10.1515/acsc-2017-0010

Archives of Control Sciences
Volume 27(LXIII), 2017
No. 2, *pages 169–181*

A fine-grained parallel algorithm for the cyclic flexible job shop problem

WOJCIECH BOŻEJKO, JAROSŁAW PEMPERA and MIECZYSŁAW WODECKI

In this paper there is considered a flexible job shop problem of operations scheduling. The new, very fast method of determination of cycle time is presented. In the design of heuristic algorithm there was the neighborhood inspired by the game of golf applied. Lower bound of the criterion function was used in the search of the neighborhood.

Key words: job shop, cyclic scheduling, parallel algorithm.

1. Introduction

Universal globalization, and thus increasing competition forces lowering of the cost of production, which can be achieved through mass production. In practice, production, in which a set of products is manufactured in large quantities, is carried out in a cyclic manner. It is a very effective method because once fixed schedule is repeated over many periods of time. Such method enables delivery of the batch of products, at predetermined intervals, resulting from the demand of consumers. It provides a systematic replenishment of small inventories and generates a systematic demand for raw materials and components from suppliers. In this way it is much easier to manage the logistics of supply. Moreover, it is relatively easy to detect certain anomalies that may indicate a deterioration of the operating parameters of the production system. New technologies, materials and rapidly changing customers' needs enable manufacturers to frequent modernization of the machinery. As a result of this process companies have machines with different parameters. In this case, production planning requires determining of not only the allocation of tasks to individual machines but also determining of the order of their execution. This leads to a complex, strongly NP-hard optimization problems, in particular to, known in the literature, flexible job shop problem. Due to the large number of decision variables and requirements of the above mentioned management methods it

W. Bożejko (corresponding author), and J. Pempera are with Department of Automatics, Mechatronics and Control Systems, Faculty of Electronics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland. E-mails: {wojciech.bozejko, jaroslaw.pempera}@pwr.edu.pl. M. Wodecki is with Institute of Computer Science, University of Wrocław, Joliot-Curie 15, 50-383 Wrocław, Poland, e-mail: mieczyslaw.wodecki@uwr.edu.pl

Received 15.12.2016. Revised 28.04.2017.

is necessary to use advanced algorithms to support scheduling at the operational level. Flexible production planning systems enable the implementation of challenges posed by modern management methods such as JIT (just in time) or JIS (just in sequence). In the work there is considered a flexible job shop problem in which a set of tasks to be performed on machines grouped into production cells is given. One should assign tasks to the appropriate machines and determine the order of their execution on each machine so as to optimize some criterion (e.g. all tasks execution time $C_{\max}$). In most of the works on a flexible job shop problem, as an optimization criterion there is completion date of all tasks execution considered (Yan and Xu [11], Gonzales et al. [6], Pezella et al., [8], Bożejko et al. [2]). Much less papers were devoted to the cyclic version of this problem. In the works of Brucker and Kampmeyer [5] and Kampmeyer [7] there was presented a cyclical job shop problem (i.e. a special case of the considered in this work problem in which each slot contains only one machine). In turn, in the work by Bożejko et al. [4] there were not only certain properties proved but also algorithms for solving the cyclic job shop problem presented. The main problem that exists in the design of efficient algorithms to solve NP-hard cyclic scheduling problems is time-consuming determination of the cycle time. In this paper we present not only some properties of a cyclic job shop problem but also an efficient method of determining the cycle time in which parallel processing was used. They were applied in the algorithm which was described in Bożejko et al. [4] where significant reduction of computation time was obtained, without worsening of the quality of designated solutions.

2. Cyclic job shop problem

In the flexible job shop problem there are given: a set of tasks $\mathcal{J} = \{1,2,\ldots,n\}$ and a set of multi-functional machines $\mathcal{M} = \{1,2,\ldots,m\}$ grouped into production slots. Each machine at any time can execute at most one task. Task $j \in \mathcal{J}$ consists of o_j operation for a set $\mathcal{J}^j = \{l_j+1,\ldots,l_j+o_j\}$ where $l_j = \sum_{i=1}^{j-1} o_i$ is the number of operations of the first $j-1$ jobs. Operations included in the tasks are performed according to the order of their numbering and form the so-called technological line. By $O = \{1,2,\ldots,o\}$, where $o = \sum_{i=1}^{j-1} o_i$ we denote the set of all operations. For each operation $v \in O$ there is defined a subset of machines $\mathcal{M}^v \subset \mathcal{M}$. The operation $v \in O$ is to be executed on any k machine from the set $\mathcal{M}^v$ in time $p_{v,k} \geqslant 0$. Execution of operations on the machine cannot be interrupted. In the cyclic production system a set of tasks (hereinafter referred to as MPS-*Minimal Part Set*), is executed repeatedly. In each of the MPS on each machine operations are performed in the same order. The problem consists in the allocation of jobs to machines from the adequate type and the schedule of jobs execution determination on each machine to minimize the cycle time. The following constrains have to be fulfilled:

(*i*) each job has to be executed on only one machine of a determined type in each moment of time,

(*ii*) machines cannot execute more than one job in each moment of time,

(*iii*) there are no idle times (i.e. the job execution must not be broken),

(*iv*) the technological line has to be obeyed,

(*v*) each operation is performed in sequence after the cycle time is completed.

Constraints (*i*)-(*iv*) define known in the literature *Flexible Job Shop* problem (in short denoted by *FJS*). If in addition we assume that each socket contains exactly one machine, then it is a classical in task scheduling theory *Job Shop* problem (abbreviated to *JS*). Descriptions of these problems and metaheuristic algorithms solving them is presented in the works by Nowicki and Smutnicki [9], Barnes and Chambers [1], González et al. [6], Yuan and Xu [11] and Bożejko et al. [3] and [4].

By $\mu = (\mu_1, \ldots, \mu_o)$ we denote the assignment of operations to machines where $\mu_a \in \mathcal{M}^a$ is a machine assigned to perform the operation $a \in O$. The set

$$O^l = \{a \in O : \mu_a = l\} \tag{1}$$

operations executed on the machine $l \in \mathcal{M}$, wherein $\cup_{i=1}^{m} O^i = O$.

Let permutation π_l be a certain sequence of operations from the set O^l on machine l ($|O^l| = n_l$) whereas Φ^l the set of all permutations of elements from O^l. The sequence of operations' execution on the machines is determined by the composition m of permutation $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$, where $\pi_i \in \Phi^i$, $i = 1, 2, \ldots, m$. Let Φ be the set of all such permutations. Let us see that a permutation $\pi \in \Phi$ unambiguously determines the allocation of operations to machines and the order of operations' execution of individual machines. For the fixed sequence $\pi \in \Phi$ (of solution to *FJS*) problem, let $\mathcal{S}^k = (S_1^k, S_2^k, \ldots, S_o^k)$ be a sequence of begining times of operations' execution in the k-th MPS, where S_i^k denotes the commencement date of operation i on machine μ_i in k-th cycle. We assumed that time schedule of the system is cyclic (constraint (*v*)). This means that there is a constant $T(\pi)$ (the so-called *cycle time*) such that

$$S_{\pi(i)}^{k+1} = S_{\pi(i)}^k + T(\pi), \quad i = 1, ..., o, \; k = 1, 2, ... \tag{2}$$

or equivalently

$$S_{\pi(i)}^{k+1} = S_{\pi(i)}^1 + (k-1)T(\pi), \quad i = 1, ..., o, \; k = 1, 2, ... \tag{3}$$

Equality (2) is an implementation of the constraint (*v*). Undoubtedly, in a feasible schedule there must be met also restrictions (*i–iv*) met, which can be written in the form of the following inequities:

$$S_{\pi(i)}^k \geqslant 0 \;\; i = 1, 2, \ldots, o, \; k = 1, 2, \ldots. \tag{4}$$

$$S_{\pi(i)}^k + p_{\pi(i),\mu(\pi(i))} \leqslant S_{\pi(i+1)}^k, \; \pi(i), \pi(i+1) \in \mathcal{J}^j, \; j \in \mathcal{J}, \; k = 1, 2, \ldots \tag{5}$$

$$S^k_{\pi_l(j)} + p_{\pi_l(j),\mu(\pi_l(j))} + \leqslant S^k_{\pi_l(j+1)},\ l \in \mathcal{M},\ j = 1,...,n_l - 1,\ k = 1,2,\ldots \tag{6}$$

$$S^k_{\pi_l(n_l)} + p_{\pi_l(n_l),\mu(\pi_l(n_l))} \leqslant S^{k+1}_{\pi_l(1)},\ l \in \mathcal{M},\ k = 1,2,\ldots \tag{7}$$

