Czesław SMUTNICKI, Zbigniew BANASZAK, Grzegorz BOCEWICZ
Wrocław University of Science and Technology
Koszalin University of Technology

SZEREGOWANIE CYKLICZNE Z OGRANICZENIEM CZASÓW WYKONYWANIA

Streszczenie. Rozważany jest gniazdowy problem szeregowania cyklicznego. Przedstawiono kilka szczególnych własności problemu. Poszukiwanie optymalnego harmonogramu przekształcono w problem poszukiwania optymalnej kolejności wykonywania operacji. Wykorzystano w tym celu grafy. Rozpatrzono i przedyskutowano zagadnienia: badania dopuszczalności, wyznaczania harmonogramu dla ustalonej kolejności operacji, efektywnego poszukiwania przybliżonej kolejności, wpływu czasów wykonywania na rozwiązanie. Fragmenty podejść ilustrowano na przykładzie liczbowym.

CYCLIC SCHEDULING WITH LIMITED PROCESSING TIMES

Summary. We deal with the cyclic job-shop scheduling problem. We show a few its special properties. We convert seeking the optimal schedule into problem of seeking optimal sequence of operations by using graphs. There have been considered and analysed subjects: checking feasibility of the solution, finding the schedule for fixed processing order of operations, efficient method of seeking approximate sequence of operations, checking the influence of operations processing times on the solution. Fragments of the approach have been illustrated by an instance.

1. Introduction

Flexibility is one of the features enumerated for Industry 4.0, see the review of problems identified in this research area [4]. Flexibility is perceived as, for example, alternative routes for manufacturing products, various scenarios of making transport activities, planning manufacturing products on request or on clients orders, adjusting processing times to improve the efficiency of manufacturing and profits of the owner.

Up to now, we have considered several special problems raised in this area, among others: fully automated and partially automated manufacturing systems without or with limited human personnel, application of typically automated, cooperated transport supporting system perceived as the fleet of Automated Guided Vehicles (AGV), see for example [10, 17]. The fundamental aim of the AGV is to transfer a job (namely, the semi-product or product) between machines. There are several various strategies of implementing transport [1, 2]. In this paper we have assumed that each product has its own unique technological route for processing. It means that there is no flexibility in the

former (technological) meaning and in the later (transport) activities. On the other hand, we are considering flexibility as well as an alternative transport strategies in the latter meaning by admitting various workloads and routes of AGVs. Referring to the optimization criterion, we are considered the cyclic schedule with the minimal cycle time criterion as well as an approximation of minimal cycle time by the makespan. The paper is an extension of recent works on cyclic scheduling, see for example achievements in [3] and our works on this topic [6], [17].

2. The problem

We consider a general manufacturing system having the park of machines represented by the set $\mathcal{M} = \{1, \ldots, m\}$, where m is the number of machines. The system provides on the output in the cyclic manner (periodically) an mixed assortment of n various products determined by the set $\mathcal{N} = \{1, \ldots, n\}$. Each product $j \in \mathcal{N}$ can be perceived as a j-th production job and requires the predefined (fixed) sequence of n_j operations numbered successively by $(l_{j-1} + 1, \ldots, l_{j-1} + n_j)$ which have to be performed in this order, where $l_j = \sum_{s=1}^{j} n_s$, $j = 1, \ldots, n$ and $l_0 = 0$. The modeling technology commonly operates on the set of operations with precedence constraints

$$\mathcal{O} = \bigcup_{j=1}^{n} \bigcup_{s=l_{j-1}+1}^{l_j} \{s\}. \tag{1}$$

instead of the set of jobs. We denote in the sequel $o = |\mathcal{O}| = l_n$. Operation $i \in \mathcal{O}$ has to be performed on the machine ν_i in the unknown yet time p_i, $\underline{p}_i \leqslant p_i \leqslant \overline{p}_i$, where $\underline{p}_i$ and $\overline{p}_i$ are given bounds (pessimistic, optimistic). Additionally, the expected (suggested) processing time is assumed to be known and is denoted by $\tilde{p}_i$. The schedule is defined by start times of operations $S = (S_1, S_2, \ldots, S_o)$ and processing times of operations $p = (p_1, p_2, \ldots, p_o)$ which fulfill the *technological routing* constraints

$$S_i + p_i \leqslant S_{i+1},\ i = l_{j-1} + 1, \ldots, l_j - 1,\ j = 1, 2, \ldots, n \tag{2}$$

as well as *sequencing* constraints for operations with conflicting resource requirements

$$(S_i + p_i \leqslant S_k) \vee (S_k + p_k \leqslant S_i),\ \nu_i = \nu_k,\ i, k \in \mathcal{O}. \tag{3}$$

The optimization goal is to minimize the cycle time, which is defined as the period of time between successive occurrence of the same operation. The approach proposed in the paper refers also to the makespan as an auxiliary criterion.

