Computers & Industrial Engineering 113 (2017) 512–524

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Parallel tabu search for the cyclic job shop scheduling problem

Wojciech Bożejko[a,*], Andrzej Gnatowski[a], Jarosław Pempera[a], Mieczysław Wodecki[b]

[a] Department of Automatics, Mechatronics and Control Systems, Faculty of Electronics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland

[b] Telecommunications and Teleinformatics Department, Faculty of Electronics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland

ARTICLE INFO

Keywords:
Job shop problem
Cyclic scheduling
Tabu search
Parallel algorithm

ABSTRACT

In this paper, we consider a cyclic job shop problem, consisting of production of a certain set of elements at fixed intervals. Optimization of the process is reduced to a minimization of a cycle time, i.e. the time, after which the next batch of the same elements may be produced. We introduce a new parallel method for the cost function calculation. The parallelization is not trivial and cannot be done automatically by the existing compilers due to the recurrent character of formulas. Since the problem is strongly NP-hard, a heuristic algorithm was designed to solve it. Computational experiments were done in a multiprocessor environment, namely – in Intel Xeon Phi.

1. Introduction

Contemporary production systems executing multi-assortment production or production in large quantities, with constant or slowly changing range of products, manufacture items exclusively in a cyclic manner. For a fixed batch of jobs (products) in the cycle, optimization of such a system usually boils down to the cycle time minimization, resulting not only in an increased system capacity, but also in an improved machine utilization. A large part of this type of systems can be classified as production models performing jobs on production slots, which correspond to the job shop task scheduling problem. In the paper we present the cyclic scheduling problem where the set of jobs is repeatedly processed by the machine, in the same order in each cycle. Described in this way, the job shop scheduling problem belongs to a wider group of cyclic optimization problems, which are recently a subject of growing interest among both: theorists and practitioners (Brucker & Kampmeyer, 2005; Brucker & Kampmeyer, 2008a; Bożejko, Gnatowski, Niżyński, & Wodecki, 2016; Bożejko, Gnatowski, Idzikowski, & Wodecki, 2017).

In this paper there is considered a solution to a cyclic job shop problem of task scheduling through the use of tabu search algorithm in the coprocessor environment, namely – Intel Xeon Phi coprocessor. The process of neighborhood search and the calculation of the value of the objective function were subjected to parallelization. The impact of the used strategy on neighborhood generation, as well as the number of used threads in achieved speedup, were examined. There are also new theoretical properties established enabling the effective determining of the cycle time, which, as it turns out, is a rational number at integral values of the times of operations execution.

In the literature, especially in the latest publications, there is a number of papers addressing cyclic scheduling problems in a variety of industrial applications: (Hitz, 1980; Méndez, Cerdá, Grossmann, Harjunkoski, & Fahl, 2006; Pinedo, 2008, 2009; Pinto, Barbosa-Póvoa, & Novais, 2005; Trautmann & Schwindt, 2009), in communication, transport and logistics (Burke, De Causmaecker, Berghe, & Van Landeghem, 2004; Gertsbakh & Serafini, 1991; Kubiak, 2005), as well as in a field of multiprocessor computing (Kats & Levner, 2003). In modeling and optimization of flexible manufacturing systems (FMS) with a cyclic production, there are also Petri nets used (Lee & Korbaa, 2004). Lee and Posner (1997) considered a special case of the cyclic job shop problem, in which the order of operations on the machines is fixed. On the other hand, an open shop problem, describing the use of the graph coloring device was considered by Kubale and Nadolski (2005). A branch and bound algorithm based on a partial selections construction is proposed in Hanen (1994). Properties of the cyclic job shop are presented in Hanen and Munier (1995). For the special case of the problem, where each job has at most two operations, Hall, Lee, and Posner (2002) propose polynomial time algorithms. Lee and Posner (1997) investigate an application of a disjunctive graph model as a solution representation for the cyclic job shop problem. Jalilvand-Nejad and Fattahi (2015) propose a mixed integer linear programming (MILP) model for small size instances of a flexible cyclic job shop problem. The authors describe the usage of two metaheuristics: a genetic algorithm and simulated annealing for solving real size problems. Brucker and Kampmeyer (2008a) consider the cyclic job shop problem with blocking. Operations are blocked on a machine if the next machine is still working (in the literature this constraint is also known as *no-store*). The authors apply the tabu search algorithm to solve the problem. The

* Corresponding author.

http://dx.doi.org/10.1016/j.cie.2017.09.042
Received 16 March 2017; Received in revised form 23 September 2017; Accepted 25 September 2017
Available online 28 September 2017
0360-8352/

subject of metaheuristics in cyclic job scheduling is presented in Brucker and Kampmeyer (2005) and Bożejko, Uchroński, and Wodecki (2016). A job shop problem was also solved by parallel programming in Bożejko (2012), where the method of parallel objective function calculation was proposed. The paper Bożejko, Uchroński et al. (2016) concerns using Intel Xeon Phi coprocessor to execute a parallel tabu search and a simulated annealing for the cyclic flow shop – a special case of the cyclic job shop problem.

2. Problem definition

The researched problem is equivalent to a variant of the *cyclic job shop with machine chain repetition*, described in the work of Brucker and Kampmeyer (2008b). However, we consider a different mathematical model of problem, with multiple new properties proven.

2.1. Mathematical model

Formally, the problem can be defined as follows: there is a manufacturing system considered which consists of m machines (with a unit capacity), constituting a set $\mathcal{M} = \{1,2,...c,m\}$. There are n jobs in the system, given by a set $\mathcal{N} = \{1,2,...c,n\}$, that should be performed cyclically (in a repetitive manner). The j-th job requires a sequence of n_j operations to be performed. Operations are indexed by successive natural numbers: $l_{j-1}+1, l_{j-1}+2,...,l_{j-1}+n_j$, where l_j is the total number of operations in the first j jobs:

$$l_j = \begin{cases} \sum_{i=1}^{j} n_i & \text{for } j = 1,2,...,n, \\ 0 & \text{for } j = 0. \end{cases} \tag{1}$$

The operations must be performed in a predefined, technological order $l_{j-1}+1 \to l_{j-1}+2 \to \cdots \to l_{j-1}+n_j$. A set $\mathcal{O} = \{1,2,...,o\}$, where $o = \sum_{i=1}^{n} n_i$ is the number of operations, contains operations from all the jobs. Each operation $i \in \mathcal{O}$, must be performed uninterruptibly on a machine $v_i \in \mathcal{M}$, in time $p_i > 0$. The set of operations $\mathcal{O}$ can be, in a natural manner, decomposed into m subsets. A subset $\mathcal{O}_k = \{j \in \mathcal{O}: v_j = k\}$ corresponds to operations performed on the machine $k \in \mathcal{M}$, $m_k = |\mathcal{O}_k|$ is the number of operations performed on this machine. Let Π_k be the set of all possible permutations of elements from $\mathcal{O}_k$. The order in which operations are performed on the machine k is described by a tuple $\pi_k = (\pi_k(1),\pi_k(2),...,\pi_k(m_k)) \in \Pi_k$; where $\pi_k(i)$ denotes the operation performed on the machine k after $i-1$ other operations, i.e. the operation $\pi_k(1)$ is performed as a first, then the operation $\pi_k(2)$, etc. The order of execution of all operations is defined as m-short $\pi = (\pi_1,\pi_2,...,\pi_m) \in \Pi = \Pi_1 \times \Pi_2 \times \cdots \times \Pi_m$.

A set of jobs in a single cycle is called MPS (*Minimal Part Set*). MPSs are processed one after another in a cyclic manner, every *cycle time* T, i.e. each operation is repeated every T time. The schedule of x-th MPS is described by two vectors

$$S^x = (S_1^x, S_2^x, ..., S_o^x), \tag{2}$$

$$C^x = (C_1^x, C_1^x, ..., C_o^x), \tag{3}$$

where S_j^x and C_j^x denote respectively the moment of start and completion of the operation j in x-th MPS. The feasible schedule must satisfy the following constraints:

(a) each machine can perform at most one operation in each unit of time;
(b) each operation can be performed only by one, determined by the production process, machine;
(c) the technological order must be preserved;
(d) performance of operations cannot be interrupted;
(e) each operation is performed every T time;
(f) within each machine, operations from $x+1$-th MPS can start only after all operations performed on the machine from x-th MPS are finished.

From the constraint (d), it is possible to represent the schedule of x-th MPS by a single vector S^x (or C^x). The constraints can be formally defined as

$$S_i^x + p_i \leqslant S_{i+1}^x, \qquad i = l_{j-1}+1,...,l_j + n_j, \quad j = 1,2,...,n, \tag{4}$$

$$S_{\pi_k(i)}^x + p_{\pi_k(i)} \leqslant S_{\pi_k(i+1)}^x, \qquad i = 1,2,...,m_k-1, \quad k = 1,2,...,m, \tag{5}$$

$$S_i^x \geqslant 0, \qquad i = 1,2,...,o, \tag{6}$$

$$C_i^x = S_i^x + p_i, \qquad i = 1,2,...,o, \tag{7}$$

$$S_i^x + T = S_i^{x+1}, \qquad i = 1,2,...,o, \tag{8}$$

$$S_{\pi_k(m_k)}^x + p_{\pi_k(m_k)} \leqslant S_{\pi_k(1)}^{x+1}, \qquad k = 1,2,...,m, \tag{9}$$

where $x = 1,2,...$ is the number of MPS, and T is the cycle time. The constraint from (5) results from the sequential nature of the machine work, whereas (4) is a result of the technological order in which operations are preformed within the jobs. The constraint (6) is obvious. The constraint from (8) imposes the periodicity of operations performance; and (7) – that the performance of operations cannot be interrupted. The constraint from (9) ensures that the MPSs are disjunctive, as described in the constraint (f).

The problem boils down to finding a schedule which minimizes the cycle time and satisfying constraints (4)–(9).

A relaxation of the constraint (9) leads to the problem with no constraints (NC) for interlacing operations from different MPSs. The cycle time of the optimal solution for the NC problem is also a lower bound for the cycle time of the problem considered in this paper.

Property 1. *The NC problem is solvable in polynomial time. The cycle time of the optimal solution T_{NC} equals*

$$T_{NC} = \max_{k \in \mathcal{M}} \{ \sum_{i \in \mathcal{O}_k} p_i \}. \tag{10}$$

Proof. Assume that $\pi \in \Pi$ is an arbitrary order of execution and T_{NC} is the cycle time calculated from (10). It can be easily shown, that the schedule S^x, C^x

$$S_{\pi_k(i)}^x = T_{NC}(\gamma(i)-1+x) + \sum_{j=1}^{i-1} p_{\pi_k(j)}, \qquad \text{for } k = 1,2,...m,\ i = 1,2,...,n_k, \tag{11}$$

$$C_i^x = S_i^x + p_i, \quad \text{for } i = 1,2,...,o, \tag{12}$$

where $\gamma(i)$ is the position of the operation i in the job j:

$$i = l_{j-1} + \gamma(i), \quad \text{for } i \in (l_{j-1}+1, l_{j-1}+2,..., l_{j-1}+n_j) \tag{13}$$

and $x = 1,2,...$ is a number of MPS, is feasible. □

Example 1. In Table 1, an instance of the cyclic job shop scheduling problem with two jobs is given. The first job consists of three operations with the processing order: $1 \to 2 \to 3$ and the second one consists of two operations: $4 \to 5$. Assuming that there are no constraints put on MPSs interlacing, the cycle time of an optimal solution can be calculated from (10), $T_{NC} = 3$. The feasible schedule for the first MPS, obtained from (11) and (12) is shown in Table 2.