Inequity (5) is a realization of the constraint (*iv*), (6) of constraint (*ii*), whereas (7) of constraint (*v*). For a fixed order of operations execution on machines π, the minimum value of $T(\pi)$, for which there is a feasible schedule that meets (2)–(7), will be called *minimal time cycle* and denoted by $T_{\min}(\pi)$. Since the order of operations for each MPS is the same, it is enough just to determine the beginning moments of execution of operations $S^1_1, S^1_2, \ldots, S^1_o$ for the first MPS and make the shift by the size of $T(\pi)$. The considered in this work flexible cyclic job shop problem (in short denoted by *CFJS*) consists in determining a permutation $\pi^* \in \Phi$ with a minimum value of cycle time, i.e. such that

$$T_{\min}(\pi^*) = \min\{T_{\min}(\pi) :\ \pi \in \Phi\}. \tag{8}$$

3. Graph model

For a fixed sequence $\pi = (\pi_1, \pi_2, \ldots, \pi_m)$ $(\pi \in \Phi)$, of operations execution in a cyclic flexible job shop problem and the first k-cycle of production $(k = 1,2,\ldots,m+1)$ we define a directed graph $G(\pi,k) = (\mathcal{V}, \mathcal{T} \cup \mathcal{E}(\pi) \cup \mathcal{C}(\pi))$ consisting of a set of vertices $\mathcal{V}$ and three sets of arcs: $\mathcal{T}$, $\mathcal{E}(\pi)$, $\mathcal{C}(\pi)$. The set $\mathcal{V}$ includes $k \cdot o$ vertices numbered with successive natural numbers. Each operation is assigned to one vertex, wherein the operation $v \in O$ executed in x-th $(x = 1,...,k)$ MPS corresponds to the vertex $v^x(v) = v + (x-1)o$ (x-th copy of vertex v) of weight $p_{v,\mu(v)}$. In the further part we will identify vertices of the graph with operations executed within the corresponding MPS. Sets of arcs represent sequence constraints and are defined as follows:

(*a*) $\mathcal{T} = \bigcup_{x=1}^{k} \bigcup_{j=1}^{n} \bigcup_{v=l_{j-1}+i}^{l_{j-1}+o_j-1} \{(v^x(v), v^x(v+1))\}$, contains arcs representing the technological line (constraint (5)),

(*b*) $\mathcal{E}(\pi) = \bigcup_{x=1}^{k} \bigcup_{l=1}^{m} \bigcup_{i=1}^{n_l-1} \{(v^x(\pi_l(i)), v^x(\pi_l(i+1)))\}$, arcs connecting the operations executed on the same machine (constraint (6)),

(*c*) $\mathcal{C}(\pi) = \bigcup_{x=1}^{k-1} \bigcup_{l=1}^{m} \left\{\left(v^x(\pi_l(n_l)), v^{x+1}(\pi_l(1))\right)\right\}$, arcs connecting the operations executed on the same machine between the MPS (constraint (7)).

Theorem 1 *If $\pi \in \Phi$ is the order of operations' execution on the machines in* CFJS *problem, then the graph $G(\pi,k)$, $k = 1,2,\ldots,m+1$ does not contain cycles.*

Proof Let permutation $\pi \in \Phi$ be the solution to *CFJS* problem. The first MPS is represented by a graph $G(\pi,1)$. It is easy to see that this is also a graph solving the *FJS* problem, whose acyclicality is easy to prove. For a fixed k $(k = 2,3,\ldots,m+1)$ we consider the graph $G(\pi,k)$. It follows from definition of a set of vertices and arcs (a), (b) that the graph is k-fold copy of the graph (components) $G(\pi,1)$. It also includes arcs between certain vertices belonging to the subsequent neighboring components (a set of $\mathcal{C}(\pi)$). Because the components are acyclic graphs and it is not possible to return to the previous components (the definition of a set of arcs (c)), so the graph $G(\pi,k)$ does not contain cycles. □

We consider the longest path in graph $G(\pi,m+1)$ from a vertex $v \in O$ in the first MPS, to the same vertex in x-th MPS, i.e. vertex v^x. By L_v^x we denote the length of this path (the length does not include the weight of vertex v^x). If S_v^1 is the moment of beginning of execution of operation v in the first MPS, and S_v^x the moment of its commencement (i.e. operation v^x) in x-th MPS, then:

$$S_v^x \geqslant S_v^1 + L_v^x. \tag{9}$$

This inequity is a direct result of the constraints (i)-(v). Before the beginning of operation v^x all the operations lying on any path (including the longest) between v and v^x must be executed.

Let

$$\Lambda^*(\pi) = \max_{v \in O} \max_{x=2,\ldots,m+1} \{\lambda_{v,x}\}, \tag{10}$$

where

$$\lambda_{v,x} = L_v^x/(x-1),\ v \in O,\ x = 2,3,\ldots,m+1. \tag{11}$$

Below we will prove two theorems showing the relationship between the minimum cycle time $T_{\min}(\pi)$, and the value $\Lambda^*(\pi)$.