3. Mathematical model

In order to define the solution as well optimization problem we use the modeling technology introduced by us originally for the conventional job-shop scheduling problem [10, 11] and employed also for cyclic job-shop [13, 14, 15, 16]. It is known in the literature for short as "permutation-and-graph". It has been considered as the most efficient, moderate comfortable, and moreover yields the representation of solutions with the smallest redundancy. The approach allows us to generate: schedule, graph for the formulation of special properties and the method of finding minimal makespan as well as cycle time. Details are given below.

Set of operations $\mathcal{O}$ can be decomposed into subsets $\mathcal{O}_k = \{i \in \mathcal{O} : \nu_i = k\}$, where $\mathcal{O}_k$ corresponds to operations which should be processed on machine k; we denote further $m_k = |\mathcal{O}_k|$, $k \in \mathcal{M}$. The processing order π of operations on all machines is defined by m-tuple $\pi = (\pi_1, \ldots, \pi_m)$, where $\pi_k = (\pi_k(1), \ldots, \pi_k(m_k))$ is a permutation on $\mathcal{O}_k$, $k \in \mathcal{M}$; $\pi_k(i)$ denotes this element of $\mathcal{O}_k$, which is on position i in π_k. Let Π_k be the set of all permutations on $\mathcal{O}_k$. Then $\pi \in \Pi = \Pi_1 \times \Pi_2 \times \ldots \times \Pi_m$. Note that, because of the feasibility condition, some π's cannot be acceptable as the legal solutions. In the sequel, we formulate a few conditions of feasibility depending of the method of finding the minimal cycle time for a fixed processing order. Cyclic behavior of the manufacturing system means that each machine k continuously repeats the sequence π_k, i.e. performs jobs in the following order $\pi_k\pi_k\pi_k \ldots$, for $k \in \mathcal{M}$. Cyclic schedule for fixed processing order π can be described by the vector $S = (S_1, \ldots, S_o)$ of operations' starting times. In successive cycles, S is increased by a constant T called the cycle time. Minimal cycle time for fixed π will be denoted by $T(\pi)$. Our fundamental aim is to find processing order π^*

$$T(\pi^*) = \min_{\pi} T(\pi), \tag{4}$$

where minimization runs only over feasible π's. It is clear that value of $T(\pi)$ depends on vector of variable processing times $p = (p_1, \ldots, p_o)$, which has an influence on the evaluation of the schedule value. Equation (4) implies an evident two-level solution algorithm which seeks π on the upper level, whereas on the lower level finds $T(\pi)$ for each fixed π. Thus, we need to resolve several research tasks: (a) how to check feasibility of π, (b) how to find $T(\pi)$, (c) how to generate only feasible π. We consider these topics hereinafter in details.

4. Linear Programming

In this section we provide the method of finding the schedule for the given π by using LP (Linear Programming). Our aim in this case is to find schedule $S = (S_1, S_2, \ldots, S_o)$, processing times $p = (p_1, p_2, \ldots, p_o)$ and the cycle time $T > 0$ for fixed given π through solving the following LP task

$$T(\pi) = \min_{S,p,T} T \tag{5}$$

with constraints

$$S_i + p_i \leqslant S_{i+1},\ i = 1, \ldots, n_j,\ i = l_{j-1} + 1, \ldots, l_j - 1,\ j = 1, \ldots, n, \tag{6}$$

$$S_{\pi_k(i)} + p_{\pi_k(i)} \leqslant S_{\pi_k(i+1)},\ i = 1, \ldots, m_k - 1,\ k = 1, \ldots, m, \tag{7}$$

$$S_{\pi_k(m_k)} + p_{\pi_k(m_k)} \leqslant S_{\pi_k(1)} + T,\ k = 1, \ldots, m, \tag{8}$$

$$\underline{p}_i \leqslant p_i \leqslant \overline{p}_i,\ i = 1, \ldots, o,\ \ S_i \geqslant 0,\ \ i = 1, \ldots, o,\ \ T \geqslant 0. \tag{9}$$

Inequalities (6) follow from technological route of jobs. Inequalities (7) follow from the fixed sequence π_k of operations on machines. Constraint (8) forces cyclic behaviour of the schedule. Constraint (9) provides bounds on the decision variables. We say that π is feasible if this LP problem has a finite solution (S, p, T). Unfortunately, such method of checking feasibility is too expensive in our opinion. On the other hand, LP belongs to P-class, then one expects moderate efficiency of this approach in finding $T(\pi)$. The first found feature is: the minimal value of T is obtainable for $p_i = \underline{p}_i$.