For the purpose of visualizing the idea behind (11) and (12), a graphic presentation how the schedule is build is shown in Fig. 1. First,

Table 1
The cyclic job shop scheduling problem instance from Example 1.

Operation (i)	1	2	3	4	5
Processing time (p_i)	1	3	1	2	2
Machine (v_i)	1	2	3	3	1
Technological pred.	–	1	2	–	4

Table 2
The schedule from Example 1.

Operation (i)	1	2	3	4	5
Position in job ($\gamma(i)$)	1	2	3	1	2
Operation start (S_i^1)	0	3	6	1	4
Operation end (C_i^1)	1	6	7	3	6

starting from 0, the timeline is divided into an infinite number of time windows, each of the width of T_{NC}. Afterwards, operations from the MPS are sequentially added to the diagram, aligned to the left to 0 – or to the end of the previous operation. The order in which operations are added is insignificant, and is determined by the arbitrarily taken $\pi = ((1,5),(2),(3,4))$. The obtained schedule is then copied into each time window. In order to build a schedule for the first MPS (marked with bold lines), for each job, each operation in the technological order is chosen from a different, consecutive time window. Here, job 1 consists of three operations: $1 \rightarrow 2 \rightarrow 3$. Therefore, the operation 1 is taken from the first time window ($S_1^1 = 0$), 2 from the second one ($S_2^1 = 3$) and 3 from the third one ($S_3^1 = 6$). Finally, the same procedure is applied to the second job.

Definition 1. For a given order of execution of π, the minimal cycle time $T(\pi)$ is the lowest value of the cycle time T, for which π is still feasible, i.e. there is at least one schedule S^x, $x = 1,2,...$, that complies with constraints (4)–(9). If such a cycle time does not exist, $T(\pi) = \infty$.

Assuming that, for a given π, the method of finding a schedule with the cycle time $T(\pi)$ is known, the problem can be reformulated to finding $\pi^* \in \Pi$ which minimizes the minimal cycle time $T(\pi)$

$$\pi^* \in \underset{\pi \in \Pi}{\operatorname{argmin}} \{T(\pi)\}. \tag{14}$$

Such a method is presented in Section 3.

2.2. Comparison to other problems

In the proposed model, despite disjunctive MPSs, there is 'interlacing' of operations possible in successive MPSs, i.e. it is possible for operations from the $x + 1$-th MPS to be executed before the completion of certain operations from the x-th MPS. The 'interlaces' may be prohibited without guarantying disjunctive MPSs by adding specific restraints to the model.

Such an approach is taken in Hanen and Munier (1995) and Brucker and Kampmeyer (2005), where the constraints are modeled by a directed graph G. The set of nodes consists of operations from $\mathcal{O}$ and two additional dummy nodes: $\circ$ and $*$ (with no processing time). The arcs represent constraints of the problem and are weighted by two functions: L, called length and H, called height. In the model, length L of an arc from a node i (representing operation $i \in \mathcal{O}$) equals p_i. The height, on the other hand, equals zero, with an exception of $H_{*\circ}$ (and the disjunctive arcs representing the order in which operations are performed on the machines). The sum of lengths in a given cycle μ in G is denoted by $L(\mu)$, and the sum of heights by $H(\mu)$. The cycle time equals to the highest value of $V(\mu) = \frac{L(\mu)}{H(\mu)}$ among the cycles in G. The cycle with the highest value is called the critical circuit. In the model, when $H_{*\circ} = 1$, all the operations $i \in \mathcal{O}$ from the x-th MPS must be finished before starting the operations from the $x + 1$-th MPS:

$$C_i^x \leqslant S_j^{x+1}, \quad i,j = 1,...,o, \quad x = 1,2,.... \tag{15}$$

Such a constraint leads to the problem equivalent to the well-researched job shop scheduling problem with a makespan minimization. In general, for $H_{*\circ} = a$, operations from a different MPSs can be scheduled in the same time window. Therefore, usually the larger the value of $H_{*\circ}$ is, the shorter the cycle times can be achieved, as more schedules become feasible (Brucker & Kampmeyer, 2005) and the value of the critical circuit may be lowered (H is inversely proportional to V). When $H_{*\circ}$ approaches infinity, the constraint on MPSs 'interlacing' is relaxed and the problem can be solved in polynomial time (as shown in Property 1).

While $H_{*\circ}$ is a versatile tool in defining various cyclic scheduling problems, it cannot express the constraint (9). Hence, the mathematical model described in this paper (and thereby the problem) is different from the cyclic job shop scheduling problem researched in Kampmeyer (2006) and Kechadi, Low, and Goncalves (2013). However, it is possible to describe our problem using the concept of *height*, as it was shown in Brucker and Kampmeyer (2008b), where *cyclic job shop with machine chain repetition* is discussed.

Example 2. Let us consider the instance of the problem from the Example 1. For the constraints introduced in this paper, the optimal schedule with the cycle time $T = 4.5$ and the order of execution $\pi = ((1,5),(2),(4,3))$ is shown in Fig. 2a. For the problem with the *hight* constraints and $H_{*\circ} = 1$, the optimal schedule with the cycle time $T = 5$ and the order of execution $\pi = ((1,5),(2),(4,3))$ is shown in Fig. 2b. For the problem without constraints on MPSs interlacing (or for $H_{*\circ}$ approaching infinity), the optimal schedule with the cycle time $T = 3$ and $\pi = ((1,5),(2),(3,4))$ is shown in Fig. 2c. Despite the same value of the cycle time, the schedule is different from the one presented in Fig. 1.

3. Graph model

Let graphs $H(\pi)$ and $G(\pi)$ be defined in the following manner (based on Nowicki & Smutnicki, 1996):

$$G(\pi) = (\mathcal{O}, \mathscr{R} \cup \mathscr{E}(\pi)), \tag{16}$$

$$H(\pi) = (\mathcal{O}, \mathscr{R} \cup \mathscr{E}(\pi) \cup \mathscr{E}^*(\pi)), \tag{17}$$

where $\mathcal{O}$ is the set of vertices and $\mathscr{R}$,$\mathscr{E}(\pi)$ and $\mathscr{E}^*(\pi)$ are sets of arcs

$$\mathscr{R} = \cup_{j=1}^{n} \ \cup_{i=l_{j-1}+1}^{l_j-1} \{(i,\ i+1)\}, \tag{18}$$

$$\mathscr{E}(\pi) = \cup_{k=1}^{m} \ \cup_{i=1}^{m_k-1} \ \{(\pi_k(i),\pi_k(i+1))\}, \tag{19}$$

$$\mathscr{E}^*(\pi) = \cup_{k=1}^{m} \{(\pi_k(m_k),\pi_k(1))\}. \tag{20}$$

The arcs representing the technological order are denoted by $\mathscr{R}$, the arcs representing the order in which operations are performed on the machines (the machine order) by $\mathscr{E}(\pi)$, whereas the arcs from the set $\mathscr{E}^*(\pi)$ – the cyclic precedence constraint

$$(S_{\pi_k(m_k)} + p_{\pi_k(m_k)}) - T \leqslant S_{\pi_k(1)}, \quad k = 1,...,m. \tag{21}$$

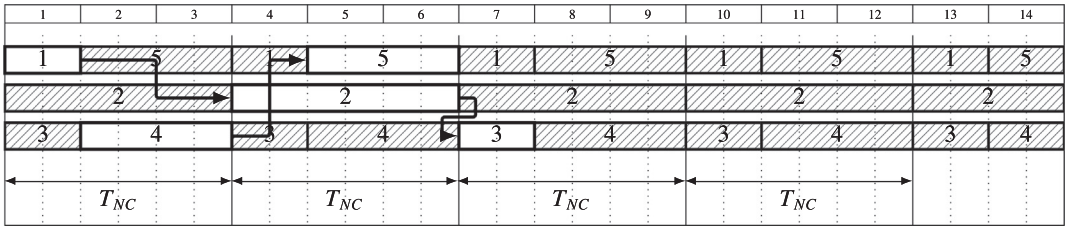
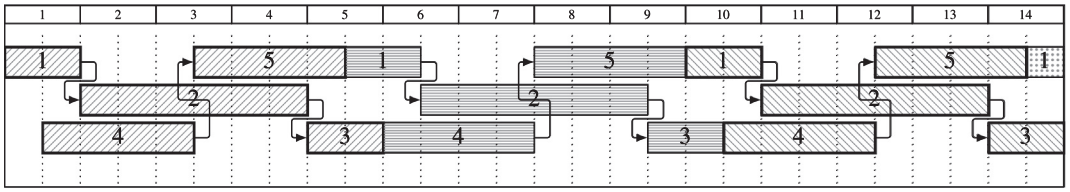
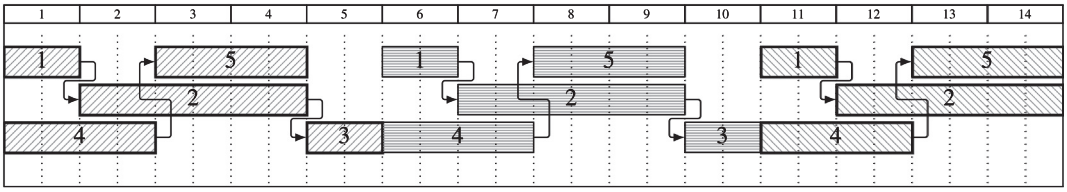


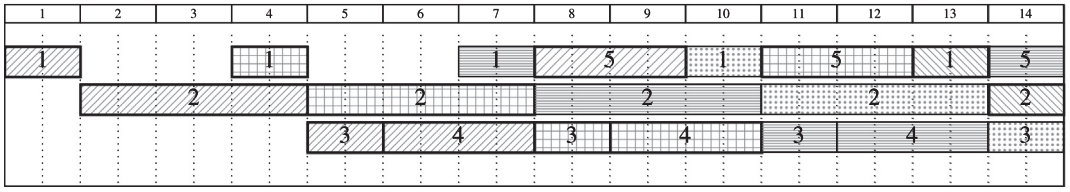
Fig. 1. Schedule from Example 1. First MPS is marked with bold lines.



(a) Constraints introduced in this paper, $T = 4.5$.



(b) Literature constraints, $H_{*\circ} = 1$, $T = 5$.



(c) No constraints on interlacing, $T = 3$. Arrows representing the technological order are omitted to increase the readability

Fig. 2. Optimal schedules for the problem instance from Example 2 and different constraints.

