International Journal of Engineering and Techniques - Volume 3 Issue 6, Nov - Dec 2017

RESEARCH ARTICLE OPEN ACCESS

Study on an Advancement to Genetic Algorithm for Flexible Job-Shop Scheduling with Overlapping in Operations

[1]Ms. Waste Mansi Nagnath

D. Y. Patil Institute of Engg & Tech, Pimpri, Pune, Maharashtra, India

Abstract:

Flexible job-shop scheduling is a type of scheduling which is extension of Job-shop scheduling problem. In FJSP, operations are processed on different machines, which means operations are break down to sublots, and these sublots are processed by machines independently. In previous research, mathematical model was developed along with implementation of Genetic Algorithm. This paper gives an overview of improved methods to Flexible Job-Shop scheduling Problem with overlapping in operation[1].

Keywords — **Flexible Job-shop scheduling, Genetic Algorithm, overlapping in operation.**

INTRODUCTION :

The flexible job-shop scheduling is extended scheduling problem of job-shop scheduling, in which operations are processed on several different machines instead of single machine. To perform operations on different machines, operations are break down to sublots. These sublots are processed individually and transferred to the next processing stage after completion of the current one. The sublot is an assumption, proposed by Fantuhan et al[2] and later discussed by Demir & Isleyen[3]. This assumption is for overlapping in operation approach. In this approach, there is no need to complete the previous operation to start processing of new operation[4]. Application of this approach can be found in multistage manufacturing industry.

As job shop scheduling is NP-hard problem, its further improved problems are also NP-hard. That is FJSP is also NP-hard type. FJSP deals with more complex cases than job shop problem. The reason behind using Genetic Algorithm as main method is, it gives better results in ample of time. In addition to this, in recent years meta-heuristics gives better results than classical dispatching rules. Meta-heuristics are the heuristics that provides good enough solution to an optimization problem, where limited information is available.

This paper gives study on proposed techniques & methodologies as well as previously used methodology. The improvement focuses on, to increase diversity & avoid falling into local optimum. In proposed method, selection operator of GA is improved by allowing single entry for each individual. The crossover operator is also modified by maintaining relationship between machine selection(MS) and operation sequence(OS). Moreover, the mutation operator is improved so that the chances of changing schedule of makespan increases and it is done by reassigning the operations that are in critical path.

The problem is same as that of in previous research paper[3]. The motive is to process n jobs on m machines with minimum makespan. Let M be m machines i.e. $M=\{M_1, M_2, \ldots, M_m\}$ and n jobs be J i.e. $J=\{J_1, J_2, \ldots, J_n\}$. Job J_i consist of n_i operations with predefined sequence : $O_{i1}, O_{i2}, \ldots, O_{ij}$ which means operation j of job i. When all operations are finished in their predefined order then that particular job is completed.

Each operation O_{ij} can be processed on k machines with processing time t_{kij}, where M_{ij} is subset of M ,which is set of available machines. There should not be an idle time between two adjacent sublots of same operation. And all sublots belonging to same operation must be operated on same machine one after another. FJSP is used here is with overlapping in operation approach. There are 3 different cases of overlapping discussed below.

Case 1: The sublot of operation O_{ij+1} starts after completion of every operation O_{ij} sublot. Because of this, the idle time gets generated between two adjacent sublots of same operation, and as we are not considering this pattern in the approach this case is excluded in this methodology. Figure 1 shows this case.

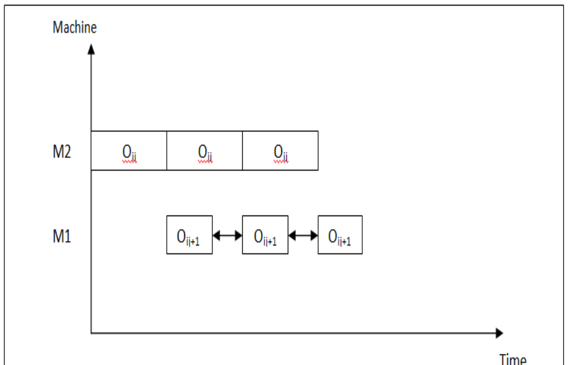

Figure 1: Case 1

Case 2: Sublots of operation O_{ij+1} starts their processing immediately after completion of first sublot of O_{ij}. And it continue its processing till all sublots get processed. In this case, there is no chance of generation of idle time, operation O_{ij+1} completes processing of its all sublots without waiting for another operation to finish. Therefore this case is suitable for proposed approach, and this is included in it. This case is suitable, when the processing time of operation O_{ij+1} is equal or larger than that of operation O_{ij} i.e. $t_{kij} <= t_{kij+1}$. Figure 2 shows this case.