Theorem 2 *If $\pi \in \Phi$ is allowable execution order of operations in a cyclic flexible job shop problem, then the minimum cycle time $T_{\min}(\pi) \leqslant \Lambda^*(\pi)$.*

Proof Let

$$T'(\pi) = \Lambda^*(\pi) = \max_{v \in O} \max_{x=2,\ldots,m+1} \{\lambda_{v,x}\} = \lambda_{a,t} = L_a^t/(t-1).$$

We will show that the so defined $T'(\pi)$ is the cycle time for solution π. Let $(S^1_{\pi(1)}, S^2_{\pi(2)}, \ldots, S^o_{\pi(o)})$ be a sequence of the commencement moment of operations of the first MPS. We will show that

$$S^{k+1}_{\pi(i)} = S^k_{\pi(i)} + (k-1)T'(\pi),\ i = 1,2,\ldots,o,\ k = 2,3,\ldots,m+1,$$

are the beginning moments of separate operations in subsequent MPS, i.e. they meet the constraints (3). By definition (10)

$$S^{k+1}_{\pi(i)} = S^1_{\pi(i)} + (k-1)T'(\pi) = S^1_{\pi(i)} + (k-1)\Lambda^*(\pi) =$$

$$S^1_{\pi(i)} + (k-1)(L^t_a/(t-1)) \geqslant S^1_{\pi(i)} + (k-1)(L^k_{\pi(i)}/(k-1)) = S^1_{\pi(i)} + L^k_{\pi(i)}.$$

The last inequity follows from the fact that $\lambda_{a,t} = L^t_a/(t-1)$ is a maximum element, thus $\lambda_{a,t} \geqslant \lambda_{v,j}$ $(v = 1,2,\ldots,o,\ j = 2,3,\ldots,m+1)$. We have shown this way, that $T'(\pi)$ is cycle time (i.e. it satisfies the inequity (3)), which completes the proof of the theorem.

□

Theorem 3 *If $\pi \in \Phi$ is allowable execution order of operations in a cyclic flexible job shop problem, then the minimum cycle time $T_{\min}(\pi) \geqslant \Lambda^*(\pi)$.*

Proof For the solution $T_{\min}(\pi)$, let $\pi \in \Phi$, be minimum cycle time, whereas $(S^k_{\pi(1)}, S^k_{\pi(2)}, \ldots, S^k_{\pi(o)})$ a sequence of beginning moments of operation in k-th MPS. According to (3)

$$S^k_{\pi(i)} = S^1_{\pi(i)} + (k-1)\cdot T_{\min}(\pi), \tag{12}$$

for $i = 1,2,\ldots,o,\ k = 2,3,\ldots,m+1$.

Let $\pi(l)$ be any operation from the set O. Rusing from (9) and (12) we obtain

$$S^1_{\pi(i)} + (k-1)\cdot T_{\min}(\pi) \geqslant S^1_{\pi(i)} + L^k_{\pi(l)},$$

hence

$$T_{\min}(\pi) \geqslant L^k_{\pi(l)}/(k-1) = \lambda_{k,\pi(l)}.$$

Since this inequality is valid for every $l = 1,2,\ldots,o$ and $k = 2,3,\ldots,m+1$, then

$$T_{\min}(\pi) \geqslant \max\{\lambda_{k,\pi(l)} :\ k = 1,2,\ldots,m,\ k = 1,2,\ldots,o\} = \Lambda^*(\pi),$$

which completes the proof of the theorem. □

Designation of the minimum value of the cycle time $T_{\min}(\pi)$ (i.e. value $\Lambda^*(\pi)$) requires the calculation of $m\cdot o$ values of the coefficients $\lambda_{i,j}$. In the work of Bożejko et al. [4] there was a theorem proven enabling much faster calculation of minimum cycle time.

Let

$$\mathcal{A} = \{v : v = \pi_l(1),\ l \in \mathcal{M}\}$$

be the set of all operations executed as first on the individual machines in the first MPS.

Theorem 4 *For any solution $\pi \in \Phi$ minimum cycle time*

$$T_{\min}(\pi) = \Lambda^*(\pi) = \max_{v\in\mathcal{A}} \max_{x=2,\ldots,m+1} \{\lambda_{v,x}\}.$$

Proof See theorem 2 and 3.

Using this theorem the number of determined coefficients $\{\lambda_{v,x}\}$ can be reduced from $m \cdot o$ to $m \cdot m$, where o is the number of operations and m the number of machines.