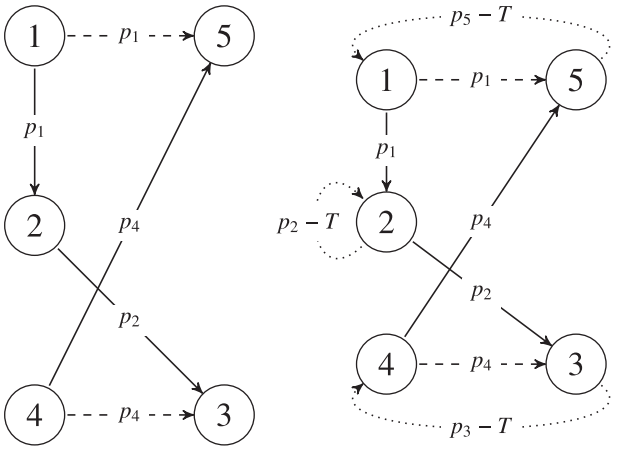
Each arc $(i,j) \in \mathscr{R} \cup \mathscr{E}(\pi)$ is loaded with weight p_i, whereas the arcs $(i,j) \in \mathscr{E}^*(\pi)$ have weight $p_i - T$.

Example 3. Let us consider the problem instance from the Example 1 and the optimal execution order $\pi = ((1,5),(2),(4,3))$. The corresponding graphs $G(\pi)$ and $H(\pi)$ are shown in Fig. 3.

In the further part of the paper, several properties of the graphs $G(\pi)$ and $H(\pi)$ related to the feasibility of solutions they represent are shown. The properties will be then used to calculate the minimal cycle time during neighborhoods generation and also as an elimination technique in the parallel tabu search algorithm (as block property).

Property 2. *An order of execution π is feasible if and only if the graph $G(\pi)$ does not contain a cycle.*

Property 3. *For a feasible order of execution π, a feasible cyclic schedule with the cycle time T exists if and only if the graph $H(\pi)$ does not contain a cycle with a positive weight. In this case, we say that π is feasible for the cycle time T.*



(a) Graph $G(\pi)$ (b) Graph $H(\pi)$

Fig. 3. Graphs $G(\pi)$ and $H(\pi)$ from the Example 2. The technological arcs from the set $\mathscr{R}$ are denoted with solid lines, the machine arcs from the set $\mathscr{E}(\pi)$ with dashed lines and the cyclic arcs from the set $\mathscr{E}^*(\pi)$ with dotted lines.

In order to introduce further properties, several new symbols and functions are defined firstly. Let $\mu = (\mu(1),\mu(2),\ldots,\mu(s))$ be a cycle in the graph $H(\pi)$, consisting of arcs from a multiset

$$A(\mu) = \bigcup_{i=i}^{s-1} \{(\mu(i),\mu(i+1))\} \cup \{(\mu(s),\mu(1))\}, \tag{22}$$

and $\mathscr{C}(H(\pi))$ be the set of all possible cycles in the graph $H(\pi)$. A set $C'(H(\pi)) \subset C(H(\pi))$ is the set of cycles with unique arcs

$$C'(H(\pi)) = \{\mu \in C(H(\pi)): \forall_{i,j\in\{1,\ldots,|\mu|-1\}:i\neq j} ((\mu(i),\mu(i+1)) \neq (\mu(j),\mu(j+1)) \wedge (\mu(i),\mu(i+1)) \neq (\mu(|\mu|),\mu(1)))\}. \tag{23}$$

For a given cycle time T and a cycle $\mu \in \mathscr{C}(H(\pi))$, the sum of arcs weights in the cycle (in short – *weight of the cycle*) can be expressed by the equation:

$$L_\mu(T) = \sum_{i\in\mu} p_{\mu(i)} - T\cdot|\mathscr{K}_\mu|, \tag{24}$$

$$\mathscr{K}_\mu = \{(i,j) \in A(\mu): (i,j) \in \mathscr{E}^*(\pi)\}, \tag{25}$$

where $\mathscr{K}_\mu$ is the multiset of the cycle arcs in the cycle μ. Finally, let $L(T,\pi)$ denote, for a given cycle time T, the highest weight of the cycle from the graph $H(\pi)$

$$L(T,\pi) = \max_{\mu\in\mathscr{C}(H(\pi))} \{L_\mu(T)\}. \tag{26}$$

Property 4. *For a given feasible π and the cycle time $T \in [T(\pi),+\infty)$ the function $L(T,\pi)$ of a T variable is continuous and strictly decreasing.*

Proof. For $T < T(\pi)$, from Definition 1, the order of execution π is not feasible. From Property 3

$$0 < L(T,\pi) = \infty \Leftrightarrow T < T(\pi), \tag{27}$$

and therefore

$$L(T,\pi) \leqslant 0 \Leftrightarrow T \geqslant T(\pi). \tag{28}$$

From Eq. (28), for $T \geqslant T(\pi)$, we have $\forall\ \mu \in C(H(\pi))\,(L_\mu(T) \leqslant 0)$ and from Eq. (27), for $T < T(\pi)$,$\exists\ \mu \in \mathscr{C}(H(\pi))\,(L_\mu(T) = +\infty)$. Function $L_\mu(T)$ is strictly decreasing and continuous, therefore $L(T,\pi)$ as defined in Eq. (26) is also strictly decreasing and continuous for $T \in [T(\pi),+\infty)$. □

Property 5. *For a given feasible order of execution π, the weight of the cycles with the highest weights in the graph $H(\pi)$ equals to 0 if and only if the cycle time is minimal:*

$$L(T,\pi) = 0 \Leftrightarrow T = T(\pi). \tag{29}$$

Proof. From Eqs. (24) and (26) it is obvious that: $T = T(\pi) \Rightarrow L(T,\pi) = 0$. Function $L(T,\pi)$ is strictly monotonic for $T > T(\pi)$ (Prop. 4), therefore: $L(T,\pi) = 0 \Leftrightarrow T = T(\pi)$.

Property 6. *For a feasible order of execution π and the cycle time $T = T(\pi)$, there is at least one cycle μ^* in the graph $H(\pi)$, consisting of the unique arcs only, that satisfies the condition $L_{\mu^*}(T) = 0$.*

Proof. For $T = T(\pi)$, let us consider the cycle with the highest weight comprising the repeating arcs: $\mu_p \in C(H(\pi))\backslash C'(H(\pi))$. Then

$$\begin{aligned}\exists_{i,j\in\{1,\ldots,|\mu_p|-1\}:i\neq j}\ (\mu_p(i),\mu_p(i+1)) &= (\mu_p(j),\mu_p(j+1)) \vee (\mu_p(i),\mu_p(i+1)) \\ &= (\mu_p(|\mu_p|),\mu_p(1)),\end{aligned} \tag{30}$$

where $(\mu_p(i),\mu_p(i+1))$ are the repeated arcs. It is possible to construct a subcycle μ_{sub} of μ_p, defined as $\mu_{sub} = (\mu_p(i),\mu_p(i+1),\ldots,\mu_p(j-1))$ or $\mu_{sub} = (\mu_p(i),\mu_p(i+1),\ldots,\mu_p(|\mu_p|-1))$. If the designated subcycle $\mu_{sub} \notin C'(H(\pi))$, then the following subcycles are being build until $\mu_{sub} \in C'(H(\pi))$. Without a loss of generality, it is possible to assume that $\mu_{sub} \in C'(H(\pi))$. Hereby the following cases should be taken into consideration:

(1) $L_{\mu_{sub}} > 0$ – contradiction, the Property 3 is not fulfilled.
(2) $L_{\mu_{sub}} = 0$ – from the Property 5, the subcycle μ_{sub} is one of the cycles of the highest weight.
(3) $L_{\mu_{sub}} < 0$ – contradiction, the cycle $\mu_p(i)$, whose subcycle is μ_{sub}, is not the cycle of the highest weight. It is possible to design the cycle of the highest weight by removing the arcs of the subcycle μ_{sub} from the cycle $\mu_p(i)$, at the same time creating the cycle $\mu_{\widehat{sub}}$ with the weight of $L_{\mu_{\widehat{sub}}} = L_{\mu_p(i)} - L(\mu_{sub})$ in the following way:

$$\mu_{\widehat{sub}} = (\mu_p(1),\mu_p(2),\ldots,\mu_p(i),\mu_p(j+1),\ldots,\mu_p(s_p)). \tag{31}$$

Only in case of (2) there is no contradiction. Therefore, when $T = T(\pi)$, there are infinitely many cycles with the highest weight equal to 0 and at least one of them belongs to the set $C'(H(\pi))$. □

Property 7. *For the cyclic job shop scheduling problem with operations processing times expressed by the natural numbers $\forall\ i \in \mathscr{O}\ (p_i \in \mathbb{N}\backslash\{0\})$ and a feasible order of execution π, the minimal cycle time is a rational number satisfying the condition*

$$T(\pi) = \frac{a}{b}, \quad a,b \in \mathbb{N}\backslash\{0\}, \quad 1 \leqslant b \leqslant m.$$

Proof. Let μ^* be the shortest cycle with the highest weight in the graph $H(\pi)$ (the length is defined as the number of the arcs that make up the cycle). From Property 6, for the cycle time $T = T(\pi)$ we have:

$$L_{\mu^*}(T) = 0. \tag{32}$$

By substituting (32) into (24), we obtain:

$$L_{\mu^*}(T) = \sum_{i\in\mu^*} p_i - T\cdot|\mathscr{K}_{\mu^*}| = 0. \tag{33}$$

Thus, after transformations

$$T = \frac{\sum_{i\in\mu^*} p_i}{|\mathscr{K}_{\mu^*}|} = \frac{a}{b}, \tag{34}$$

where

$$a = \sum_{i\in\mu^*} p_i, \quad 1 \leqslant b = |\mathscr{K}_{\mu^*}| \leqslant |\mathscr{E}^*(\pi)| = m. \tag{35}$$

Since $\forall\ i \in \mu^*\ (p_i \in \mathbb{N})$, therefore $a \in \mathbb{N}$. □

In order to determine the value of the minimal cycle time, there will be a graph required which is a m-fold duplication of the graph $G(\pi)$, taking into account the arcs modeling the cyclic constraints linking individual "copies" of the graph. Hence, the graph $H^{(m)}(\pi)$ was defined as follows:

$$H^{(m)}(\pi) = \left(\bigcup_{x=1}^{m}\{\mathscr{O}^{(x)}\} \cup \mathscr{O}', \bigcup_{x=1}^{m}\{\mathscr{R}^{(x)} \cup \mathscr{E}^{(x)}(\pi) \cup \mathscr{E}^{*(x)}(\pi)\}\right), \tag{36}$$

where $\mathscr{O}^{(x)}$ and $\mathscr{O}'$ are the sets of vertices, whereas $\mathscr{R}^{(x)}$, $\mathscr{E}^{(x)}(\pi)$, $\mathscr{E}^{*(x)}(\pi)$ are the sets of arcs. A vertex $j^{(x)} \in \mathscr{O}^{(x)}$ corresponds to S_j^x – the start of execution of the operation j in the x-th MPS. The set of vertices $\mathscr{O}'$ models the first operations of the $m+1$-th MPS and "end" moment of the m-fold duplication of the graph $G(\pi)$

$$\mathscr{O}' = \bigcup_{k=1}^{m}\{\pi_k^{(m+1)}(1)\}. \tag{37}$$

The arcs $(i^{(x)},j^{(x)})$ from the sets $\mathscr{R}^{(x)}$ and $\mathscr{E}^{(x)}(\pi)$ have weight p_i and represent precedence: the technological (inequality (4)) and the machine (inequality (5)) constraints in x-th MPS

$$\mathscr{R}^{(x)} = \bigcup_{j=1}^{n}\bigcup_{i=l_{j-1}+1}^{l_j-1}\{(i^{(x)},(i+1)^{(x)})\}, \tag{38}$$

$$\mathscr{E}^{(x)}(\pi) = \bigcup_{k=1}^{m}\bigcup_{i=1}^{m_k-1}\{(\pi_k^{(x)}(i),\pi_k^{(x)}(i+1))\}. \tag{39}$$

The arcs $(i^{(x)},j^{(y)})$ from the set $\mathscr{E}^{*(x)}(\pi), x = 0,1,\ldots,m$ represent the cyclic constraints between x-th and $x+1$-th MPS

$$\mathscr{E}^{*(x)}(\pi) = \bigcup_{k=1}^{m}\{(\pi_k^{(x)}(m_k),\pi_k^{(x+1)}(1))\} \tag{40}$$

and have weight equal to $p_i - T$.

