PARALLEL LINE AND MACHINE JOB SCHEDULING USING GENETIC ALGORITHM

Dr.V.Selvi
Assistant Professor,
Department of Computer Science
Mother Teresa women's University
Kodaikanal.
Tamilnadu,India.

Abstract - *Parallel line and machine job scheduling involves the optimal allocation and scheduling of jobs in multiple processing lines. All jobs allocated in a line based on the processing time, setup time and processed in the particular order of machines or different order of machines .The parallelism is achieved through increasing the number of processors. The main objective of this paper is to find the minimize makespan for the maximum completion time of all jobs using genetic algorithm.*

Keywords: Parallel job scheduling, genetic algorithm, MATLAB.

1. INTRODUCTION

Most theoretical models assume that machines can be operated in a long period of time without maintenance. In the real world, this assumption may not be true under industrial conditions. According to practical experience, machine breakdowns are found prevalently in many manufacturing systems. Therefore, the question of how to schedule maintenance to reduce the breakdown rate becomes a common one in many companies.[1] In many production systems, we usually find that periodic repair, periodic inspection and preventive maintenance are conducted in the shops. These maintenance works are performed regularly in the production systems. Although the problem is important, it is relatively unexplored in the scheduling literature. Thus, the periodic maintenance scheduling problem is discussed in this paper.

The jobs are parallel in the sense that each of them specifies the number of processors, in this case 1 or 2, required for simultaneous processing. The jobs are presented one by one. Upon receiving a job, we must assign the job to a time slot in the schedule before the next job is presented. No re-assignment is allowed. Parallel jobs may require more than one machine simultaneously. They are characterized by two parameters, processing time and size (the number of requested machines). Jobs {J1, J2, · · · , Jn} arrive over list, which are not known in advance. Upon arrival of job Ji, its processing time and the number of machines required become known, and it must immediately and irrevocably be scheduled. The next job Ji+1 appear only after job Ji has been assigned. The goal is to minimize the makespan, i.e., the largest completion time over all the jobs.

The rest of this paper is organized as follows: Section 2 describe manufacturing scheduling. In section 3 problem descriptions with makespan. In section 4 Methods and materials and its objective function, genetic implementation and its operators. In section 5 results and its discussion .In section 6 conclusion and future work.

2. MANUFACTURING SCHEDULING

2.1 Parallel Machine Scheduling

Parallel machine scheduling involves scheduling a set of tasks on two or more machines that work in parallel with each other. The machines perform identical operations and may or may not operate at the same pace. In this type of problem, the tasks are assigned to either machine for processing, and flow between machines is not allowed.

The Job Shop is linked in some way to Parallel Machine Job Shop. Indeed as can see in the Figure 1.1, a Job Shop is composed of different stages which can contain one single machine or parallel machines.

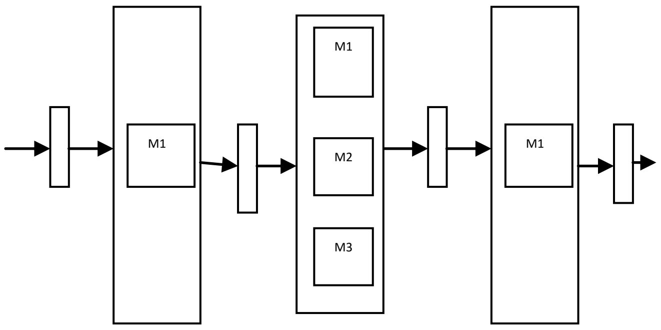


Figure 1.1

So this type of problem can be described as a sequence of parallel machine problem. Moreover the Parallel Job Shop problem has been widely studied especially for the minimization of the total tardiness.

2.2 Parallel line scheduling

A parallel line scheduling problem is one where there is more than one processing line for jobs to be processed. There are a number of machines in each line but the number of machines may be different. The processing time of a job is different for each machine. Each job is allocated to only one particular line and is machined to completion in that line. In other words, a job cannot move between lines. Further, the jobs enter their respective lines simultaneously and are processed in parallel. The time when the last job is completed is the completion time of the problem. In other words, the highest makespan among all lines is the
makespan of the process.

This paper has been worked under the following assumptions:
(i). All lines contain the same number of machines and different number of machines.
(ii). The set up time of a job with respect to another job is the same for all the lines.
(iii). Any job can be machined in any line.
(iv). Breakdown of machine is not allowed.
(v). Machines of each line are identical and in the same order.

The objective considered in this work is minimization of makespan. Makespan includes the processing times of the jobs and their respective set up times. The genetic algorithm has been used to find the optimal schedule with minimum makespan .

So this type of problem can be described as a sequence of parallel machine problem. Moreover the Parallel Job Shop problem has been widely studied especially for the minimization of the total tardiness.

