Vidya Journal of Engineering & Technology
ISSN 2277-8978, Vol. 3, No. 1, Jan-Dec 2017, pp. 55-62

HYBRID AND MULTISTAGE BASED GENETIC ALGORITHM FOR THE DISTRIBUTED AND FLEXIBLE MANUFACTURING UNITS

*Md. Iqbal**

ABSTRACT

The Distributed and Flexible Job-shop Scheduling problem (DFJS) considers the scheduling of distributed manufacturing environments, where jobs are processed by a system of several Flexible Manufacturing Units (FMUs). Distributed scheduling problems deal with the assignment of jobs to FMUs and with determining the scheduling of each FMU, in terms of assignment of each job operation to one of the machines able to work it (job-routing flexibility) and sequence of operations on each machine. The objective is to minimize the global make span over all the FMUs. This synopsis proposes Hybrid and Multistage based Genetic Algorithm for the Distributed and Flexible Job Shop Scheduling to solve the Distributed and Flexible Job-shop Scheduling problem. With respect to the solution representation for non-distributed job-shop scheduling, gene encoding is extended to include information on job-to FMU assignment, and a greedy decoding procedure exploits flexibility and determines the job routings.

Keywords: *Scheduling, DFJS, FMU, JSP.*

I. INTRODUCTION

The classical Job-shop Scheduling Problem (JSP) concerns determination of a set of jobs on a set of machines so that the make span is minimized. It is obviously a single criterion combinational optimization and has been proved as a NP-hard problem [4] with several assumptions as follows: each job has a specified processing order through the machines which is fixed and known in advance; processing times are also fixed and known corresponding to the operations for each job; set-up times between operations are either negligible or included in processing times (sequence-independent)

* IIMT Engineering College, Meerut

[5], each machine is continuously available from time zero; there are no precedence constraints among operations of different jobs; each operation cannot be interrupted; each machine can process at most one operation at a time.

The flexible Job-shop Scheduling Problem (f-JSP) extends JSP by assuming that a machine may be capable of performing more than one type of operation [1]. That means for any given operation, there must exist at least one machine capable of performing it. In this paper two kinds of flexibility are considered to describe the performance of f-JSP [2].

In this paper, we propose a more efficient algorithm called multistage-based hybrid GA to solve f-JSP (including total flexibility and partial flexibility) compared with Kacem's approach. The considered objective is to minimize the make span, total workloads of the machines and the maximum workloads of machines. This multi-objective optimization will be done by a multistage-based GA which including K stages (the total number of operations for all the jobs), and m state (total number of machines). Computational experiments will be carried out to evaluate the efficiency of our methods with a large set of representative problem instances based on practical data.

II. MATHEMATICAL MODEL

In this paper, the flexible Job-shop Scheduling Problem [3] is presented. We are treating it to minimize the make span and balance the workload for all machines. Before defining the problem concretely .We should add several assumptions to the problem.

1. There is a set of jobs and a set of machines.
2. Each job consists of one fixed sequence of operations.
3. Each machine can process at most one operation at a time.
4. Each machine becomes available to other operations once the operations which are currently assigned to be completed.
5. All machines are available at t = 0.
6. All jobs can be started at t = 0.
7. There are no precedence constraints among operations of different jobs.
8. Any operation cannot be interrupted.
9. Neither release times nor due dates are specified.

The f-JSP we considering here is a problem which including n-jobs operated on m machines.

Some symbols and notations have been defined as follows:

i : index of jobs, i = 1, 2, … n

Ji : the ith job

n : total number of jobs
k : index of operations, k = 1, 2, … Ki
oik : the kth operation of job i (or Ji)
Ki : total number of operations in job i (or Ji)
j : index of machines, j = 1, 2, … m
Mj : the jth machine
m : total number of machines
pikj: processing time of operation oik on machine j (or Mj)
U : a set of machines with the size m
Uik : a set of available machines for the operation oikF
ik t : completion time of operation oik
Wj : workloads (total processing time) of machine Mj

The objective function can be described as the following three equations. Eq. (1) gives the first objective makespan and also means to minimize the maximum finishing time considering all the operations. Eq. (2) gives the second objective which is to minimize the maximum of workloads for all machines. Eq. (3) gives the objective total workloads.

$$\min tm = \max\{\max tfk\} \quad \ldots(1)$$

$$\min Wm = \max\{Wj\} \quad \ldots(2)$$

$$\min Wt = f(Wj) \quad \ldots(3)$$

Eq. (2) combining with Eq. (3) give a physical meaning to the f-JSP, which referring to reducing total processing time and dispatching the operations averagely for each machine.