Let $\mu(i^{(x)}j^{(y)})$ denote the path with the highest weight between the vertices $i^{(x)}$ and $j^{(y)}$. The weight of the path $\mu(i^{(x)}j^{(y)})$ is

$$L_{\mu(i^{(x)}j^{(y)})} = \sum_{k\in\mu(i^{(x)}j^{(y)})} p_k - T(y-x). \tag{41}$$

It is obvious that:

$$\bigcup_{\mu\in\mathscr{C}^m(H^m(\pi))}\{L_\mu(T(\pi))\} = \bigcup_{\mu\in\mathscr{C}'(H(\pi))}\{L_\mu(T(\pi))\}, \tag{42}$$

$$\mathscr{C}^m(H^m(\pi)) = \bigcup_{i\in\mathscr{M}}\bigcup_{y=2}^{m}\{\mu(\pi_i^{(1)}(1),\pi_i^{(y)}(1))\}. \tag{43}$$

From Property 6, the shortest cycle μ^* that satisfies the condition $L_{\mu^*}(T(\pi)) = 0$ is included in the set $\mathscr{C}'(H(\pi))$. Therefore, the task of determining the cycle μ^* in the graph $H(\pi)$ is equivalent to the problem of determining the shortest path with the largest weight from the set $\mathscr{C}^m(H^m(\pi))$.

Let us consider the value of the minimal cycle time for which the path $\mu(i^{(x)}j^{(y)})$ in the graph $H^m(\pi)$ has a weight less than or equal to zero (which corresponds to a cycle with non-positive weight in the graph $H(\pi)$)

$$T^{min}_{\mu(i^{(x)}j^{(y)})} = \frac{\sum_{k\in\mu(i^{(x)}j^{(y)})} p_k}{y-x}. \tag{44}$$

Knowing T^{min}_{μ} value for every path μ from the set $\mathscr{C}^m(H^m(\pi))$, it is possible to calculate the minimal cycle time

$$T(\pi) = \max_{\mu \in \mathscr{C}^m(H^m(\pi))} \{T_\mu^{min}\} = \max_{i \in \mathscr{M}} \left\{ \max_{y=2,\ldots,m} \left\{ \frac{\sum_{k \in \mu(\pi_i^{(1)}(1), \pi_i^{(y)}(1))} p_k}{y-1} \right\} \right\}. \tag{45}$$

In the further part of the paper, a complexity analysis of the objective function of determination algorithms for their implementations for the Parallel Random Access Machine (PRAM model) is done (see Cormen, Leiserson, Rivest, & Stein, 1990). PRAM consists of many cooperating processors – RAM machines, usually used in theoretical computer science in algorithms complexity analysis. Each processor can make local calculations – additions, subtractions, shifts, conditional and unconditional jumps and indirect addressing. All the processors are synchronized and have access to a shared global memory (uniformly accessible from any processor) in a constant time $O(1)$. CREW (Concurrent Read Exclusive Write) is a type of PRAM, where multiple processors can read from the same memory cell concurrently, but only a single processor can write the cell at the same time. This model resembles, for example, the GPU programming model and therefore is used in this paper to analyze the algorithms.

Theorem 1. *The value of the minimal cycle time in the cyclic job shop scheduling problem can be determined in $O(mo)$ time, using $O(m)$-processor CREW PRAM.*

Proof. In order to determine the cycles with the highest weights in the graph $H(\pi)$, the corresponding graph $H^m(\pi)$ is considered. The minimal cycle time $T(\pi)$ can be calculated using Eq. (45). The set $\mathscr{C}^m(H^m(\pi))$ can be decomposed into m disjoint subsets in such a manner, that for each subset there is a different common first operation in the path:

$$\mathscr{C}^m(H^m(\pi)) = \bigcup_{i \in \mathscr{M}} \{\mathscr{C}_i^m(H^m(\pi))\}, \tag{46}$$

$$\mathscr{C}_i^m(H^m(\pi)) = \bigcup_{y=2}^{m} \{\mu(\pi_i^{(1)}(1), \pi_i^{(y)}(1))\}. \tag{47}$$

Substituting (47) into (45) we have:

$$T(\pi) = \max_{i \in \mathscr{M}} \{\max\{T_\mu^{min} : \mu \in \mathscr{C}_i^m(H^m(\pi))\}\}. \tag{48}$$

The value of $\max\{T_\mu^{min} : \mu \in \mathscr{C}_i^m(H^m(\pi))\}$ can be calculated using a method based on a review of the graph nodes in the topological order. This method has the sequential complexity $O(mo)$. Therefore, obtaining the value of $T(\pi)$ has the sequential complexity $O(m^2o)$, resulting from the fact that it is necessary to perform the procedure of revision of the nodes in the topological order m times, each time starting from different source – node representing execution of the first operation on the machine. This procedure can be easily parallelized using m processors, resulting in the parallel operation time $O(mo)$. □

Example 4. Let us consider the cyclic job shop scheduling problem instance defined in Example 2 and the order of execution $\pi = ((1,5),(2),(4,3))$. The corresponding graph $H^m(\pi)$ is shown in Fig. 4a. In order to calculate $T(\pi)$, the first weights of the paths from the set $\mathscr{C}_1^m(H^m(\pi))$ are computed. In the topological order, the longest paths from the node $1^{(1)}$ to each other node are found as shown in Fig. 4b. Three different paths are taken into consideration: $\mu(1^{(1)},1^{(2)})$, $\mu(1^{(1)},1^{(3)})$ and $\mu(1^{(1)},1^{(4)})$. For each path, T^{min} is calculated. The path:

$$\mu(1^{(1)},1^{(3)}) = (1^{(1)},2^{(1)},3^{(1)},4^{(2)},5^{(2)},1^{(3)})$$

has the highest value of $T^{min}, T^{min}_{\mu(1^{(1)},1^{(3)})} = 4.5$. Using the same method, paths with the highest values of T^{min} are found for the rest of possible starting points – for the node $2^{(1)}$:

$$\mu(2^{(1)},2^{(3)}) = (2^{(1)},3^{(1)},4^{(2)},5^{(2)},1^{(3)},2^{(3)}), \quad T^{min}_{\mu(2^{(1)},2^{(3)})} = 4.5,$$

and for the node $4^{(1)}$:

$$\mu(4^{(1)},4^{(3)}) = (4^{(2)},5^{(2)},1^{(3)},2^{(3)},3^{(3)},4^{(3)}), \quad T^{min}_{\mu(4^{(1)},4^{(3)})} = 4.5.$$

Finally, $T(\pi) = \max\{4.5,4.5,4.5\} = 4.5$. Because $T^{min}_{\mu(1^{(1)},1^{(3)})} = T(\pi)$, the shortest cycle μ^* that satisfies the condition $L_{\mu^*}(T(\pi)) = 0$ from $H(\pi)$ is $\mu^* = (1,2,3,4,5)$.

Any cycle that meets constraints form Property 6 will be called a *critical cycle*, whereas the elements of the cycle are called *critical operations*. It is easy to see that the increase in the processing time of any critical operation increases the cycle time (and the minimal cycle time).

Let μ be a critical cycle. Any subsequence $B_{a,b} = (\mu(a),\ldots c,\mu(b))$ of the cycle μ consisting of the vertices from the connected arcs from the set $\mathscr{E}(\pi)$ will be called an *operation block*, operations $\mu(a)$ and $\mu(b)$ will be called respectively the first and the last operation of the block $B_{a,b}$, whereas $\mu(a+1),\ldots,\mu(b-1)$ will be called internal operations of the block $B_{a,b}$.

Property 8. *If $\alpha,\pi \in \Pi$ and $T(\alpha) < T(\pi)$, then in the permutation α at least one operation of at least one block from permutations π is executed before the first or the last operation of the block.*

Proof of the Property 8 is carried out in a manner analogous to the proof of the block property for the job shop scheduling problem with the makespan minimization criterion C_{max} (see Nowicki & Smutnicki, 1996; Grabowski & Wodecki, 2005).

Property 9. *If the permutation α was generated from π by changing the order of execution of two neighboring operations in the critical cycle, then α is a feasible permutation.*

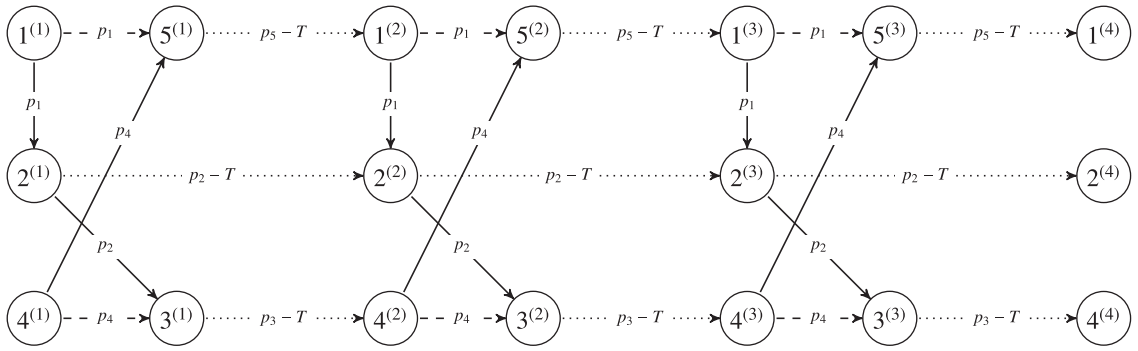
4. Tabu Search

Tabu Search (TS) was originally proposed by Glover and Laguna (1997). TS is a modification of the local search method. To improve the chance of reaching a global extreme, Glover and Laguna added the possibility to increase the value of the objective function in consecutive algorithm iterations. In order not only to avoid visiting the same solution multiple times, but also to explore more promising regions of search space and to enable the exit from the local extreme, there is a tabu mechanism introduced. After generating a neighborhood, the solutions that are forbidden by the tabu mechanism are not considered, unless they meet the aspiration criteria, i.e. the conditions under which the tabu constraints may be omitted.

