BULETINUL Universităţii Petrol – Gaze din Ploieşti	Vol. LXII No. 1/2010	113 - 118	Seria Matematică - Informatică - Fizică

Using Genetic Algorithms for Production Scheduling

Florentina Alina Chircu

Petroleum-Gas University of Ploiesti, Informatics Department, Ploieşti, Romania
email: chircu_florentina@yahoo.com

Abstract

Scheduling is an important tool in the manufacturing area since productivity is inherently linked to how well the resources are used to increase efficiency and reduce waste. This paper presents the implementation of a genetic algorithm in manufacturing. The genetic algorithm's goal is to obtain a detailed plan that represents the tasks' order and the completion time for each resource. The application aim is the identification of the best allocation of resources that minimizes the makespan for a predetermined quantity of products, considering the system restrictions.

Keywords: *artificial intelligence, genetic algorithm, production scheduling, resource allocation*

Introduction

Production scheduling generally involves organizing and planning of processes; it can be defined as a technique that organizes each step in a long sequence of separate operations, each step being done at the right time and place, each operation being carried out with maximum efficiency [8]. This helps establishing the exact amount of necessary labor, machinery and money to produce a predetermined quantity of products in a given time, providing also a general idea over the economic efficiency of the production.

A set of inputs, a set of constraints and a set of finite resources define a classic scheduling problem. Usually the resource constraints in a scheduling problem consist of: a set of jobs that requires to be executed, a limited set of resources, a set of procedural and temporal constraints and a set of objectives to evaluate the scheduling performance [5]. The job-shop scheduling model refers to the problems described by the fact that each product must visit the machines in a given sequence, but the sequence is different for each products type. Let $O = \{0, 1, \ldots, n\text{-}1\}$ be the set of jobs to be scheduled and $M = \{1,..., m\}$ the set of machines. The jobs are organized by two constraints [4]: (1) each task j of a job must be scheduled after all the predecessor tasks in that job are completed (the precedence constraints), and (2) the task j can only be scheduled if the machine it requires is idle. In other words, one machine can only process one task at a time. The objective is to minimize the makespan.

The traditional methods and algorithms designed to solve job-shop scheduling problems returned specific results limited to the case of the two-machine and two-job problems. The problem of M machines job-shop is a combinatorial optimization problem. The most difficult combinatorial optimization problems are NP-complete [5]. There are no efficient algorithms implemented for these types of problems. In this case, the implementation of a

heuristic method is completely justified. The performance of the heuristic method depends on the quality of the obtained solution [6].

Over time, different types of techniques have been implemented to solve the job-shop scheduling problem. In 1996, Vaessens suggests the implementation of the local search technique in this area [9]. Simulated annealing and genetic algorithms are examples of such techniques. Simulated annealing technique was proposed as a solution for the job shop scheduling problem by Laurenco in 1995 [4]. The first genetic algorithm was applied to the job-shop scheduling problem in 1985 by Davis [2]. Since then, many authors, such as Croce [1] and more others, have proposed different approaches for this problem using genetic algorithms. A comprehensive survey of job shop scheduling techniques along with a comparative analysis can be found in a study recommended by Jain and Meeran [3].

The genetic algorithm is a technique inspired by the biological mechanism, based on the principles of natural evolution and selection. Although the genetic algorithms have been designed to respond to specific needs in the biology field, they have quickly adapted to a large variety of problems. The genetic algorithm presented in this paper searches a resource allocation for obtaining a predetermined quantity of products with a minimal makespan. The results are represented by the production sequences.

The rest of this paper is structured as follows: the next section briefly describes the genetic algorithms, the third section contains a description of the problem proposed to be solved, the fourth section includes the experimental results, and the last section is dedicated to conclusions.

Genetic Algorithms

Genetic algorithms are evolutionary search techniques used to identify approximate solutions for optimization problems. They represent a computer simulation of a population of abstract representation (called chromosomes) of the candidate solutions (called individuals) to an optimization problem that evolves toward better solutions [7].

The algorithm starts with a complete or partial randomly generated population. The evolution is simulated in generations. Each individual in this population has attached a fitness function that represents the individual performance based on a number of criteria. The new population is obtained from the old population by following three important steps [7]: selecting the best individuals to become parents, performing crossover on the parents to obtain new individuals, and performing mutation to some very few individuals. The selection of individuals that will become parents is an important stage of the algorithm. Based on the fitness function value attached to each candidate, the individuals are chosen to become parents in order to increase the solution's quality.

Crossover operation is a genetic operator that aims to obtain the propagation of the best genetic material. The two chosen parents are combined and the resulted individuals are included in the new generation in order to increase the population performance. The mutation is represented by a chromosome modification applied to one or more genes.

