Parallel Tabu Search Algorithm with Uncertain Data for the Flexible Job Shop Problem

Wojciech Bożejko[1(✉)], Mariusz Uchroński[2], and Mieczysław Wodecki[3]

[1] Department of Control Science and Mechatronics, Faculty of Electronics, Wrocław University of Technology, Janiszewskiego 11-17, 50-372 Wrocław, Poland
wojciech.bozejko@pwr.edu.pl
[2] Wrocław Centre of Networking and Supercomputing, Wyb. Wyspiańskiego 27, 50-370 Wrocław, Poland
mariusz.uchronski@pwr.edu.pl
[3] Institute of Computer Science, University of Wrocław, Joliot-Curie 15, 50-383 Wrocław, Poland
mwd@ii.uni.wroc.pl

Abstract. In many real production systems the parameters of individual operations are not deterministic. Typically, they can be modeled by fuzzy numbers or distributions of random variables. In this paper we consider the flexible job shop problem with machine setups and uncertain times of operation execution. Not only we present parallel algorithm on GPU with fuzzy parameters but also we investigate its resistance to random disturbance of the input data.

1 Introduction

In the process of solving classical tasks scheduling problems all tasks parameters (e.g. execution times) are known. In case of uncertain data, first – assuming a certain size (from many possible), then – finding the solution to obtained in this way deterministic scheduling problem – results in reaching not very stable solutions. Therefore, data uncertainty is modeled either with the use of stochastic methods or with the theory of fuzzy numbers. The first attempt [8] requires knowledge of certain statistic data from the past. The main disadvantage of this method is the difficulty in obtaining and verifying not only probability distribution of parameters but also their moments (e.g. mean). In case of research on new problems or the use of new technologies, knowledge concerning statistic data from the past usually does not exist. That is why the approach based on the use of the theory of fuzzy numbers is in this case fully justified. The first studies concerning solving scheduling problems with fuzzy data relate to PERT method [10], and also to classical single and multi-machine problems [1,11].

2 Problem Formulation

Flexible job shop problem can be defined as follows. There is a set of tasks $\mathcal{J} = \{1, 2, \ldots, n\}$ given, which should be executed on the machines from the set

L. Rutkowski et al. (Eds.): ICAISC 2016, Part II, LNAI 9693, pp. 419–428, 2016.
DOI: 10.1007/978-3-319-39384-1_36

$\mathcal{M} = \{1, 2, \ldots, m\}$. There is a breakup of a set of machines into types, i.e. machines of the same functional properties. The task is a sequence of some operations. Each operation should be performed on the suitable type of machine in a fixed period of time. Between sequentially executed operations there should be setup of machines performed. The solution to the problem relies in the allocation of operations to the appropriate type of machines and determination of the order of operations on each machine, to minimize the time of execution of all tasks.

Let $\mathcal{O} = \{1, 2, \ldots, o\}$ be the set of all operations. It can be divided into the subsequences of operations corresponding to the tasks, where the task $j \in \mathcal{J}$ is a sequence of o_j operations which will be successively performed on the respective machines (in the so-called technological order). These operations are indexed by numbers $(l_{j-1} + 1, \ldots, l_{j-1} + o_j)$, where $l_j = \sum_{i=1}^{j} o_i$ is the number of the first operation of the task j, $j = 1, 2, \ldots, n$, whereas $l_0 = 0$, and $o = \sum_{i=1}^{n} o_i$.

The set of machines $\mathcal{M} = \{1, 2, \ldots, m\}$ can be partitioned into q subsets of machines of the same type (*nests*), whereas i-th ($i = 1, 2, \ldots, q$) type $\mathcal{M}^i$ includes m_i machines, which are indexed by numbers $(t_{i-1} + 1, \ldots, t_{i-1} + m_i)$, where $t = \sum_{j=1}^{i} m_j$ is the number of machines in the first i types, $i = 1, 2, \ldots, q$, and $t_0 = 0$, and $m = \sum_{j=1}^{m} m_j$.

