www.czasopisma.pan.pl PAN www.journals.pan.pl

ARCHIVES of FOUNDRY ENGINEERING

ISSN (2299-2944)
Volume 2022
Issue 1/2022

5 – 12

10.24425/afe.2022.140210

1/1

Published quarterly as the organ of the Foundry Commission of the Polish Academy of Sciences

Comparison of the Classical Algorithm with the Training Algorithm in Scheduling Problem ADI Production

D. Wilk-Kołodziejczyk [a, b, *], K. Chrzan [b], K. Jaśkowiec [b], Z. Pirowski [b], R. Żuczek [b], A. Bitka [b], D. Machulec [a]

[a] AGH University of Science and Technology, Kraków, Poland
[b] Łukasiewicz Research Network - Kraków Institute of Technology, Poland
* Corresponding author. E-mail address: wilk.kolodziejczyk@gmail.com

Received 18.08.2021; accepted in revised form 04.10.2021; available online 20.01.2022

Abstract

A classical algorithm Tabu Search was compared with Q Learning (named learning) with regards to the scheduling problems in the Austempered Ductile Iron (ADI) manufacturing process. The first part comprised of a review of the literature concerning scheduling problems, machine learning and the ADI manufacturing process. Based on this, a simplified scheme of ADI production line was created, which a scheduling problem was described for. Moreover, a classic and training algorithm that is best suited to solve this scheduling problem was selected. In the second part, was made an implementation of chosen algorithms in Python programming language and the results were discussed. The most optimal algorithm to solve this problem was identified. In the end, all tests and their results for this project were presented.

Keywords: Production scheduling, Austempered Ductile Iron, Iron castings, Foundry industry, Training algorithm, Tabu search

1. Introduction

In the production of castings, the problem of scheduling tasks concerns both the manufacturing process of a given product and the queuing of accepted orders. This allows to increase efficiency or optimize the production process itself. The development of artificial intelligence is also used in the production of steel. It allows you to create an intelligent knowledge system that allows for task scheduling and production planning in an optimal way. Integrated planning of steel production is a problem of high complexity, therefore intelligent algorithms are used, which will allow you to obtain more accurate results than the classic algorithms, in a shorter time [1]. The tasks performed by training algorithms are often used for optimization. Their use requires less calculation power than the use of classical algorithms, especially for problems with a large number of possible solutions [2]. Scheduling is the process of assigning limited resources to tasks in order to obtain the optimal value due to the selected criterion [3]. In production systems, resources are understood as objects such as workstations, machines, means of transport, as well as fuel, energy and people. However, the tasks are the processes of processing, transporting or producing a given product [3] [4]. In order to present and systematize the scheduling problems, the three-field notation $\alpha \mid \beta \mid \gamma$ is used, where α is the task processing system, β is the task characteristics, γ is optimization criterion [3] [4] [5] [6]. The symbol α can be represented by equation 1, where $\alpha 1$ describes the way in which tasks are performed by the system, and the symbol α_2 describes the amount of resources in the system, in a particular case it is the number of machines or stands. The α_2 symbol can take

the empty character "o", which means that the number of resources is part of the problem and is not predetermined [3].

$$\alpha = \alpha_1\alpha_2 \quad (1)$$

The way in which tasks are performed by the system can be divided into two main groups: systems with parallel machines, where each task can be performed on one of the machines, and systems with dedicated machines, where each task must go through by specific positions. The second group can be further divided into the following subgroups: flow shop systems, where each task must pass through all machines in the same order, open systems, where each task must be performed at all workstations, but the order is arbitrary and general systems (called job shop), where each job position and the order is arbitrary. Table 1 shows examples of the values that the symbol α1 can take [3].

2. Learning algorithms in scheduling tasks

2.1. Description of the problem

Learning methods are also used in scheduling problems [7] [8] [9] [10]. The most commonly used solutions are neural networks. An example of this is the Hopfield network. This is a self-associative non-linear searching network that aims to minimize the function representing activation in the network units {system energy}. The operation of this network can be compared to an electrical circuit with an operational amplifier [11]. The simplest form of a Hopfield network is a single-layer feedback network, an example of which is shown in Figure 1. In this figure, each neuron n_i is excited by external signals such as threes holds or signals entering a_{in}, additionally, it is excited by internal signals in the form of the feedback of the output signal $a_{out,j}$ multiplied by the respective weights in ij. The output signals are connected to all other neurons except the own [2]. With other learning algorithms, decision trees are used for solving problems. In the considered example, a decision tree was used, for real-time planning of the flow system problem. In this case, the decision tree selects the scheduling rule from a previously prepared list. The applied solution is presented graphically in Figure 2 [12].