5. Decomposition

This decomposition is oriented on the usage of graphs in order to find $T(\pi)$. Already in [16] it has been suggested a decomposition of the problem (5), namely

$$T(\pi) = \min_{T}(\min_{S,p} T). \tag{10}$$

In the considered case (10) is supplemented by constraints (6) – (9). Equation (10) enforces a two-level method of finding $T(\pi)$ for the given π. Since the goal function (10) remains constant for T, so the internal problem is to check whether exists or not exists for the given T feasible S, p satisfying constraints (6) – (9). As it was mentioned in [16], in the context of metaheuristics, only $T(\pi)$ is essential for the search over space Π. Thus, we need to settle a few problems: (A) fast checking if for the given T exists feasible S, p satisfying (6) – (9), (B) solving $T(\pi)$ without LP. There are at least two ideas for (A) – (B). The first approach seeks minimal T for which $\min_S T < \infty$ still exists; any one-dimension minimization method can used in this aim, which means that (A) needs to be applied many times, see [15] for details. The method has relatively large pseudo-polynomial computational complexity. The second approach evaluates $T(\pi)$ by graph paths and applies (A) only once.

6. Graph models

We operate on two graphs. The first, planar graph $G(\pi)$ is associated with classical job-shop scheduling problem for the set of operations $\mathcal{O}$ and is used to formulate, among others, the sufficient condition for π to be feasible for this case of the job-shop, see for example Fig. 1. Being more precisely, π is feasible if the graph $G(\pi)$

$$G(\pi) = (\mathcal{O}, \mathcal{R} \cup \mathcal{E}(\pi)) \tag{11}$$

where $\mathcal{O}$ is the set of nodes, and $\mathcal{R} \cup \mathcal{E}(\pi)$ are sets of arcs

$$\mathcal{R} = \bigcup_{j=1}^{n} \bigcup_{i=l_{j-1}+1}^{l_j-1} \{(i, i+1)\}, \quad \mathcal{E}(\pi) = \bigcup_{k=1}^{m} \bigcup_{i=1}^{m_k-1} \{(\pi_k(i), \pi_k(i+1))\} \tag{12}$$

has no cycles. Node $i \in \mathcal{O}$ represents in Activity-on-Node (AoN) description style the operation $i \in \mathcal{O}$ and has weight $p_i = \underline{p}_i$.Weight of the node has no meaning for checking feasibility, but it is valid for calculation of the makespan. With the i-th node we associate the event S_i which denotes the earliest starting time of this operation. Arcs from sets $\mathcal{R}$ as well as from the set $\mathcal{E}(\pi)$ have weight zero. Profits from the application of the graph $G(\pi)$ is as follows. Let us define length of a paths in $G(\pi)$ between nodes i and j (including p_j) as

$$d_{i,j} = \max_{v \in B_j} d_{i,v} + p_j, \tag{13}$$

where B_j is the set of immediate predecessors of node j in $G(\pi)$. The makespan is

$$C_{\max}(\pi) = \max_{j \in \mathcal{O}} \max_{i \in \mathcal{B}_j} d_{i,j}. \tag{14}$$

$C_{\max}(\pi)$ can be found in $O(o)$ time by using conventional Bellman's algorithm. Relation between $C_{\max}(\pi)$ and $T(\pi)$ one can find in the following property; the proof will be skipped because of the paper size.

