International Journal of Industrial Engineering and Technology.
ISSN 0974-3146 Volume 2, Number 1 (2010), pp. 279--290
© International Research Publication House
http://www.irphouse.com

A Genetic Algorithm for Job Scheduling in Two Machine Flow Shop Problem

A.O. Odior[1]*, F.A. Oyawale[2] and J.A. Akpobi

[1,3]Department of Production Engg., University of Benin, Nigeria.
[2]Dept. of Industrial and Production Engg., University of Ibadan, Nigeria.
**Corresponding Author E-mail: waddnis@yahoo.com.*

Abstract

This paper considers the problem of scheduling in flow-shop by Johnson's Algorithm method, Branch and- Bound Algorithm method and Genetic Algorithm method to find an optimal sequence for n jobs m-machine problem based on minimum elapsed time.In scheduling the two machine flow shop problem $F2||\sum C_i$ one has to determine a schedule that minimizes the sum of finishing times of an arbitrary number of jobs that need to be executed on two machines, such that each job must complete processing on machine 1 before starting on machine 2. We propose a heuristic for approximating the solution for the $F2||\sum C_i$ problem using a genetic algorithm.

Keywords: Scheduling, Flow-Shop, Genetic Algorithm, Optimal Sequence.

Introduction

Sequencing problems are most commonly encountered in production shops where different products are to be processed over various combinations of machines. The selection of appropriate order in which jobs are to be performed is called job sequencing. The objective is to determine an appropriate sequence or order for jobs to be done on a finite number of service facilities in some pre-assigned order, so as to minimize the total involved resources. There are total (n!)m possible ways by which n jobs can be processed on m-machines. Here, the aim is to find out one sequence out of (n!)m that minimizes the total elapsed time, (Smita and Paheli, 2009).

According to Blazewicz, et al. (2005), Johnson's Rule has been the basis of many flow shop scheduling heuristics. Palmer first proposed a heuristic for the flow shop scheduling problem to minimize makespan (Odior *et al.*, 2010). The heuristic generates a slope index for jobs and sequences them in a descending order of the index. Campbell et al. (1970) proposed Campbell, Dudek, Smith (CDS) heuristic which is a generalization of Johnson's two machine algorithm; it generates a set of m-1 artificial two-machine problems from an original m-machine problem, then each of the generated problems are solved using Johnson's algorithm.. Du (1993) proposed an AIS approach for solving the permutation flow shop scheduling problem while Liaw, (2008) developed a two-phase heuristic to solve the problem of scheduling two-machine no-wait job shops to minimize makespan.

Holland (1992) conceived of genetic algorithms in the early 1970 in order to solve optimization problems, by using random search. Genetic algorithms are a class of adaptive heuristic search techniques which exploit gathered information to direct the search into regions of better performance within the search space. In terms of time complexity, compared with other optimization techniques such as integer linear programming, branch and bound, tabu search, they may offer a good approximation for the same big-O time when the state-space is large, (Golden, 1996). Flow shop problems are a distinct class of shop scheduling problems (Du, 1993), where n jobs (i = 1, . . . , n) have to be performed on m machines (j = 1, . . . ,m) as follows. A job consists of m operations, the jth operation of each job must be processed on machine j and has processing time pij . A job can start only on machine j if its operation is completed on machine (j − 1) and if machine j is free. The completion time of job i, Ci, is the time when its last operation has completed. This problem is denoted in α|β|γ-notation as Fm||$\sum C_i$, (Brucker, 2004). Consider an example of flow shop with three machines with the following data (Table 1).

Table 1: Three Machine Flow Shop.

Job i	P_{i1}	P_{i2}	P_{i3}
J1	1	2	3
J2	1	2	1
J3	1	1	1

Two possible and feasible schedules for the three machine flow shop presented in Table 1 are presented in Figures 1 and 2.

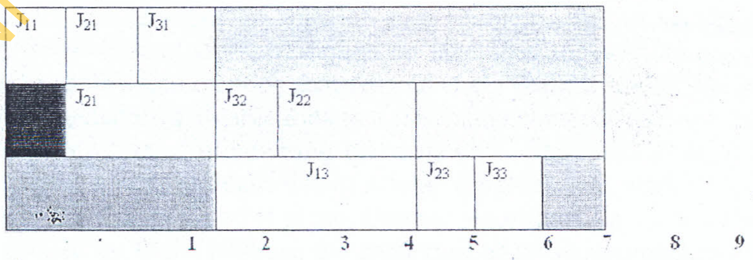


Figure 1: Flow Shop for 3 Machines: Case i.

