IMPROVING JOB-SHOP PRODUCTION SCHEDULE WITH USE OF GENETIC ALGORITHM IN SMALL BATCH AND MULTI ASSORTMENT PRODUCTION COMPANY

Stanisław GACEK

ANGA Uszczelnienia Mechaniczne sp. z o. o. Kozy, Poland, EU,
sgacek@anga.com.pl

Abstract

The paper presents a genetic algorithm being used for production scheduling in a real-life small batch and multi assortment production company in Poland. The algorithm has been implemented in an already functioning scheduling module of an ERP system. The use of the algorithm generated better quality production schedules and significantly limited staff engagement in scheduling preparation and its control. The details of the algorithm, production system, the range of improvement, conditions which allowed the improvement as well as situation before it are shown in the article. These include planning procedures used before and after the improvement, rationale behind the genetic algorithm development and problems in its every-day use.

Keywords: Scheduling, genetic algorithm, job-shop, small batch, production

1. INTRODUCTION

The aim of the paper is to publish the genetic algorithm details and share experiences gathered during its use. The paper is a result of an improvement made in scheduling in a small batch and multi assortment production company located in Poland - ANGA Uszczelnienia Mechaniczne sp. z o. o. It is a medium company with a job-shop workshop, that allows machining of wide high-tech product portfolio, both in terms of product type and their size. The average batch is 5.2 items and there are usually 2÷3 set-ups a workstation a shift. In the production department of interest, there is approx. 43 workstations and 3,958 operations to plan. In 2010, scheduling method switched from manual to computer-aided thanks to introduction of a new ERP system which included a scheduling module. The ERP and the above mentioned module were developed in tight collaboration between ANGA's staff and a software company KLL.

The computer scheduling method was based on simulation. The scheduling module included a deterministic production department model. Sole scheduling relied on heuristic algorithms, due-dates, constraints within the production system, data accumulated and assumed in ERP and orders' priorities. The scheduling involved two consecutive phases: initial planning forwards and detailed planning backwards in which orders' priorities were paramount variables. The smaller priority value the higher order's importance in the scheduling module. The software allowed to obtain a feasible production plan. The scheduling solution generated with the module is never illegal in terms of violating the model constraints, e.g. operations sequence. More details on the scheduling module can be found in [1] and [2]. Afterwards, the plan was reviewed by a production planner, who modified it to minimise obvious mistakes and resulting delays' risk. An official production schedule has been released each Monday morning.

The computer scheduling method was initially a significant improvement in comparison to the previous one, but in 2017 the schedules' quality expectations were higher than initially, the production planner modifications were too time-consuming, and production staff started to lose their confidence in the method. As a result, the author and Mr Rafał Laszczak (KLL) developed a genetic algorithm which was successfully built-in into the scheduling module of ERP. The algorithm use allowed to achieve a substantial improvement in scheduling within the company.

Applicability of genetic algorithms is extensive, and includes among others scheduling, jobs grouping, design and lay-out problems [3]. Job-shop scheduling problems belong to the NP-hard class and genetic algorithms may be used to address these problems. The problem is one of the best known of the difficult combinatorial optimisation problems [4]. The problem is an area of interest for many researchers and many new approaches using genetic algorithms in determining its solution can be found, e.g. [5-8].

The genetic algorithm is a search and optimisation technique which is based on natural selection and genetics. The technique allows an evolution of many individuals of a population. The evolution is governed by specified selection rules to a state that minimises (or maximises) the "fitness" function [9]. Genetic algorithms GA have - in general - five basic components [10]:

- an evaluation function which rate solutions on the basis of their fitness,
- a genetic representation of solutions to the problem,
- a way to create an initial population of solutions,
- genetic operators that change the genetic composition of children (offspring) during reproduction,
- genetic algorithm parameters' values.

The GA starts by defining a genetic representation of a solution - a chromosome. The chromosome is an array of variables to be optimised. The variables can be represented as binary, continuous or discrete values. Each chromosome has its corresponding fitness function value [9]. In the job-shop scheduling problem the fitness function generally represents the production time and GA is used to minimise it - to evolve an appropriate permutation.