Property 1. *Let* $\kappa = (\kappa_1, \ldots, \kappa_s)$, $s \leqslant m$ *be the sequence so that* $\kappa_i \in \mathcal{M}$, $\kappa_i \neq \kappa_j$, $i, j = 1, \ldots, s$, $i \neq j$, $\kappa_{s+1} \equiv \kappa_1$, $d_{\pi_{\kappa_i}(1),\pi_{\kappa_{i+1}}(m_{\kappa_{i+1}})} < \infty$, $i = 1, \ldots, s$. *Then we have*

$$T(\pi) \leqslant C_{\max}(\pi), \quad T(\pi^*) \leqslant C_{\max}(\pi^*) \leqslant \min_{\pi \in \Omega \subseteq \Pi} C_{\max}(\pi), \tag{15}$$

and

$$T(\pi) \geqslant \max_{\kappa} T_\kappa \geqslant T_{\kappa,|\kappa|=2} \geqslant T_{\kappa,|\kappa|=1} \tag{16}$$

where

$$T_\kappa = \frac{1}{s} \sum_{i=1}^{s} d_{\pi_{\kappa_i}(1),\pi_{\kappa_{i+1}}(m_{\kappa_{i+1}})}. \tag{17}$$

Because the computational complexity of finding $\max_\kappa T_\kappa$ from (16) is non-polynomial, one can consider the limited sequences of κ, namely for $s = 2, 1$, see [16] for details, leading to polynomial-time algorithm, but with weaker lower evaluation.

The second, non-planar graph $H(\pi)$, will be used to find cyclic schedule S, p having given correct value $T(\pi)$ or its upper evaluation $T \geqslant T(\pi)$. In order to cover all possible cases of T, then the graph analysis for the value $T < T(\pi)$ should provide the answer that no feasible schedule exists. We define $H(\pi)$ as follows

$$H(\pi) = (\mathcal{O}, \mathcal{R} \cup \mathcal{E}(\pi) \cup \mathcal{F}(\pi)). \tag{18}$$

where sets $\mathcal{O}$, $\mathcal{R}$, $\mathcal{E}(\pi)$ are the same that as for the graph $G(\pi)$; the same are also the node and arcs weights. The graph $H(\pi)$ can be perceived as an extension of $G(\pi)$ created by adding set of additional arcs wrapping around the cylinder, see Fig. 1 (right). Set $\mathcal{F}(\pi)$ closing up the cylinder is defined as follows

$$\mathcal{F}(\pi) = \bigcup_{k=1}^{m} \{(\pi_k(m_k), \pi_k(1))\} \tag{19}$$

where arc $(i, j) \in \mathcal{F}(\pi)$ has weight minus $-T$ and represents preceding constraint (8).

We recall fundamental properties formulated previously, see [16]. They provide necessary and sufficient conditions for π to create the feasible schedule. The former states that "processing order π is feasible if and only if $G(\pi)$ does not contain a graph cycle and T is chosen so that $H(\pi)$ does not contain graph cycle of positive length". The latter states that "If graph $G(\pi)$ does not contain graph cycles, there always exists feasible schedule S, p for this π and with minimal cycle time $T(\pi)$".

Let us consider now the problem of finding S, p for fixed $T \geqslant T(\pi)$ by using $H(\pi)$. With each node $i \in \mathcal{O}$ we associate the event S_i having the sense of the longest path length going to this node i, without this node weight. Notice, $H(\pi)$ contains graph cycles (they go around the cylinder one or several times) as well as positive and negative weights on arcs. Therefore finding S, p can be perceived as certain labeling algorithm, which assigns to nodes values satisfying constraints (6) - (8). Such labeling procedure AC with the computational complexity $O(o^2)$ built on Bellman-Ford relaxation has been proposed in [16]. Paper [16] also provided the advanced version of AC called AC* which runs in shorter time $O(mo)$.

7. Instance

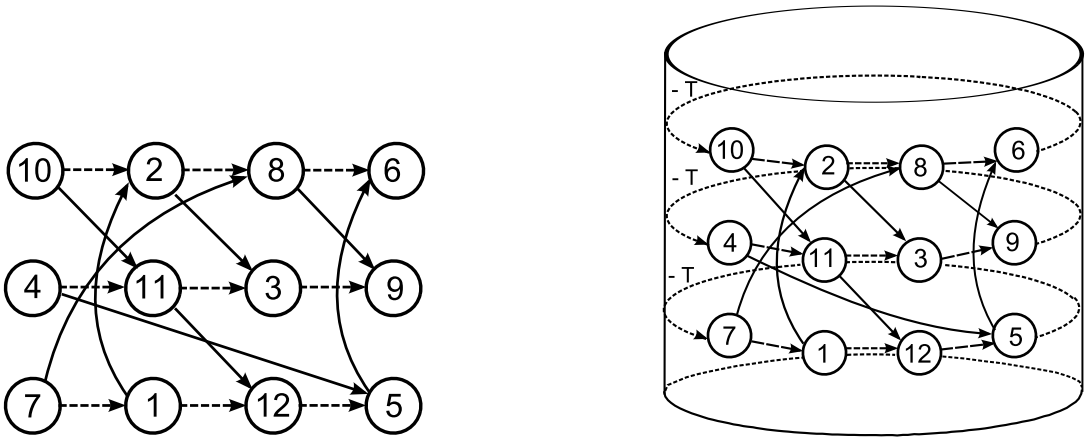
For the Instance from Tab. 1 and processing order