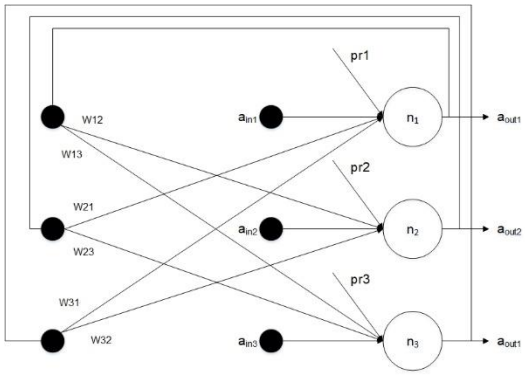


Fig. 1 Hopfield network

Three main elements can be distinguished in the applied solution: a real-time controller, a scheduler and a decision tree. The real state controller is receiving data from the flow system and sends jobs to be executed according to the rule issued by the scheduler. Additionally, it monitors the state of the system. The scheduler decides when a new scheduling rule is selected. It then fills in and releases the rules selected by the decision tree. The decision tree selects a new scheduling rule based on the state of the system. In addition, the flow system itself, the learning data, the decision tree and the system state, which stores current information about the system, are also distinguished. It is updated with every change.

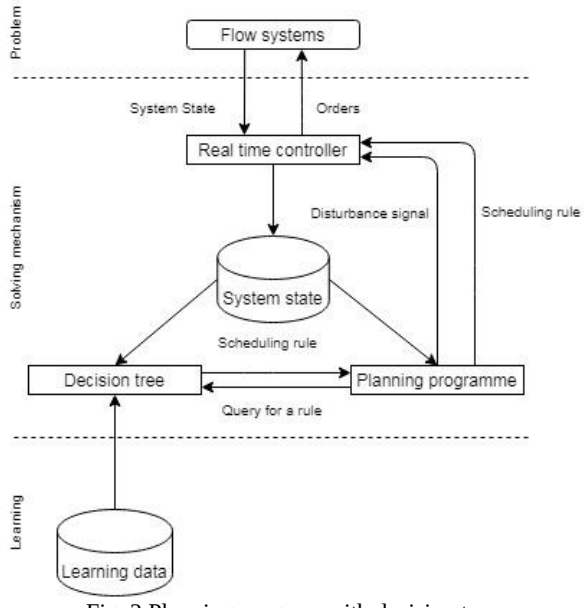


Fig. 2 Planning program with decision tree

The ADI cast iron manufacturing process can be represented as scheduling problem $F_n \mid 6 \mid E_{min}$. It is a flow system with 6 stations and n tasks, the optimization criterion of which the purpose is to minimize the energy consumption of the process. The first station (M1) corresponds to the processes taking place in the furnace, the second station (M2) of the molding and casting line, the third station (M3) is optional technical processing depending on the order, the fourth station (M4) corresponds to the annealing process, station five (M5) to the cooling process and station six (M6) to the final technical treatment. In order to simplify the problem, it has been assumed that the machining is fully automated. The total energy consumption during the production of E_c was assumed as the optimization criterion. In the case under consideration, not only the duration of the process itself is of great importance for the costs incurred, but also the change of temperature parameters between individual tasks. For stations M2, M3 and M6, the power of devices and the duration of the process will be important factors. The power of these devices in turn is P2 equal to 350 kW, P3 equal to 185 kW, P6 equal to 140 kW. For stations M1, M4 and M5, an additional factor influencing energy consumption will be the influence of temperature on the consumed power. In the paper, it was assumed that in the considered temperature ranges this relationship will be linear as shown in equation 2.

$$P(\theta) = c\,\theta \quad (2)$$

Where P (θ) - power is dependent on temperature, c- dependence coefficient, θ - process temperature expressed in Kelvin. The dependence coefficient is: 1.119 for the M1 station kW / K, for M4 0.1063 kW / K, for M5 0.2972 kW / K. Contrary to previous positions, the time from the beginning of the first tp1 task to the end of the last tkn task will be important here, not just the duration. For the remaining machines, it was assumed that they only consume energy while working. The optimization criterion for the entire process is presented in Equation 3, where Ei is the energy w consumed in the i-th station and is described by Equation 4.

$$E_c = \sum_{i=1}^{n} E_i \quad (3)$$

$$E_i = \sum_{j=1}^{m} P_i(\theta_{ij})(t_{p(j+1)} - t_{pj}), i \in \{1,4\};\ \sum_{j=1}^{m} P_i t_j \quad (4)$$

Where Pi is the power consumed by the i-th station, θij is the temperature at the i-th station during the j-th task, and ie is the duration of the j-th task.
The number of tasks depends on the number of ADI cast-iron variants with specific properties, listed in Table 1. It was assumed that one task allows for the production of 20 elements with a total weight of 3 tons. The work assumes the implementation of a maximum of 3 orders from each cast iron variant. Table 2 summarizes the required work parameters for each position for each task. Machining on an M3 machine depends on the requirements of the order and can be applied to any task. This work assumes that only the first two orders of each variant require this processing.

