Journal of Engineering for Development

Journal of Engineering for Development Vol. 9, April (2010) pp.65-73

A GENETIC ALGORITHM FOR FLOW SHOP SCHEDULING PROBLEM

Odior.A.O[a], Oyawale.F.A[b], and Orsarh.E.S[c]

Department of Production Engineering, University of Benin, Nigeria, And
Department of Industrial and Production Engineering, University of Ibadan, Nigeria.
Email: waddnis@yahoo.com

ABSTRACT

This paper considers the problem of scheduling in flow-shop by Johnson's Algorithm method and Genetic Algorithm method to find an optimal sequence for n jobs m-machine based on minimum elapsed time. It has been shown that the method for finding an optimal sequence for n jobs, m-machine based on minimum make span using genetic algorithm provides a better result.

Keywords: Johnson's Algorithm, Scheduling, Flow-Shop, Genetic Algorithm, Optimal Sequence

1. INTRODUCTION

An important function of management is the coordination and control of complex activities, including optimum allocation of resources in the performance of those activities. In scheduling problems we must determine the order or sequence for processing a set of jobs through several machines in optimal manner. The flow shop problem is a scheduling problem which considers m different machines and all the jobs are processed in the same processing order. 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 shops 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.

In many manufacturing and assembly facilities, a number of operations have to be done on every job. Often these operations have to be done on all jobs in the same order implying that the jobs follow the same route. These machines are assumed to be set up in series, and the environment is referred to as a flow-shop. The assumption of classical flow-shop scheduling problems that each job visits each machine only once (Baker, 1974) is sometimes violated in practice. A new type of manufacturing shop, the re-entrant shop has recently attracted attention. The basic characteristic of a re-entrant shop is that a job visits certain machines more than once. The re-entrant flow-shop (RFS) means that there are n jobs to be processed on m machines in the shop and every job must be processed on machines in the order of $M1$, $M2$, …, Mm, $M1$, $M2$, …, Mm, …, and $M1$, $M2$, …, Mm, (Bispo and Tayur, 2001).

According to Smita and Paheli, (2009), Holland 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 and Leung,

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).

2.0 FLOW SHOP SCHEDULING PROBLEM

Flow shop problems are a distinct class of shop scheduling problems , where n jobs (i = 1, . . . , n) have to be performed on m machines, < M_1, ..., Mm >, where m ≥ 1, as follows, (Gangadharan and Rajendran, 1993): A job consists of m operations; the jth operation of each job must be processed on machine j and has processing time Tij, 1 ≤ j ≤ n. 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, Ti, is the time when its last operation has completed. Each of n jobs J1,..., Jn has to be processed on m machines M_1, ..., Mm in that order. Job Ji for i=1... n, thus consists of a sequence of m Process Pi1, ..., Pim; where Pik corresponds to the processing of Ji on machine Mk during an uninterrupted processing time Tik. This problem is denoted in the literature in α | β |γ - notation as Fm||PCi (Gangadharan and Rajendran, 1993).

Consider an example of flow shop with three machines with the following data presented in 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

Figure 1 and Figure 2 show two feasible schedules for the three machine flow shop presented in Table 1

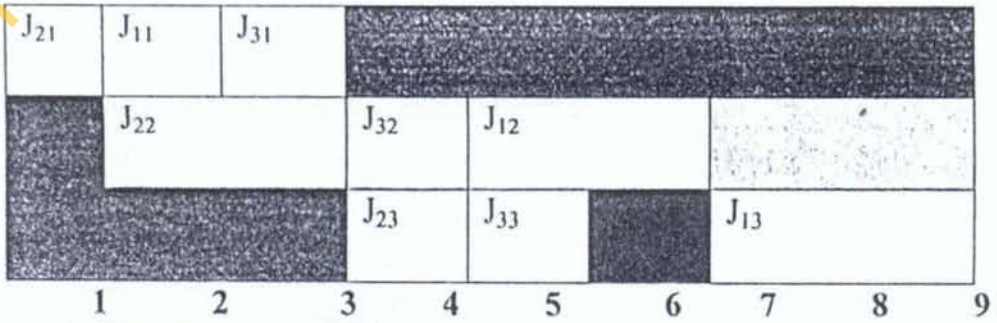

Figure 1. Flow Shop for 3 Machines: Case i.