Description of the Application

The problem proposed is a manufacturing system consisting of 6 machines M_1, M_2, …, M_6 for which several suppositions are made. The machines cannot substitute each other and the processing units are non-preemptive. In this ideal system no machine breakdown occurs and the passing times of jobs between machines are neglected. In addition, job-specific setup times of machines are not considered.

Generally, a production plan consists of n jobs, and each job consists of m_i jobs, each of them having to be processed by a single machine. The production schedule represents an order of the tasks and the starting times for each task considering the technological machine order of jobs. The completion time of a task can be obtained by adding the processing time to its starting time.

The input data for this problem is represented by the technological machine order of the task for each job and the corresponding processing time on the machines. The problem's constraints are represented by the operation sequence that is different for each product and by the fact that the machine can operate only one product at the time. The objective is to determine a schedule for the tasks on the machines in minimum time.

The genetic algorithm parameters are: initial population size, maximum number of generations, maximum number of individuals, and crossover probability and mutation probability. The main objective of this algorithm is to identify the best plan, i.e. to perform all necessary tasks in order to minimize the makespan.

The genetic representation for each candidate solution is:

$$(G_1, G_2, G_3, G_4, G_5, G_6, G_7, G_8, G_9, G_{10}, G_{11}, G_{12}, G_{13}, G_{14}, G_{15}),$$

where G_i is a structure that encodes the series number, the type and the starting times for each product series (there are 5 series for each type of products).

For example $G_i=(s_i, p_i, [t_1, t_2, t_3, t_4, t_5])$, represents a structure that refers to the s_i series of the product type noted with $p_{i,}$, which accesses the first machine form the operation sequence at the time t_1, the second machine form the operation sequence at the time t_2, the third machine form the operation sequence at the time t_3, and so on.

An individual's performance will be evaluated by his fitness function value, which needs to be minimized. At the beginning, the fitness function is initialized with 0 (because the execution time at the beginning of the execution is equal to 0).

The objective function minimizes the finishing time of tasks *n-1* (the last task), and therefore minimizes the makespan. The function value is updated according to both the ending time on the last machine, of the last product and several restrictions on total waiting times at machines and job sequence, by applying a set of specific rules.

The genetic operators selected for this implementation are:

- roulette selection;
- one-point crossover;
- rotation mutation.

By using the roulette selection, the future parents are chosen by simulating the launching of a roulette needle on the field of fitness values for the current population.

The one-point crossover copies the string from the beginning of the chromosome to the crossover point from one of the parents and the rest is copied from the other parent. The construction of the new individual is made considering the restrictions imposed by the problem.

The rotation mutation performs a small chromosome modification by selecting a bloc of genes with random length and by inversing the gene order. This modification is made considering the restrictions imposed by the problem.

The genetic algorithm stops when either the upper limit of generation or the maximum number of individuals has been reached. The individuals with the lowest value of the fitness function represent the solution returned by this algorithm. This individual represents the production scheduling with minimum makespan.

Experimental Results

To test the genetic algorithm results, we propose the production simulation of 15 series of products (5 for each of the three products types). The machines sequences for each product and the processing time on each machine for each product type are presented in Table 1.

The tasks sequence and the necessary time for each job are different for each of the three types:

- The task sequence for the first type of products (called Product 1) is represented by (M1, M2, M3, M5, M6), where Mi represents machine number i;
- The task sequence necessary to obtain the second type of products (called Product 2) is represented by (M1, M3, M4, M5, M6);
- The task sequence necessary to obtain the third type of products (called Product 3) is represented by (M1, M4, M5, M6);

Table 1. Machine sequences and processing time

		Product 1	**Product 2**	**Product 3**
Machine 1	Task Number	1	1	1
	Execution Time(minutes)	30	10	20
Machine 2	Task Number	2	-	-
	Execution Time(minutes)	10	-	-
Machine 3	Task Number	3	2	-
	Execution Time(minutes)	10	10	-
Machine 4	Task Number	-	3	2
	Execution Time(minutes)	-	10	40
Machine 5	Task Number	4	4	3
	Execution Time(minutes)	30	30	20
Machine 6	Task Number	5	5	4
	Execution Time(minutes)	10	10	10

Three sets of experimental values for the specific parameters of the genetic algorithm (initial population size, maximum population size, maximum generation number, crossover probability, and mutation probability) are specified in Table 2. 20 tests have been repeated for each set.

Table 2. Experimental Values Set

Values Set / **Parameters**	**1**	**2**	**3**
Number of tests	20	20	20
Initial population size	10	15	20
Maximum population size	20	30	50
Maximum generation number	10	15	20
Crossover probability	0.5	0.3	0.6
Mutation probability	0.07	0.02	0.15

Twenty tests have been executed for each set of input values, and the final results are synthesized in Table 3.