III. HEURISTIC METHOD

To demonstrate f-JSP model clearly, we first prepare a simple example. Table 1 gives the data set of an f-JSP including 3 jobs operated on 4 machines. It is obviously a problem with total flexibility because all the machines are available for each operation (Uik =U) [10]. There are several traditional heuristic methods that can be used to make a feasible schedule.

In this case, we use the SPT (select the operation with the shortest processing time) as selective strategy to find an optimal solution, and the algorithm is based on the procedure given in section II. Before selection we first make some initialization:

- starting from a table D presenting the processing times possibilities
- on the various machines, create a new table D' whose size is the same one as the table D;

- create a table S whose size is the same one as the table D (S is going to represent chosen assignments); initialize all elements of S to 0 (Sikj=0)
- recopy D in D'

Table 1: Data set of a 3-job 4-machine Problem

Jobs	Machine	M1	M2	M3	M4
J1	o11	1	3	4	1
	o12	3	8	2	1
	o13	3	5	4	7
J2	o21	4	1	1	4
	o22	2	3	9	3
	o23	9	1	2	2
J3	o31	8	6	3	5
	o32	4	5	8	1

Procedure: SPT Assignment
Input: dataset table D
Output: best schedule S
begin for (i=1; i<=n)
for (k=1; k<=Ki)
min=+80;
pos=1;
for (j=1; j<=m)
if (p'ikj<min) then {min=p'ikj; pos=j;}
Si,k,pos=1(assignment of oik to the machine Mpos);
//updating of D';
for (k'=k+1; k'<=Ki')
p'i',k,pos= p'i',k,pos+ pi,k,pos;
for (i'= i +1; i'<=n)
for (k'= 1; k'<=Ki')
p'i',k',pos= p'i',k',pos+ pi,k,pos;
end
end
output best schedule S
end
Furthermore, we can denote the schedule based on job sequence as:
S={(o11, M1), (o12, M4), (o13, M1), (o21, M2), (o22, M2), (o23, M1), (o31, M3), (o32, M4)}
= {(o11, M1: 0-1), (o12, M4: 1-2), (o13, M1: 2-5), (o21, M2: 0-1),

(o22, M2: 1-4), (o23, M1: 4-6), (o31, M3: 1-3), (o32, M4: 3-4)}

Finally we can calculate the solution by Eq.1, Eq. 2 and Eq. 3 as follows:

tm = max{ t F 11 , t F 12 , t F 13 , t F21 , t F22 , t F23 , t F 31 , t F 32 }
= max{1, 2, 5, 1, 4, 6, 3, 4}= 6

WM = max{(1+3), (1+3), (3+2), (1+1)}=5

WT = 4+4+5+2=15

IV. GENETIC ALGORITHM APPROACH

There are three parts in this section, firstly some traditional representation [5-6]. Secondly Imed Kacem's approach (Kacem, Hammadi & Borne, 2002), and thirdly multistage operation-based representation.

Traditional Representation of GA

Parallel Machine Representation (PM-R)

The chromosome is a list of machines placed in parallel (Table 2). For each machine, we associate operations to execute. Each operation is coded by three elements [7]:

Operation k, job Ji and S ikj t (starting time of operation oik on the machine Mj).

M1 (k , J i , t ik 1S)	J1 (Mj , t 1kj s)…
M2 (k , J i , t ik 2S)	J2 (Mj , t 2kj s)…
M3(k , J i , t ik 3S)	J3 (Mj , t 3kj s)…
……	……
Mm (……)	Jn (Mj , t nkj s)…

Parallel Jobs Representation (PJ-R)

The chromosome is represented by a list of jobs showed in Table 2. Information of each job is shown in the corresponding row where each case is constituted of two terms:

machine Mj which executes the operation and corresponding starting time tikj(s).

V. IMED KACEM'S APPROACH

Imed Kacem proposed Operations Machines Representation (OM-R) approach [8], which based on a traditional representation called Schemata Theorem Representation (ST-R) [9]. It was firstly introduced in GAs by Holland.

In the case of a binary coding, a schemata is a chromosome model where some genes are fixed and the other are free, Positions 4 and 6 are occupied by the symbol:"*". This symbol indicates that

considered genes can take "0" or "1" as value. Thus, chromosome C1 and C2 respect the model imposed by the schemata S.