Operation $v \in \mathcal{O}$ should be performed in the nest $\mu(v)$, i.e. on one of machines from the set $\mathcal{M}^{\mu(v)}$ in time p_{vj}, where $j \in \mathcal{M}^{\mu(v)}$.

Let $\mathcal{O}^k = \{v \in \mathcal{O} : \mu(v) = k\}$ be a set of operations executed in k-th ($k = 1, 2, \ldots, q$) nest. A sequence of disjoint sets of operations will be executed in k-th ($k = 1, 2, \ldots, q$) nest. A sequence of disjoint sets of operations $\mathcal{Q} = [\mathcal{Q}_1, \mathcal{Q}_2, \ldots, \mathcal{Q}_m]$, such that for every $k = 1, 2, \ldots, q$, $\mathcal{O}^k = \bigcup_{i=t_{k-1}+1}^{t_{k-1}+m_k} \mathcal{Q}_i$, is called the *allocation of operations from the set* $\mathcal{O}$ *to machines from the set* $\mathcal{M}$. In turn, the sequence $[\mathcal{Q}_{t_{k-1}+1}, \mathcal{Q}_{t_{k-1}+2}, \ldots, \mathcal{Q}_{t_{k-1}+m_k}]$ is the allocation of operations to machines in the i-th nest.

Any feasible solution of the flexible job shop problem is a pair $(\mathcal{Q}, \pi(\mathcal{Q}))$, where $\mathcal{Q}$ is the allocation of operations to machines and $\pi(\mathcal{Q}) = (\pi_1, \pi_2, \ldots, \pi_m)$ is a m-tuple of permutations (or permutation in short) denoting the order of performing operations on individual machines. In the further part, for simplicity of notation, any feasible solution will be denoted by $\Theta = (\mathcal{Q}, \pi)$. By Φ there is a set of all feasible solutions for the flexible job shop problem denoted.

2.1 Graph Model

Any solution $\Theta = (\mathcal{Q}, \pi) \in \Phi$ can be represented by the directed graph with burdened vertices and arcs $G(\Theta) = (\mathcal{V}, \mathcal{R} \cup \mathcal{E}(\Theta))$, where $\mathcal{V}$ is a set of vertices, and $\mathcal{R} \cup \mathcal{E}(\Theta)$ is a set of arcs, whereby:

1. The set $\mathcal{V} = \mathcal{O} \cup \{s, c\}$, where s and c are additional operations representing respectively: 'start' and 'end'. The vertex $v \in \mathcal{V} \setminus \{s, c\}$ is characterized by two features:
   - $\lambda(v)$ – the number of a machine $v \in \mathcal{O}$ on which the operation is to be executed,
   - $p_{v,\lambda(v)}$ – the time of execution of the operation $v \in \mathcal{O}$ on machine $\lambda(v)$.

The weighs of additional vertices s and c equal zero.
2. The set $\mathcal{R}$ contains arcs connecting successive operations of the same task. Weight of an arc is equal to the time of machine setup. The set also includes arcs from the vertex s for the first operation of each task and arcs for the last operation of each task to the vertex of the c. The weight of the arcs is zero.
3. Arcs from the set $\mathcal{E}(\Theta)$ combine operations performed on the same machine. The weight of this arc is zero.

Remark 1. The pair $\Theta = (\mathcal{Q}, \pi)$ is a feasible solution for the flexible job shop problem if and only if the graph $G(\Theta)$ does not contain cycles.

2.2 Block Eliminating Properties

The sequence of vertices $(\upsilon_1, \upsilon_2, \ldots, \upsilon_k)$ of the graph $G(\Theta)$ such that $(\upsilon_i, \upsilon_{i+1}) \in \mathcal{R} \cup \mathcal{E}(\Theta)$ for $i = 1, 2, \ldots, k-1$ is called *a path* from the vertex υ_1 to υ_k. Let $C(\nu, \upsilon)$ denote the longest path (called *critical path*) in the graph $G(\Theta)$ from the vertex ν do υ $(\nu, \upsilon \in \mathcal{V})$, and $L(\nu, \upsilon)$ the length (the sum of weighs of vertices) of the path.