Table 1.
Cast iron variants ADI

Material index	Tensile strength, R_m, [MPa]	Yield point, $R_{0,2}$, [MPa]	Elongation, A_5, %	Hardness, HB
1	800	500	8	260-320
2	1000	700	5	300-360
3	1200	850	2	340-440
4	1400	1100	1	380-480

Table 2.
Parameters of workplaces for especially tasks

Task design ation	Material designat ion	M1		M2	M3	M4		M5		M6
		t_1, s	θ_1, °C	t_2, s	t_3, s	t_4, s	θ_4, °	t_5, s	θ_5, °	t_6, s
a	1	7200	900	3600	10800	90000	375	6000	375	5400
b		7200	900	3600	8800	90000	375	6000	375	5400
c		7200	900	3600	0	90000	375	6000	375	5400
d	2	7200	900	3000	11800	90000	330	10000	330	6500
e		7200	900	3000	7800	90000	330	10000	330	6500
f		7200	900	3000	0	90000	330	10000	330	6500
g	3	7200	900	2600	8900	10800	270	16000	270	4400
h		7200	900	2600	6000	10800	270	16000	270	4400
i		7200	900	2600	0	10800	270	16000	270	4400
j	4	7200	871	4000	10800	14400	260	17000	260	8800
k		7200	871	4000	6000	14400	260	17000	260	8800
l		7200	871	4000	0	14400	260	17000	260	8800

2.2. Classic algorithm

The search algorithm was selected from among the classic algorithms presented in this paper, due to the typical and self-defined target function. Additionally, this algorithm is used, inter alia, for flow problems in scheduling tasks [14]. In the problem under consideration, the algorithm will schedule tasks on the first machine. Then, in accordance with the ADI cast iron production process, will determine the energy consumed on both individual stations and across the entire process. The process of determining the energy consumption will be determined during each iteration. In order to fully describe the algorithm for this solution, the following should be defined: initial scheduling, traffic, final condition and the length of the prohibition for given traffic [13]. In a given study, due to the lack of information on the optimal method of selecting the initial solution, the algorithm will be performed for two different initial orderings. In each of them, the criterion will be the total energy consumption of a single task, assuming that only that one task is involved in the process. In the first case, the tasks are ranked in ascending order of energy consumption, and in the second case, in descending order. The move will be to bring the selected item to the top of the list and move it the remaining elements to the empty space in the order [13]. The final condition is that the maximum number of iterations, which is the product of the number of tasks and the number of stations, has been exceeded. Two values of the forbidden length were used and compared: 10% of the maximum number of iterations and half of the number of tasks rounded up because in the literature this value is not clearly defined for this problem. The algorithm receives the tasks with the data as input, as shown in Table 2. The results are presented in the form of a sequence of tasks on the first machine, the total energy consumption of this ranking, and a graphical representation of the ranking in the form of a Gantt chart. In addition, it will return data from subsequent iterations to determine which variant of the algorithm parameters gave the best results.4

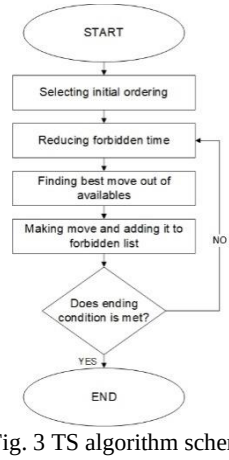


Fig. 3 TS algorithm scheme

The scheme of the algorithm's operation is shown in Figure 3, where the subsequent elements of the algorithm's operation include selecting the initial solution, searching for the best possible motion along with its execution, and checking the final condition. In addition, the diagram shows the elements that allow you to control the list of prohibited moves by adding more moves and reducing

the forbidden time at the beginning of the loop, as well as controlling the improvement in subsequent moves to determine one of the conditions completion of the algorithm [13].

2.3. Learning algorithm

The learning algorithm was chosen due to the architecture properties that allow it to be used for optimization processes, without the need to create an additional database of learning examples. The algorithm checks the successive possibilities of ranking only the given problem and selects the best combination found. In the problem under consideration, the algorithm will change the order of tasks only on the first machine, and then, in accordance with the described process, it will determine energy consumption, the same as in the classic algorithm. The scheme of the algorithm's operation, adapted to the problem of scheduling tasks in the flow system, is shown in Figure 4. The algorithm uses the Q table to store values for each pair of environmental states (si) and actions that can be performed (a). The action to be performed is to select one of the tasks, which means the total number of possible actions is equal to the number of tasks to be performed, in this case, it is 12. The total number of environmental states is given by equation 5, which for the described case gives 4096 states. It follows that the size of the array Q is 4096 x 12. At the beginning of the algorithm, the array is initialized with the values 0 in each cell [14].

