ISSN 1330-3651 (Print), ISSN 1848-6339 (Online)
https://doi.org/10.17559/TV-20150527133957

APPLYING IMPROVED GENETIC ALGORITHM FOR SOLVING JOB SHOP SCHEDULING PROBLEMS

Gordan Janes, Mladen Perinic, Zoran Jurkovic

Preliminary communication

The Job Shop Scheduling Problem (JSSP) is one of the most general and difficult of all traditional scheduling combinatorial problems with considerable importance in industry. When solving complex problems, search based on traditional genetic algorithms has a major drawback - high requirement for computational power. The goal of this research was to develop fast and efficient scheduling method based on genetic algorithm for solving the job-shop scheduling problems. In proposed GA initial population is generated randomly, and the relevant crossover and mutation operation is also designed. This paper presents an efficient genetic algorithm for solving job-shop scheduling problems. Performance of the algorithm is demonstrated in the real-world examples.

Keywords: *genetic algorithms; optimization; scheduling; serial production*

Učinkoviti genetski algoritam za planiranje proizvodnje

Prethodno propćenje

Problem planiranja proizvodnje je jedan od najvažnijih ali i najkompleksnijih kombinatornih problema, koji je ujedno i veoma bitan u proizvodnji. Kod rješavanja kompleksnih problema korištenjem uobičajenih genetskih algoritama javlja se potreba za značajnom upotrebom računalnih resursa. Cilj ovog istraživanja je bio razvoj brze i efikasne metode određivanja redoslijeda u proizvodnji bazirane na genetskom algoritmu. U promatranom GA početna populacija se kreira slučajnim odabirom, a predloženi su i modificirani operatori križanja i mutacije. Svojstva testiranog algoritma provjerena su na problemima iz prakse.

Ključne riječi: *genetski algoritam; optimizacija; planiranje rasporeda; serijska proizvodnja*

1 Introduction

Genetic algorithms were first proposed by Holland in the 1970s [1] and have been successfully used in a variety of problems. Genetic algorithm (GA) is a heuristic search that mimics the process of natural evolution. It is inspired by Darwin's theory of evolution: problems are solved by using techniques of natural evolution: inheritance, mutation, selection and crossover. Genetic algorithms belong to the larger class of evolutionary algorithms (EA). Since their introduction, they have been extensively studied and successfully applied to a large variety of optimization problems over finite (discrete) domains. L. Davis [2] first applied a genetic algorithm to the JSSP in 1985 successfully, and now genetic algorithms have been proved to be an effective approach for the JSSP.

Scheduling is one of the critical issues when planning and managing manufacturing processes and represents a significant problem for many companies [3÷5]. One of the most difficult problems in this area is the job shop scheduling problem (JSSP), which is well known as one of the most difficult NP-hard problems [6, 7]. In this paper we tested efficiency of a proposed algorithm for solving JSSP problems and for this purpose a dedicated software was written in C#.

2 The problem formulation

The flexible job-shop scheduling problem can be formulated as follows: There is a set of NP products $P = \{P_1, P_2,\ldots, P_i,\ldots, P_N\}$. To complete each of these products we need to complete corresponding job from a set of K jobs $J = \{J_1, J_2,\ldots, J_j,\ldots, J_K\}$. Each job J_j consists of a predetermined sequence of operations. Each operation requires one machine M selected from a set of available machines $M = \{M_1, M_2,\ldots, M_k,\ldots, M_M\}$.

The aim is to optimize production by using a genetic algorithm or in other words to find a schedule of the operations on the machines with the shortest makespan taking into account the precedence statements:

- All machines are available at time 0
- All jobs are released at time 0
- Every job is a chain of operations and order of operations has to be maintained
- All operations of a given job have to be processed in a given order.
- Two operations of a job cannot be processed at the same time;
- Each machine can process at most one operation at a time
- Once processing starts on a given machine, it must complete on that particular machine without any interruption.
- Each operation has a fixed duration.
- Each machine cannot process more than one operation at a time
- All machines are available at zero time and machine efficiency is 100 %
- There is no break time.

For the production, there are available nine machines: two touring machines, one induction hardening machine, two milling machines and three grinders.

3 Genetic algorithm

3.1 Chromosome representation and decoding

Genetic algorithm is an effective meta-heuristic method to solve combinatorial optimization problems and in this paper we used it to solve this scheduling problem. When solving problems with genetic algorithms, the first step is to represent a solution to a problem as

chromosome. A good representation is crucial because it significantly affects all the steps of the algorithm and has a great impact on computational time. The genes of the chromosomes describe the type of product and assignment of operations to the machines.

It is assumed that a potential solution to a problem may be represented as a set of parameters. Based on the analysis of the problem approach from the literature, we design an improved chromosome representation to reduce the cost of decoding (computational time). Our chromosome representation has three components: Product Code Component (PCC), Series Size Component (SSC) and Machine Code Selection Component (MCSC).