2.3 Job Shop Scheduling

A job shop consists of two or more machines that perform specific operations, and a set of tasks that must be processed on some or all of these machines. Unlike the flow shop problem, there is no fixed path that products must follow through the system therefore the order of operations is not fixed. This type of layout is typically used when product variety is high and product volume is low.

2.4 Scheduling Problem Objectives

Some of the common objectives amongst scheduling problems are minimizing makespan, minimizing average flow time, minimizing the number of tardy jobs and maximizing the utilization of each machine. Minimizing makespan means that the goal is to reduce the amount of time it takes to complete all of the jobs, and minimizing average flow time means that the goal is to reduce the amount of time each job stays in the system.

3. PROBLEM DESCRIPTION

Problem variables the various problem variables are:
(i) Number of lines ‘l’
(ii) Number of machines on each line ‘m’
(iii) Number of jobs ‘n’
(iv) Quantity of each job to be processed ‘Q’
(v) Processing times of all jobs in all machines in each line ‘alnm’
(vi) Set up times of each job with respect to all other jobs

The processing times for all jobs in all machines in each line is obtained from this matrix. The set up time of each job with respect to all other jobs can be obtained from this matrix.

Makespan is calculated as follows:

MAKESPAN = PROCESSING TIME + SET UP TIME
Processing time is calculated for each job separately. For example, the processing time of J1 in Line 1 is
PTJ1 = Q1*T11
Where,

T11 The processing time of J1 in Line 1 = a111 + a112 +.....+ a11m
Q1 Quality of J1 required.

Similarly the processing time for all jobs in all lines is calculated.

Now, the total processing time of all jobs in a line is obtained by adding the individual processing times of all jobs in that line. For example, if J1,J2 and J3 are in line 1, the total processing time of all jobs in line 1 is

PTline1 = T11 + T21 + T31

Finally, the respective set up times of the jobs(in order) are added to the processing time of that line. The sum is called makespan of that line. Let the order of the jobs in
Line 1 is J1 followed by J2 and by J3.

Makespanline1 = PTline1 + S21 + S32

Similarly makespans of all lines are calculated. The largest of all makespans is taken as the makespan of the entire process. The objective of this paper is to find the allocation and schedule of jobs so as to minimize this makespan.

4. METHODS AND MATERIALS

4.1GENETIC ALGORITHM

Genetic algorithms are used for solving complex problems. Genetic algorithms use the process of simulated evolution to intelligently search a solution space of a problem. By using the Darwinian principles of classical biological evolution, genetic algorithms seek to produce better solutions based on past performance information. The advantages of using a genetic algorithm versus traditional solution approaches, such as mathematical modeling, branch and bound, and hill climb, include the ability to avoid convergence on local optima, ability to search a large sample space, and comparably faster solution times.

This section outlines some of the basics of genetic algorithm. The three most important aspects of using genetic algorithms are:

(i) Definition of the objective function,
(ii) Definition and implementation of the genetic representation,
(iii) Definition and implementation of the genetic operators.

Once these three have been defined, the generic genetic algorithm should work fairly well.

4.1 Definition of the objective function

To use genetic algorithm, one must encode solutions to the problem in a structure that can be stored in the computer. This object is a genome. The genetic algorithm creates a population of genomes then applies crossover and mutation to the individuals in the population to generate new individuals. It uses various selection criteria so that it picks the best individuals for mating.

The genetic algorithm is very simple, yet it performs well on many different types of problems. Basically, if the objective function is right, the representation right and the operators right, then variations on the genetic algorithm and its parameters will result in only minor improvements.

4.2 Definition and implementation of the genetic representation

In order to implement a genetic algorithm, a chromosome definition, fitness function, and evolution strategy must first be defined. A chromosome is a representation of the independent variables that can be manipulated to determine how well the solution solves the problem (fitness) and to produce alternate solutions (reproduction). Typically a chromosome will consist of an array of values.

The fitness function of a genetic algorithm evaluates each candidate solution, chromosome, to determine how well the solution performs with respect to the desired objectives. This information is used to choose which solutions to keep and which ones to discard (survival of the fit). The fitness of a candidate is used to determine how likely a solution is to reproduce (selection probability).

4.3 MATERIALS

4.3.1Definition and implementation of the genetic operators

The evolutionary strategy of a genetic algorithm defines how new solutions are formed from the previous solutions. The evolutionary strategy consists of the crossover, mutation, repair, selection, and termination operators. These are the core of the genetic algorithm, and their definitions vastly impact both computation time and solution quality.

4.3.2 Crossover

Crossover is the process of making new candidate solutions from previous candidate solutions. This process uses the principals of biological evolution to form children (new candidates) from parents (old candidates). The first step in crossover is to select a mating pair from the parent population. These solutions are chosen based on their performance (fitness). Solutions that perform better have a greater chance of being selected for mating, this idea come from the Darwinian principle of "survival of the fit."