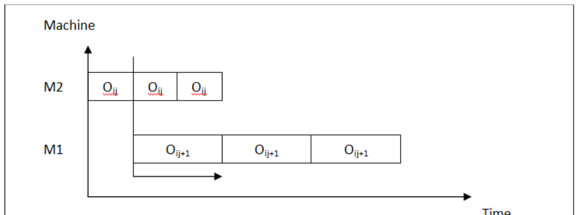

Figure 2 : Case 2

Case 3 : The last sublot of operation O_{ij+1} starts processing, when entire operation O_{ij} completes its processing. The operation Oij+1 starts processing along with last sublot of operation Oij, but on separate machines. This will led to avoid the idle time in between sublots of same operation. So this is also applicable case to the approach, hence it is included. This particular case is used, when the processing time of operation O_{ij+1} is smaller than that of operation O_{ij} i.e. $t_{kij} > t_{kij+1}$. Figure 3 shows this case.

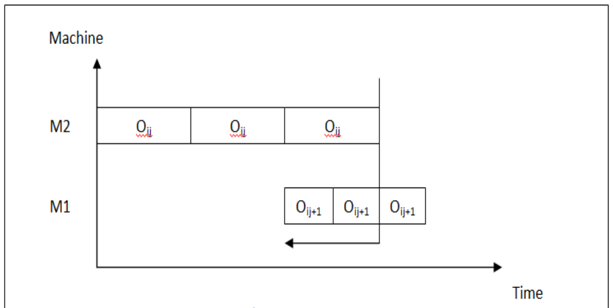

Figure 3 : Case 3

The last two cases described above helps to minimize the completion time of the operation, and they are used in the proposed method. Previously used techniques and improvements over it are as follows.

A. Chromosome Encoding :

Representation of chromosome for improved version and the previous research[5] are same. It includes two parts : machine selection (MS) and operation sequence (OS). The MS part is the array of integers which represents index of the machine that are available for processing. The OS shows all operations of same job with particular job index. All operations of same job, are different than each other operation of same job and this is based on time represented by job index.

B. Chromosome Decoding :

While decoding the chromosome, only active schedules are considered. Active schedules means schedules that allows only global right shift, i.e. to preserve feasibility no operation is allowed to put earlier in the schedule. Optimal schedule [3] is the most important property of active scheduling. Priority based scheduling was used to convert the chromosome into active schedules, and it includes searching earliest available time interval for each operation.

C. Improved Population Initialization :

The generation of initial population of MS and OS part of chromosome was considered separately. To obtain MS and OS parts of chromosomes, assignment algorithm & random selection strategy [6] and three dispatching rules were used respectively. The dispatching rules are : random selection, most work remaining, most operations remaining [7]. The reason behind not using previous strategy for population initialization is it was time consuming. And here we are considering that it doesn't matter that initial population is best or not, rather it should take minimum time to generate population.

D. Improved Selection Operator :

The previous method used was three sized tournament selection operator [5]. In this particular method, as the size of tournament was large, and there was high possibility to select individuals for many times, this condition is avoided in the improved method.

In improved method the size of tournament is decreased by 3 to 2. And in each tournament selection the individual with high fitness is removed from parent population and added to new population. This results into only one time selection for individual from one population and helps to achieves the diversity of population.

E. Improved Crossover Operator :

In original method, crossover operator for the MS and OS parts of chromosome were applied separately. For MS part two point crossover was applied. And for OS part a precedence preserving order based crossover(POX) [5] was applied. Figure 4 shows the example of crossover operator.