We used an array of integer values to represent Product Code Component and Machine Code Selection Component. Size of array equals five times number of products that are in the production queue, since the description for each job takes six elements (Fig. 1). First element is reserved for product code and the order the products appear in the chromosome describes the sequence of manufacturing. Second element is used for size of the series, and next four are used for operations descriptions (maximum number of operations in tested scheduling problems is four).

2	27	3	5	4	9

Figure 1 Chromosome representation and decoding (example)

As can be seen from example in Fig. 1 the first element in array (PCC) represents product code (product code value is "2"). Value of the second element (SSC) is 27 – that means that there are 27 products in the series. Next four elements in the array are used for description of operations and machines (MCSC). Codes for machines are used according to Fig. 2.

Operation	Code	Machine
Buffer 1	1	-
Turning	2	Turing machine 1
	3	Turing machine 2
Induction hardening	4	Induction hardening machine
Milling	5	Milling machine 1
	6	Milling machine 2
Grinding	7	Grinder 1
	8	Grinder 2
	9	Grinder 3
Drilling	10	Drilling machine
Buffer 2	11	-

Figure 2 Machine codes

As can be seen in Fig. 2 there are two turning machines, one milling machine, one induction hardening machine, three grinder machines.

3.2 Population initialization

The initial population in proposed algorithm is produced randomly. Random generation of the initial population can require more computational power to obtain a solution, but ensures high genetic diversity and this is important to avoid premature convergence and to escape the local optimum [8].

Each chromosome (individual) represents a possible solution to the optimization problem. After the creation of the initial population, fitness of each chromosome in the population is calculated. In this research size of the population was between 100 and 1000 individuals.

3.3 Selection and crossover

The goal of the crossover operator in GA is to obtain better chromosomes to improve the result by exchanging genetic material contained in the population [9]. Process of selection and crossover in proposed algorithm is based on the following rules:

- Number of individuals in a habitat is designated by resources of habitat, or in other words number of individuals in the population is limited and cannot increase or decrease. Crossover and selection used in GA are consistent with the theory of evolution: In real world number of living beings in a certain habitat is limited by its resources: food, water, a shelter…
- Individuals that are not successful enough to gain access to the resources will not survive. Process of selection and crossover is based on tournament elimination and can be divided into four steps:

Step 1. In first step, Ts individuals are selected, and they enter into tournament competition.

Step 2. Individual with the lowest fitness is found among individuals in the tournament and it is eliminated from the population.

Step 3. Within whole population, two individuals are selected randomly. To avoid premature convergence (and loss of genetic material) all individuals in the population have the same chance to be selected for mating except individual with the lowest fitness which will be removed from the population. These two individuals will become parents and their genetic material will be used in the creation of a new individual. The individual that was eliminated in the previous step is replaced with a newly created individual

Step 4. Newly created individual replaces the one that was eliminated during the process of selection.

3.4 Mutation

Mutation is a process of changing one randomly chosen gene of randomly chosen individual. Mutation is a genetic operator used to maintain the genetic diversity in a genetic algorithm assuring that the final result of the algorithm would not finish in a local optimum that is not necessarily a global optimum. Like in biological mutation, mutation in GA can alter one or more genes.

After the genome in the process of mutation is changed, fitness for individual that mutated has to be recalculated. To avoid loss of the best yet found solution, in each generation individual with the best fitness (lowest makespan) cannot mutate.

Mutation introduces some new genetic material into the population to enhance the diversity of a population. In GA, a mutation is applied with small probability since large probability of mutation may lead to loss of the good genes and, as a result, has slowdown of algorithm. To ensure that mutation will fulfill its purpose in the genetic algorithm two types of mutation were adopted: product sequence and machine selection mutation.

3.4.1 Product sequence mutation

The role of mutation is to attempt to form a better schemata [10], the shorter makespan by changing sequence order that products enter into production. Since there are fixed quantities of each type of products that entered production, during the process of mutation those numbers must not be changed during mutation. That means that during this type of a mutation we cannot assign random values to altered genes, or in other words, we can only change the product sequence in the production queue (Fig. 3). Product sequence mutation is described as follows:

Step 1. Select one individual form population
Step 2. Select first product
Step 3. Select second product
Step 4. Swap position of selected products in production queue.

