Single Objective Evolutionary Algorithm for Job Shop Scheduling Problem

G. Uma Sankar[1]
Lecturer in Mathematics
Ranipettai Engineering College
Ranipettai, Chennai, India

Dr. D. Saravanan
Professor of Mathematics
Karpaga Vinayaga College of Engineering and Technology
Chennai, India

[1] *Corresponding author*

***Abstract*— Job shop scheduling belongs to the class of NP-hard problems. There are a number of algorithms in literature for finding near optimal solution for the job shop scheduling problem. Many of these algorithms exploit the problem specific information and hence are less general. However, single objective evolutionary algorithms for job shop scheduling are general and produce better results. In this paper, we discuss approaches to developing single objective evolutionary algorithm for solving the job shop scheduling problem. Initial population is generated randomly. Two-row chromosome structure is adopted based on working procedure and machine distribution. The relevant crossover and mutation operation is also designed. It jumped from the local optimal solution, and the search area of solution is improved. Finally, the algorithm is tested on instances of 8 working procedure and 5 machines. The result shows that the evolutionary algorithm has been successfully applied to the job shop scheduling problems efficiency.**

***Index Terms*—job shop scheduling, evolutionary algorithm**

I. INTRODUCTION

Evolutionary algorithms (EAs) such as evolution strategies and evolutionary algorithms have become the method of choice for optimization problems that are too complex to be solved using deterministic techniques such as linear programming or gradient (Jacobian) methods. The large number of applications Beasley (1997)) and the continuously growing interest in this field are due to several advantages of EAs compared to gradient based methods for complex problems. EAs require little knowledge about the problem being solved, and they are easy to implement, robust, and inherently parallel. To solve a certain optimization problem, it is enough to require that one is able to evaluate the objective (cost) function for a given set of input parameters. Because of their universality, ease of implementation, and fitness for parallel computing, EAs often take less time to find the optimal solution than gradient methods. However, most real-world problems involve simultaneous optimization of several often mutually concurrent objectives. Single objective EAs are able to and optimal trade off in order to get a set of solutions that are optimal in an overall sense. Computational results show that Evolutionary algorithms can find shorter makes span than the other recent approximation algorithms [1]. This is however at the cost of large execution times.

Job-shop scheduling is usually a strongly NP-complete problem of combinatorial optimization problems and is the most typical one of the production scheduling problems. Since it is an important practical problem, some researchers have formulated various JSP models based on different production situations and problem assumptions [1,2]. Dessouky and Leachman developed two integer programming formulations which could easily handle high-volume manufacturing such as multiple machines of the same type, demand size greater than one unit for a particular product type and repeat visit to the same machine type [3]. Makoto and Hiroshi considered the JSP problem to minimize the total weighted tardiness with job-specific due dates and delay penalties, and a heuristic algorithm based on the tree search procedure was developed for solving the problem [4]. Gomes and Barbosa presented an integer linear programming model to schedule flexible job shop, which considered job re-circulation and parallel homogeneous machines [5]. Loukit and Jacques dealt with a production scheduling problem in a flexible job shop with particular constraints-batch production [6]. Jason presented mix integer linear programming, which considered job re-circulation. The objective was to minimize the completion time [7]. Liu regarded the JSP problem with dynamic shop scheduling problem [8]. Borstjan and Peter proposed an alternative way to avoid infeasibility by incorporating a repairing technique into the mechanism for applying moves to a schedule [9]. Hiroshi, Toshihiro considered the job shop scheduling problem of minimizing the total holding cost of completed and in-process products subject to no tardy jobs [10]

.

Since Davis (1985) proposed the first EA-based technique to solve scheduling problem, EA has been used with increasing frequency to address scheduling problems. The EA utilizes a population of solution in its search, giving it more resistance to premature convergence on local minima [11]. Hong Zhou and Yuncheng Feng proposed a hybrid heuristics EA for $n / m / G / C_{max}$, where the scheduling rules, such as

shortest processing time(SPT) and MWKR, were integrated into the process of genetic evolution . Byung developed an efficient method based on evolutionary algorithm to address JSP. The scheduling method based on single evolutionary algorithm and parallel evolutionary algorithm was designed. Dirk and Christian considered a job shop scheduling problems with release and due-dates, as well as various tardiness objectives. The evolutionary algorithm can be applied to solve this kind of problem

.

In this paper, a university mathematical model for agile job shop scheduling problem is constructed. The objective of this model is to minimize make span. In order to solve this mixed- and multi-product scheduling problem, a new evolutionary optimization process based on EA will be developed, which includes initialization population and two kinds of coding and crossover and mutation operators based on work procedure and machine distribution. The neighborhood structure is extended and the globally optimal solution is obtained. The algorithm will then be used to solve the JSP problem of 8 working procedure and 5 machines. The Gantt chart of processing route is draw, and minimizes the completion time of all the jobs.