$$s = 2^{\alpha} \tag{5}$$

The next steps of the algorithm are cycled over a specified number of iterations. Based on the analysed literature, the number of iterations was determined as the product of the number of tasks and machines, in this case, it is 72 iterations. The first step in this loop is to initialize a randomly sorted set of tasks. In this implementation, it will be the alphabetical order of the task designation. Primary rankings in this algorithm have no effect on the final rankings [14]. The next step is, the next loop and this is performed until all tasks from the set are completed. In this loop, a number between 0 and 1 is first drawn and compared with the 'greedy strategy' coefficient ε to decide whether the model will search arrays. If the value is greater than ε then the action is selected based on the Q table, such an action is called an array search because it is already based on the known actions in a given state and their values in the Q table. If the value is smaller than the action is selected randomly, it is called this is discovery because the model selects actions without checking the values in the Q table for that state. This situation is advantageous when, for example, there are no values other than 0 [14] in the table yet. This solution provides two values of the ε coefficient to choose from. In the first solution, it is a value that is constant and unchanging throughout the duration of the program and amounts to 0.1. In the latter case, the value decreases from 1 by 10% for each iteration. This allows the algorithm to initially discover the values of the Q array by randomizing actions, and the more information the model has about that array, the more likely it will be to choose an action based on the values of the Q array. However, there will always be a low probability of taking a random action. After selecting the action, it is performed, in this case, it consists in assigning a given task to the spot list. After completing the entire loop, the program has full task scheduling for a given iteration. After the action is performed, the values of the predicted future state (s) and the reward (r) determined according to equation 6 are examined. At the end of the loop, table Q is updated according to equation 7 and the task set by deleting the task that has been selected [14].

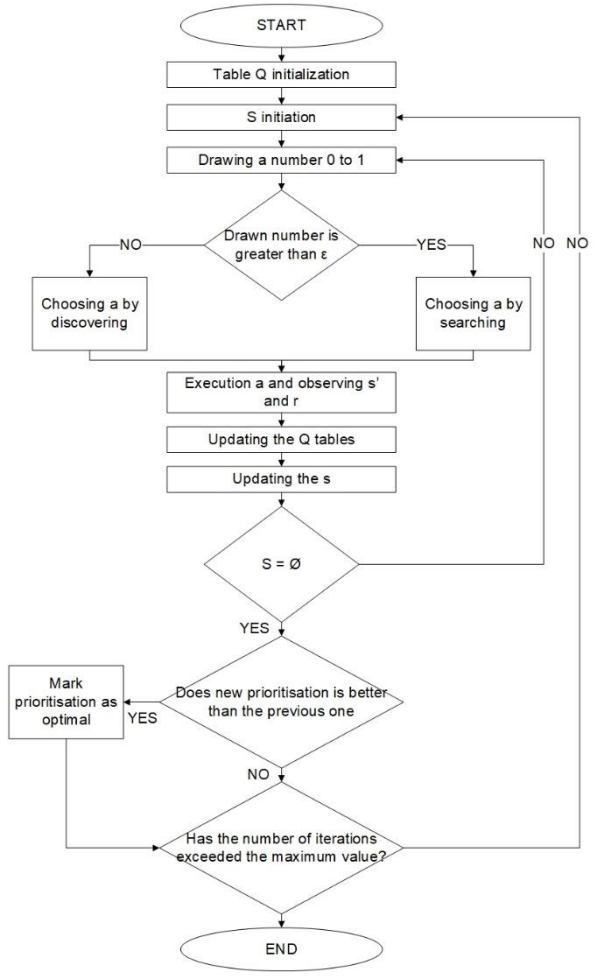


Fig. 4 Learning algorithm scheme

$$r = \frac{1}{f(S_{opt})} \tag{6}$$

$$Q'(s.a) = Q(s,a) + \alpha_{ql}\left(r + \gamma_{ql} maxQ(s',a') - Q(s,a)\right) \tag{7}$$

Where, f (sopt) - the value of the objective function for a list of ordered tasks, Q '(s, a) - the value in the Q table after the update, Q (s, a) - the value of the Q table before the update, α ql - the constant learning coefficient, γql - constant discount factor [14]. The learning coefficient αql can be interpreted as the degree of updating the values in the Q table, for the case considered in this paper its value was assumed to be 0.1. The discount factor γql determines

the significance level of the future, anticipated reward, here adopted at the level of 0.8 [14]. If the set of tasks in this iteration is already empty, it means that each task has been ranked. Then the obtained value of the objective function for this ranking is compared with the currently best ranking, and if the new ranking is better, it is stored as the currently best ranking. After all, iterations are completed, the algorithm returns the best-found solution. Additionally, the program has the ability to print out the solution after each iteration [14]. Works on similar solutions have also been described in other literature books.[15 - 18].