$$\pi = ((10, 2, 8, 6), (4, 11, 3, 9), (7, 1, 12, 5)) \tag{20}$$

Tabela 1

Illustrative instance

job	operation	machine	$\underline{p}_i$	$\tilde{p}_i$	$\overline{p}_i$	p_i	
	0						$\leftarrow l_0$
1	1	3	35	61	75	59	
1	2	1	39	67	83	65	
1	3	2	72	96	108	94	$\leftarrow l_1$
2	4	2	65	89	95	86	
2	5	3	48	56	88	60	
2	6	1	5	10	15	10	$\leftarrow l_2$
3	7	3	40	48	62	49	
3	8	1	38	42	52	43	
3	9	2	4	8	12	8	$\leftarrow l_3$
4	10	1	47	75	79	71	
4	11	2	16	24	38	25	
4	12	3	88	96	116	98	$\leftarrow l_4$



Rys. 1. Graphs $G(\pi)$ (left) and $H(\pi)$ (right).

graph $G(\pi)$ is shown in Fig. 1 (left), whereas $H(\pi)$ in Fig. 1 (right). Arcs from $\mathcal{R}$ are drawn by solid line, arcs from $\mathcal{E}(\pi)$ by dashed line, arcs from $\mathcal{F}(\pi)$ by dotted line.

Applying graph $G(\pi)$ for processing order (20) and fixed p we obtain $C_{\max}(\pi) = 289$ with $S = (49, 108, 173, 0, 219, 279, 0, 173, 216, 0, 86, 111)$. Applying graph $H(\pi)$, for fixed p (see last column in the table), we seek with the help of AC procedure [15] the minimal value of T so that $H(\pi)$ does not contain cycle of positive length. This condition is fulfilled for $T = 272$ but not for $T < 272$. Thus $T(\pi) = 272 \geqslant C_{\max}(\pi)$. Starting initially $S_i = 0,\ i = 1, 2, \ldots, o$ we finally get $S = (49, 108, 173, 3, 212, 272, 0, 173, 267, 10, 89, 114)$ from the analysis of paths in this graph. Evaluation of $T(\pi)$ appeared in the Property 1 provides $T_{\kappa,|\kappa|=1} = 266$ and $T_{\kappa,|\kappa|=2} = 272$. Note that solution of LP task also provides $T(\pi) = 272$.

8. Proposed approach

The solution algorithm is "hybrid" and contained two phases. In the first phase, we seek for the feasible approximate solution π^A of the problem with the makespan criterion, namely $C_{\max}(\pi^A) = \min_{\pi\in\Omega\subseteq\Pi} C_{\max}(\pi)$, see (15). To this aim algorithms TSAB [10] or i-TSAB [11] are recommended. Since the processing times are variable, therefore in order to speed up the search process and eliminate the influence of these variables on the choice of π^A, we assume that the processing time p_i is fixed and approximated by

$$p_i = \frac{\underline{p}_i + 4\tilde{p}_i + \overline{p}_i}{6}. \tag{21}$$

where evaluations are: $\underline{p}_i$ – optimistic, $\overline{p}_i$ – pessimistic and $\tilde{p}_i$ – average.

The second phase employs inequality $T(\pi^A) \leqslant C_{\max}(\pi^A)$, see (15). In this phase we solve only once the problem (5)–(9) for $\pi = \pi^A$. This can be done either by solving appropriate LP task or by seeking $T \geqslant T(\pi)$, see (16), for which graph $H(\pi)$ is feasible. Such the approach ensures a balance between calculation cost and quality of obtained solution.