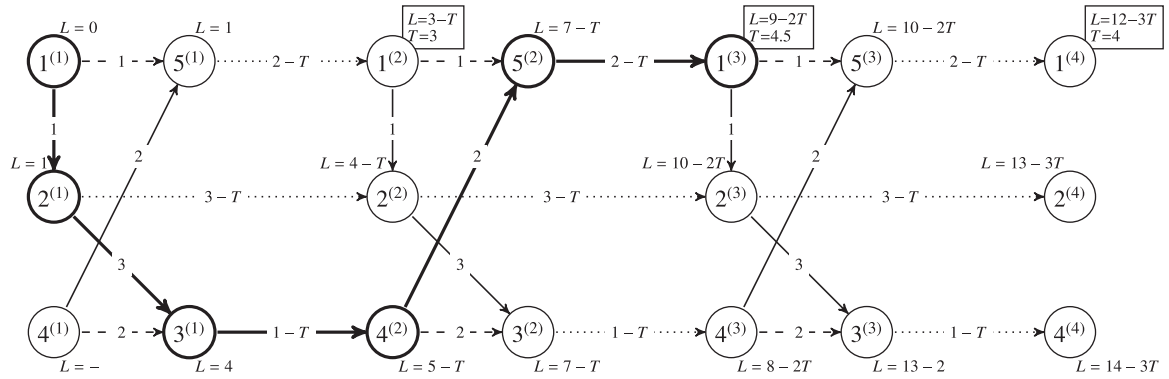
4.1. Neighborhood

A neighborhood is a set of solutions (neighbors) created by a specific operator; a move is a function transforming a solution into one of its neighbors. In this paper, a swap move is used, i.e. the move that swaps positions of two given operations in the order of their execution. The attributes of the move are described by an unordered pair of swapped operations $\{i,j\}$, $v_i = v_j = m$ (on the same machine). In the algorithm, the strategy of choosing the neighbor with the lowest value of the objective function was adopted.

First, the neighborhood N_1, generated by the operator proposed in the work Nowicki and Smutnicki (1996), was considered. The results of computational experiments prove its great effectiveness in the design of the algorithms for solving the job shop scheduling problem with $C_{\max}$ criterion. The elements of the neighborhood are generated by swapping (see Bożejko, Pempera, & Smutnicki, 2013) the first operation of each block with the second operation of the block and the last operation of each block with the penultimate operation. One can easily notice that the generated solutions fulfill constraints from Properties 8 and 9. Therefore, the received solutions are not only feasible but their minimal cycle times are potentially lower. The experimental studies in Brucker and Kampmeyer (2005) examined several neighborhoods. The described neighborhood yielded the best results, hence it was decided to use N_1 as the neighborhood for the sequential algorithm implementation.



(a) Graph $H^m(\pi)$



(b) Graph $H^m(\pi)$ used in the calculating paths with the highest weights (denoted as L) form the node $1^{(1)}$. The critical cycle is marked with bold lines.

Fig. 4. Graphs for the Example 4. The technological arcs from the sets $\mathscr{R}^{(x)}$ are denoted with solid lines, the machine arcs from the sets $\mathscr{E}^{(x)}(\pi)$ with dashed lines and the cyclic arcs from the sets $\mathscr{E}^{(x)*}(\pi)$ with dotted lines.

Since the parallel environment provides more computing power, a larger neighborhood N_2 was also defined. N_2 extends the neighborhood N_1 by swapping each internal operation of the block, with the first and the last operation of this block. With the larger size of neighborhood N_2, the likelihood of a situation when all the moves are forbidden by the tabu list is reduced. However, some solutions obtained from the neighborhood N_2 do not fulfill conditions from Properties 9 and 8. The process of generating the neighborhood of a solution π is summarized in the Procedure 1.

Procedure 1. Neighborhood generation

Step 1.	Generate the graph $H(\pi)$.
Step 2.	Determine the critical cycle and blocks of operations.
Step 3.	Generate neighborhood N_1 or N_2 by swapping the operations from the blocks.

4.2. Tabu mechanism

A tabu mechanism is implemented by using a tabu list (LT), which is a short-term search history. LT stores the attributes of the moves which were made. When the length of the list reaches the maximum fixed value L_{max}, before adding the next element, the oldest one is deleted. When, in a given step, all the moves are forbidden by the tabu list, the oldest elements are being deleted from the list, until at least one move is allowed. In addition, the aspiration criterion is introduced: each move that leads to the neighbor with the value of the objective function less than the objective function of the current best solution is allowed, even if it is forbidden by the tabu list.

4.3. Start procedure

In the paper, there are two different TS start procedures used. The first one, later referred to as *simple start procedure*, is designed only to provide a feasible solution. Therefore, the quality of the initial solution is not taken into a consideration. The procedure is summarized in Procedure 2.

Procedure 2. Simple start procedure

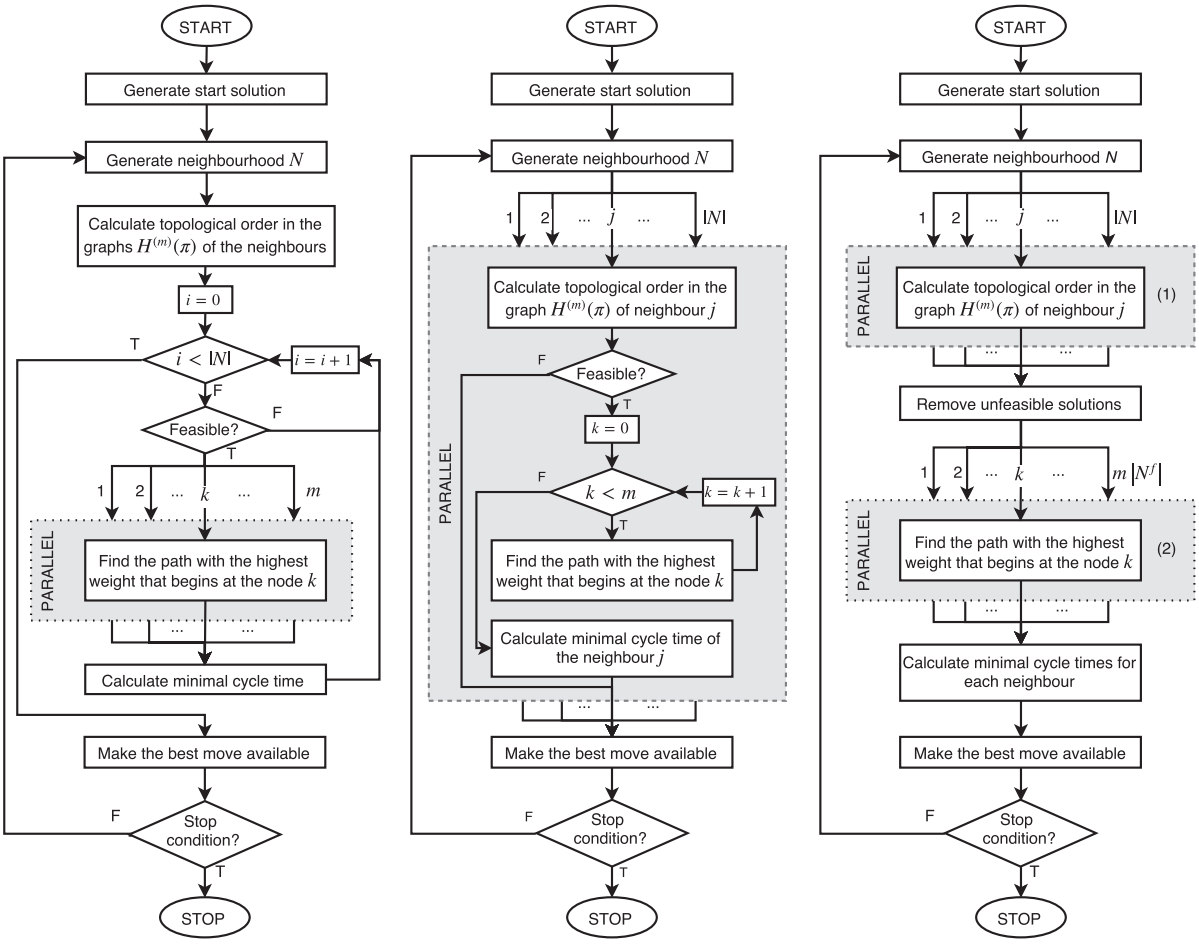
```
for i = 1,2,...,m do
    π_i ← ()
for j = 1,2,...,n do
    for i = l_{j-1} + 1, l_{j-1} + 2,...,l_{j-1} + n_j do
        append operation i to the end of π_{v_i}
```

The procedure starts with an empty order of execution π. Then, each operation (from 1 to o) is being put at the end of π_{v_i}, so that technological constraints are satisfied.

The second procedure is the TSAB algorithm proposed in Nowicki and Smutnicki (1996). In order to obtain the initial solution, an instance of the problem is treated as it were the job shop scheduling problem instance.

5. Parallelization

The most time-consuming part of a single TS iteration is calculating the value of the objective function for each neighbor. Therefore, this part of the algorithm was chosen for parallelization. Three different



(a) OF strategy (b) N strategy (c) $OF + N$ strategy

Fig. 5. Overview of the algorithm parallelization strategies.

approaches were considered: the parallel method of the objective function calculation, the parallel evaluation of the neighbors and the approach combining the two mentioned before. An outline of the TS parallelization strategies is shown in Fig. 5.

5.1. Objective function parallelization (OF strategy)

The strategy is a direct implementation of the Theorem 1 and it is shown in Fig. 5a. First, in a sequential manner, the neighbors are generated and the topological orders are determined. Then, the feasible neighbors are evaluated. For one neighbor at a time (in Fig. 5a – neighbors $i = 0,1,...,|N|-1$), the paths with the highest weights, starting from m different nodes, are computed simultaneously. Hence, an upper bound of the speedup can be expressed by the equation

$$UB_{OF}(p,m) = \frac{m}{\left\lceil \frac{m}{p} \right\rceil}. \tag{49}$$

The theoretical speedup is therefore affected by the number of machines m and the number of processors p.

5.2. Neighborhood parallelization (N strategy)

The second strategy revolves around evaluating the neighbors simultaneously. In each iteration of the TS algorithm, a set of solutions (neighbors) is generated. Then, for each neighbor at once, the value of the objective function is computed sequentially; as shown in Fig. 5b. For a neighbor, first the topological order is determined. If there are cycles in the graph $H^{(m)}(\pi)$, a neighbor is rejected. Next, paths with the highest weights starting from m different nodes are found (in Fig. 5b – paths $k = 0,1,...,m-1$). The minimal cycle time is finally obtained by analyzing the weights of the paths. For obvious reasons, the speedup of the parallel algorithm cannot exceed $|N(\pi)|$, with $p = |N(\pi)|$ processors used. Therefore, the upper bound of the speedup for this parallelization strategy is

$$UB_{N}(p,\pi) = \frac{|N(\pi)|}{\left\lceil \frac{|N(\pi)|}{p} \right\rceil}. \tag{50}$$

The degree of parallelization depends on the sizes of the generated neighborhoods and therefore on: the neighborhood definition, the problem instance, the order of execution π and the number of processors p.

5.3. Mixed parallelization (OF + N strategy)

The last strategy combines both previously described approaches, allowing for the highest theoretical speedups.