3. Description of achieved results of own research

The results of assigning tasks from the described problem were presented separately for each algorithm, and then their comparison was made. Each algorithm was performed for three data sets. The materials for checking the operation of the algorithms came from item [19].

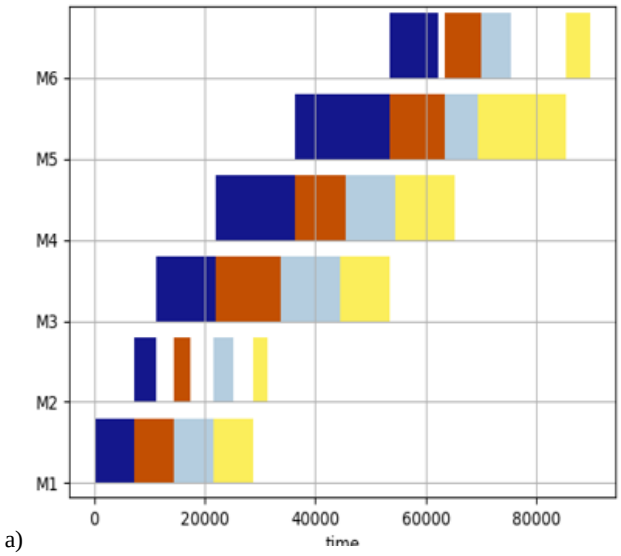
3.1. Classical algorithm results

The classic algorithm allows the selection of three work parameters, two of which may affect the value of the result. These are the length of the prohibition and the primary alignments. The last parameter determines whether all results are saved or only one. Table 3 summarizes the results for all three data sets, performed for all combinations of parameters affecting the result. The first column contains the designation of sets, where 1 is a set with four tasks, 2 is a set with eight tasks, and 3 is a set with twelve tasks. In the second column, the selected prohibition length. Primary alignments are selected in the third column. The fourth column shows the final rankings. In the fifth, the value for this ranking. The last column shows the execution time of the algorithm. Only the versions with the best ranking were selected for these results.
The table shows that different orderings give the same results, which is additionally confirmed by the analysis of the results after each iteration. Results after each iteration were also generated for each variant and each combination of parameters. By analysing the results after each iteration, it can be concluded that the rankings presented in the table are optimal. On the basis of the obtained results, there is no visible influence of the original ranking and the length of the prohibition on the solution. This may result from the characteristics of the considered example, and not the characteristics of the issue itself. For the first two data sets, there is no additional impact on the length of the algorithm's execution. In the option with the prohibition length equal to half the number of tasks, the time is significantly longer than with the 10% number of iterations. You can see the effect of the forbidden length on the algorithm by analysing the results after each iteration. For the bypass length equal to half the number of tasks, the algorithm more often returned the result already achieved earlier than for the bypass length equal to 10% of the number of iterations. Additionally, extreme results were achieved more often.

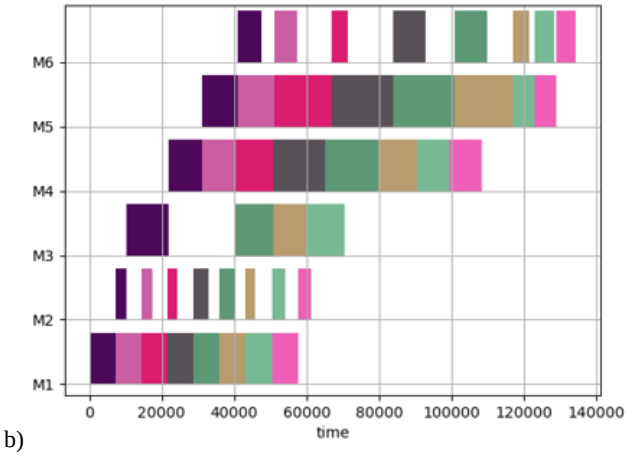
Table 3.
Ranking results for the classical algorithm

Set designation	Length of forbiddance	Primary ordering	Final ordering	Ordering value [MJ]	Time of performance [ms]
1	10 % amount of iteration	From higher energy consumption	j, d, a, g	64387	249
	Half of tasks amount	From higher energy consumption	j, d, a, g	64387	253
	10 % amount of iteration	From lower energy consumption	j, g, a ,d	64387	244
	Half of tasks amount	From lower energy consumption	j, g, a, d	64387	249
2	10 % amount of iteration	From higher energy consumption	d, f, i, l, j, g, a, c	120949	306
	Half of tasks amount	From higher energy consumption	g, i, l, j, d, a, f, c	120949	323
	10 % amount of iteration	From lower energy consumption	l, e, i, f, g, a, d, j	120949	305
	Half of tasks amount	From lower energy consumption	j, l, d, g, c, I, f, a	120949	321
3	10 % amount of iteration	From higher energy consumption	k, g, c,a, b, d, c, f, I, l, j, h	182801	334
	Half of tasks amount	From higher energy consumption	g, j, d, k, a, e, b, h, l, f, I, c	182801	487
	10 % amount of iteration	From lower energy consumption	l, e, i, f, h, b, e, g, a, k, d, j	182801	334
	Half of tasks amount	From lower energy consumption	l, e, i, f, h, b, e, g, a, k, d, j	182801	494