It is easy to notice that the execution time of all operations $C_{max}(\Theta)$ consistent with the assignment of operations $\mathcal{Q}$ and the sequence π is equal to the length $L(s, c)$ of the critical path $C(s, c)$ in graph $G(\Theta)$. Solution of the flexible job shop problem is reduced to designation of $\Theta = (\mathcal{Q}, \pi)$ in the set Φ, for which the corresponding graph $G(\Theta)$ has the shortest critical path, i.e. minimizes $L(s, c)$.

Critical path $C(s, c) = (s, \upsilon_1, \upsilon_2, \ldots, \upsilon_w, c)$, where $\upsilon_i \in \mathcal{O}$ $(1 \leqslant i \leqslant w)$, in graph $G(\Theta)$ can be divided into the subsequences of vertices $\mathcal{B} = [B_1, B_2, \ldots, B_r]$ called *components* of the critical path $C(s, c)$, where:

(a) component contains further operations performed directly one after the other, on the same machine,
(b) intersection of any two components is empty,
(c) component is the maximum (due to the inclusion) subset of operations from the critical path fulfilling the constraints (a)–(b).

In the further part there will be considered only the components containing at least two elements. The component B_k $(k = 1, 2, \ldots, r)$ on machine M_i $(i = 1, 2, \ldots, m)$ from the nest t $(t = 1, 2, \ldots, q)$ will be denoted by $B_k = (\pi_i(a_k), \pi_i(a_{k+1}), \ldots, \pi_i(b_k))$, where $1 \leqslant a_k < b_k \leqslant |\mathcal{Q}^i|$. The operations $\pi(a_k)$ i $\pi(b_k)$ are called respectively *first* and *last*.

For any component $B_k = (\pi_i(a_k), \pi_i(a_k + 1), \ldots, \pi_i(b_k))$, by Φ^k we denote the set of all permutations from the set $\{\pi_i(a_k + 1), \pi_i(a_k + 2), \ldots, \pi_i(b_k - 1)\}$. Let $\beta^* \in \Phi^k$ be the permutation such that

$$\Psi(\beta^*) = min\{\Psi(\gamma) : \gamma \in \Phi^k\}, \tag{1}$$

where $\Psi(\gamma) = s_{\pi_i(a_k),\gamma(1)} + \sum_{i=a_k+1}^{b_k-1} s_{\gamma(i),\gamma(i+1)} + s_{\gamma(b_k-1),\pi_i(b_k)}$ is the length of the path $(\pi_i(a_k), \gamma(1), \gamma(2), \ldots, \gamma(b_k - 1), \pi_i(b_k))$.

Remark 2. Permutation β^* represents the shortest path between $\pi_i(a_k)$ and $\pi_i(b_k)$) in $G(\Theta)$ including vertices from the set $\{\pi_i(a_k+1), \pi_i(a_k+2), \ldots, \pi_i(b_k-1)\}$.

The sequence of operations executed on k-th machine $\widehat{B}^k = (\pi_i(a_k), \beta^*, \pi_i(b_k))$ is called k-th **block**, whereas permutation β^* – internal block. By designating blocks on critical path of the solution Θ we generate some new solution of value of not more than the value of solution Θ. This procedure can be seen as some form of local optimization (improvement of Θ).

Theorem 1. *If $\widehat{B}_k$ is a block from critical path, then any change of the order of operation from internal block does not generate a solution of lower value of the objective function.*

The proof is similar to that of Theorem 3 from the work by Bożejko et al. [5].