Determination of value $\lambda_{v,x}$ dla $x = 2,...,m+1$, $v \in \mathcal{A}$ requires construction of a graph $G(\pi, m+1)$ consisting of $(m+1)o$ vertices and the same order of arcs. In turn, to calculate the value $\{\lambda_{v,x}\}$, for a given $v \in \mathcal{A}$, one must designate the length of the longest paths from v to other vertices which requires $O(mo)$ time. Ultimately, we get the computational complexity designation $\Lambda^*(\pi)$, $|\mathcal{A}|O(mo) = O(om^2)$.

A path in a graph $G(\pi, m+1)$, whose length $L_v^x/(x-1) = \Lambda^*(\pi)$ will be called *a critical path*. In turn, the maximum subsequence of vertices of the path representing the operations executed one after another on the same machine will be called *a block*. In case of the considered in the work *CJFS* tasks problem, one can use the so-called a *blocks* eliminating properties'. The theory was successfully used in the construction of the best optimization algorithms for a wide class of scheduling problems with the criterion $C_{\max}$, e.g. by Nowicki and Smutnicki [9] or Bożejko et al. [3].

Theorem 5 *If the solution $\beta \in \Phi$ was generated from $\pi \in \Phi$ and $T(\beta) < T(\pi)$ at least one operation of at least one block of tasks is executed*

(a) *before the first operation of this block, or*

(b) *after the last operation of this block, or*

(c) *on another machine.*

This theorem will be used when generating elements of the neighborhood in the tabu search algorithm to solve the considered in the work problem.

4. Effective determination of cycle time

Currently the best optimization algorithms for a wide class of scheduling problems are based on iterative methods of local search solution space. Quality of solutions determined by these algorithms depends on the number of directly considered solutions, which with the limited time of the algorithm action depends on the computational complexity of the procedure for calculating the value of the criterion function. Described in the previous chapter method for determining the cycle time has a computational complexity of $O(om^2)$ and is $O(m^2)$ times bigger than the time of determination of the value $C_{\max}$. Acceleration of calculations determining cycle time is possible by use of a parallel processing. For this purpose, the method of parallel vector processing will be used.

We consider a graph $G(\pi, 1)$, $\pi \in \Phi$ corresponding to the first MPS. For any vertex (operation) $v \in O$, by $\tau(v)$ i $\eta(v)$ we successively denote two successors: technological and sequential (in permutation π executed on the same machine). If the operation v has not a corresponding successor, then after $\tau(v)$ or $\eta(v)$ we assume zero. Since the graph

$G(\pi,1)$ is directed acyclic and weakly consistent (i.e. for each pair of distinct vertices x,y there exists the path from x to y or from y to x), so its vertices can be sorted topologically. We can therefore number the vertices in such a way that the beginning of a given arc has a smaller number than its end. In particular the successors $\tau(v)$ or $\eta(v)$ of vertex v have greater numbers than v. Sorting Algorithm topologically sorting vertices of the graph $G(\pi,1)$ has a complexity $O(o)$.

If $\sigma=(\sigma(1),\sigma(2),\ldots,\sigma(o))$ is the topological order of the vertices of the graph $G(\pi,1)$, then it is easy to extend it to any of the graphs $G(\pi,k)$, $k=2,3,\ldots,m+1$. In such a case vertex v^i (from i-th MPS) is given the number of $\sigma(v)+(i-1)o$.

Procedure SeqTC

π - feasible solution;

σ - topological ordering of the vertices of the graph $G(\pi,1)$;

1. For $k=1,\ldots,m$ do
2. Set $L_i^x=-\infty$ for $i\in O, x=1,\ldots,m+1$.
3. Set $L^1_{\pi_k(1)}=0$.
4. For $x=1,\ldots,m+1$ do
5. For $v=\sigma(1),\ldots,\sigma(o)$ do
6. Set $L_v^x=L_v^x+p_v$, $L^x_{\tau(v)}=L_v^x$ **and** $L^x_{\eta(v)}=L_v^x$.
7. For $l=1,\ldots,m$ do $L^{x+1}_{\pi_l(1)}=L^x_{\pi_l(n_l)}$
8. For $k=1,\ldots,m$ **and** $x=1,\ldots,m+1$ do
9. Set $\lambda_{k,x}=L^{x+1}_{\pi_k(1)}-L^x_{\pi_k(1)}$.

Figure 1: Sequential cycle time computing procedure.