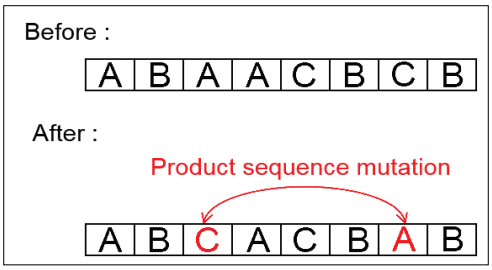

Figure 3 Product sequence mutation (example)

3.4.2 Machine selection mutation

Since in some operations we can use more than one machine in this type of mutation we alter genes that represent machines, or in other words machine for a certain operation is altered. Newly assigned machine must be included in the machine set of the corresponding operation. All individuals in the population have the same chance for mutation except the best individual which is protected from alteration of its genes. Machine sequence mutation is described as follows:

1. Select one individual form population
2. Select one product
3. Select one operation
4. Assign new machine for selected operation.

For example, there are three machines that can be used during the process of grinding of product "A". In the example shown in Fig. 4 instead of grinding machine GRND_2 machine GRND_1 is used.

product "A"				
Operation	**before** mutation		**after** mutation	
	code	machine	code	machine
1.	1	-	1	-
2.	2	Turing machine 1	2	Turing machine 1
3.	4	Induction hardening machine	4	Induction hardening machine
4.	8	Grinder 2	7	Grinder 1
5.	11	-	11	-

Figure 4 Machine selection mutation (example)

3.5 Fitness evaluation

The scheduling problem is an optimization problem where the ultimate objective is to reduce the total cost of production by reducing the time of production, so we adopted time of makespan as a fitness value. Fitness function is used to determine which will survive and go into the next generation and reproduce. The lower value of the makespan is considered as better fitness of a chromosome. Better fitness enhances the probability of chromosome to survive.

The makespan is calculated for each individual and this value is used as the fitness, reflecting the way each individual is, in comparison to the others, more desirable final solution and nearer to the optimum. Since late delivery is a pervasive problem in today's highly competitive market, we are looking for a chromosome with lower value. Genetic evolution during selection will prefer chromosomes with a better fitness (a lower makespan).

3.6 Evolution program pseudo-code

Genetic algorithms are inspired by nature [11]. The environment acts as a selector: the less adapted chromosomes die off producing less offspring than the better adapted ones. The occasional mutations insure that genes that are not present in the population improve characteristic of mutated chromosomes and enter the genetic pool of population. Many of the commonly used genetic algorithms have problems with speed of convergence or with high demand for computer resources. In order to overcome these drawbacks, in this paper, a modified crossover and selection operators were designed. Furthermore, to improve the speed of the algorithm only one population was used and we tried to keep calculations data movement in computer memory and as low as possible.

The proposed effective genetic algorithm terminates when a maximal number of iterations is reached, and the best individual, together with the corresponding schedule, is output (Fig. 5).

```
Genetic Algorithm
{
    Generate initial population P0
    Evaluate population P0
    While (stopping GA criteria not satisfied) Repeat
        {
        For 1 to (number of tournaments)
            {
                Select Nt chromosomes for tournament
                Find chromosome with lowest fitness
                Remove chromosome with lowest fitness
                Crossover (Create new chromosome)
                Evaluate new chromosome
            }
            Mutation
            Evaluate (mutated chromosomes)
        }
}
```

Figure 5 Machine selection mutation (example)

4 Computational results

We tested the algorithm on three scheduling problems of different sizes. We used the same setting for all tests. In order to obtain meaningful results, we ran our algorithm twenty times on the same instance and used the best. The parameters used in the GA are chosen

experimentally in order to get a satisfactory solution in an acceptable time. Parameter chosen for GA and result were as follows:

GA parameters	
Number of generation:	100
Size of population:	100
Crossover probability:	80 %
Product sequence mutation probability:	5 %
Machine selection mutation probability:	5 %

Figure 6 Case 1 – GA parameters - machines

4.1 Case 1

In this case input parameters were: nine machines and total of 1924 products, as can be seen from Fig. 8 and nine available machines Fig. 9.

Machine ($M_P = 9$)	pcs
Turing machine	2
Induction hardening machine	1
Milling machine	2
Grinder	3
Drilling machine	1

Figure 7 Case 1 – Machines used in scenario 1.

Products ($N_P = 3$)			
product	number of products in the series	size of series	Total
P_1	21	27	**577**
	10	1	
P_2	33	29	**962**
	5	1	
P_3	15	25	**385**
	10	1	

Figure 8 Case 1 – Input parameters

We ran our algorithm twenty times with the same input parameters and the best result was 63,54 hours. Machine utilization for this scenario can be seen in Fig. 9.

Operation	Code	Machine	Utilization
Turning	2	Turing machine 1	84,214 %
	3	Turing machine 2	84,273 %
Induction hardening	4	Induction hardening machine	39,306 %
Milling	5	Milling machine 1	50,590 %
	6	Milling machine 2	46,422 %
Grinding	7	Grinder 1	68,694 %
	8	Grinder 2	61,052 %
	9	Grinder 3	72,031 %
Drilling	10	Drilling machine	22,026 %

Figure 9 Case 1 – Utilization

4.2 Case 2

Since the best result in the first scenario is lower than 80 hours (or working hours per one week) we tried to process the same amount of products but without using two machines: one milling and one grinding machine. As can be seen from Fig. 10, fewer machines were used with the same quantities of products (Fig. 8.) The same GA parameters were uses as in the first – Fig. 6.