This property (i.e. *block elimination property*) will be used to generate the neighborhood in both sequential algorithm and parallel algorithm solving the flexible job shop problem with machine setups. The above theorem implies that any improvement of the value of the current solution can only be achieved by: (1) swapping some operations from some internal block before the first or the last operation of this block, or (2) moving the operation to another, from the same nest, machine.

3 Tabu Search Algorithm

To solve the problem considered in this work there will be tabu search algorithm with golf neighborhood used (Bożejko et al. [3]). It is generated by the combination of insert type (i-move) and transfer (t-move) moves (see [2]). The first – changes the order of operations on a machine, whereas the second – transfers operation to another machine (from the same nest). Ideas for generating the neighborhood solution Θ can be represented as follows:

1. generate graph $G(\Theta)$,
2. determine critical path in graph $G(\Theta)$, then perform partitioning of the set of operations from the path into components,
3. in accordance with block definition determine the sequence of operations for each component,
4. generate golf neighborhood using block elimination properties (Theorem 1).

If B is a component of the critical path in the graph $G(\Theta)$, then determination of the order of an operation that fulfills the constraints of the block definition requires the determination of the shortest path (permutation) between the first and the last operation in B.

It is easy to notice that in case of the considered problem the process comes down to solving the traveling salesman problem (TSP) in the graph, wherein vertices are the operations of B and the distance between vertices are setup times. This is an NP-hard problem, this is why to find a good approximate solution the 2-*opt* algorithm will be used, one of the most popular approximate

algorithms for the TSP. The use of heuristic algorithm results in the fact that one of the conditions from the block definition cannot be fulfilled. As a consequence, 'good' elements can be eliminated from the neighborhood.

3.1 Parallel Algorithm on GPU

In the tabu search algorithm, for any solution $\Theta = (\mathcal{Q}, \pi)$, there should be created a graph $G(\Theta) = (\mathcal{V}, \mathcal{R} \cup \mathcal{E}(\Theta))$, then there should be a critical path, components and blocks designated. Next, using the block eliminating properties there should be generated the neighborhood from which the best element is chosen. As the graph contains o vertices, the parallel algorithm designating the longest path between all pairs of vertices needs time of $O(o)$ using o^2 processors.

Theorem 2. *The sequence of blocks $\mathcal{B} = (B_1, B_2, \ldots, B_r)$ on the critical path $C(s, c)$ for any solution $\Theta \in \Phi$ can be determined on CREW PRAM machine in time $O(1)$ with the use of $O(o)$ processors.*

Proof. Let each of the processors be assigned to the vertex v from critical path. It is sufficient if the processor checks whether the number of machines allocated to its vertex $\lambda(v)$ is the same as the number of the machine $\lambda(u)$ assigned to the next vertex u from the critical path. If the numbers of machines are different, the next block starts from the vertex u.

Theorem 3. *Neighborhood $\mathcal{N}(\Theta)$ of the solution Θ of the flexible job shop problem generated by t-moves can be searched in time $O(o)$ with the use of $O(om)$ processor CREW PRAM machine.*

Proof. The considered neighborhood consists of $O(om)$ elements. Let each of the processors be assigned to one element from the neighborhood. Designation of the objective function for any solution requires time $O(o)$. Therefore, the whole search process of the neighborhood requires time $O(o)$.

Proposal 1. *Speedup of the method based on Theorem 3 is $O(om)$, whereas the cost equals $O(o^2m)$. The presented method is cost-optimal. Its efficiency equals $O(1)$.*

Theorem 4. *The neighborhood of the solution generated by t-moves can be searched in the time $O(mo)$ with the use of $O(o)$ processor CREW PRAM machine.*

Proof. The considered neighborhood consists of $O(om)$ elements. Let each of the processors be assigned to one of the operations $i \in \mathcal{O}$, $i = 1, 2, \ldots, o$. For each operation it is possible to generate $O(m)$ solutions using t-moves. Determination of the value of the objective function of a single solution takes time $O(o)$. Therefore, the whole process takes time $O(mo)$.