The search for the solution in the search space involves exploration and exploitation and is conducted by genetic operators: crossover and mutation. The operators can work in many ways depending on the specific problem and researcher's choice. The operators - in combination with solution selection based on its fitness function value evaluation - are employed to generate offspring [4]. In each generation worst chromosomes are eliminated. Decision how many chromosomes to keep is somewhat arbitrary. For crossover operator at least two chromosomes must be chosen to create at least one offspring. There are many ways one can choose the chromosomes, and one of them is a roulette wheel in which each chromosomes has a probability of being selected. The probability P_n can be set basing on the number of chromosomes to keep N_{keep} and the rank n of a chromosome in the fitness function ranking [9]:

$$P_n = \frac{N_{keep}-n+1}{\sum_{n=1}^{N_{keep}} n} \quad (1)$$

In mutation, a chosen chromosome is modified in a defined and stochastic way. A ratio of chromosomes to mutate, scope of mutation, and the ratio/scope change in generations may all be set to match the problem specifics. The change of the mutation rate and its scope as the algorithm run are called the parameter adaptation. The parameter adaptation is used to increase an algorithm efficiency [4,9].

Population size and a way to create an initial population impact a GA efficiency. Both are problem specific, but traditionally large populations have been used to explore complicated cost surfaces. The initial population should ensure an appropriate degree of diversity in chromosome values, therefore many approaches include some kind of randomisation of initial chromosome values. All GA parameters are interrelated in terms of an algorithm efficiency [9].

2. IMPELEMENTED GENETIC ALGORITHM DESCRIPTION

The developed GA has been implemented into an already working production module of the ERP system described in the introduction. The code was written in PHP. The algorithm is initiated early in the Saturday and runs - with breaks due to some other server tasks - until Monday early morning. Then the evolved schedule is set as an official production schedule.

The use of algorithm is possible due to free computational power on one of the company servers. The server is IBM 7915G3G M4 with 2 processors Intel Xeon E5-2650v2 2.6 GHz 64GB RAM 8 cores/processor. The algorithm fully engages one of the processors for 35 hours a weekend.

Depending on production orders and the shop constraints it is able to evolve an official production schedule in 88÷160 generations. It takes ~10 min 30 s a generation. **Figure 1** presents the general overview of the algorithm.

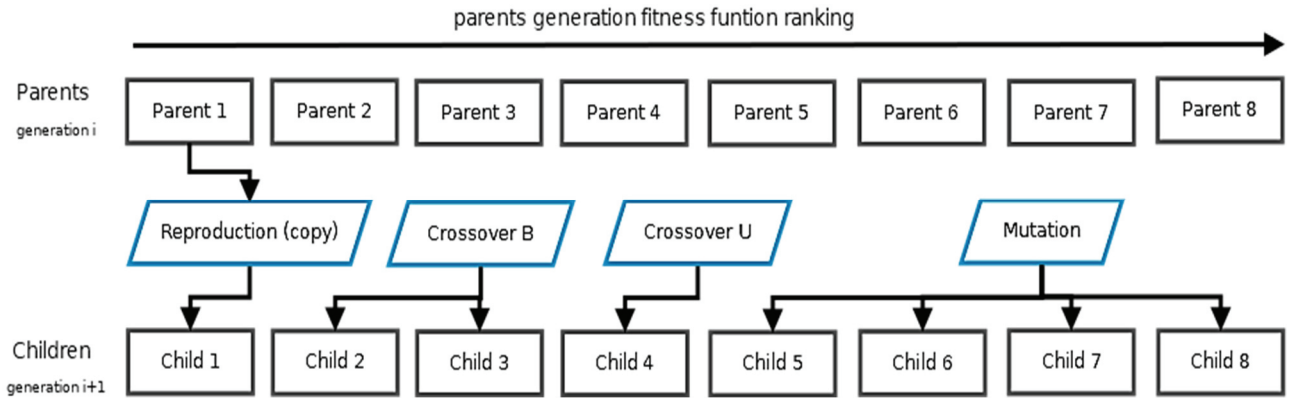

Figure 1 Implemented GA overview

2.1. Fitness function

The fitness function *KOH* - which is minimised as the algorithm run - has been formulated in such a way that it incorporates delays L_i and wages W_i of all n production orders in the job-shop. A production order usually consists of many consecutive operations.

$$KOH = \sum_{i=1}^{n} L_i \cdot W_i \tag{2}$$

The wages values depend on an absolute difference between a production order official due-date and a schedule preparation date. The wages values depending on the difference are shown in **Table 1**.

