
</tr>
<tr>
<td>Processing time</td>
<td>U[05-25]</td>
</tr>
<tr>
<td>Quantity of each job</td>
<td>U[40-60]</td>
</tr>
</tbody>
</table>

Table 6Makespan Corresponding to Lot Sizes (Case study 3)

<table>
<thead>
<tr>
<th>Lot Size</th>
<th>Optimal Makespan (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2974</td>
</tr>
<tr>
<td>2</td>
<td>2759</td>
</tr>
<tr>
<td>3*</td>
<td>2646</td>
</tr>
<tr>
<td>4</td>
<td>2698</td>
</tr>
<tr>
<td>5</td>
<td>2723</td>
</tr>
<tr>
<td>6</td>
<td>2795</td>
</tr>
</tbody>
</table>

* indicates the optimal lot size

Figure 3 Convergence Curve for Lot Size 3 (case study 2)

6 Conclusions and Future Scope

In the present work, an attempt is made to optimize flexible flow shop scheduling problem with sequence dependent setup times and to assess the affect of lot splitting on makespan using genetic algorithm approach. From the analysis of the case studies considered, it can safely be concluded that there is an optimal lot size which results in the minimum makespan. For the considered case studies, the optimal lot size is three.

References


Baker, Kenneth and Trietsch. (2009), Principal of Sequencing and Scheduling, John Wiley & Sons.

Deb, K., (2003), Optimization for Engineering design, Algorithm and Examples, Prentice Hall, New Delhi, India.


Li, Dongni, Meng, Xianwen, Liang, Qiqiang, Zhao, Junqing, (2014), A heuristic-search genetic algorithm for multi-stage hybrid flow shop
OPTIMIZATION OF FLEXIBLE FLOW SHOP SCHEDULING WITH SEQUENCE DEPENDENT SETUP TIME AND LOT SPLITTING

scheduling with single processing machines and batch processing machines, Journal of Intelligent Manufacturing, DOI 10.1007/s10845-014-0874-y.