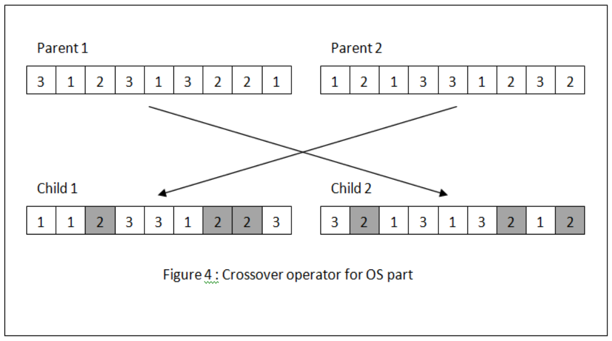

Figure 4 : Crossover operator for OS part

All 1's are the operations of job 1, 2's are all operations of job 2, 3's are all operations of job 3. Child 1 and child 2 preserves the location of job 2, from their respective parents. Then operation of job 1 and job 3 are copied in cross pattern, i.e. child 1 copied the sequence of operations of job 1 and job 3 from parent 2, and for child 2 sequence is copied from parent 1.

Improved method includes to treat the MS and OS part of chromosome as a whole. This helps to identify which changes gives better result and which does not in one generation. Therefore POX crossover is applied to OS part to produce children and it inherits the MS part from its parents. Figure 5 shows the example. In every chromosome, first three genes are of job 1, next three genes are of job 2 and last three genes are of job 3. The integer value of every gene represents the machine assigned to that operation. According to figure 4 for MS part, child 1 and child 2 preserves the location for job 2 same as of its parent. Now in figure 5, child 1 preserves the MS part of job 2 from parent 1 i.e. third, fourth, fifth gene and MS part of job 1 and job 3 from parent 2 i.e. first three and last three genes of child 1.

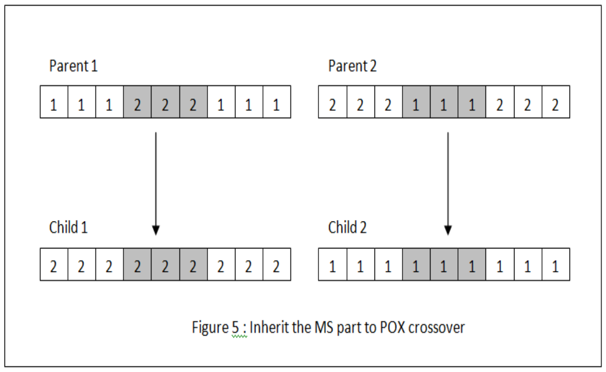

Figure 5 : Inherit the MS part to POX crossover

F. Improved Mutation operator in MS part : In previous research[5], the mutation operator for MS and OS are also performed separately. For MS part first random operation is selected and then machine gets changed to the machine having shortest processing time among alternative machine set.

An improvement over this is done by reassigning the operation to its adjacent machines, i.e. if original assigned machine is machine1, then reassigned to machine2, if original machine is machine2 then reassign it to machine3 and so on. This is because assigning an operation to the machine that has shortest processing time may not always be the best choice for the whole schedule to achieve a smallest makespan. The reassignment of operation to its adjacent machine helps to achieve diversity.

New mutation method in MS part is proposed which contains steps as : Identifying critical path and reassigning operations in it. The purpose is only the moving operations in critical path affect the makespan. While identifying critical path, start from the latest finished operation. If several operations are finished at the same time, then select one of them randomly. Then there might be two paths, one we call it as machine path and other one as job path. Machine path is path that goes to previous operation of same machine. And job path is path that goes to previous operation of same job. In proposed method we always select the machine path first, and if it not available then move to the job path.

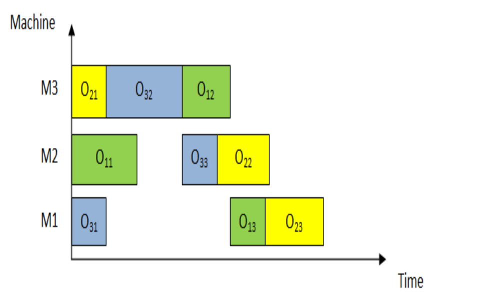

Figure 6 : Finding critical path

Figure 6 shows the example for finding a critical path. As mentioned above start from the latest finished operation and in figure last completed operation is O_{23}. Operation O_{23} starts immediately after completion of O_{22} and O_{13}. Now here there are two paths, O_{13} is machine path and O_{22} is job path. As discussed above we always select machine path first in this proposed method, so we select O_{13}. Then O_{13} starts processing after completion of O_{12} only, which is job path. So we select O_{12} and its path is decided by O_{32}. After selecting O_{32}, again there are two paths. One is O_{21} is machine path, and the other one is O_{31} which is job path. So according to our rule we select O_{21}, which is the very first job operated on one of the machines.