Table 1 Fitness function wage values depending on a difference between a production order official due-date and a schedule preparation date.

difference between a production order official due-date and a schedule preparation date [days]	**wage**
≤ 7	5
from 8 to 14	3
from 15 to 21	1.5
from 22 to 28	1
≥ 29	0.5

The rationale behind the wage values decrease for less urgent production orders lies in a fact, that orders uncertainty increases with time, the schedule is being issued weekly and that average lead time in the job-shop is approx. 14 days. In addition, the most urgent production orders are usually those already delayed. With wages presented in **Table 1**, the delayed production orders - and the most urgent and not delayed yet orders - get the highest priority in the scheduling process with the algorithm.

2.2. Initial population

Initial population of 15 chromosomes starts with a single source: a schedule described in the introduction - the old method schedule generated using heuristic algorithms, in which priorities are pre-set for each production order. The priorities values are integers ranging from 1 to 9999. This is the first chromosome. To ensure the algorithm has an appropriate search space - priorities are not too close to their boundaries 1, 9999 - an average priority is calculated. If it does not fit between 4000 and 6000, all priorities in the source of the initial population are being increased by 5000 minus the average priority. The effect of the increase for the source is shown in **Figure 2**.

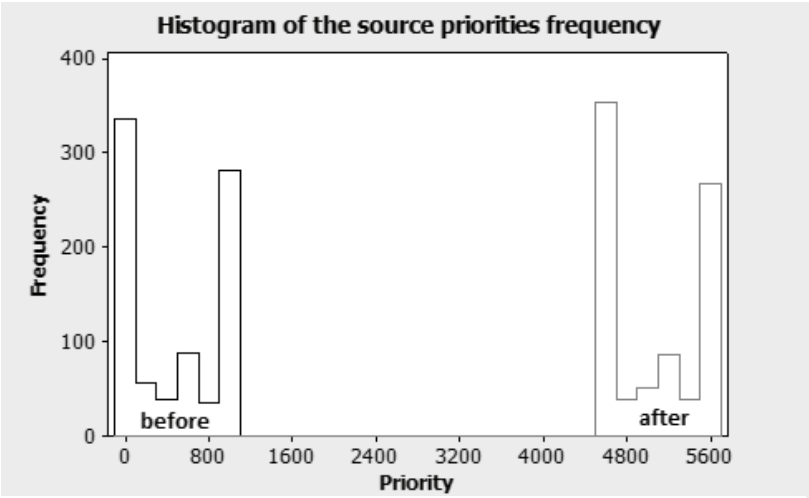

Figure 2 Histogram of the source priorities frequency before and after the increase

Next 5 chromosomes are results of the first one modifications in such a way, that all production orders with due-date in next 14 days have their priorities modified ± 70 with uniform distribution. The other production orders in these 5 chromosomes have their priorities modified ± 300 with uniform distribution.

Next 9 chromosomes are also a result of the source modification. All production orders have their priorities randomised ± 200 with uniform distribution.

2.3. Selection

Starting from the initial population, every chromosome is an input to the job-shop workshop model embedded in the company ERP system. The model allows to determine the chromosomes corresponding production schedule and their fitness function value - KOH. The fitness function values are used to create a chromosome ranking. The solution with the smallest KOH value achieves the highest rank. In every generation 3÷6 best solutions are used to generate an offspring. The number of solutions to be used depends on the selection rules and genetic operators. Except for the initial population, there are 8 children in each generation.

The developed genetic algorithm incorporates both crossover and mutation operators. The chromosome selection for both operators rely on the roulette wheel, where probabilities of being selected are defined by the equation (1). The probabilities decrease with the rank - from 40 % for the best solution in a parent generation to 10 % for the fourth best solution in the generation.

2.4. Crossover

Two crossover operators have been implemented in the GA. Both rely on parents selection presented in section 2.3. The selected parent or parents are excluded from the ranking determining the choice of next particular parent or parents for a particular child. However, those parents are not excluded from the ranking when it comes to another child.

Crossover B is a rounded arithmetic mean of two selected parents. Crossover U is a rounded arithmetic mean of three selected parents. The mean is calculated for alleles (production order priorities) on the same position on a chromosome (the mean of the same production order in parent chromosomes).