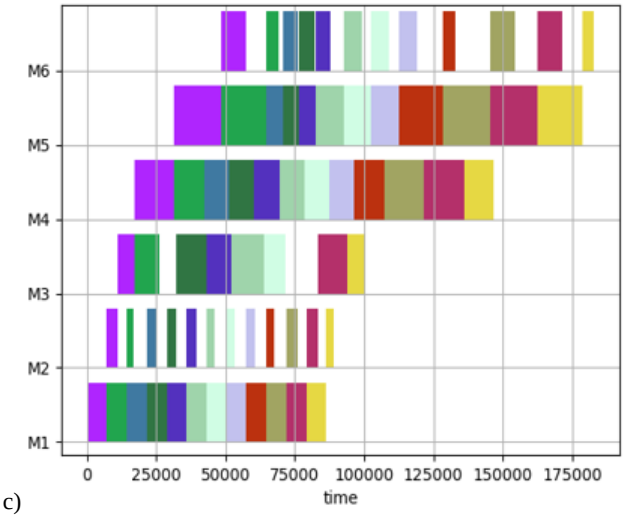
Figure 5 shows the Gantt charts for the optimal ordering. In Figure 5 a) for option 1, at 5 b) for the 2nd option and at 5 c) for the 3rd option. In all examples, the forbidden length was chosen to be 10% of the number of iterations and the primary ordering decreasing with respect to energy consumption. The drawings clearly show the breaks in the work of individual machines and the desire to avoid breaks in the machines that draw power not only during the performance of tasks. Additionally, one can observe in Figures 5 b) and 5c) interruptions resulting from the lack of performance of some tasks on the M3 machine.



a)



b)



c)

Fig. 5. Gantt chart for a) 1st variant, b) 2nd variant, c) 3rd variant

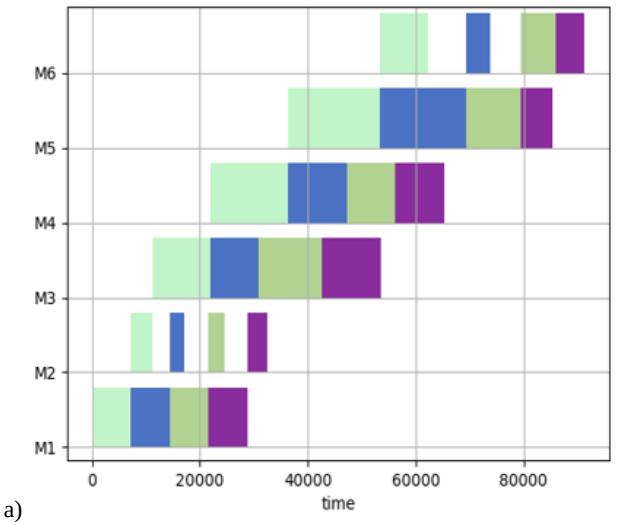
3.2. Learning algorithm results

The learning algorithm requires the selection of only two parameters: the ε coefficient and the result variant. Only the first factors can have an impact on finding the optimal solution. In Table 4, as for the classical algorithm, the results for all three data variants were collected for both options of the ε coefficient. Only the versions with the best ranking were selected for these results. When analyzing the ranking for this algorithm, it is worth highlighting its random nature. The results do not need to repeat when the algorithm is called again. The results after each iteration were also generated for each variant and each option of the ε coefficient. By analyzing these results, it can be concluded that the rankings presented in the table are optimal. No influence of the ε coefficient on the final solution was observed, nor on the algorithm execution time. However, you can see its effect on the operation of the algorithm by analyzing the results after each iteration. For the ε coefficient decreasing by 10% from 1, it can be seen how the algorithm initially selects solutions that are far from optimal, and only from a certain moment selects the optimal ones. For a constant value coefficient, optimal results started to appear earlier, leading to fewer extremely bad results appearing.