Machine ($M_P = 9$)	case 1.	case 2.
Turing machine	2	2
Induction hardening machine	1	1
Milling machine	**2**	**1**
Grinder	**3**	**2**
Drilling machine	1	1

Figure 10 Case 2 – Machines

The best achieved result was 77,152 hours and that is shorter than weekly production hours. That means that we can process the same amount of products with fewer machines and as a consequence lower production costs. Machine utilization for this scenario can be seen in Fig. 11.

Operation	Code	Machine	Utilization
Turning	2	Turing machine 1	87,040 %
	3	Turing machine 2	73,281 %
Induction hardening	4	Induction hardening machine	37,401 %
Milling	5	Milling machine 1	92,311 %
	6	Milling machine 2	-
Grinding	7	Grinder 1	96,653 %
	8	Grinder 2	95,344 %
	9	Grinder 3	-
Drilling	10	Drilling machine	20,959 %

Figure 11 Case 2 – Utilization

5 Conclusion

In this paper we proposed the genetic algorithm for solving the real life job shop scheduling problems. Two main operations of the genetic algorithm were modified in comparison with traditional GA: selection and crossover operators. Obtained results showed that the proposed algorithm is capable to find out a satisfactory solution in a relatively short time span, thereby it is reliable for using.

We demonstrated the viability of the proposed method for real-world problems, suggesting its readiness for use in industry and results obtained from this simulation will help managers to evaluate the performance of the system by knowing machines utilization and other resources, average waiting time of jobs, and average idle time for each machine.

In future works the proposed genetic algorithm will be modified to work in highly parallel computer environment and tested on more complex and other types of problems.

6 References

[1] Holland, J. H. Genetic algorithm. // Scientific American. 266, (1992), pp. 44-50. https://doi.org/10.1038/scientificamerican0792-66

[2] Davis, L. Job shop scheduling with genetic algorithms. // Proceedings of the first international conference on genetic algorithms. 5, (1985), pp. 136-140.

[3] Haider, A.; Mirza, J. An implementation of lean scheduling in a job shop environment. // Advances in Production Engineering & Management. 10, 1(2015), pp. 5-17. https://doi.org/10.14743/apem2015.1.188

[4] Nidhiry, N. M.; Saravanan, R. Scheduling optimization of a flexible manufacturing system using a modified NSGA-II algorithm. // Advances in Production Engineering & Management. 9, 3(2014), pp. 139-151. https://doi.org/10.14743/apem2014.3.183

[5] Yinan, Q.; Tang, M.; Zhang, M. Mass customization in flat organization: The mediating role of supply chain planning and corporation coordination. // Journal of Applied Research and Technology. 12, 2(2014), pp. 171-181. https://doi.org/10.1016/S1665-6423(14)72333-8
[6] Aydemir, E; Koruca, H. I. A new production scheduling module using priority-rule based genetic algorithm. // International journal of simulation modelling. 14, 3(2015), pp. 450-462. https://doi.org/10.2507/IJSIMM14(3)7.299
[7] Chen, Y. X. Integrated optimization model for production planning and scheduling with logistics constraints. // International journal of simulation modelling. 15, 4(2016), pp. 711-720. https://doi.org/10.2507/IJSIMM15(4)CO16
[8] Gen, Mitsuo; Cheng, Runwei. Genetic Algorithms and Engineering Design, ISBN: 978-0-471-12741-3, Wiley, February 1997.
[9] Goldberg, D. E. Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company Co., Inc. Boston, MA, USA, 1989, ISBN:0201157675
[10] Simon, D. Evolutionary Optimization Algorithms, ISBN: 978-0-470-93741-9, Wiley, May 2013
[11] Holland, J. H. Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, (1975).

Authors' addresses

Gordan Janes
Center for advanced computing and modelling
University of Rijeka
Radmile Matejčić 2, HR-51000 Rijeka, Croatia
E-mail: gordan.janes@cnrm.uniri.hr

Full. Prof. Mladen Perinic, Ph.D.
Department of Industrial Engineering and Management
Faculty of Engineering, University of Rijeka
Vukovarska 58, HR-51000 Rijeka, Croatia
E-mail: mladen.perinic@riteh.hr

Assoc. Prof. Zoran Jurkovic, Ph.D.
Department of Industrial Engineering and Management
Faculty of Engineering, University of Rijeka
Vukovarska 58, HR-51000 Rijeka, Croatia
E-mail: zoran.jurkovic@riteh.hr