2.5. Mutation

The mutation starts with a parent selection as presented in section 2.3. The percent of production orders to mutate *m* depends on a generation number n_{gen} as shown in relation (3) - e.g. 6.5 % in the 2nd generation and 3.05 % in the 25th generation. This parameter adaptation proved to generate better results in comparison to a fixed parameter in the GA.

$$m = \begin{cases} -0{,}15 \cdot n_{gen} + 20{,}3\ [\%], & n_{gen} < 134 \\ 0{,}2\ \%\ , & n_{gen} \geq 134 \end{cases} \quad (3)$$

Determination of the particular priorities to be mutated in a chromosome is based on a lottery. Production order probability P_k of being drawn depends on a difference between the production order official due-date and the due-date resulting from a schedule corresponding with the chromosome, and the order's wage. The probability P_k increases both with lateness and earliness, as well as with production order *k* urgency. It is shown in equation (4) and **Table 2**.

$$P_k = \frac{O_k \cdot G_k}{\sum_{i=1}^{n} O_i \cdot G_i} \quad (4)$$

where:

O - difference between a production order official due-date and its due-date resulting from the chromosome corresponding schedule

G - wages as defines in **Table 2**

Table 2 Mutation wages G depending on a production order urgency

difference between a production order official due-date and a schedule preparation date [days]	wage for production orders scheduled for too late	wage for production orders scheduled for too early
≤ 7	20	1
from 8 to 14	10	2
from 15 to 21	5	5
from 22 to 28	2	10
≥ 29	1	20

The determination of the production orders to mutate is followed by the priorities modifications. Priorities of orders scheduled for too late are decreased, and priorities of orders scheduled for too early are increased. It is designed this way due to the fact, that in the scheduling module the smaller priority value, the higher order's importance. The modification depends on the $O_k \cdot G_k$ product, as presented in **Table 3**. The probability of being drawn for an order which is scheduled on-time is null.

A one criteria scheduling optimisation does not take into account some other criteria that are always present in reality. E.g. it is unacceptable within the company to delay one production order for a two months only to minimise the fitness function by 1. The presented mutation mechanism prevents such a situation. It works as a controller which does not allow a production order to be greatly delayed in return for a miniscule fitness function value decreased.

Table 3 Priorities modifications in mutation depending on $O_k \cdot G_k$ product

$O_k \cdot G_k$ product	Priority modification range (uniform distribution)
≤ 50	$[1; 100]$
from 51 to 100	$[1; 200]$
from 101 to 200	$[1; 400]$
from 201 to 500	$[1; 600]$
≥ 501	$[1; 800]$

2 RESULTS

The algorithm has been in use since 2017. The version presented in the article version is a result of extensive testing of different versions of genetic algorithms. Among them, the presented GA proved to provide the highest efficiency in terms of fitness function value minimisation in the first 80 generations.

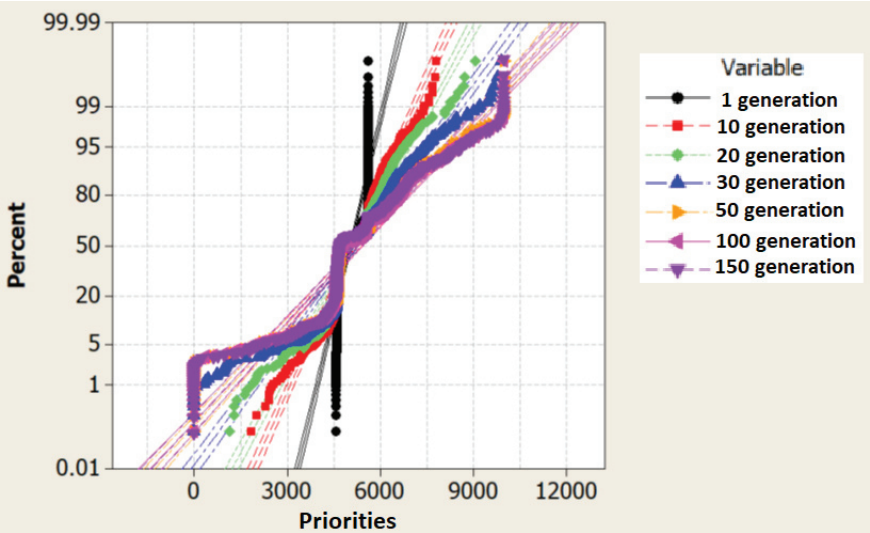

Figure 3 Change in priorities cumulative distribution as the GA runs

The use of the presented GA in practice significantly improved the official schedules quality and reduced the effort needed to produce an acceptable schedule before the GA implementation. As the algorithm runs, priorities are modified to minimise the fitness function. An exemplar change in the priorities cumulative distribution is presented in **Figure 3**.