Let $\pi\in\Phi$ be the order of operations' execution on the machines in *CFJS*, problem, whereas $\sigma=(\sigma(1),\sigma(2),\ldots,\sigma(o))$ topological order of vertices in the graph $G(\pi,1)$. Figure 1 depicts **SeqTC** procedure of the sequential determining the length of the longest paths L_v^x in the graph $G(\pi,m+1)$ used in the computation of coefficients $\lambda_{v,x}$ (formula(11)). On this basis we determine the value Λ^* (10), i.e. minimum cycle time $T_{\min}(\pi)$. In the description of the procedure the sources are vertices of the graph $G(\pi,1)$, who are not the end of any arc (being the first operations executed on machines). The length of paths $L_v^{(x)}$, $x=1,\ldots,m+1$, $v\in O$ are to be interpreted in two ways. Until step 6 they are lower estimate of the length of the longest path coming out of the vertex being the source to the vertex representing an operation v in x-th MPS (without the weight of the vertex). The source is determined in Step 3. Then, in step 6 the exact value of the length of the longest path to vertex v (with the weight of that vertex) are computed and lower estimate of the length of the longest paths to vertices being successors v, i.e. $\tau(v)$ and $\eta(v)$. are updated. Finally, in step 7, the lower estimate of the length of the longest paths to vertices representing the operations performed on the first machine, in the next MPS are updated.

Based on the analysis of the code it is easy to see that the computational complexity of the procedures for the designation of sequential cycle time **SeqTC** is $O(om^2)$. This

time can be significantly reduced by using techniques of parallel search based on the vector processing. Then, in a vector processor cycle there are performed logical, arithmetic operations or data movements, etc. on one or two multi-element vectors. Vector operations are implemented in hardware in all modern processors, both desktops and laptops, and above all, in programmable graphics cards.

Procedure ParTC

π - feasible solution;

σ - topological ordering of the vertices of the graph $G(\pi, 1)$;

1. Set $\overrightarrow{L_v^x} = -\infty$ for $v \in O$, $x = 1, \ldots, m+1$.
2. For $k = 1, \ldots, m$ do
3. Set $\overrightarrow{L^1_{\pi_k(1)}}(k) = 0$.
4. For $x = 1, \ldots, m+1$ do
5. For $v = \sigma(1), \ldots, \sigma(o)$ do
6. Set $\overrightarrow{L_v^x} = \overrightarrow{L_v^x} + \overrightarrow{p_v}$, $\overrightarrow{L^x_{\tau(v)}} = \overrightarrow{L_v^x}$ **and** $\overrightarrow{L^x_{\eta(v)}} = \overrightarrow{L_v^x}$.
7. For $l = 1, \ldots, m$ do $\overrightarrow{L^{x+1}_{\pi_l(1)}} = \overrightarrow{L^x_{\pi_l(n_l)}}$.
8. For $k = 1, \ldots, m$ **and** $x = 1, \ldots, m+1$ do
9. Set $\lambda_{k,x} = \overrightarrow{L^{x+1}_{\pi_k(1)}}(k) - \overrightarrow{L^x_{\pi_k(1)}}(k)$.

Figure 2: Parallel cycle time computation.

In Figure 2 there is shown a diagram of **ParTC** procedure effectively determining the cycle time using a vector parallel processing. Designations are the same as in the sequential procedure. In the vector

$$\overrightarrow{L_i^{(x)}} = (\overrightarrow{L_i^x}(1), \overrightarrow{L_i^x}(2), \ldots, \overrightarrow{L_i^x}(m)) \tag{13}$$

there is remembered the length of the longest path reaching to the vertex representing operation i in x-th MPS. For various elements of the vector the values differ from one another due to a different source vertex assigned to each vector element. The most time-consuming are iterative instructions in row 4 and 5. Assuming that operations on vectors are executed in one tact we get superior computational complexity $O(om)$, in case of CPU processors transforming m- element vectors.

5. Neighborhood viewing

In the algorithms of local search the adjacent solutions (neighborhood) can be viewed in two ways, ie. by generating: (*i*) all neighboring solutions (*ii*) subset containing only some solutions. Undoubtedly, the first method is much more time-consuming. However, it usually enables determination of good solutions with fewer iterations of the whole

algorithm. Regardless of the method of the neighborhood viewing, for any solution there should be the value of the objective function determined. In the algorithms viewing the whole neighborhood, for many optimization problems, it is possible to construct the accelerator. This ensures, with the use of partial results, a considerable reduction of the computation time of goal function value for all solutions of the neighborhood. An effective example of the accelerator use was described, among others, by Nowicki and Smutnicki [9]. Below, we present a new two-phase neighborhood search method. In the first phase, for each solution in the neighborhood, there is a lower bound of the value of the objective function determined. At the beginning of the second phase there is created a list of solutions ordered non-decreasingly in reference to lower bound (determined in the first phase). Then, for solutions in the sequence they appear in the ordered list, there is the exact value of the objective function calculated. The computation process is terminated, as soon as the solution whose exact value of the objective function is determined, not greater than the lower bound of the remaining on the solutions list. It is worth noting that the better the lower bound of the objective function value, the less solutions will be verified by calculating the exact value.