Once the parents have been selected, a cut point must be determined. This can be accomplished by choosing a random number. The first child is formed by using the all of the genes from the first parent up to the cut point, and all of the genes from the second parent after the cut point. The second child is then formed using the remaining genes from each parent. The idea here is that parents with high fitness values have good genes, and these good genes then can be used to form good children.

4.3.3 Mutation

Mutation is the process of manipulating individual genes in a candidate solution. This is a completely random process that does not use any information from the parent chromosomes or the fitness of the candidate solution. Mutation uses random numbers to select genes, and then modifies the selected genes according to the strategy applied by the developer of the algorithm. This is a local search technique that tries to improve candidate solutions by looking at other candidates nearby in the solution space.

4.3.4 Selection

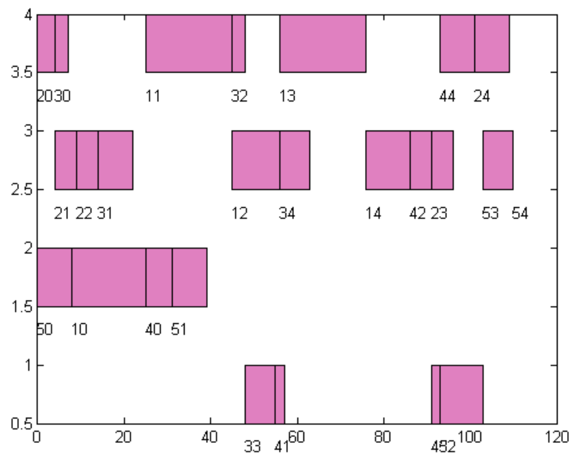
The process of selection is used to form each subsequent population during the evolutionary process. This process is done by computing the fitness function of each candidate solution, and then selecting the individuals that will form the next generation's parents. There are many techniques used in the selection process, some of which include tournament, elitist, and roulette wheel selection. During tournament selection a subsection of the population is randomly selected and the candidate with the highest fitness value is passed to the next generation. This process is repeated until the parent population of the subsequent generation is formed. The elitist strategy selects the top N individuals from the current child and parent population to form the parent population of the next generation. Finally, roulette wheel selection involves constructing a cumulative probability distribution based on the fitness function of each candidate and then drawing random numbers to choose the parents of the next generation.

4.3.5 Termination

The termination operator is the mechanism for stopping the evolutionary process of a genetic algorithm. Termination can occur when the algorithm has reached a predetermined number of generations, an acceptable solution has been found, or when no solution improvement has occurred for a set period of time. Additionally it should be noted that manual termination can also be a termination operator, this occurs when the user stops the evolutionary process.

5. RESULTS AND DISCUSSION

The entire process has been automated with the help of a user-friendly MATLAB program coded in WINDOWS platform. The program has been tested for various data. The quantity of each job processed is given in Table below.



6. CONCLUSION

This paper solves the problem of parallel line job shop scheduling problem using the genetic algorithm optimization technique. The MATLAB program code can be used to achieve the optimal allocation and schedule of given jobs in parallel line. It also gives minimum makespan for a given problem. It can be executed for any number of lines, jobs, and different number of machines per line. It also gives the minimum makespan for a given problem.

References:

1. Glazebrook KD, Niño-Mora J (2001) Parallel scheduling of multiclass M/M/m queues: Approximate and heavy-traffic optimization of achievable performance. Oper Res 49(4):609–623

2. Fu Z, Golden BL, Lele S, Raghavan S, Wasil EA. (2003) A genetic algorithm-based approach for building accurate decision trees. INFORMS J Comput 15(1):3–22

3. Drezner Z (2003) A new genetic algorithm for the quadratic assignment problem. INFORMS J Comput 15(3):320–330

4. Ataei M, Osanloo M (2004) Using a combination of genetic algorithm and the grid search method to determine optimum cutoff grades of multiple metal deposits. Int J Surf Min Reclam Environ 18(1):60–78

5. Lin Z-Z, Bean JC, White CC (2004) A hybrid genetic/optimization algorithm for finite-horizon, partially observed Markov decision processes. INFORMS J Comput 16(1):27–38

6. A.Noorul Hag& k.Balasubramanian (2007) Parallel line job shop scheduling using genetic algorithm #Springer –Verlog London limited.

7. Allard ,David .M A MULTI-OBJECTIVE GENETIC ALGORITHM TO SOLVE SINGLE MACHINE SCHEDULING PROBLEMS USING A FUZZY FITNESS FUNCTION(thesis), College of Engineering and Technology of Ohio University.