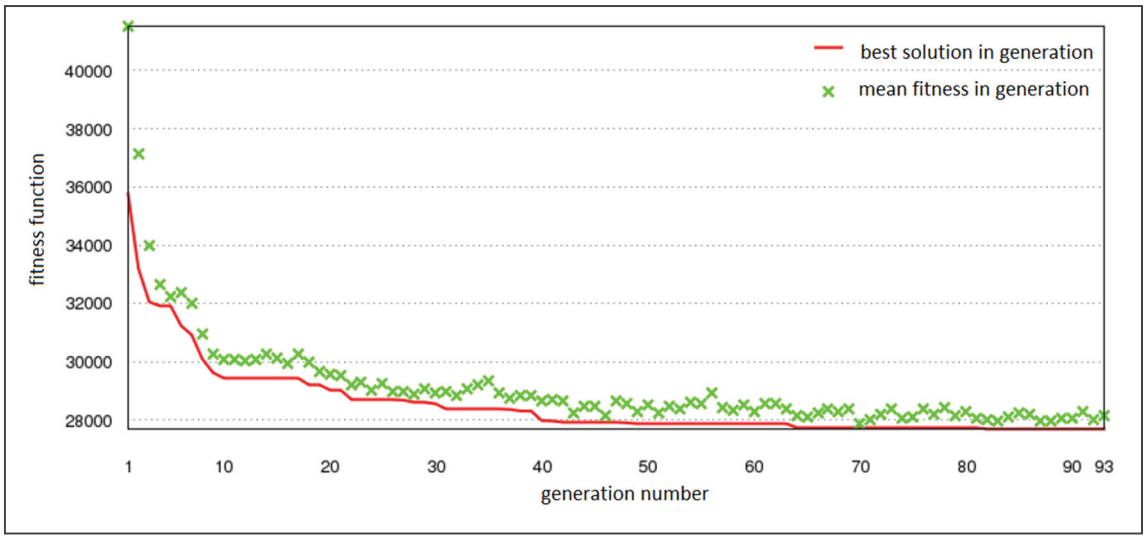

Figure 4 Fitness function change in generations

The **Figure 3** shows the priorities distribution change. The more generations the higher spread of the priorities. The spread increases until some priorities reach the boundaries - either 1 or 9999. This is a result of the mutation mechanism, which in general decreases delayed production orders priorities, and increases priorities of production orders scheduled for too early.

A typical progress in terms of fitness function minimisation with the algorithm is shown in **Figure 4**. It is visible the most significant improvement happens in the first 10 generations and it is mainly because of the mutation operator which is responsible for most of the breakthroughs.

The fitness function formula proved to be a good indicator of production schedule quality within the company. The scope of improvement and differences between old and current procedures of generating production schedules is shown in **Figure 5**. A measure of improvement is presented as an average ratio. The ratio is

calculated as a fitness function value divided by a benchmark's fitness function value. The benchmark is a schedule with random priorities. The average ratio has been calculated for 51 production schedules.

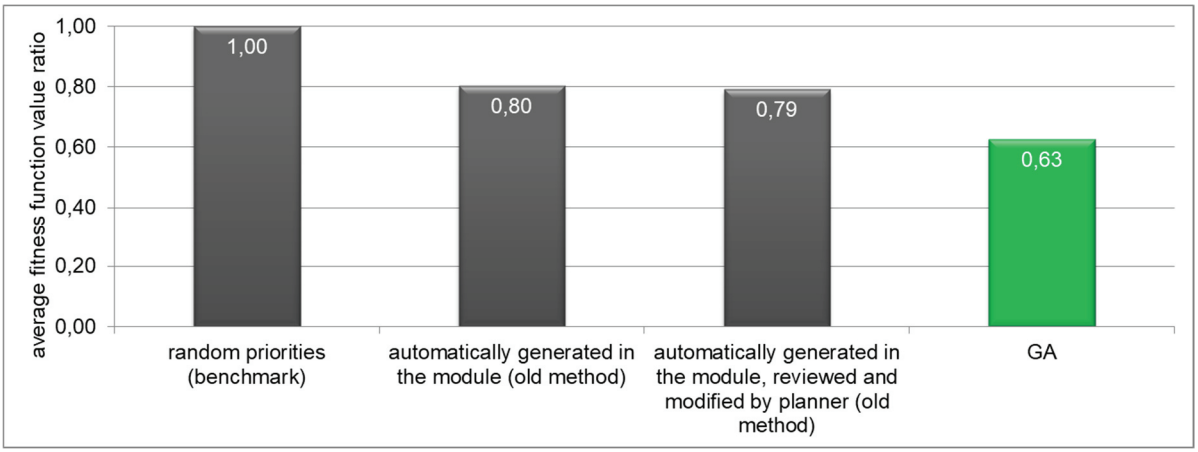

Figure 5 Scope of improvement and differences between current and previous scheduling procedures

Results in **Figure 5** show the scope of improvement after implementing the GA. The scheduling with the genetic algorithm use visibly generates best results within the company. One may wonder why random priorities are not worse in terms of the fitness function value. This is the result of the production module in the company ERP, which generate only legal solutions even if priorities are random and also the result of the fact that not all materials needed for production are already in the shop. Some operations are in progress and even with random priorities they are often in the right queues for workstations.