For the considered in this paper cyclic flexible job shop problem lower bound can be determined by considering only the first MPS (i.e. graph $G(\pi,1)$), wherein as the lower bound we assume:

$$LB(\pi) = \max_{v\in\mathcal{A}}\{\lambda_{v,1}\}. \tag{14}$$

The computational complexity of determining the value $LB(\pi)$ is in sequential version $O(om)$, whereas in parallel $O(o)$.

6. Computational experiments

In order to evaluate the acceleration of computations relating to the proposed neighborhood viewing method and the use of vector processing there were computational experiments carried out. The results of golf *AGF* algorithm presented in the work of Bożejko et al. [4] and its two of modifications A_S and A_V were compared. In both algorithms there was a two-phase search of neighborhood applied. In A_S algorithm the search was executed sequentially (procedure **SeqTC**), whereas in A_V algorithm in parallel (procedure **ParTC**). Algorithms were programmed in C++ in Visual Studio 2010. The computations were performed on a PC with an Intel I7-core 2.4GHz on a single core of the processor. Parallel processing was carried out on 128-bit registers using SSE2 commands. Each of them was an 8-element vector consisting of 16-bit representations of the data processed in parallel. Vector processing based on SSE instructions was used, among many others, in the work of Smutnicki et al. [10]. Comparative studies of algorithms were carried out on the instances presented in the works of Barnes and Chambers [1]. For all three algorithms the adopted number of iterations (stop condition) equaled 10 000, whereas the length of tabu list was 15.

Table 1: The operating times and acceleration of algorithms.

Instance	$n \times m$	o	$t(AGF)$	$t(A_S)$	$\frac{t(AGF)}{t(A_S)}$	$t(A_V)$	$\frac{t(A_S)}{t(A_V)}$	$\frac{t(AGF)}{t(A_V)}$
setb4c9	15×11	150	19.59	4.91	4.0	1.49	3.3	13.1
setb4cc	15×12	150	30.22	6.63	4.6	1.97	3.4	15.3
setb4x	15×11	150	18.56	4.92	3.8	1.48	3.3	12.5
setb4xx	15×12	150	23.45	5.78	4.1	1.64	3.5	14.3
setb4xxx	15×13	150	69.33	6.19	11.2	1.73	3.6	40.1
setb4xy	15×12	150	7.47	5.38	1.4	1.63	3.3	4.6
setb4xyz	15×13	150	4.06	0.75	5.4	0.23	3.3	17.7
seti5c12	15×16	225	67.6	12.67	5.3	3.52	3.6	19.2
seti5cc	15×17	225	98.2	15.86	6.2	4.41	3.6	22.3
seti5x	15×16	225	62.86	12.09	5.2	3.38	3.6	18.6
seti5xx	15×17	225	77.13	14.09	5.5	3.81	3.7	20.2
seti5xxx	15×18	225	107.27	17.03	6.3	4.52	3.8	23.7
seti5xy	15×17	225	98.1	15.75	6.2	4.36	3.6	22.5
seti5xyz	15×18	225	121.14	17.97	6.7	4.81	3.7	25.2

In Table 1 there were the results on the time of computations of algorithms $t(A)$, $A \in \{AGF, A_S, A_V\}$. presented. The first column shows the name of example, in the subsequent ones: the number of tasks (n), the number of machines (m) and the number of operations (o). The next three columns include the running times of AGF and A_S. algorithms. In turn, in the last three columns there were the operating times of sequential algorithms AGF and A_S. compared with the running time of the parallel algorithm A_V.

The results of experimental studies clearly indicate that the use of the two-phase method of viewing the neighborhood significantly reduces computation times. For the sequential version of the algorithm the running time is shorter - from 4.0 to 11.2 times (the quotient of $\frac{t(AGF)}{t(A_S)}$). On the other hand, the use of parallel processing realized by vector computing enables additional reduction of time from 3.3 to 3.8 times. Ultimately, the simultaneous use of both methods of computation acceleration allows its users for additional time reduction from 4.6 to 40.1 times.