As shown in Fig. 5c, there are two parallel sections. In the first Section (1), the topological orders of the graphs $H^{(m)}(\pi)$ for each neighbor are simultaneously computed. Then, the neighbors with cycles in graphs $H^{(m)}(\pi)$ are rejected. As a result, a set of feasible neighbors N^f is formed. In the second Section (2), the paths with the highest weights are determined. There are m starting points for each neighbor, and $|N^f(\pi)|$ neighbors. Since the calculations are independent, $m|N^f(\pi)|$ paths are computed at once. Hence, the upper bound of the speedup is

$$UB_{OF+N}(p,m,\pi) = \max\left\{\frac{m}{\left\lceil \frac{m}{p} \right\rceil}, \frac{m\ |N^f(\pi)|}{\left\lceil \frac{m\ |N^f(\pi)|}{p} \right\rceil}\right\}. \quad (51)$$

A pool of the path finding tasks (Section (2)), shared between neighbors, is preferred over the nested parallelism (a straightforward combination of *N* and *OF* strategies), due to the implementation reasons. However, both approaches have the same theoretical performance.

5.4. Implementation

Due to hardware architecture on which the experiments were conducted, the chosen implementation technology was OpenMP. It enables creation of multi-platform applications for multiprocessor systems with shared memory.

As a method of parallelization, the **#pragma omp parallel for** clause was used. This kind of loop can be executed simultaneously by multiple threads. OpenMP uses the fork-join model for a parallel execution. When a thread encounters a parallel block of code, it creates a group of one or more threads (inducing the original thread, called a *master thread*). After finishing the work within a parallel block, the *slave threads* are released and the *master thread* continues execution. Each gray *PARALLEL* block on Fig. 5 corresponds to the one **#pragma omp parallel for** loop.

One of the main strengths of the Xeon Phi accelerator is SIMD (Single Instruction, Multiple Data) parallelization, namely vectorization. However, Xeon Phi was chosen as a platform of an implementation exclusively because it resembles PRAM, where the concept of vectorization is not present in a strict sense. Therefore, no special effort was undertaken to fine-tune the algorithm to the hardware. Nevertheless, 18 different loops were (partially) vectorized by the compiler. The following fragments of the code were vectorized:

(a) loading the data from the instances;
(b) initialization of various data structures;
(c) finding the path with the highest weight from *m* different starting points (objective function calculation);
(d) neighborhood generation.

Most notably, the bottleneck of the algorithm was vectorized (fragment c).

6. Computational experiments

In this section, computational experiments to implement and evaluate the proposed methods are presented. First, the test setup, including the selection of the benchmarks and the target machine is outlined. Then, the quality of the results obtained by the algorithm is briefly discussed. Finally, the implementation of the algorithm, with different parallelization strategies used, is tested on test (benchmark) instances.

6.1. Experiment setup

The test data were taken from the OR-library, http://people.brunel.ac.uk/mastjjb/jeb/info.html (online; accessed 10-June-2017). Since there are no benchmarks dedicated to the variant of the cyclic problem researched in this paper, instances of the job shop scheduling problem were chosen – proposed in Fisher and Thompson (1963) FT06, FT10, FT20; and the instances LA01–LA40 from Lawrence (1984). The benchmarks have been already used before in testing scheduling algorithms for cyclic problems e.g. in Brucker and Kampmeyer (2005), Kampmeyer (2006), and Kechadi et al. (2013). The instances were adapted to the cyclic scheduling problem, by assuming that they define single MPSs and then grouped according to their parameters.

The tests were carried out on a workstation with an Intel(R) Core (TM) i7-3820 CPU @ 3.60 GHz, 4 GB RAM and Intel(R) Xeon Phi with 57 cores (4 threads each, one core is usually reserved for the system, leaving 56 usable cores) and 5952 Mb GDDR @ 2.5Ghz. The code was compiled with icpc (ICC) 14.0.4 20140805 compiler and run in a native mode (the program was executed entirely on the coprocessor).

6.2. Effectiveness of the proposed algorithm

Although the quality of the solutions provided by the heuristic is not the main goal of this research, it was necessary to check if the algorithm is able to provide *sufficiently good* results. For each instance from the test set, the sequential variant of the proposed algorithm was launched with the time limit of 30 s. The initial solutions were generated by TSAB algorithm. For each instance, the following parameters were calculated:

- T – the minimal cycle time of the best obtained solution.
- LB – lower bound of the cycle time.
- $Gap = \frac{T-LB}{LB}\cdot 100\%$ – the relative gap between the obtained cycle time T and the lower bound.
- $Gap' = \frac{T-T^{Best}}{T^{Best}}\cdot 100\%$ – the relative gap between the obtained cycle time T and the best known value of the cycle time.

The value of the lower bound was computed based on Property 1, from the equation

$$LB = \max_{k\in M} \sum_{i\in O_k} p_k. \quad (52)$$

Additionally, we show the efficiency of some existing heuristics for cyclic job shop scheduling problem. Two heuristics are compared: the tabu search proposed in Brucker and Kampmeyer (2005) (TS-BK) and one of the most recent and advanced metaheuristics, based on the recurrent neural network LRNN, proposed in Kechadi et al. (2013).

As shown in Table 3, the algorithm proposed in this paper provided the results better or equally good as TS-BK for all the tested instances. 27 solutions with the cycle time equal to the lower bound (marked with a star) were obtained. In the remaining 16 cases, the *Gap* to the lower bound varies from 1.4% to 28.5% with the average of 4.0%. For TS-BK algorithm, the *Gap* varies from 0% to 33%, in average 4.9%. The superiority of PTS over TS-BK is particularly evident when analyzing the values of Gap'. The proposed algorithm PTS has found the best solutions for all the instances (TS-BK has found worse ones for 14 instances). Comparing the results for the cyclic job shop scheduling problem (without machine chains), it is worth noting that TS-BK algorithm generated better solutions than LRNN. The value of the *Gap* for LRNN varies from 0.0% to 8.4%, in average 1.7%; whereas for TS-BK *Gap* changes from 0.0% to 6.2%, on average 1.2%.

6.3. Evaluation of the parallelization strategies

The proposed PTS is the best known sequential algorithm for the problem, therefore the orthodox speedup was measured in the experiments (i.e. the computation times of the algorithm on 1 and p processors were compared). The upper bounds of the speedups for different number of threads (including a theoretical, infinite number of threads) were calculated as described in Section 5. Considering the maximum number of available threads was 224 (56 cores), experiments for: 1, 2, 4, 8, 16, 32, 56, 64, 96, 128, 160, 192 and 224 threads used were conducted. Three different parallelization strategies were tested: the objective function parallelization, the neighborhood parallelization (for two different neighborhoods) and the mixed parallelization (for two different neighborhoods). Since the PTS is a deterministic algorithm, the tests consisted of a single algorithm run with 3200 iterations for each experiment setup. Initial solutions were obtained by the *simple start procedure*, as the quality of results were not measured.

The detailed results of the numerical experiments are shown in

Table 3
Performance analysis LRRNN, TS-K and PTS approaches.

	Cyclic Job Shop with machine chains							Cyclic Job Shop				
		TS-BK			PTS				TS-BK		LRNN	
Name	*LB*	*T*	Gap	Gap′	*T*	Gap	Gap′	*LB*	*T*	Gap	*T*	Gap
LA01	666	666*	0.0	0.0	666*	0.0	0.0	666	666*	0.0	666*	0.0
LA02	635	635*	0.0	0.0	635*	0.0	0.0	655	655*	0.0	655*	0.0
LA03	588	588*	0.0	0.0	588*	0.0	0.0	597	603	1.0	597*	0.0
LA04	537	556*	3.5	0.0	556	3.5	0.0	590	590*	0.0	590*	0.0
LA05	593	593*	0.0	0.0	593*	0.0	0.0	593	593*	0.0	593*	0.0
LA06	926	926*	0.0	0.0	926*	0.0	0.0	926	926*	0.0	926*	0.0
LA07	869	869*	0.0	0.0	869*	0.0	0.0	890	890*	0.0	890*	0.0
LA08	863	863*	0.0	0.0	863*	0.0	0.0	863	863*	0.0	863*	0.0
LA09	951	951*	0.0	0.0	951*	0.0	0.0	951	951*	0.0	951*	0.0
LA10	958	958*	0.0	0.0	958*	0.0	0.0	958	958*	0.0	958*	0.0
LA11	1222	1222*	0.0	0.0	1222*	0.0	0.0	1222	1222*	0.0	1222*	0.0
LA12	1039	1039*	0.0	0.0	1039*	0.0	0.0	1039	1039*	0.0	1039*	0.0
LA13	1150	1150*	0.0	0.0	1150*	0.0	0.0	1150	1150*	0.0	1150*	0.0
LA14	1292	1292*	0.0	0.0	1292*	0.0	0.0	1292	1292*	0.0	1292*	0.0
LA15	1207	1207*	0.0	0.0	1207*	0.0	0.0	1207	1207*	0.0	1207*	0.0
LA16	660	781	18.3	0.1	780	18.2	0.0	945	962	1.8	957	1.3
LA17	683	713	4.4	1.4	703	2.9	0.0	784	785	0.1	784*	0.0
LA18	623	768	23.3	0.7	763	22.5	0.0	848	861	1.5	848*	0.0
LA19	685	794	15.9	1.4	783	14.3	0.0	842	852	1.2	842*	0.0
LA20	744	769	3.4	0.0	769	3.4	0.0	902	902*	0.0	908	0.7
LA21	935	963	3.0	1.5	949	1.5	0.0	1046	1070	2.3	1074	2.7
LA22	830	875	5.4	1.6	861	3.7	0.0	927	960	3.6	932	0.5
LA23	1032	1032*	0.0	0.0	1032*	0.0	0.0	1032	1032*	0.0	1054	2.1
LA24	857	924.5	7.9	1.9	907	5.8	0.0	935	955	2.1	944	1.0
LA25	864	902	4.4	3.0	876	1.4	0.0	977	996	1.9	984	0.7
LA26	1218	1218*	0.0	0.0	1218*	0.0	0.0	1218	1218*	0.0	1224	0.5
LA27	1188	1197	0.8	0.8	1188*	0.0	0.0	1235	1293	4.7	1249	1.1
LA28	1216	1216*	0.0	0.0	1216*	0.0	0.0	1216	1242	2.1	1235	1.6
LA29	1105	1105*	0.0	0.0	1105*	0.0	0.0	1152	1212	5.2	1249	8.4
LA30	1335	1355	1.5	1.5	1335*	0.0	0.0	1355	1355*	0.0	1355*	0.0
LA31	1784	1784*	0.0	0.0	1784*	0.0	0.0	1784	1784*	0.0	1855	4.0
LA32	1850	1850*	0.0	0.0	1850*	0.0	0.0	1850	1850*	0.0	1947	5.2
LA33	1719	1719*	0.0	0.0	1719*	0.0	0.0	1719	1719*	0.0	1840	7.0
LA34	1721	1721*	0.0	0.0	1721*	0.0	0.0	1721	1721*	0.0	1849	7.4
LA35	1888	1888*	0.0	0.0	1888*	0.0	0.0	1888	1888*	0.0	1994	5.6
LA36	1028	1204.5	17.2	3.9	1159	12.7	0.0	1268	1305	2.9	1299	2.4
LA37	980	1326	35.3	5.2	1260	28.6	0.0	1397	1483	6.2	1405	0.6
LA38	876	1157.5	32.1	5.4	1098	25.3	0.0	1196	1267	5.9	1257	5.1
LA39	1012	1172	15.8	2.3	1146	13.2	0.0	1233	1274	3.3	1302	5.6
LA40	1027				1146	11.6		1222			1320	8.0
ft06	43				46	7.0		55			55	0.0
ft10	631				631*	0.0		930			930	0.0
ft20	1119				1119*	0.0		1165			1165	0.0
Average			4.9	0.8		4.0	0.0			1.2		1.7

Table 4. Besides the empirical speedups, the speedups upper bounds for different number of threads and parallelization strategies are presented. The results for 64, 96, 128, 160 and 190 threads are omitted from the table, as Xeon Phi provides only 56 physical cores. The speedups for each tested number of threads are presented in Fig. 6.