Note that in both schedules the order of the jobs differs across machines. For the case (i), we have Cmax = 9 and $\sum C_i = 18$. In case (ii), Cmax = 8 and $\sum C_i = 21$. Note that $\sum C_i$ is better than Cmax in case (i) whereas it is the opposite in case (ii). The example suggests that jobs and their scheduling often very much depend on the objective function.

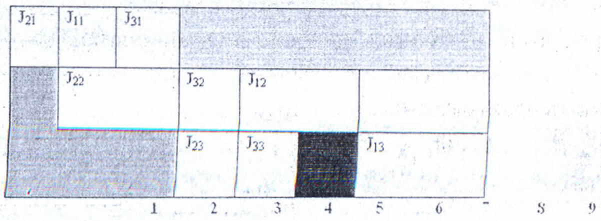


Figure 2: Flow Shop for 3 Machines.

Theory

In a *flow shop* problem, there are *m* machines that should process *n* jobs. All jobs have the same processing order through the machines. The order of the jobs on each machine can be different however the objective is that of minimising the makespan.

- If there are *m*=2 machines, then the problem can be solved in *O*(*n* log*n*) time by Johnson's algorithm.
- If there are *m*=3 machines or more, then the problem is NP-hard. We discuss the properties of an optimal schedule for the general case with *m* machines and describe two approximation algorithms.

Two Machine Flow Shop Problem

The work here focuses on the case m = 2 where the objective is to minimize the sum of completion time ($\sum C_i$) or the average completion time; thus we consider the flow shop problem F2||$\sum C_i$ with n jobs. Oulamara (2007) considered makespan minimization for no-wait flow shop problems on two batching machines. (For Batching machines the completion time of a job is the completion time of the batch the job is part of.). Independently, Liaw (2008) developed heuristic for minimizing the makespan for two-machine no-wait job shop problems. In this setting operations must be performed without any interruption on machines and without any waiting in between machines. We also mention that Allaoui et al., (2008), studied the problem of scheduling n immediately available jobs in a flow shop composed of two machines in series with the objective of minimizing the makespan. Blazewics *et* al. (2005) have studied the variant of the problem where a total weighted late work criterion and a common due date (F2|di = d|Yw) is given. Genetic algorithms for shop problems were extensively studied by Wall (1996) in the context of adaptive approaches to resource-constrained scheduling.

Thus an optimal schedule may be represented by a job permutation and a permutation fully describes the solution. Computing the order is NP–hard (Garey *et al.*, 1976). Still, the fact that in the case of two machines the search space is restricted to permutations makes the construction of effective genetic operators more feasible. It should be noted that the problem of F2||Cmax is to find a schedule, which minimizes the Cmax = max{Ci, i = 1, . . . , n} (the so called *makespan*). For arbitrary processing times, this problem is the only flow shop problem that is polynomially solvable. The optimal solution is given by Johnson's algorithm (Johnson, 1954).

Using Johnson's Algorithm

Johnson's algorithm gives an optimal solution to the F2||Cmax problem and all the jobs are scheduled on the same order for both machines. It creates two partial schedules, L and R. The final schedule T (the same for the both machine) is obtained by concatenating L and R (see Algorithm 1).

In order to schedule the processing of customers' orders such that maximum profit is obtained, the principles guiding flow shop scheduling are adopted as presented in the mathematical frame work. In this case customers are free to bring their jobs at any time. However, each customer's order passes through the machines in the same order.

Single Machine Sequencing

A single machine sequencing is a flow shop in which the jobs visit the machines in the same sequence. The shop characteristics of a single machine shop is given as:

$$n / m // F / \bar{F}$$

where n is the number of jobs in the shop
m is the number of machines in the shop
F is the flow shop
$\bar{F}$ is the mean flow time.
n / m is referred to as the hardware and F / $\bar{F}$ is referred to as the software of the system.