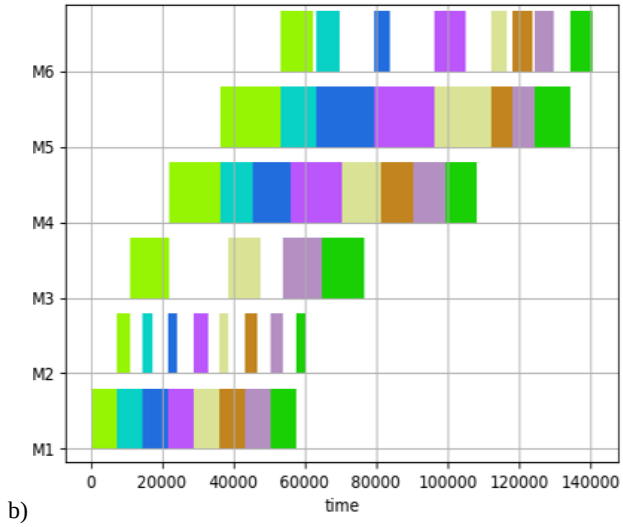
Table 4.
Classification results for the learning algorithm

Set designation	ε factor	Final ordering	Ordering value [MJ]	Time of performance [ms]
1	Constant 0.1	j, a, g, d	64387	276
	Decreasing every iteration by 10%	j, a, g, d	64387	246
2	Constant 0.1	g, l, j, i, c, a, f, d	120949	302
	Decreasing every iteration by 10%	l, f, g, j, i, c, d, a	120949	286
3	Constant 0.1	j, i, f, l, a, c, e, h, g, b, k, d	182801	373
	Decreasing every iteration by 10%	l, h, k, f, e, d, b, c, a, i, j, g	182801	380

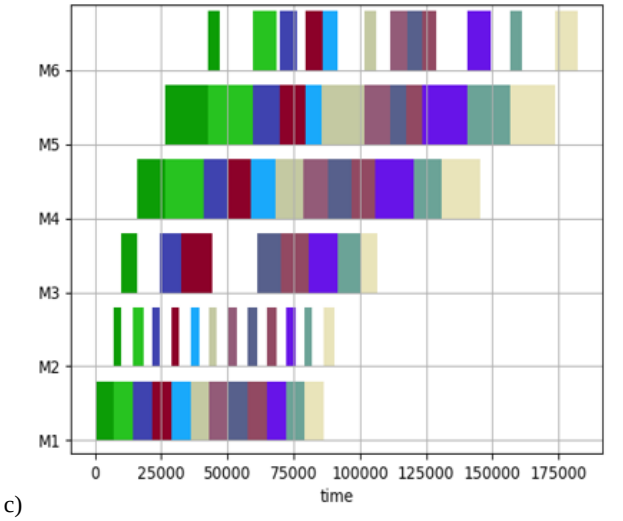
Figure 6 shows the Gantt charts for the optimal ordering. In Figure 6 a) for variant 1, to 6 b) for the 2nd variant and to 6 c) for the 3rd variant. In all examples, a constant value of the ε coefficient equal to 0.1 was chosen. The charts have the same characteristics as the classical algorithm.



a)



b)



c)

Fig. 6 Gantt chart for a) 1st variant, b) 2nd variant, c) 3rd variant

3.3. Comparison of algorithms

The comparison of algorithms was also made for all three data sets, but one set of algorithm operation parameters was selected. For comparison, the following were selected: for the classical algorithm, the length of the sequence equal to 10% of the number of iterations and the primary ordering from the highest energy consumption, and for the learning algorithm, the value of the ε coefficient equal to 0.1. It was decided to set these parameters because they were recognized
for the most advantageous. For the classical algorithm, this parameter value gives a shorter algorithm execution time and greater differentiation during the search. In addition, this value more often allowed to find various optimal solutions. The comparison of the results is presented in Table 5, where the first column is the number of the data variant, in the second column is the name of the algorithm, in the third is the optimal sequence, in the fourth column is the objective function value for this ranking, and in the last column is the algorithm execution time. Algorithms return different orderings, but with similar values. After analysing the results after each iteration, these rankings can be considered optimal. The only difference between the algorithms shown in the table is the execution time for the set with the most tasks. Classic algorithms here return results much faster than the learning algorithm. When additionally analysing the results obtained after each iteration for the described parameters of the algorithms, it can be observed that the learning algorithm returns more differentiated orderings. No line-up repeats itself. This gives the possibility of finding a greater number of optimal orderings. The downside of this algorithm, in this case, is its randomness, which causes it to be returned in an extremely unfavourable case the end result will not be the optimal ranking.

Table 5.
Comparison of the obtained results for the classical and learning algorithms

Designation of the set	Algorithm name	Final ranking	The value of the ranking [MJ]	Execution time [ms]
1	Classic	j, d, a, g	64387	249
	Learning	g, j, a, d	64387	257
2	Classic	d, f, i, l, j, g, a, c	120949	306
	Learning	g, j, l, d, i, f, c, a	120949	317
3	Classic	k, g, c, a, b, d, e, f, i, l, j, h	182801	334
	Learning	g, k, l, d, e, h, c, i, b, a, f, j	182802	436

4. Conclusions

The goals of the work were fully achieved. Two algorithms were implemented and compared: Classic and Learning. An Algorithm has been implemented to solve the scheduling problem for the simplified ADI cast iron manufacturing process. The operation of the algorithms has been tested for various tasks.