3 CONCLUSION AND DISCUSSION

The use of the presented GA was a significant improvement to the previous scheduling procedures. The algorithm was developed within the company and successfully built-in into the scheduling module of an ERP system by a software company. The algorithm use allowed to shorten the schedule preparation, greatly reduced production planners' engagement in the scheduling and provided better quality production schedules. The quality is measured by the fitness function value and partially by subjective opinions of the company production department staff - employees responsible for the schedule preparation and also employees responsible for the schedule execution.

Although the genetic algorithm implementation has been a success, it did not eliminate all problems related to internal production planning. It is difficult to completely stick to even best schedule, and not because of scheduling mistakes but e.g. due to unplanned workers absence or material delivery delays. Additionally, employees do not always understand the concept of optimisation and therefore may not trust the improved scheduling method. There is also a matter of a responsibility dilution after shifting scheduling from an employee to a computer. It is worth to add that the scheduling output depends on input (data) and the job-shop model. In practice, sometimes data is not up-to-date or include mistakes. The model is always a simplification of a reality and this simplification may result in scheduling mistakes. One may also wonder whether 35 hours of computation on a company server is a satisfactory period of time, even though the server' computational power had not been extensively used before the GA implementation.

The algorithm presented in the paper can be used in the original or modified form in other medium size, small batch and multi assortment production companies. Particularly, the developed mutation operator proved to be very effective and might be useful in other GAs as well. It works as a controller which does not allow a production order to be greatly delayed in return for a miniscule fitness function value minimisation.

The algorithm is going to be modified due to the natural changes within the company in time - especially constant changes in the company production methods, means and organisation. Consequently, more research, the algorithm development and its adjustment to the reality will take place in the future.

ACKNOWLEDGEMENTS

I would like to thank the production department staff (ANGA) and Mr Rafał Laszczak (KLL) for their effort and support in the algorithm implementation.

REFERENCES

[1] GACEK, S. *Komputerowe metody planowania i sterowania na przykładzie firmy "ANGA" Uszczelnienia Mechaniczne sp. z o. o.,* Master's thesis, Cracow: AGH Management Department, 2010. p. 133.

[2] GACEK, S. CNC machine group scheduling in a multitasking system. In *CLC 2012: Carpathian Logistics Congress*. Jesenik: Tanger, 2012

[3] KNOSALA, R. *Zastosowania metod sztucznej inteligencji w inżynierii produkcji*. Warsaw: WNT, 2002, p. 455.

[4] GEN, M., CHENG, R. *Genetic Algorithms & Engineering Optimization*, New York: Wiley-Interscience, 2000. p. 495.

[5] MOU, J., GAO, L., LI, X., LU, C., HU, H. Optimisation of the reverse scheduling problem by a modified genetic algorithm. In: *International Journal of Production Research.* 2015. 53, 23, pp. 6980-6993.

[6] ISHIKAWA, S., KUBOTA, R., HORIO, K. Effective hierarchical optimization by a hierarchical multi-space competitive genetical gorithm for the flexible job-shop scheduling problem. In: *Expert Systems with Applications.* 2015. 42, pp. 9434-9440.

[7] MAY, G., STAHL, B., TAISCH, M., PRABHU, V. Multi-objective genetic algorithm for energy-efficient job shop scheduling. In: *International Journal of Production Research.* 2015. 53, 23, pp. 7071-7089.

[8] ZHANG, W., WEN, J., B., ZHU Y., C., HU, Y. Multi-Objective Scheduling Simulation of Flexible Job-Shop Based on Multi-Population Genetic Algorithm. In: *International Journal of Simulation Modelling.* 2017. 16, 2, pp. 313-321.

[9] HAUPT, R., L., HAUPT, S., E. *Practical genetic algorithms*. Hoboken, New Jersey: Wiley-Interscience, 2004. p 253.

[10] MICHALEWICZ, Z. *Genetic Algorithm + Data Structure = Evolution Programs*. 3rd edition, New York: Springer-Verlag, 1996. p. 387.