Johnson's 2- Machine Algorithm

Johnson's 2 – machine algorithm is a process in which the jobs are scheduled in the machines in such a sequence that gives the minimum makespan. A typical case of Johnson's 2-machine algorithm with n jobs is presented in Figure 3.

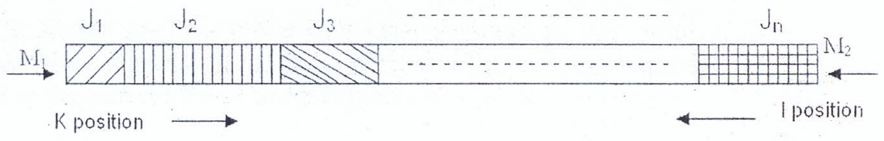


Figure 3: A typical chart for johnson's 2-machine. algorithm.

The flow time for job J in the kth position is given by

$$F(k) = P(1) + P(2) + P(3) + \ldots\ldots\ldots + P(k)$$

$$\therefore F(k) = \sum_{i=1}^{k} P(i) \qquad 1$$

where P(i) is the processing time for the job in the ith position in the sequence.

This algorithm supposes that we have (n) jobs to be scheduled on two machines i.e. J1, J2, ..., Jn,

Then n positions are possible.

$$\text{Total flow time } F_T = \sum_{k=1}^{n} F(k) = \sum_{k=1}^{n}\sum_{i=1}^{k} P(i)$$

$$\text{Mean flow time } \bar{F} = \frac{\sum_{k=1}^{n}\sum_{i=1}^{k} P(i)}{n} \qquad 2$$

Generally, for n position we have;

$$\sum_{i=1}^{n} (n-i+1)P(i)$$

$$\frac{\sum_{k=1}^{n}\sum_{i=1}^{k} P(i)}{n} = \frac{\sum_{i=1}^{n} (n-i+1)P(i)}{n} \qquad 3$$

The optimizing sequence can be obtained from the following process:

In this case we have (n) jobs to be scheduled on two machines i.e. J1, J2, ..., Jn. The optimal solution by Johnson algorithm is obtained as follows:

Step 1: Set k =1, l = n

Step 2: Set the list of unscheduled jobs = {J1, J2, ..., Jn}

Step 3: Find the smallest processing times on first and second machines for the currently unscheduled jobs

Step 4: If the smallest processing time obtained in step 3 for Ji is on the first machine then schedule Ji in kth position of processing sequence. Then delete the Ji job from the list of unscheduled and decrease k by 1.

Step 5: If the smallest processing time obtained in step 3 for Ji is on the second machine then schedule Ji in the lth position of processing sequence. Then delete the Ji job from the current list of unscheduled jobs and decrease l by 1.

Step 6: Repeat steps 3 to 5 for the remaining unscheduled jobs until all the J jobs are scheduled.

Summing up the various processing times gives the makespan for the optimum scheduling.

Algorithm 1: Johnson's Algorithm

```
1. X := {1, . . . , n}; L := $\phi$; R := $\phi$
2. while X $\neq \phi$ do
BEGIN
3. Find job i that has smallest pi1 or pi2.
4. if pi1 is the smallest then L := L ∘ i else R := i ∘ R;
5. X := X \ {i}
END
6. T := L o R
```

From algorithm 1, we have the set X of all jobs that are not scheduled yet, at time t consider the job i that has the smallest processing time for either machine: the smallest value of pi1 or pi2 where i ∈ {1, . . . , n}. If job i has smallest pi1 value then job i is removed from X and added to the tail of L i.e, L_{oi} and if otherwise job i is added to the front of R i.e, i o R.

This is done until X becomes empty (all the jobs have been scheduled in T and R).

Initially let X = {1, . . . , i, . . . , n} be the set of all jobs. The example in Figure 4 shows how Johnson's algorithm works for a set of 5 jobs, where i represents the job number and j represents the machine.