Best performances average represents the mean of lowest values of the fitness function obtained in the 60 sets of results. Worst performances average at last generation represents the average of highest values of the fitness function. The average performance averages is calculated considering the values of the medium performances obtained during the experimental tests.

Table 3. Final results

Final results	
Best performance average at last generation	455
Worst performance average at last generation	703
Average performance	579
Best performance	470
Best performance solutions number	1
Best performance solution	(0, 10, 3, 14, 13, 6, 5, 9, 8, 1, 12, 2, 4, 7, 11)

After completing all the experimental tests, the lowest value obtained for the fitness function is 470 and only one solution has managed to reach this value. This solution represents a scheduling plan that manages to respect both the planned production and the 480 minutes limit imposed by the problem.

The solution with the minimum fitness function returned is: (0, 10, 3, 14, 13, 6, 5, 9, 8, 1, 12, 2, 4, 7, 11) and represents the technological order of production series. For each series a set of detailed specifications (the starting times on each machine, the total execution time and the completion times on each resource) is stored in a text file. The solution returned by the algorithm can be decoded as: first series of products that enters the system is Series 0 (that corresponds to the Product 1), after that follows the Series 10 (that corresponds to Product 3), the Series 3 (that corresponds to Product 2), and so on. The list of values that correspond to the moments of time in which each series accesses the machines is presented in a text file that contains the detailed solution.

Conclusions

This paper proposes an implementation of an artificial intelligence technique in the production planning area. Thus, we have implemented a genetic algorithm to solve a problem from the manufacturing domain. The proposed problem is represented by a manufacturing system that consists of six machines for which several suppositions are made. The problem has two constraints: the task sequence is different for each product, and the machine can operate only one product at the time. The objective is to determine a schedule for products on each machine, respecting the time limit.

The solution proposes a resource allocation that minimizes the makespan and guarantees the obtaining of a specific number of products. It is important to note that the solution was formulated after a sixty tests because the algorithm starts with a partial randomly generated population. The results can be translated as a production plan, which includes both the task sequences on the six machines and the execution time for each product type.

As further research, the implementation of a better fitness function can be mentioned aiming to obtain a better solution and to minimize the makespan. A comparison with other artificial intelligence techniques applied to the proposed problem is also envisaged.

References

1. Croce, F., Tradei, R., Volta, G. - A Genetic Algorithm for Job Shop Scheduling Problem, *Computer Operations Research*, 1995
2. Davis, L. - Job Schop Scheduling with Genetic Algorithms, *Procedings of The First International Conference on Genetic Algorithms and their Applications*, 1985
3. Jain, A., Meeran, S. - A State-of-the-art Review of Job Shop Scheduling Thechniques, *European Journal of Operations Research*, Vol. 113, 1999
4. Laurenco, H.R. - Local Optimization and the Job Shop Sheduling Problem, *European Journal of Operational Research*, 1995
5. Lopez, P., Roubellat, F. - *Production Scheduling,* ISTE Ltd, 2008
6. Nazif, H., Lee, L.S. - Algorithm on Single Machine Scheduling Problem to Minimise Total Weight Completion Time, *European Journal of Scientific Research* Vol. 35, No. 5, 2009
7. Oprea, M., Nicoara, S. - *Artificial intelligence,* Petroleum–Gas University of Ploiesti, 2005
8. Sawik, T. - *Production Planning and Scheduling in Flexible Assembly Systems,* Springer, 1999
9. Vassesens, R.J.M., Asrts, E.H.L, Lenstra, J.K.L. - Job Shop Scheduling by Local Search, *Informs Journal on Computing*, Vol. 8, No. 3, 1996

Utilizarea algoritmilor genetici în planificarea producției

Rezumat

Planificarea este unul dintre cele mai importante instrumente în cadrul producției având în vedere faptul că productivitatea este legată in principal de modul în care resursele sunt utilizate pentru a creste eficiență şi pentru a reduce pierderile. Această lucrare prezintă implementarea unui algoritm genetic în acest domeniu. Algoritmul genetic este proiectat pentru a obține o planificare detaliată care are ca obiectiv prezentarea ordinii în care se execută taskurile şi timpul de finalizare pe fiecare resursă în parte. Scopul aplicației este reprezentat de identificarea celei mai bune modalități de repartizare a resurselor pentru a minimiza timpul de terminare al ultimii operații în cadrul sistemului pentru o cantidate predefinită de produse şi respectând restricțiile impuse de sistemul de producție.