II. MODELING THE JOB-SHOP SCHEDULING PROBLEM

The job shop scheduling problem is a generalization of the classical job shop problem. In the static job-shop scheduling problem, finite jobs are to be processed by finite machines. Each job consists of a predetermined sequence of task operations, each of which needs to be processed without preemption for a given period of time on a given machine. Tasks of the same job cannot be processed concurrently and each job must visit each machine exactly once. Each operation cannot be commenced until the processing is completed, if the precedent operation is still being processed. A schedule is an assignment of operations to time slots on a machine. The make span is the maximum completion time of the jobs and the objective of the JSSP is to find a schedule that minimizes the make span.

We consider the flexible case where stages might be skipped. Every job is a chain of operations and every operation has to be processed on a given machine for a given time. The task is to find the completion time of the very last operation is minimal. The chain order of each job has to be maintained and each machine can only process one job at the same time. No job can be preempted; once an operation starts it must be completed; two operations of a job cannot be processed at the same time; no more than one job can be handled on a machine at the same time; the same priority level at each operation; there is no setup and idle time; there is no break time; all machines are available at zero in the usage time; machine efficiency is 100%; the money value is not considered. The following additional definitions and notation will help in formulating the problem:

i : number of machines;

j_i : number of operations of machine i;

p_{ij} : processing time of operation j on machine i;

o_j : sequence and technique restriction of job;

t_{ij} : starting time of operation j on machine i;

t_j : completion time of operation j.

$$X_{ijk} = \begin{cases} 1, & \text{if operation } j \text{ precades operation} \\ 0, & \text{otherwise} \end{cases}$$

$$Z_{ij} = \begin{cases} 1, & \text{if operation } j \text{ is allocated on machine} \\ 0, & \text{otherwise} \end{cases}$$

$C_{\max}$: makespan

According to above suggestion, parameter and decision variable of problem, the mathematical model is identified as followed:

$$\min C_{max} \ s.t.$$

$$\sum_{j=i}^{n} z_{ij} = 1 \qquad t_{ij} + p_{ij} \le t_{ik} + (1 - X_{ijk})M$$

$$t_{ij} + p_{ij} \le t_{i+1,j} \sum_{j=1, j \neq k}^{n} \sum_{i=1}^{n} X_{ijk} = 1$$

$$\sum_{i=1,}^{m} \sum_{j=1}^{n} (X_{ijk} + X_{ikj}) \le 1$$

III. EA FOR JOB SHOP SCHEDULING PROBLEM

The EA was first introduced by Holland (1975). It is a stochastic heuristics, which encompass semi-random search method whose mechanism is based on the simplifications of evolutionary process observed in nature. As opposed to many other optimization methods, EA works with a population of solutions instead of just a single solution. EA assigns a value to each individual in the population according to a problem-specific objective function. A survival-of-the-fittest step selects individuals from the old population. A reproduction step applies operators such as crossover or mutation to those individuals to produce a new population that is fitter than the previous one. EA is an optimization method of searching based on evolutionary process. In applying EA, we have to analyze specific properties of problems and decide on a proper representation, an objective function, and a construction method of initial population, a evolutionary operator and a evolutionary parameter. The following sub-sections describe in detail how the EA is developed to solve the above JSP problem.

A. Chromosome Representation and Decoding

The first step in constructing the EA is to define an appropriate evolutionary representation (coding). A good representation is crucial because it significantly affects all the subsequent steps of the EA. Many representations for the JSP problem have been developed. In this study, two representations based on working sequence and machine distribution are constructed. If the number of the machine type t (t>1), the genes in each chromosome will be divided into t parts in turn. Each part represents one type of machine. Each operation can only be assigned to the machines which can handle it. For example, suppose a chromosome is given as [4221134233114234] in 4 job×4 machines problem. Here, 1 implies operation of job J_1, and 2 implies operation of job J_2 and so on. Because there are four operations in each job, it appears the four times in a chromosome. Such as number 2 being repeated the forth in chromosome, it implies four operations of job J_2. The first number 2 represents the first operation of job J_2 which processes on the machine 1. The second number 2 represents the second operation of J_2 which processes on the machine 2, and so on. The representation of such problem is based on two row structure as follows:

Figure-1 Chromosomes genes and operations

Chromosome gene	4	2	2	1	1	3	4	2	3	3	1	1	4	2	3	4
Machine-operation	1-1	1-2	1-3	1-4	2-1	2-2	2-3	2-4	3-1	3-4	3-2	3-3	4-2	4-1	4-3	4-4

Figure-2 Chromosomes genes and machines

Chromosome gene	4	2	2	1	1	3	4	2	3	3	1	1	4	2	3	4
Job-operation	4-1	2-1	2-2	1-1	1-2	3-1	4-2	2-3	3-2	3-3	1-3	1-4	4-3	2-4	3-4	4-4

So the chromosome of machine 1 is 4 1 3 2, and the chromosome of machine 2 is 2 3 1 4, and the chromosome of machine 3 is 2 4 1 3, and the chromosome of machine 1 is 1 2 3 4.