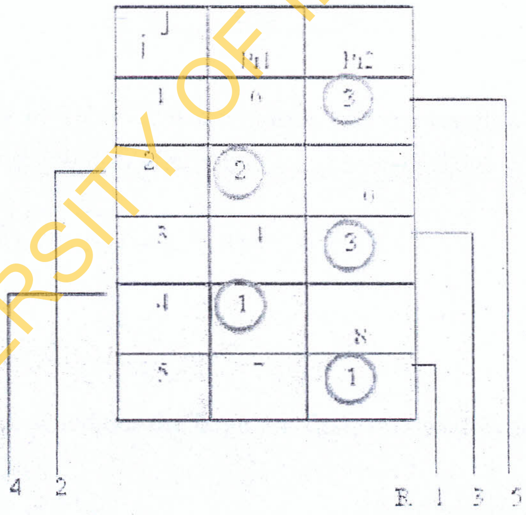
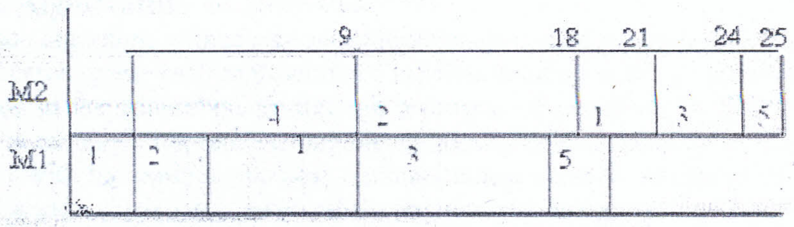


Figure 4: Johnson's Algorithm for n = 5 (Selecting the Jobs).

The optimal schedule for the set of 5 jobs, where i represents the job number and j represents the machine is now presented in Figure 5.



$C_{max} = 25$

Figure 5: Johnson's Algorithm for n = 5, (Optimal Schedule T of the Jobs).

To show that Johnson's algorithm gives an arbitrarily large solution for the $F2||\sum C_i$ problem, consider the following flow shop that has n jobs given in Table 2. The value $\in$ is considered very small and the value k very large. We refer to the nth job as the "large" job.

Table 2: A 2-Machine Flow Shop Problem.

Job i	Pi1	Pi2
1	$\in$	$\in$
2	$\in$	$\in$
⋮	⋮	⋮
n	$\in/2$	k

For the data given in Table 2, it is obvious that the optimal schedule for $\sum C_i$ would schedule the large job last, after jobs 1, . . . , (n − 1). Thus $\sum C_i$ is equal to

$$\sum C_i = C_1 + C_2 + \ldots + C_n$$
$$= 2\in + 3\in + \ldots + n\in + (n\in + \in/2 + k)$$
$$= \frac{n(n+3)-1}{2}\in + k$$
$$= \text{lower order terms} + k$$

Johnson's algorithm schedules the large job first, followed by jobs 1, . . . , (n − 1). Thus $\sum C_i$ is equal to

$$\sum C_i = C_1 + C_2 + \ldots + C_n$$
$$= (\in/2 + k) + (k + (\in)) + (k + (\in + \in)) + \ldots + (k + (\in + \ldots + \in))$$
$$= nk + \frac{n(n+1)+1}{2}\in$$
$$= nk + \text{lower order terms}$$

If n is arbitrarily large, then Johnson's algorithm gives an arbitrarily bad solution.

Genetic Algorithms

In a genetic algorithm a fixed size set of individuals (called generation) is maintained within a search space, each representing a possible solution to the given problem. The individuals in the generation go through a process of evolution. A fitness score is assigned to each solution representing the abilities of an individual to "compete". The individual with the optimal (or near optimal) fitness score is sought (Kumar *et al.*, 2007). The individuals with lower values are removed and newer ones, added by the "breeding" process – by combining information from the parents' components – are added. After an initial population is randomly generated, the algorithm evolves through three operators: selection represents the paradigm of survival of the fittest, crossover mimics mating between individuals, and mutation introduces random modifications.

A genetic algorithm has the following structure:

1. Randomly initialize population (at time t).
2. Determine fitness of population (at time t).
3. Repeat the following until the best individual is found:
   a. Select parents from population (at time t).
   b. Perform crossover on parents creating population (at time t + 1).
   c. Perform mutation of population (at time t + 1).
   d. Determine fitness of population (at time t + 1).

In the case of the 2-machine flow shop problem, an individual is represented by a permutation. The fitness of a permutation is the $\sum C_i$ -value of the corresponding schedule.