S = 1 0 0 * 1 * 0 0
C1 = 1 0 0 0 1 1 0 0
C2 = 1 0 0 1 1 1 0 0

Based on the ST-R approach, Kacem[10] expanded it to Operations Machines Representation (OM-R). It consists in representing the schedule in the same assignment table S. We replace each case Sikj=1 by the couple (Fikt, Fikt), while the cases Sikj = 0 are unchanged. To explain this coding, we present the same schedule introduced before (Table 2). Furthermore, operation based crossover [7] and the other two kinds of mutation (operator of mutation reducing the effective processing time, operator of mutation balancing work loads of machines) are included in this approach.

Table 2: Representation of OM-R.

Jobs	machine	M1	M2	M3	M4
J1	o11	0, 1	0	0	0
	o12	0	0	0	1,2
	o13	2, 5	0	0	0
J2	o21	0	0,1	0	0
	o22	0	1,4	0	0
	o23	0, 1	0	0	0
J3	o31	0	0,3	0	0
	o32	0	0	0	0,4

VI. HYBRID AND MULTISTAGE BASED GENETIC APPROACH

Procedure: Hybrid Genetic Algorithm [6]
Input: fJSP data set, GA parameters
Output: best schedule
Begin
t←0;
initialize P(t) with two-vector Multistage Operation-based representation;
fitness eval(P) by priority-based decoding;
reorder operation sequence according to operation starting time;
while (not termination condition) do
crossover P(t) to yield C(t) by exchange crossover;

mutation P(t) to yield C(t) by allele-based mutation;
improve P(t) and C(t) to yield P'(t) and C'(t) by bottleneck shifting;
fitness eval(P', C') by priority-based decoding;
select P(t+1) from P'(t) and C'(t) by mixed sampling;
reorder operation sequence according to operation starting time;
$t \geq t + 1$;
end
output a best schedule;
end

VII. CONCLUSION

Some GA approaches have been used for solving f-JSP recently. However the efficiency is mainly affected by the complexity of chromosome representation. In this paper, a new multistage operation-based representation of GA (moGA) approach is proposed to solve f-JSP.

The proposed algorithm is designed for optimizing the 3 objectives including the make span tM, total workloads of all machines W, and maximum of workloads for all machines WM.

REFERENCES

[1] **Haipeng Zhang, and Mitsuo Gen,** (2014), "Multistage-based genetic algorithm for flexible job-shop scheduling problem", Vol. 11, pp. 223-232.

[2] **I. Kacem, S. Hammadi, P. Borne,** (2002), Approach by Localization and Multiobjective Evolutionary Optimization for Flexible Job-shop Scheduling Problems. IEEE Transactions on Systems, Man and Cybernetics, Part C, pp. 1-13.

[3] **K. Mesghouhi, S. Hammadi and P. Brone,** (2015), "Production Job-Shop Scheduling Using Genetic Algorithms", Vol. 1, pp. 23-32.

[4] **N. Chaiyaratana and A. M. S. Zalzala,** (2017), "Recent Devalopments in Evolutionary and Genetic Algorithms: Theory and Applications", IEEE Transaction on Evolutionary Computing Vol. 4(2), pp. 93-113.

[5] **Q. C. Meng, T. J. Feng, Z. Chen, C. J. Zhou and J.H. Bo,** (1999), "Genetic Algorithms Encoding Study and a sufficient Convergence Condition of Gas", IEEE Conference SMC 99, pp. 649-652.

[6] **Takeshi Yamada and Ryohei Nakano,** (1997), "Genetic Algorithms for Job Shop Scheduling Problem", In Proceedings of Modern Hemistic for Decision Support, pp. 474-479.

[7] **Anderson, E.J., Glass, C.A., Potts, C.N.,** (1997). Machine scheduling. In: Aarts, E.H.L., Lenstra, J.K. (Eds.), Local Search Algorithms in Combinatorial Optimization. Wiley, pp. 361–414.

[8] **Bierwirth, C., Mattfeld, D.C., Kopfer, H.,** (1996), "On permutation representations for scheduling problems", International Conference on Parallel Problem Solving from Nature, pp. 310-318.

[9] **Anderson, E.J., Glass, C.A., Potts, C.N.,** (1997), Machine scheduling. In: Aarts, E.H.L., Lenstra, J.K. (Eds.), Local Search Algorithms in Combinatorial Optimization. Wiley, pp. 361–414.

[10] **Bierwirth, C., Mattfeld, D.C.,** (1999), "Production scheduling and rescheduling with genetic algorithms", Evolutionary Computation, Vol. 7, pp. 1–17.

[11] **Berlin Anderson, E.J., Glass, C.A., Potts, C.N.,** (1997), "Machine scheduling. In : Aarts, E.H.L. Lenstra, J.K. (Eds), Local search Algorithms in Combinational Optimization", Proceedings of Parallel problem solving from Nature IV, Springer, pp. 361-419.