B. Fitness and Selection

Fitness function is defined of each chromosome so as to determine which with reproduce and survive into the next generation. It is relevant to the objective function to be optimized. The greater the fitness of a chromosome is, the greater the probability to survive. In this study, the fitness function is defined as the function of the objective function as $f = 1/u(f(x))$.

C. Crossover

Since sequencing as well as assignment problems allow a permutation encoding, various permutation crossover operators have been developed. The crossover process is used to breed a pair of children chromosome from a pair of parent chromosomes using a crossover method. A binary vector of equal length as the permutation is filled at random. This vector defines the order in which the operations are successively drawn from parent 1 and parent 2. We now consider the parent and offspring permutations and the binary vector as list.

(1) The cross over based on the working procedure

Parent1 and Parent2 are selected as parent:

Parent 1 = $O_{11}, O_{13}, O_{12}, O_{21}, O_{22}, O_{24}, O_{31}, O_{32}, O_{33}, O_{34}, O_{42}, O_{43}$
Parent 2 = $O_{14}, O_{12}, O_{13}, O_{21}, O_{21}, O_{22}, O_{31}, O_{33}, O_{32}, O_{41}, O_{43}, O_{44}$

In this example, supposing that the processing order of job J_1 is O_{11}, O_{21}, O_{31} and the processing order of job J_1 is O_{11}, O_{21}, O_{31} and the processing order of job J_2 is O_{12}, O_{22}, O_{32}, O_{42}, and the processing order of job J_3 is O_{13}, O_{23}, O_{43} and the processing order of job J_4 is O_{24}, O_{34}. The position of O_{11}, O_{21}, O_{31} are provided from Parent1, and the position of O_{11}, O_{22}, O_{32} is provided from parent 2. Then the Offspring1 position of O_{11}, O_{21}, O_{31} is drawn from the parent 1 and deleted from parent2, the other position are filled with parent2, so as to offspring 2. Example of crossover is given as following:

offspring 1 = $O_{11}, O_{14}, O_{12}, O_{21}, O_{13}, O_{23}, O_{31}, O_{22}, O_{33}, O_{32}, O_{41}, O_{43}$
offspring 2 = $O_{11}, O_{12}, O_{13}, O_{21}, O_{24}, O_{22}, O_{31}, O_{33}, O_{32}, O_{34}, O_{42}, O_{43}$

(2) The crossover based on machine distribution

Here, we consider one-point crossover. A crossover point is selected from two parents randomly. Example of crossover is given in figure 3 and figure 4.

D. Mutation

The mutation operation is critical to the success of the EA since it diversifies the search directions and avoids convergence to local optima. We select a parent, and an operation is get randomly. As an example that an

E. Evolutionary algorithm Flow Chart

Step1 Initialization population is generated randomly, and it is feasible schedule.
Step2 The fitness is defined by objective of JSP model, and individual adaptive value is evaluated.
Step3 The crossover is operated in the population according to probability of crossover Pc, so the offspring is generated.
Step4 The individual is selected randomly according to probability of mutation P_m, so the offspring is generated.
Step5 The new individual adaptive value is calculated, parent and offspring are taken part in survival competition together.
Step6 Adjusting the termination criterion, then the optimal solution is obtained, otherwise going back to Step3.

Parent1	working sequence	O_{11}	O_{13}	O_{12}	O_{21}	O_{32}	O_{34}	O_{31}	O_{32}	O_{33}	O_{34}	O_{42}	O_{43}
	machine	1	3	2	4	3	1	2	4	2	1	3	4
Parent2	working sequence	O_{14}	O_{12}	O_{13}	O_{21}	O_{23}	O_{22}	O_{31}	O_{33}	O_{32}	O_{41}	O_{43}	O_{44}
	machine	4	2	3	1	3	2	4	1	2	1	3	4

IV. SIMULATION STUDY

In order to investigate the effectiveness of the proposed algorithm, the experiments were conducted based on the production data. The experiment were conducted under five machines and eight processes, and each process has P=[2,5,2,3,4,4,2,3] machines number. We consider initial population that is 50, and iteration is 50, and probability of mutation is 0.2. The process time is brought randomly. The result is as followed:

Figure-3 Distributed machine from parent

Figure-4 Distributed machine from offspring

offspring1	working sequence	O_{11}	O_{13}	O_{12}	O_{21}	O_{22}	O_{24}	O_{31}	O_{32}	O_{33}	O_{34}	O_{42}	O_{43}
	machine	1	3	2	1	2	1	4	2	1	1	3	3
offspring2	working sequence	O_{14}	O_{12}	O_{13}	O_{21}	O_{23}	O_{22}	O_{31}	O_{33}	O_{32}	O_{41}	O_{43}	O_{44}
	machine	4	2	3	4	3	3	2	1	4	1	4	4

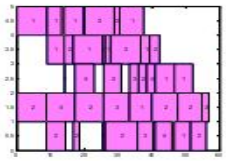

Figure 5. Gantt chart

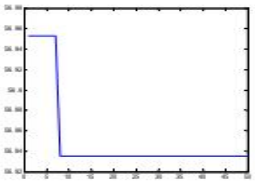

Figure 6. Minfitness convergence curve.

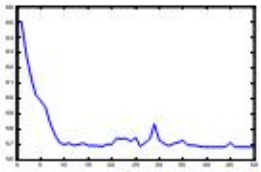

Figure 7. Minfitness convergence curve

The above example shows the makespan is 56.8354. The minfitness is attained 56.8354, when the iterative number is 7 in the figure 2. The mean fitness swings around 56.8354, when the iterative number is 39 that the mean fitness attains stabilization.

V. CONCLUSIONS

This paper proposed a EA for solving the agile job shop scheduling to minimize the makespan. Two-row chromosome structure is designed based on working procedure and machine distribution. The relevant crossover and mutation operation is also given. Finally, the Gantt chart is drawn based on five machines and eight processes. The computational result shows that EA can obtain better solution. The minfitness and mean fitness validate the solution that is validity.

VI. REFERENCES

[1] Lars Monch, Rene Schabacker, Detlef Pabst et al,*"Evolutionary algorithm-based Sub-problem Solution Procedures for a Modified Shifting Bottleneck Heuristic for Complex Job Shops"*, European Journal of Operational Research, Vol.3 No.177,pp.2100-2118 , 2007

[2] Young Su Yun, *"Genetic Algorithm with Fuzzy Logic Controller for Preemptive and non-Preemptive Job Shop Scheduling roblems"*, Computers & Industrial Engineering, Vol.3 No.43 pp.623-644 , 2007

[3] Dessouky, M.M, Leachman et al, *"Dynamic models of production with multiple operations and general processing times"*, Journal of the Operational Research Society, Vol.6 No.48 pp:983-997 , 1997

[4] Makoto Asano, Hiroshi Ohta, *"A Heuristic for Job Shop Scheduling to Minimize Total Weighted Tardiness"*, Computers & Industrial Engineering, Vol. 2-4 No 42 pp:137-147 , 2002

[5] Gomes, M.C., Barbosa-Povoa et al, *"Optimal scheduling for*

flexible job shop operations", International Journal of Production Research, Vol.11 No.43 pp:2323-2353，2005

[6] Taicir Loukil, Jacques Teghem, Philippe Fortemps ，*"A multi-objective production scheduling case study solved by simulated annealing"*, European Journal of Operation Research, Vol.3 No.179 pp:709-722，2007

[7] Jason Chao-Hsien Pan, Jen-Shiang Chen, *"Mixed–Binary Integer Programming Formulations for the Reentrant Job Shop Scheduling Problem"*, Computers & Operations Research, Vol.5 No.32 pp:1197-1212，2005

[8] S.Q.Liu, H.L.Ong,K.M.Ng(2005), *"Metaheuristics for Minimizing the Makespan of the Dynamic Shop Scheduling Problem"*, Advances in Engineering software, Vol.3 No.36 pp: 199-205

[9] Bortjan Murovec, Peter Suhel, *"A repairing technique for the local search of the job-shop problem"*, European Journal of Operational Research, Vol.1 No.153

[10] Chen Hua-ping, Gu Feng, Lu Bing-yuan et al, *"Application of Self-adaptive Multi-objective Evolutionary algorithm in Flexible Job Shop Scheduling"*, Journal of System Simulation, Vol.8 No.18 pp: 2271-2274，2006

[11] Hong Zhou, Yuncheng Feng, Limin Han, *"The Hybrid Heuristic Evolutionary algorithm for Job Shop Scheduling"*, Computers & Industrial Engineering, Vol.3 No.40 pp:191-200, 2001