For the described problem, the algorithms returned different orderings, but with the same value of the objective function. All the solutions returned were considered optimal. The work of the algorithms for different parameter values was checked and the sets

best suited to this problem were selected. For the learning algorithm, the ε coefficient with a constant value equal to 0.1 more often, during the work of the algorithms, it found the optimal ordering and found different orderings, which reduces the chances of finding the local optimum instead of the global one. In the classical algorithm, for a sequence length equal to 10% of the number of iterations, the algorithm was faster and found more differentiated orderings during operation. The original rankings in this case did not matter. Of these two algorithms, Classic algorithms were found to be better for this problem.

Acknowledgements

The works were financed under the TECHMATSTRATEG 1 project with the acronym INNOBIOLAS entitled "Development of innovative working elements for forestry machines and biomass processing based on high-energy technologies of surface modification of the surface layer of cast elements"; contract no.TECHMATSTRATEG1 / 348072/2 / NCBR / 2017 and POIR.04.01.04-00-0027 / 18-00 Development of innovative technical and material solutions in the construction of an autonomous Agrobo under Measure 4.1 of the Intelligent Development Operational Program 2014-2020 co-financed by the European funds Regional Development Fund and TANGO2 / 340100 / NCBR / 2017..

Managing production processes in the enterprise

References

[1] Yang, L., Jiang, G., Chen, X., Li, G., Li, T. & Chen, X. (2019). Design of integrated steel production scheduling knowledge network system. *Claster Comput.* 10197-10206.
[2] Żurada, J. Barski, M., Jędruch, W. (1996). *Artificial Neural Networks. Fundamentals of theory and application*. Warszawa: PWN. (in Polish).
[3] Janiak, A. (2006). *Scheduling in computer and manufacturing systems.* Warszawa: Wydawnictwa Komunikacji i Łączności.
[4] Smutnicki, C. (2002). *Scheduling algorithms*. Warszawa: Akademicka Oficyna Wydawnicza EXIT. (in Polish).
[5] Coffman, E.G. (1980). *Task scheduling theory*. Warszawa: Wydawnictwa Naukowo-Techniczne. (in Polish).
[6] Janczarek, M. (2011). *Managing production processes in the enterprise*. Lublin: Lubelskie Towarzystwo Naukowe. (in Polish).
[7] Szeliga, M. (2019) *Practical machine learning*. Warszawa: PWN. (in Polish).
[8] Raschka, S. (2018) *Python machine learning.* Gliwice: Helion. (in Polish).
[9] Choi, H-S, Kim, J-S. & Lee, D-H. (2011). Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to a TFT-LCD line. *Expert System with Application*. 38, 3514-3521.
[10] Agarwal, A., Pirkul, H. & Jacob, V.S. (2003). Augmented neutral network for task scheduling. *European Journal of Operational Research*. 151, 481-502.
[11] Jain, A.S. & Meeran, S. (1998). Jop-shop scheduling using neutral networks. *International Journal of Production Research*,
[12] Fonseca-Reyna, Y.C., Martinez-Jimenez, Y. & Nowe, A. (2017). Q-Learning algorithm performance for m-machine, n-jobs flow shop scheduling problems to minimize makespan, *Revista Investigacion Operacional*. 38(3), 281-290.
[13] Dewi, Andriansyah, & Syahriza, (2019). Optimization of flow shop scheduling problem using classic algorithm: case study, IOP Conf. Series: Materials Science and Engineering 506.
[14] Putatunda, K. (2001) Development of austempered ductile cast iron (ADI) with simultaneous high yield strength and fracture toughness by a novel two-step austempering process. *Material Science and Engineering A.* 315, 70-80.
[15] Dayong Han, Hubei Key, Qiuhua Tang; Zikai Zhang; Jun Cao, (2020). Energy-efficient integration optimization of production scheduling and ladle dispatching in steelmaking plants. *IEEE Access.* 8, 176170-176187.
[16] Perzyk, M. (2017). The use of production data mining methods in the diagnosis of the causes of product defects and disruptions in the production process. *Utrzymanie Ruchu*. 4, 45-47. (in Polish).
[17] Perzyk, M., Dybowski, B. & Kozłowski, J. (2019). Introducing advanced data analytics in perspective of industry 4.0 in a die casting foundry. *Archives of Foundry Engineering.* 19(1), 53-57.
[18] Yescas, M. (2003). Prediction of the Vickers hardness in austempered ductile irons using neural networks. *International Journal of Cast Metals Research*. 15(5), 513-521.
[19] Report on the contract no. U / 227/2014 implemented at the Foundry Research Institute. (in Polish).