Proposal 2. *Speedup of the method based on Theorem 4 is $O(o)$. The cost is $O(o^2m)$. The presented method is cost-optimal.*

4 Fuzzy Times of Tasks Execution

Let $p_{\upsilon j}$ ($j \in \mathcal{M}^{\mu(\upsilon)}$) be the time of execution of operation $\upsilon \in \mathcal{O}$ in the nest $\mu(\upsilon)$, i.e. on one of the machines from the set $\mathcal{M}^{\mu(\upsilon)}$. We assume that the uncertain times of operations execution will be represented by the membership function γ represented by four numbers $\widehat{p}_{\upsilon,j} = (p^{min}_{\upsilon,j}, p^{med1}_{\upsilon,j}, p^{med2}_{\upsilon,j}, p^{max}_{\upsilon,j})$ such that:

- $(p^{\min}_{\upsilon,j} \leqslant p^{med1}_{\upsilon,j} \leqslant p^{med2}_{\upsilon,j} \leqslant p^{\max}_{\upsilon,j})$,
- $\gamma(x) = 0$ for $x \leqslant p^{\min}_{\upsilon,j}$ or $x \geqslant p^{\max}_{\upsilon,j}$,
- $\gamma(p^{med1}_{\upsilon,j}) = \gamma(p^{med2}_{\upsilon,j}) = 1$,
- γ is increasing in the interval $[p^{min}_{\upsilon,j}, p^{med1}_{\upsilon,j}]$ and decreasing in the interval $[p^{med2}_{\upsilon,j}, p^{max}_{\upsilon,j}]$,

which can be represented as:

$$\gamma(x) = \begin{cases} \frac{x - p^{min}_{\upsilon,j}}{p^{med1}_{\upsilon,j} - p^{min}_{\upsilon,j}} & \text{for } x \in [p^{\min}_{\upsilon,j}, p^{\text{med1}}_{\upsilon,j}), \\ 1 & \text{for } x \in [p^{\text{med1}}_{\upsilon,j}, p^{\text{med2}}_{\upsilon,j}), \\ \frac{p^{max}_{\upsilon,j} - x}{p^{max}_{\upsilon,j} - p^{med2}_{\upsilon,j}} & \text{for } x \in [p^{med2}_{\upsilon,j}, p^{\max}_{\upsilon,j}]. \end{cases} \tag{2}$$

Graphical representation of the function $\gamma(x)$ has a trapezoidal shape, therefore, fuzzy numbers represented by it are called trapezoidal fuzzy numbers. A special case of trapezoidal fuzzy number is a triangular fuzzy number. By substituting $p^{med}_{\upsilon,j} = p^{med1}_{\upsilon,j} = p^{med2}_{\upsilon,j}$ to (2) we obtain a membership function for the triangular fuzzy number. Definitions of arithmetic activities and computing maximum/minimum of fuzzy numbers is defined in the work of Dubois [7]. Therefore, when the times of operations execution are fuzzy numbers, then the completion times of operations and completion date the of all tasks executions are also fuzzy numbers.

In the optimization algorithms certain values (for instance goal function values) are repeatedly compared with one another. Therefore, there is a need in mapping of the number of fuzzy into the real numbers (*defuzzification*). In literature the most often cited features of defuzzification are: last of maximum, mean of maxima values and center of area.

Last of Maximum Defuzzification Function. Let $(m_1, m_2, \ldots m_l)$ be a sequence of local maximum values of the fuzzy number $\widehat{a}$ of the membership function $\gamma(x)$, $x \in \mathbb{R}$. First of maximum defuzzification function is defined as $LOM(\widehat{a}) = m_i$, where $i = \max_{1 \leqslant j \leqslant l} \arg m_j$.