In the case of the F2||PCi problem, we use the branch and- bound algorithm presented in Brucker (2004). A natural way to branch is to choose the first job to be scheduled at the first level of branching tree, the second job at the next level, and so on. Thus the basic idea of this algorithm is to consider subproblems, where r jobs have been scheduled. Algorithm Branch-and-Bound summarizes these basic ideas. As an example, consider Figure 6, here, the number of jobs is 4. For example the node (23) represents the fact that jobs 2 and 3 are fixed in this order and jobs 1 and 4 could still be in any order after jobs 2, 3. In general, suppose we are at a node at which the jobs in the set M $\subseteq$ {1, · · · , n} have been scheduled, where |M| = r. The cost of this schedule, which we wish to bound, is

$$S = \sum_{i \in M} C_i + \sum_{i \notin M} C_i$$

For the second sum, Brucker, (2004) derives two possible lower bounds based on assumptions:

1. Every job i $\notin$ M completes its processing without delay from machine 1
2. Every job i $\notin$ M starts its processing on machine 2 without delay from machine 2.

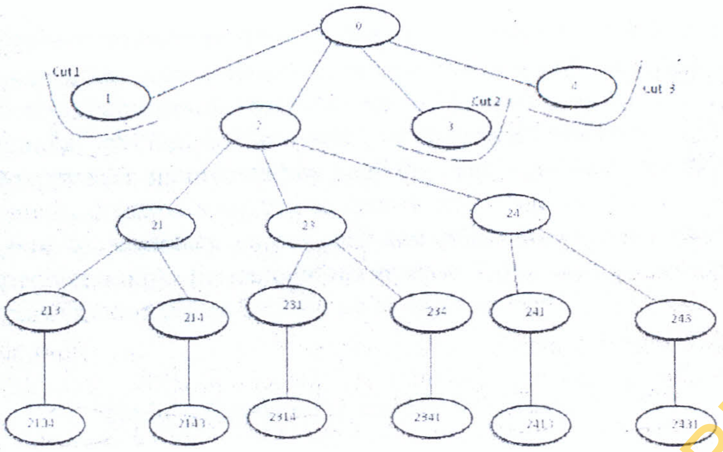


Figure 6: Branch and Bound Tree after Pruning (n = 4).

Algorithm 2: The Branch-and-Bound Algorithm

```
1.  lowerBound, upperBound = feasible solution GENERATE NODES (a, i)
2.  IF i = n THEN current Solution = calsch(n, a) END IF
3.  IF current Solution < upperBound THEN UPDATE upperBound
ELSE
a.  CALCULATE lowerBound
b.  IF lowerBound ≥ upperBound THEN CUT
ELSE
i.  FOR i + 1 TO n DO
BEGIN
ii. SWAP
iii. CALL GENERATE NODES(a, i+1) END FOR
END IF
END IF
```

We note that the branch-and-bound algorithm is exponential in its run time, and, unlike the genetic algorithm cannot be used for larger values of n. But it is useful to calibrate the genetic algorithm.

Simulations and Results

The following results are developed using Johnson's algorithm (JA), branch-and-bound (BB), and a genetic algorithm (GA) for two machine flow shop scheduling problem. Two assumptions are made:

1. When implementing branch-and-bound, we calculate an initial feasible solution which is the sum of completion time for all the processes in the ascending order.
2. When implementing a genetic algorithm, the mutation probability is 0.01 and

the crossover probability is 0.85. These parameters were found after extensive experimentation. Lower crossover probabilities slowed convergence and other mutation probabilities did not work well. The choice of these parameters was also guided by our earlier work on traveling salesman problems. The following results are obtained by applying The following results are obtained by applying Johnson's algorithm, branch-and-bound algorithm and a genetic algorithm to randomly chosen pi1 and pi2 values. When more runs are executed for a GA, the results are separated by commas. Table 3 contains randomly selected pi1 and pi2 for up to 20 jobs.

Table 3: Random Pi1 and Pi2 for n up to 20.

Job i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
Pi1	6	2	4	1	7	4	7	9	6	7	5	9	2	4	6	3	9	4	8	8
Pi2	3	9	3	8	1	5	6	3	4	2	8	3	6	3	5	6	2	6	5	1

For n = 5 and randomly selected pi1 and pi2 given in Table 3, by running JA (Johnson's Algorithm),
BB (Branch and- Bound Algorithm) and GA (Genetic Algorithm), the results for the objective function $\sum C_i$ are presented in Table 4.