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.

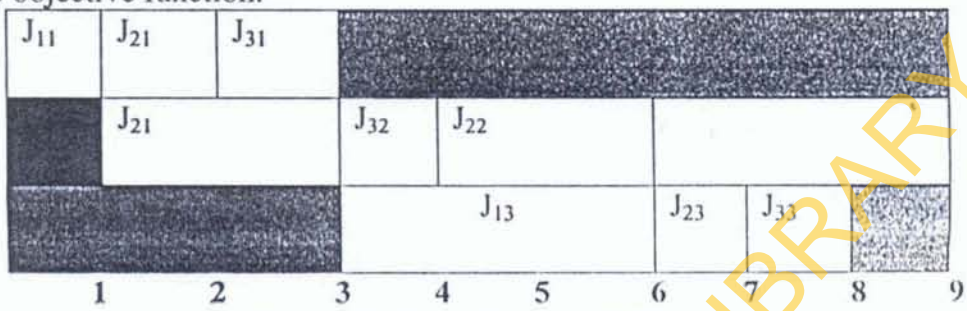

Figure 2. Flow Shop for 3 Machines,

JOHNSON'S ALGORITHM

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.

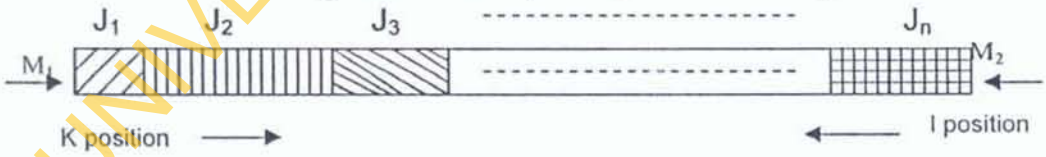

Fig. 3: A typical chart for johnson's 2-machine. algorithm.

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

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

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

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.

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

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

Generally, for n position we have;

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

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

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.

Assumptions AndProcedure

The following assumptions are made while solving a sequencing problem:
(a) No machine can process more than one job at a time
(b) Each operation, once started, must be performed till completion
(c) Each operation must be completed before any other operation, which it mustprecede, can begin
(d) All machines are of different types
(e) A job is processed as soon as possible subject to ordering requirements
(f) Processing times are independent of order of performing the operations
(g) The time involved in moving a job from one machine to other is negligiblysmall

(h) All jobs are completely known and are ready for processing before the periodunder consideration begins

Johnson's Rule And Extensions

Johnson's rule which is the most well known optimal rule applicable to alarge class of flowshop problems says that job i precedes job j in an optimal sequence if: $\min\{t_{i1}, t_{j2}\} \leq \min\{t_{i2}, t_{j1}\}$. Implementing Johnson's rule, the flowshop problem form = 2 and makespan as performance criterion or n/2/F/c_{max} can be optimally solvedby the following famous algorithm:

Step 1: Find $\min_i \{t_{i1}, t_{j2}\}$.

Step 2a: If the minimum processing time requires machine 1, place the associatedjob in the first available position in sequence. Go to Step 3.

Step 2b: If the minimum processing time requires machine 2, place the associatedjob in the last available position in sequence. Go to Step 3.

Step 3: Remove the assigned job from consideration and return to Step 1 until allpositions in sequence are filled. Another shape of this algorithm may bedescribed as follows:

Step 1: Let U = $\{j / t_{j1} < t_{j2}\}$ *and* $V = \{j / t_{j1} \geq t_{j2}\}$.

Step 2: Arrange the members of set U in non-decreasing order of t_{j1} and membersof set V in non-increasing order of t_{j2}.

Step 3: An optimal sequence is the ordered set U followed by the ordered set V.