Mean of Maxima Defuzzification Function. Let $(m_1, m_2, \ldots m_l)$ be a sequence of local maximum values of the fuzzy number $\widehat{a}$ of the membership function $\gamma(x)$, $x \in \mathbb{R}$. First of maximum defuzzification function is defined as $MOM(\widehat{a}) = \left\lceil \frac{1}{l} \sum_{i=1}^{l} m_i \right\rceil$.

Center Area Defuzzification Function. Let $\widehat{a}$ be a fuzzy number of membership function $\gamma(x)$, $x \in \mathbb{R}$. Center area defuzzification function is defined as

$COA(\widehat{a}) = \left\lceil \frac{\int x\gamma(x)\,\mathrm{d}x}{\int \gamma(x)\,\mathrm{d}x} \right\rceil$. In order to determine the 'center area' defuzzification function for trapezoidal fuzzy number (2) let us adapt the following simplification:

$$\gamma(x) = \begin{cases} \frac{x-a}{b-a} & \text{for } x \in [a,b), \\ 1 & \text{for } x \in [b,c), \\ \frac{d-x}{d-c} & \text{for } x \in [c,d]. \end{cases} \tag{3}$$

For each of the intervals we calculate the corresponding integrals.

$$\int_a^b x\frac{x-a}{b-a}\,\mathrm{d}x = \frac{1}{b-a}\int_a^b x^2 - ax\,\mathrm{d}x = \frac{1}{b-a}\left[\frac{x^3}{3} - \frac{ax^2}{2}\right]_a^b = -\frac{1}{6}(a-b)(a+2b).$$

$$\int_c^d x\frac{d-x}{d-c}\,\mathrm{d}x = \frac{1}{d-c}\int_c^d dx - x^2\,\mathrm{d}x = \frac{1}{d-c}\left[\frac{dx^2}{2} - \frac{x^3}{3}\right]_c^d = -\frac{1}{6}(c-d)(2c+d).$$

$$\int_a^b \frac{x-a}{b-a}\,\mathrm{d}x = \frac{1}{b-a}\int_a^b x - a\,\mathrm{d}x = \frac{1}{b-a}\left[ax - \frac{x^2}{2}\right]_a^b = \frac{b-a}{2}.$$

$$\int_c^d \frac{d-x}{d-c}\,\mathrm{d}x = \frac{1}{d-c}\int_c^d d - x\,\mathrm{d}x = \frac{1}{d-c}\left[dx - \frac{x^2}{2}\right]_c^d = \frac{d-c}{2}.$$

$$\int_b^c x\,\mathrm{d}x = \left[\frac{x^2}{2}\right]_b^c = \frac{1}{2}(c^2-b^2), \int_b^c 1\,\mathrm{d}x = [x]_b^c = c-b.$$

Using the calculated integrals we ultimately obtain:

$$COA(\widehat{a}) = \left\lceil \frac{\int x\gamma(x)\,\mathrm{d}x}{\int \gamma(x)\,\mathrm{d}x} \right\rceil = \left\lceil \frac{a^2+ab+b^2-c^2-cd-d^2}{3(a+b-c-d)} \right\rceil. \tag{4}$$

4.1 Block Properties for the Fuzzy Times of Operation Execution

Since the operations execution times and moments of their completion are fuzzy numbers, there can be a **fuzzy critical path** $\widehat{C}(s,c)$ determined with the use of defuzzification function and then its division into blocks

$$\widehat{B} = [\widehat{B}_1, \widehat{B}_2, \ldots, \widehat{B}_r]. \tag{5}$$

In deterministic version of the problem block properties enable the elimination of worse solutions from the search process. This can significantly reduce the size of the analyzed solution space. In case of modeling of uncertain times of operations execution with the use of fuzzy numbers there can be two models, using eliminating block properties, taken into consideration.