Table 4: $\sum C_i$ Results for n = 5.

n = 5	JA	BB	GA with gen =150, pop=50
$\sum C_i$	97	83	83

For n = 7 and randomly selected pi1 and pi2 given in Table 3, by running JA, BB, and GA algorithms the results for the objective function $\sum C_i$ are presented in Table 5.

Table 5: $\sum C_i$ Results for n = 7.

n = 7	JA	BB	GA with gen =150, pop=50
$\sum C_i$	182	150	150

For n = 10 and randomly selected pi1 and pi2 given in Table 3, by running JA, BB, and GA algorithms the results for the objective function $\sum C_i$ are presented in Table 6.

Table 6: $\sum C_i$ Results for n = 10.

n = 10	JA	BB	GA with gen =150, pop=50
$\sum C_i$	331	292	297, 292, 294, 295

Conclusion

As noted before the It is seen that the branch-and-bound algorithm is exponential in its run time, and, unlike the genetic algorithm it cannot be used for larger values of jobs (n). Its purpose is to calibrate the genetic algorithm using relatively small values of n. A heuristic model based on genetic algorithms to approximate the two machine flow shop problem F2$||\sum C_i$ has been proposed. To calibrate our genetic algorithm we show that for smaller numbers of jobs (n) the results are comparable with the optimal schedule (obtained by using branchand- bound technique). In our simulations and for small values of n, almost the same results were obtained by our genetic algorithm and branch-and-bound algorithm while the schedule produced by Johnson's algorithm gave different results for weighted average completion time, $\sum C_i$ for the F2$||$ $\sum C_i$ problem.

References

[1] Smita V. and Paheli S., 2009, "Flow-shop Sequencing Model using Genetic Algorithm", International Journal of Computational and Applied Mathematics. 4(2), pp 111–114.

[2] Blazewics, J., Erwin P., Margozata S. and Frank W., 2005, "The two-machine flow-shop problem with weighted late work criterion and common due date". European Journal of Operational Research, 165, pp.408–415,

[3] Odior A.O., Charles-Owaba, O. E. and Oyawale, F.A., 2010, "Application of Job Scheduling in Small Scale Rice Milling Firm", ARPN Journal of Engineering and Applied Sciences. 5(1), pp 1-5.

[4] Campbell, H G; Dudek, R A; Smith, M L., 1970, "A heuristic algorithm for n-job, m-machine sequencing problem", J. Management Science. 16: 630-637.

[5] Du J., 1993, "Minimizing mean flow time in two-machine open shops and flow shops", Journal of Algorithms. 14, pp. 24-44.

[6] Liaw C. F. 2008, "An efficient simple metaheuristic for minimizing the makespan in two-machine no-wait job shops", Journal of Computers and Operations Research, 35(10), pp. 3276-3283.

[7] Holland, J., 1992, Adaptation in Natural and ArtificialSystems. MIT Press, Cambridge.

[8] Golden, R. M. 1996. Mathematical Methods for NeuralNetwork Analysis and Design. MIT Press, Cambridge, Massachusetts.

[9] Brucker, P., 2004. Scheduling algorithms (Fourth edition). Springer-Verlag, Heidelberg, Germany.

[10] Oulamara, A., 2007, "Makespan minimization in a nowait flow shop problem with two batching machines", Computers and Operations Research, 34, pp. 1033–1050.

[11] Allaoui, H, Artiba, A and Aghezzaf, E,, 2008, "Simultaneoulsy scheduling n jobs and the preventive maintenance on the two-machine flow shop to minimize the makespan. International Journal of Production Economics, 112(1), pp 161-167

[12] Wall, M., 1996. A Genetic Algorithm for Resource-Constrained Scheduling. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachussetts.

[13] Garey, M. R., Johnson, D. S. and Sethi, R., 1976, "The complexity of flowshop and jobshop scheduling" Mathematics of operations research, 1, pp.117–129.

[14] Johnson, S. M. 1954, "Optimal two-and-three-stage production schedules with set-up times included" Naval Research Logistic Quaterly, 1, pp. 61–68.

[15] Kumar A., Doina B. and Wolfgang B., 2007. A Genetic Algorithm for the Two Machine Flow Shop Problem. School of Computer Science University of Nevada Las Vegas, NV 89154