The results of the experiment on the objective function parallelization strategy (*OF strategy*) are shown in Table 4 and in Fig. 6e. Since the upper bound of the speedup depends on the number of machines m, three groups of instance are clearly visible in the figure ($m \leqslant 6$,$m = 10$ and $m = 15$). As shown in Table 4, for small instance sizes, the obtained speedups are more than two times lower than the upper bound of the speedup. With the increase in the problem size, the empirical speedup also increases (even within the groups of the instances with the same number of machines). For the instance groups: 20×10, 30×10 and 15×15, the empirical speedups are less than two times lower than the upper bounds. As it was expected, no significant further speedup increase for the number of threads greater than the number of machines was observed. What is more, increasing the number of threads above the number of physical cores caused a constant decrease in the measured speedup (as shown in Fig. 6e).

The neighborhood parallelization (*N strategy*) was the second strategy to be tested. The results are presented in Table 4 and in Figs. 6a and b for the two tested neighborhoods N_1 and N_2 respectively. The mean size of the neighborhood N_2 was larger than N_1 (as can it be seen in column $t = \infty$), allowing for the higher speedups (up to 17 times). The upper bound of the speedup again proved to be an effective tool for estimating the number of threads required to maximize the speedup. Also in this experiment, increasing the number of threads beyond the number of physical cores caused a constant decrease in the measured speedup (as shown in Figs. 6a and b). This is particularly noticeable in

Table 4
Speedups obtained during the computational experiments for various parallelization strategies, for different number of threads t used.

Strat.	Group	Speedup							Speedup upper bound							
	$n \times m$	$t = 2$	$t = 4$	$t = 8$	$t = 16$	$t = 32$	$t = 56$	$t = 224$	$t = 2$	$t = 4$	$t = 8$	$t = 16$	$t = 32$	$t = 56$	$t = 224$	$t = \infty$[a]
OF	6 × 6	1.40	1.66	1.78	1.77	**1.84**	1.67	1.35	2.00	3.00	**6.00**	**6.00**	**6.00**	**6.00**	**6.00**	**6.00**
	10 × 5	1.29	1.54	1.54	1.57	**1.62**	1.49	1.17	1.67	2.50	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**
	15 × 5	1.31	1.58	1.62	1.67	**1.72**	1.61	1.25	1.67	2.50	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**
	20 × 5	1.33	1.62	1.69	1.77	**1.84**	1.76	1.34	1.67	2.50	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**	**5.00**
	10 × 10	1.69	2.42	2.63	3.83	**4.17**	4.04	3.05	2.00	3.33	5.00	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	15 × 10	1.78	2.62	2.88	4.41	**4.85**	4.75	3.53	2.00	3.33	5.00	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	20 × 10	1.80	2.47	3.19	4.52	**5.05**	4.99	3.67	2.00	3.33	5.00	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	30 × 10	1.81	2.31	3.60	4.80	**5.24**	5.19	3.81	2.00	3.33	5.00	**10.00**	**10.00**	**10.00**	**10.00**	**10.00**
	15 × 15	1.80	2.87	4.97	**7.99**	**7.99**	7.91	5.90	1.88	3.75	7.50	**15.00**	**15.00**	**15.00**	**15.00**	**15.00**
N_1	6 × 6	1.29	1.48	**1.50**	1.49	1.49	1.47	1.09	1.67	2.48	**2.63**	**2.63**	**2.63**	**2.63**	**2.63**	**2.63**
	10 × 5	1.31	1.57	**1.65**	1.64	1.63	1.56	1.19	1.77	2.90	3.46	**3.52**	**3.52**	**3.52**	**3.52**	**3.52**
	15 × 5	1.24	1.42	**1.47**	**1.47**	**1.47**	1.43	1.07	1.87	2.72	3.03	**3.17**	**3.17**	**3.17**	**3.17**	**3.17**
	20 × 5	1.25	1.43	1.60	**1.62**	**1.62**	1.58	1.18	1.90	2.37	3.09	3.37	**3.43**	**3.43**	**3.43**	**3.43**
	10 × 10	1.41	1.81	2.18	**2.24**	**2.24**	2.16	1.65	1.91	2.74	4.07	4.48	**4.50**	**4.50**	**4.50**	**4.50**
	15 × 10	1.56	2.14	2.93	3.21	**3.22**	3.07	2.38	1.87	3.18	5.30	6.61	**6.65**	**6.65**	**6.65**	**6.65**
	20 × 10	1.65	2.44	3.44	4.26	**4.30**	4.11	3.20	1.89	3.43	5.99	8.68	**8.98**	**8.98**	**8.98**	**8.98**
	30 × 10	1.74	2.54	3.89	5.63	6.12	**6.26**	4.65	1.92	3.58	6.30	10.97	**13.23**	**13.23**	**13.23**	**13.23**
	15 × 15	1.67	2.59	3.55	4.61	4.54	**4.80**	3.56	1.90	3.44	6.00	9.29	**10.00**	**10.00**	**10.00**	**10.00**
N_2	6 × 6	1.11	1.02	**1.28**	**1.28**	**1.28**	1.27	0.92	1.54	2.99	3.34	3.50	**3.51**	**3.51**	**3.51**	**3.51**
	10 × 5	1.31	1.55	2.19	2.47	**2.52**	2.49	1.85	1.65	3.01	4.50	6.86	7.18	**7.19**	**7.19**	**7.19**
	15 × 5	1.51	2.05	3.23	4.02	**4.55**	4.54	3.38	1.82	3.40	5.84	9.48	13.03	**13.28**	**13.28**	**13.28**
	20 × 5	1.64	2.27	3.99	5.09	6.54	**6.57**	4.91	1.90	3.59	6.55	11.08	17.96	18.40	**18.43**	**18.43**
	10 × 10	1.60	2.28	3.47	4.14	4.64	**4.66**	3.42	1.89	3.47	5.91	9.33	11.09	**11.18**	**11.18**	**11.18**
	15 × 10	1.64	2.60	3.81	4.80	5.75	**5.78**	4.23	1.88	3.53	6.08	10.39	13.31	**13.62**	**13.62**	**13.62**
	20 × 10	1.59	2.67	3.83	5.47	7.36	**7.48**	5.51	1.94	3.67	6.61	11.49	18.38	19.18	**19.45**	**19.45**
	30 × 10	1.71	2.85	4.29	7.22	9.74	**11.45**	8.58	1.96	3.79	7.14	12.90	22.30	30.53	**30.82**	**30.82**
	15 × 15	1.60	2.71	3.61	5.69	6.68	**7.05**	5.13	1.89	3.59	6.28	10.74	15.54	16.84	**16.90**	**16.90**
$N_1 + OF$	6 × 6	1.52	2.12	2.38	2.85	2.76	**2.86**	2.18	1.99	3.80	6.09	10.83	13.52	**13.66**	**13.66**	**13.66**
	10 × 5	1.38	2.03	2.31	**2.74**	2.73	2.52	2.13	1.87	3.32	6.01	10.23	12.64	**13.19**	**13.19**	**13.19**
	15 × 5	1.30	1.96	2.32	**2.69**	2.63	2.48	2.08	1.81	3.05	5.62	8.69	9.69	10.39	**10.55**	**10.55**
	20 × 5	1.33	2.04	2.53	**3.03**	3.01	2.91	2.41	1.76	2.85	5.41	7.25	9.79	10.94	**11.54**	**11.54**
	10 × 10	1.60	2.74	3.89	6.50	7.86	**8.47**	7.57	1.98	3.59	6.32	11.69	17.75	24.56	**31.61**	**31.61**
	15 × 10	1.66	3.20	5.55	9.06	11.95	**13.68**	13.14	2.00	3.90	7.42	13.73	24.78	36.56	**57.96**	**57.96**
	20 × 10	1.81	3.37	5.39	10.15	14.15	16.58	**16.94**	2.00	3.94	7.66	14.49	26.46	40.75	**76.32**	**76.32**
	30 × 10	1.84	3.52	5.68	11.06	17.11	21.43	**24.03**	2.00	3.96	7.76	14.97	27.70	44.65	109.33	**113.63**
	15 × 15	1.68	3.09	6.07	11.23	18.12	23.90	**27.21**	1.99	3.94	7.72	14.98	27.56	43.86	111.97	**114.92**
$N_2 + OF$	6 × 6	1.27	1.61	2.07	2.35	**2.44**	2.18	1.84	1.98	3.12	6.13	7.12	8.74	10.17	**10.29**	**10.29**
	10 × 5	1.39	2.11	2.87	3.78	**3.88**	3.74	3.28	1.80	3.20	6.15	9.64	15.18	22.28	**23.73**	**23.73**
	15 × 5	1.57	2.36	3.83	5.89	**6.98**	6.82	6.65	1.91	3.60	6.86	12.16	20.51	31.36	49.04	**49.20**
	20 × 5	1.69	2.67	4.56	7.34	9.83	**10.17**	**10.17**	1.95	3.83	7.43	14.03	25.15	39.70	**73.01**	**73.01**
	10 × 10	1.72	2.88	5.00	8.49	12.10	14.34	**14.46**	1.98	3.89	7.54	14.24	25.74	40.26	78.87	**79.43**
	15 × 10	1.75	2.83	5.78	9.40	14.99	18.46	**21.02**	2.00	3.89	7.47	14.47	26.14	41.78	110.03	**113.46**
	20 × 10	1.73	2.94	5.73	9.79	16.69	22.04	**25.95**	2.00	3.97	7.81	15.20	28.74	46.28	138.33	**151.35**
	30 × 10	1.74	3.17	5.93	10.93	18.28	26.77	**34.14**	2.00	3.98	7.88	15.47	29.88	49.89	154.45	**242.24**
	15 × 15	1.76	3.08	6.04	10.93	19.68	27.40	**33.89**	1.99	3.90	7.56	14.79	27.45	44.91	135.65	**193.20**

The highest values of empirical speedups and upper bounds for each group and the strategy are marked in bold font.
[a] Upper bound calculated for infinite number of threads.