Moreover, there are some extensions of Johnson's rule, one is for m = 3 and makespancriterion. He showed that a generalization is possible when the second machineis dominated (i.e. when no bottleneck could possibly occur on the second machine).This extension is described below:

1. If $\min_k \{t_{k1}\} \geq \max_k \{t_{k2}\}$ then job i precedes job j in an optimal schedule if:

   $\min_k \{t_{i1} + t_{i2}, t_{j2} + t_{j3}\} \leq \min\{t_{i2} + t_{i3}, t_{j1} + t_{j2}\}$

2. If $\min \min_k \{t_{k3}\} \geq \max_k \{t_{k2}\}$ then job i precedes job j in an optimal schedule if:

   $\min_k \{t_{i1} + t_{i2}, t_{j2} + t_{j3}\} \leq \min\{t_{i2} + t_{i3}, t_{j1} + t_{j2}\}$

To apply these results in an algorithm, it is possible to use the main algorithmimplementing Johnson's rule, described previously, with first step of seeking a minimumin the form of $\min \min\{t_{i1} + t_{i2}, t_{i2} + t_{i3}\}$ instead of seeking minimum processingtime. Additionally, if there is no dominance present, it is also known that if thatmain algorithm implementing Johnson's rule, yields the same optimal sequence for two-machine sub problems represented by the set $\{t_{i1}, t_{i2}\}$ *and* $\min\{t_{i2}, t_{i3}\}$. then that sequence is optimal for the full three-machine problem.

GENETIC ALGORITHM

The Genetic Algorithm (GA) is an optimization and search technique based on theprinciples of genetic and natural selection. A GA allows a population composed ofmay individuals to evolve under specific selection rules to a state that maximizes thefitness (i.e., minimizes the cost function). John Holland et. al. [17] developed Genetic

Algorithm (GA) as a search algorithm based on the mechanics of natural selection [18] in order to find optimal or near optimal solution. The main idea of GA is that in order for a population of individuals to adapt to some environment, it should behave like a natural system [19]. This means that survival and reproduction of an individual is promoted by the elimination of useless or harmful traits and by rewarding useful behavior. 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 (i.e. integer linear programming, branch and bound, tabu search), GA may offer a good approximation for the same big-O time when the state-space is large [20 & 21]. GA belongs to the family of evolutionary algorithms, along with genetic programming, evolution strategies, and evolutionary programming. Evolutionary algorithms can be considered as a broad class of stochastic optimization techniques. An evolutionary algorithm maintains a population of candidate solutions for the problem at hand. The population is then evolved by the iterative application of a set of stochastic operators.

Genetic Algorithm for Processing n-jobs through 2-machines:

The genetic algorithm for processing n-jobs through 2-machines begins by reading the number of jobs and processing times for each job on first machine (m1) and second machine (m2) as presented in Figure 4.