9. Conclusion

Properties have been designed in order to find or, at least evaluate, the minimal cycle time $T(\pi)$ for the given job processing order π without necessity of solving auxiliary LP task or even without necessity of generating the exact schedule S, p. In this paper we propose and discuss an approach which allow us to reduce the size of the LP problem as well as the size of graph model due to skillful relaxation and aggregation of jobs. This reduction moves the appropriate LP graph model from pseudo-polynomial time complexity to polynomial time complexity. These findings extend significantly former results, found so far in our research, see [15]. All properties can be built into conventional metaheuristic approaches, e.g Simulated Annealing (SA), Tabu Search (TS), Evolutionary Search (EA) with particular evaluation of a processing order during the run. Actually, one can perform the complex procedure of finding the exact schedule only for final processing order generated by the output of local optimization algorithm. Note, $T(\pi)$ can be found more efficiently than LP task by using special graphs, being the extension of graphs formulated originally for the conventional job-shop scheduling problem. The computational experiments is the natural continuation of the research.

LITERATURA

1. Bocewicz G., Muszynski W., Banaszak Z., Models of multimodal networks and transport processes. Bulletin of the Polish Academy of Sciences: Technical Sciences, 2015, vol. 63(3), p. 635-650.
2. Bocewicz G., Nielsen I., Banaszak Z., Majdzik P., A cyclic scheduling approach to maintaining production flow robustness. Advances in Mechanical Engineering, 2018, vol. 10(1).

3. Brucker P., Kampmeyer T., Cyclic job shop scheduling problems with blocking. Annals of Operations Research, 2008, vol. 159(1), p. 161-181.
4. Dolgui A., Ivanov D., Sethi S. P., Sokolov B., Scheduling in production, supply chain and Industry 4.0 systems by optimal control: fundamentals, state-of-the-art and applications. International Journal of Production Research, 2019, vol. 5(2), p. 411-432.
5. Grabowski J., Pempera J.: New block properties for the permutation flow shop problem with application in tabu search. Journal of Operational Research Society, 2001, vol. 52(2), p. 210-220.
6. Pempera J., Smutnicki C., Open shop cyclic scheduling. European Journal of Operational Research, 2018, vol. 269(2), p. 773-781.
7. Grabowski J., Pempera J., Sequencing of jobs in some production system. European Journal of Operational Research, 125, 2000, p. 535–550.
8. Kampmeyer T., Cyclic Scheduling Problems, Ph. D. Thesis, University Osnabrück, 2006.
9. Levner E., Kats V., De Pablo D. A. L., Cheng T. E., Complexity of cyclic scheduling problems: A state-of-the-art survey. Computers and Industrial Engineering, 2010, vol. 59(2), p. 352-361.
10. Nowicki E., Smutnicki C., A fast taboo search algorithm for the job shop problem. Management Science, 1996, vol. 6, p. 797–813.
11. Nowicki E., Smutnicki C., New algorithm for the job-shop problem. Journal of Scheduling 2005, vol. 8, p. 145–159.
12. Sawik T., A scheduling algorithm for flexible flow lines with limited intermediate buffers. Applied Stochastic Models and Data Analysis 1993, vol. 9, p. 127–138.
13. Smutnicki C., An efficient algorithm for finding minimal cycle time in cyclic job shop scheduling problem. IEEE 16th International Conference on Intelligent Engineering Systems (INES), June 13-15, 2012, Lisbon, Portugal. Piscataway, NJ: IEEE, cop. 2012. p. 381-386.
14. Smutnicki C., New features of the cyclic job shop scheduling problem. 20th International Conference on Methods and Models in Automation & Robotics, 24-27 August, 2015. [Piscataway, NJ: IEEE, cop. 2015, p. 1000-1005.
15. Smutnicki C., Cykliczny problem gniazdowy, w: (red. Bozejko W. i inni) Zaawansowane modele i algorytmy optymalizacji w systemach cyklicznych, EXIT, Warszawa, 2017.
16. Smutnicki C., A New Approach for Cyclic Manufacturing. IEEE 22nd International Conference on Intelligent Engineering Systems (INES), 21-23 June 2018, IEEE, doi: 10.1109/INES.2018.8523903.
17. Smutnicki C., Pempera J.: Job shop scheduling with transport by Automated Guide Vehicles. In: 16th International Conference on Soft Computing Models in Industrial and Environmental Applications (SOCO 2021), Hugo Sanjurjo González et al. (eds.), Cham : Springer, cop. 2022, pp. 789-799. (Advances in Intelligent Systems and Computing, ISSN 2194-5357; vol. 1401).