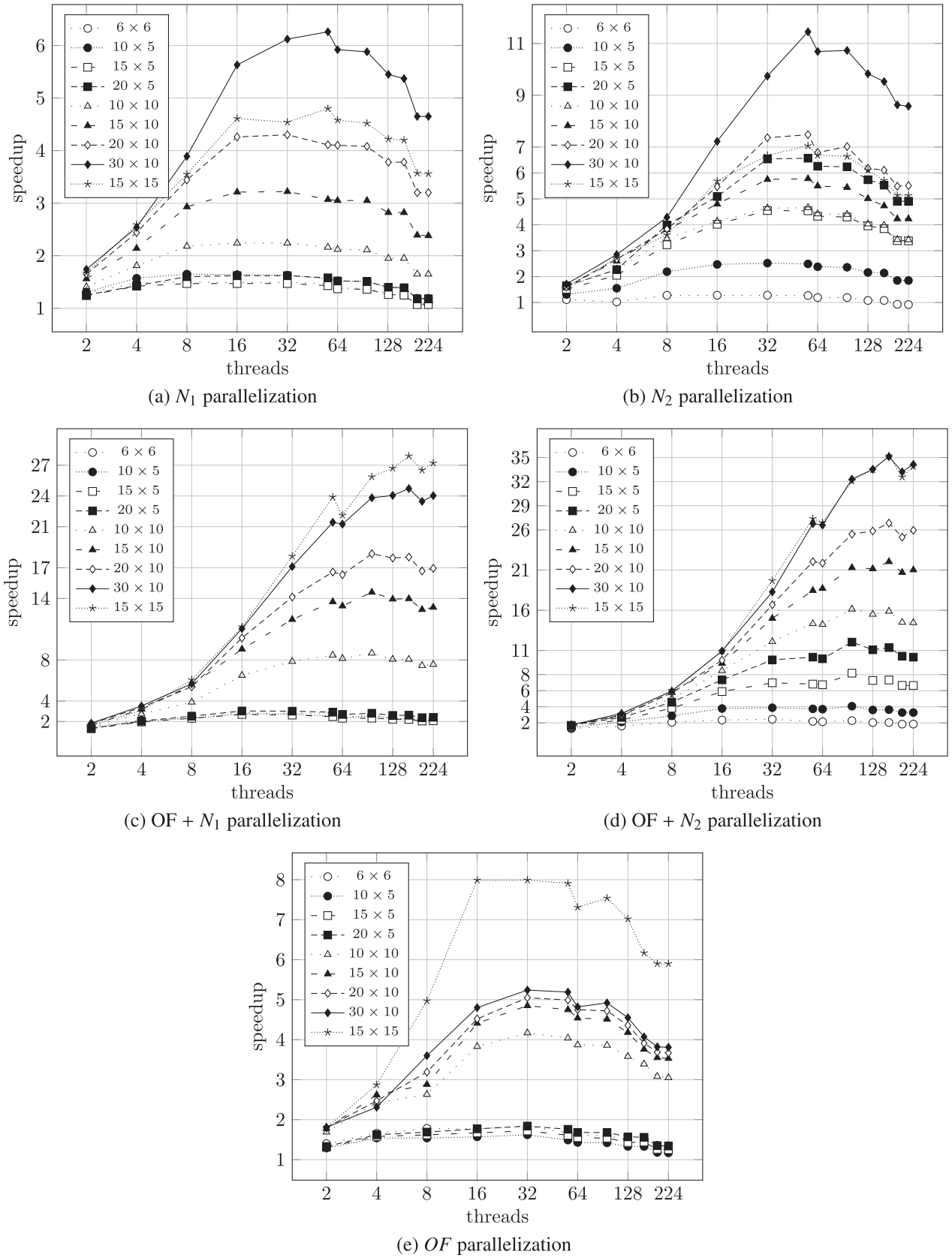
the significant speedup drop between 56 (the number of physical processors) and 64 threads.

The last test was conducted for the hybrid $OF + N$ parallelization strategy. The results for the neighborhoods N_1 and N_2 are shown in Figs. 6c (neighborhood N_1) and 6d (neighborhood N_2) and in Table 4. In this experiment, the highest speedups were achieved. The calculated speedups upper bounds indicate that the limiting factor was the number of threads. For N_2, even 224 threads would be insufficient. Such high computational power requirements resulted in the speedup increase for the number of threads above 56 (Fig. 6c and d).

7. Conclusions

The paper presents a parallel tabu search algorithm for the cyclic job shop scheduling problem. The objective function, as well as the neighborhood evaluation parallelization were designed taking into consideration the architecture of Intel Xeon Phi coprocessor. The numerical experiments were carried out on the literature test data and fully confirm the results of the theoretical research. For a given instance of the problem, the proposed speedup upper bounds can be used to determine how many processors (or threads) should be assigned to the algorithm.

In the further research, we would like to extend our considerations on the flexible job shop scheduling problem with parallel machines and setup times, as well as and no-wait or no-store and no-idle constrains.



(a) N_1 parallelization

(b) N_2 parallelization

(c) OF + N_1 parallelization

(d) OF + N_2 parallelization

(e) *OF* parallelization

Fig. 6. Speedups obtained from computational experiments for different parallelization strategies.

References

Bożejko, W., Gnatowski, A., Niżyński, T., & Wodecki, M. (2016). Tabu Search algorithm with neural tabu mechanism for the cyclic job shop problem. In *Artificial intelligence and soft computing. Lecture notes in computer science* (Vol. 9693, pp. 409–418).

Bożejko, W. (2012). On single-walk parallelization of the job shop problem solving algorithms. *Computers & Operations Research, 39*, 2258–2264.

Bożejko, W., Gnatowski, A., Idzikowski, R., & Wodecki, M. (2017). Cyclic flow shop scheduling problem with two-machine cells. *Archives of Control Sciences, 27*(2), 151–168.

Bożejko, W., Pempera, J., & Smutnicki, C. (2013). Parallel Tabu Search algorithm for the hybrid flow shop problem. *Computers & Industrial Engineering, 65*, 466–474.

Bożejko, W., Uchroński, M., & Wodecki, M. (2016). Parallel metaheuristics for the cyclic flow shop scheduling problem. *Computers & Industrial Engineering, 95*, 156–163.

Brucker, P., & Kampmeyer, T. (2005). Tabu Search algorithms for cyclic machine scheduling. *Journal of Scheduling, 8*, 303–322.

Brucker, P., & Kampmeyer, T. (2008a). Cyclic job shop scheduling problems with blocking. *Annals of Operations Research, 159*, 161–181.

Brucker, P., & Kampmeyer, T. (2008b). A general model for cyclic machine scheduling problems. *Discrete Applied Mathematics, 156*(13), 2561–2572.

Burke, E. K., De Causmaecker, P., Berghe, G. V., & Van Landeghem, H. (2004). The state of the art of nurse rostering. *Journal of Scheduling, 7*(6), 441–499.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., & Stein, C. (1990). *Introduction to algorithms.* MIT Press and McGraw-Hill.

Fisher, H., & Thompson, G. L. (1963). Probabilistic learning combinations of local job-shop scheduling rules. In J. F. Muth, & G. L. Thompson (Eds.). *Industrial scheduling* (pp. 225–251). Englewood Cliffs, NJ: Prentice-Hall.

Gertsbakh, I., & Serafini, P. (1991). Periodic transportation schedules with flexible departure times: An interactive approach based on the periodic event scheduling problem and the deficit function approach. *European Journal of Operational Research, 50*(3), 298–309.

Glover, F., & Laguna, M. (1997). *Tabu search.* Kluwer Academic Publishers July.

Grabowski, J., & Wodecki, M. (2005). A very fast tabu search algorithm for the job shop problem. In C. Rego, & B. Alidaee (Eds.). *Adaptive memory and evolution; tabu search and scatter search* (pp. 117–144). Dordrecht: Kluwer Academic Publishers.

Hall, N. G., Lee, T. E., & Posner, M. E. (2002). The complexity of cyclic shop scheduling problems. *Journal of Scheduling, 5*(4), 307–327.

Hanen, C. (1994). Study of a NP-hard cyclic scheduling problem: The recurrent job-shop. *European Journal of Operational Research, 72*, 82–101.

Hanen, C., & Munier, A. (1995). Cyclic scheduling on parallel processors: An overview. In P. Chretienne, E. G. CoffmanJr., J. K. Lenstra, & Z. Liu (Eds.). *Scheduling theory and its applications* (pp. 94–226). New York: Wiley Chapter 4.

Hitz, K. L. (1980). *Scheduling of flow shops II. Report no. LIDS-R-879.* MIT, Cambridge, Massachusetts: Laboratory for Information and Decision Systems.

Jalilvand-Nejad, A., & Fattahi, P. (2015). A mathematical model and genetic algorithm to cyclic flexible job shop scheduling problem. *Journal of Intelligent Manufacturing, 26*, 1085–1098.

Kampmeyer, T. (2006). *Cyclic scheduling problems.* Ph.D. DissertationUniversitat Osnabruck.

Kats, V., & Levner, E. (2003). Polynomial algorithms for periodic scheduling of tasks on parallel processors. In L. T. Yang, & M. Paprzycki (Vol. Eds.), *Practical applications of parallel computing: Advances in computation theory and practice: Vol. 12*, (pp. 363–370). Canada: Nova Science Publishers.

Kechadi, M., Low, K., & Goncalves, G. (2013). Recurrent neural network approach for cyclic job shop scheduling problem. *Journal of Manufacturing Systems, 32*(4), 689–699.

Kubale, M., & Nadolski, A. (2005). Chromatic scheduling in a cyclic open shop. *European Journal of Operational Research, 164*(3), 585–591.

Kubiak, W. (2005). Solution of the Liu¢Layland problem via bottleneck just-in-time sequencing. *Journal of Scheduling, 8*(4), 295–302.

Lawrence, S. (1984). *Resource constrained project scheduling: An experimental investigation of heuristic scheduling techniques, GSIA.* Pittsburgh, PA: Carnegie Mellon University.

Lee, J., & Korbaa, O. (2004). Modelling and scheduling of ratio-driven FMS using unfolding time Petri nets. *Computers & Industrial Engineering, 46*(4), 639–653.

Lee, T. E., & Posner, M. E. (1997). Performance measures and schedules in periodic job shops. *Operations Research, 45*(1), 72–91.

Méndez, C.a., Cerdá, J., Grossmann, I. E., Harjunkoski, I., & Fahl, M. (2006). State-of-the-art review of optimization methods for short-term scheduling of batch processes. *Computers and Chemical Engineering, 30*, 913–946.

Nowicki, E., & Smutnicki, C. (1996). A fast taboo search algorithm for the job shop problem. *Management Science, 42*(6), 797–813.

Pinedo, M. L. (2008). *Scheduling: Theory, algorithms and systems.* New York: Springer.

Pinedo, M. L. (2009). *Planning and scheduling in manufacturing and services* (2nd ed.). New York: Springer.

Pinto, T., Barbosa-Póvoa, A. P. F. D., & Novais, A. Q. (2005). Optimal design and retrofit of batch plants with a periodic mode of operation. *Computers and Chemical Engineering, 29*(6 Spec. iss.), 1293–1303.

Trautmann, N., & Schwindt, C. (2009). A cyclic approach to large-scale short-term planning in chemical batch production. *Journal of Scheduling, 12*(6), 595–606.