The important benefit of selecting machine path first is it avoids the situation that some machines are crowded with operations. Therefore, to balance the workload makespan could be reduced. It would be preferred that to include all possible operations, it makes easy to reassign them later. Finding available both paths at the same time makes it easier to determine whether available path is machine or job path. After finding the critical path select an operation randomly, and reassign to its adjacent machine.

G. Improved Mutation Operator in the OS part : In previous method, to obtain mutation operator in OS swapping technique was used[5]. As figure 7 shows, two random genes are selected and then values of these genes get swapped with each other. This technique might break the structure of original schedule, therefore another technique is introduced.

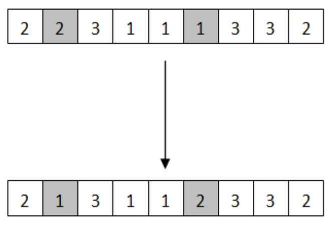

Figure 7 : Original mutation in the OS part

In improved method the precedence preserving order based shift technique is introduced which is proposed by Lee et al[6].In this method a gene is selected randomly and then it is shifted to another position, while the operation represented by this gene within its job is not changed. Figure 8 shows the example of precedence preserving order based shift. Here 2 s selected which is second operation of job2, we can shift this gene only within the range of first and last operations of job 2. If position of selected gene is shifted beyond the mentioned range, then it might bring a huge influence to the structure of the original chromosome.

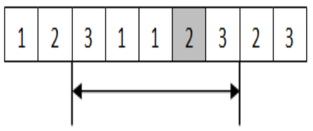

Figure 8 : Precedence preserving order based shift

H. Convergence and diversity :The proposed strategy helps to jump out from local optimum i.e. if global best result remain unchanged for certain number of generations, we will randomly generate some new individuals and replace some worst individuals with these randomly generated individuals before crossover.

CONCLUSION:

This paper gives a study on improvements over previous research. The paper includes the review on previously used methods along with improved methods with examples. The proposed methods are to improve the efficiency and diversity of Genetic Algorithm in previous research. All the improvements are based on general FJSP. If we consider the specific property of overlapping in operation, then we might get better results.

References :

[1] Tiyong He,Wei Weng, Shingeru Fujimura. 2017. “Improvements to Genetic Algorithm for Flexible Job Shop Scheduling with Overlapping in Operations.” In IEEE

[2]Fantuhan, M., C. Defersha, and C. Mingyuan. 2009. “A Coarse-grain Parallel Genetic Algorithm for Flexible Job-shop Scheduling with Lot Streaming.” In International Conference on Computational Science and Engineering, Vancouver.

[3] Yunus Demir & Selçuk Kürúat øúleyen 2014 “An effective genetic algorithm for flexible job-shop scheduling with overlapping in operations.” International Journal of Production Research, 52:13, 3905-3921.

[4] Farugi, H., Y. Y. Babak, S. Hiresh, Z. Fyagh, and N. Foruzan. 2011. “Considering the Flexibility and Overlapping in Operation in Job Shop Scheduling Based on Meta-heuristic Algorithms.” Australian Journal of Basic and Applied Sciences 5 (11): 526–533

[5] Zhang, G., L. Gao, and Y. Shi. 2011. “An Effective Genetic Algorithmfor the Flexible Job-shop Scheduling Problem.” Expert Systems withApplications 38: 3563–3573

[6] Lee, K. M., T. Yamakawa, and K. M. Lee. 1998. “A Genetic Algorithmfor General Machine Scheduling Problems.” International Journal ofKnowledge-Based Electronic 2: 60–66

[7]Pezzellaa, F, Morgantia. G, Ciaschettib, G. 2007. “A GeneticAlgorithm for the Flexible Job-shop Scheduling Problem.” Computers& Operations Research 35 (2008) 3202 – 3212

AUTHOR’S BIOGRAPHY:

My name is Waste Mansi Nagnath from Chinchwad, Pune.

DOB: 07/02/1996(DD/MM/YY)

Educational Background :

I am graduated with degree Bachelor of Engineering(Computer Science) from Savitribai Phule Pune University in year 2017. I completed my degree in Dr. D. Y. Patil Institute of Engg & Tech, Pimpri, Pune, Maharashtra, India.

Author’s Title: Ms. Mansi Waste

I was member of Association for Computing Machinery for my academic year 2015-16. I have published Implementation Paper of my degree project in June 2017 in IEEE Journal.