The quality of solutions generated by AGF algorithm was presented in Table 2 (the other two algorithms A_S and A_V determined the same solutions but in a much shorter time). The cycle time of solutions generated by AGF algorithm were compared with the best known values of the minimum execution time of all tasks $C_{\max}$. It should be noted that $C_{\max}$ is the lower bound on the length of cycle time for one of the cyclic models considered in the work of Brucker and Kampmeyer [5]. For each example, based on a minimum time of tasks execution of the first MPS $C_{\max}$ and the length of the cycle time

Table 2: The values set by the algorithms of solutions.

Instance	$n \times m$	o	$C_{\max}$	T^*	PRD
setb4c9	15×11	150	914	903	1.20
setb4cc	15×12	150	907	887.67	2.13
setb4x	15×11	150	925	878	5.08
setb4xx	15×12	150	925	879	4.97
setb4xxx	15×13	150	925	1002	-8.32
setb4xy	15×12	150	910	845	7.14
setb4xyz	15×13	150	903	838	7.20
seti5c12	15×16	225	1174	1130	3.75
seti5cc	15×17	225	1136	1064.5	6.29
seti5x	15×16	225	1198	1141	4.76
seti5xx	15×17	225	1197	1100	8.10
seti5xxx	15×18	225	1197	1136.5	5.05
seti5xy	15×17	225	1136	1064.5	6.29
seti5xyz	15×18	225	1125	1052	6.49

T there was a relative percentage improvement determined

$$PRD = \frac{C_{\max} - T}{C_{\max}} 100\%. \tag{15}$$

For 13 instances the cycle times determined by *AGF* algorithm were significantly lower than the value $C_{\max}$. This improvement ranged from 1.2 to 8.1 %. In one case (example setb4xxx) determined by *AGF* algorithm, cycle time length was about 8.32% worse than the value $C_{\max}$.

Given the fact that the best solutions were determined after a small number of iterations, it is possible to state that this algorithm can be successfully used to solve practical examples of large sizes.

7. Summary

In the work there was a cyclical flexible job shop problem considered. A graph model, for a fixed order of operations execution on individual machines was presented. The theorems enabling efficient determination of the minimum cycle time and a lower bound were proven. In order to speed up the calculations there was not only a two-phase method of neighborhood searching proposed but also a parallel method of determining the cycle time for the established order of operations that used vector processing

proposed. Ultimately, the results of computational experiments, which confirmed a significant acceleration of calculations, while maintaining the designated solutions were presented. To sum up, on the basis of the obtained results it can be concluded that the modified *AMF* algorithm designated in a short time fully accepted in practice solutions.

References

[1] J.W. BARNES and J.B. CHAMBERS: Flexible job shop scheduling by tabu search, Graduate program in operations research and industrial engineering. The University of Texas at Austin, Technical Report Series: ORP96-09, 1996.

[2] W. BOŻEJKO, M. UCHROŃSKI and M. WODECKI: Parallel hybrid metaheuristics for the flexible job shop problem. *Computers & Industrial Engineering*, **59**(2), (2010), 323-333.

[3] W. BOŻEJKO, M. UCHROŃSKI and M. WODECKI: Block approach to the cyclic flow shop scheduling. *Computers & Industrial Engineering*, **81** (2015), 158-166.

[4] W. BOŻEJKO, J. PEMPERA and M. WODECKI: The golf algorithm for the cyclic flexible job shop problem, (to appear).

[5] P. BRUCKER and T. KAMPMEYER: Cyclic job shop scheduling problems with blocking. *Annals of Operations Research*, **159** (2008), 161-181.

[6] M.A. GONZÁLEZ, C.R. VELA and R. VARELA: Scatter search with path relinking for the flexible job shop scheduling problem. *European J. of Operational Research*, **245**(1), (2015), 35-45.

[7] T. KAMPMEYER: Cyclic Scheduling Problems. Ph.D. Thesis, University Osnabrick, 2006.

[8] F. PEZZELLA, G. MORGANTI and G. CIASCHETTI: A genetic algorithm for the flexible job-shop scheduling problem. *Computers & Operations Research*, **35** (2008), 3202-3212.

[9] E. NOWICKI and C. SMUTNICKI: A fast tabu serach algorithm for the job shop problem. *Management Scence*, **42**, (1996), 797-813.

[10] C. SMUTNICKI, J. PEMPERA, J. RUDY and D. ZELAZNY: A new approach for multi-criteria scheduling. *Computers & Industrial Engineering*, **90** (2015), 212-220.

[11] Y. YUAN and H. XU: Flexible job shop scheduling using hybrid differential evolution algorithms. *Computers & Industrial Engineering*, **65**(2), (2013), 246-260.