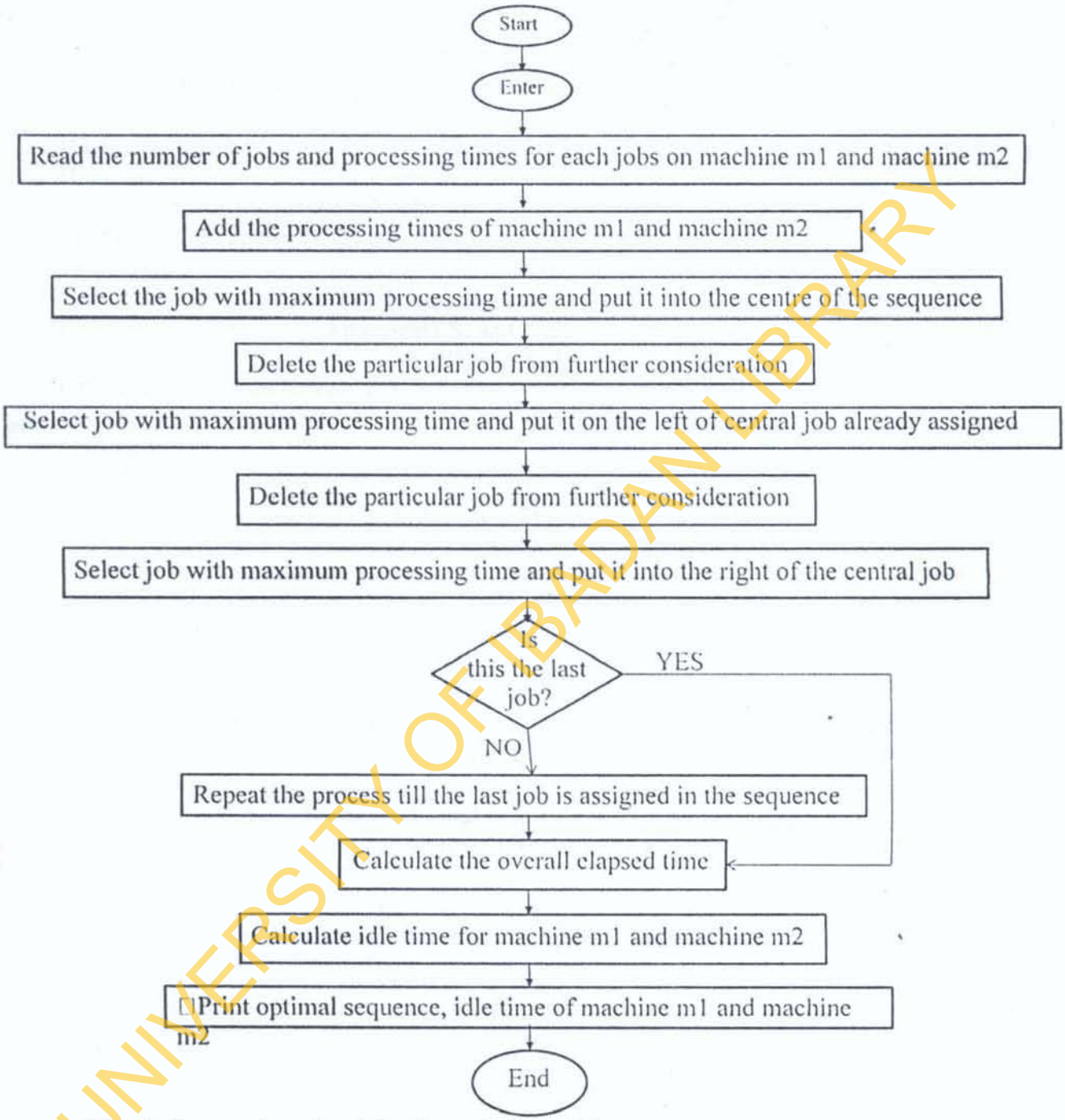

Fig. 4: Processing of n-jobs through 2-machines

CONCLUSION

This study developed a method for scheduling in flow-shop by Johnson's Algorithm method and Genetic Algorithm method to find an optimal sequence with makespan as the criterion. For processing n-jobs through 3-machines, Johnson extended the method by deriving a special condition that the smallest processing time for the first machine is at least as great as the largest processing time for the second machine. The

GA procedure mentioned above can also be applied to the n-jobs three machine problem or in general to the n-jobs m-machine problems.

REFERENCES

Baker, K. R., (1974). Introduction to sequencing and scheduling, John Wiley & Sons, ISBN: 0-471-04555-1, New York.

Bispo, C. F. & Tayur, S. (2001). Managing simple re-entrant flow lines: theoretical foundation
and experimental results. IIE Transactions, Vol. 33, No. 8, 609-623, ISSN: 0740-817X.

Blazewicz, 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, Vol. 165, pp.408–415,

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

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.

Du, J. and Leung, G. R. (1993). Minimizing mean flow time in two-machine open shops and flow shops. Journal of Algorithms. Vol. 14, pp. 24-44.

Gangadharan, R. and Rajendran, C., (193). Heuristic algorithms for scheduling in the no-wait flowshop. Int. J. Prod. Econ., 32: 285–290.

Golden, R. M. (1996). Mathematical Methods for NeuralNetwork Analysis and Design. MIT Press, Cambridge, Massachusetts.

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, Vol. 35, No.10, pp. 3276-3283.

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. Vol.5, No.1, pp1-5.

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