In the first model – determination of the starting moment of the operation execution – relies in computing of a maximum of two fuzzy numbers (termination moments of the technological $\widehat{C}_t$ and machine $\widehat{C}_m$) predecessor, resulting in obtaining the fuzzy number $\widehat{C}_x = \max\{\widehat{C}_t, \widehat{C}_m\}$. Since the number $\widehat{C}_x$ does not have to be any of the numbers $\widehat{C}_t$ and $\widehat{C}_m$, critical path determination is not possible. By introducing additional parameters describing the degree of similarity between the determined number of a maximum $\widehat{C}_x$ and the numbers $\widehat{C}_t$ and $\widehat{C}_m$ there can be *quasi critical path* determined.

In turn, in the second model, using the defuzzification function, there can be the termination moments of technological C_t and machine C_m predecessor determined. Then, it is possible to clearly determine the time of commencement of new operations as the fuzzy number C_x (defuzzification maximum):

$$C_x = \max\{defuz(\widehat{C}_t), defuz(\widehat{C}_m)\}. \tag{6}$$

In this case, it is possible to clearly determine the critical path.

4.2 Resistance of Algorithms to Data Disturbances

Resistance of an algorithm is the property that allows its users to determine the influence of data disturbances on changes in the value of the objective function. For the problem considered in the work, let $\delta = [p_{v,j}]_{o\times 1}$ be time vector of (deterministic) times of execution of separate operations. By $D(\delta)$ we denote the set of data generated from δ by the disturbance in times of operations execution. The disturbance relies in changing the elements of $p = [p_{v,j}]_{o\times 1}$ into randomly generated values.

Let $A = \{TS, TSF\}$, where TS and TSF are respectively deterministic and fuzzy algorithms (i.e. for the data represented by fuzzy numbers). By π_δ^A we denote a solution (permutation) determined by the algorithm A for data δ. Then $C_{max(\pi_\delta^A,\varphi)}$ is a termination point, when the tasks are executed in the order π (defined by algorithm A) and φ is a time vector of individual operations execution. The average relative error for the disturbed data from a set $D(\delta)$ is

$$\Delta(A,\delta,D(\delta)) = \frac{1}{|D(\delta)|}\sum_{\varphi\in D}\frac{C_{max(\pi_\delta^A,\varphi)} - C_{max(\pi_\delta^{TS},\varphi)}}{C_{max(\pi_\delta^{TS},\varphi)}}. \tag{7}$$

If Ω is a set of deterministic instances of the problem, then the resistance coefficient of an algorithm A on the set Ω is defined as follows:

$$S(A,\Omega) = \frac{1}{|\Omega|}\sum_{\delta\in\Omega}\Delta(A,\delta,D(\delta)). \tag{8}$$

The lower the value of the coefficient, the greater the resistance to random disturbances of solutions determined by the algorithm A.

5 Computational Experiments

Parallel algorithms: deterministic pTSGPU and fuzzy pFzTSGPU were implemented in C++ with the use of CUDA. The MPI library was used for communication between the GPU computing cards. Computational experiments were performed on HP server equipped with Intel i7 CPU (3.33 GHz) and Tesla GPU S2050 running at 64-bit operating system Linux Ubuntu 10.04.4 LTS. The calculations were made separately for the triangular and trapezoidal representation

of fuzzy numbers. Their goal was to determine the resistance coefficient and to investigate the effect of defuzzification function on its value.

Representation of Triangular Fuzzy Numbers. As the basis there were examples from work [4] for flexible job shop problem taken. For each deterministic instance there was 100 *disturbed* instances determined. They were generated, in accordance with the uniform distribution, in the interval$[\max\{1, \lceil p_i - p_i/3 \rceil\}, \lceil p_i + p_i/6 \rceil]$.

Fuzzy times of operations execution were generated as follows. If p_i $(i = 1, 2, \ldots, o)$ are deterministic times, then the fuzzy times are represented by three $(p_i^{min}, p_i^{med}, p_i^{max})$, where$p_i^{min} = \max\{1, \lceil p_i - p_i/3 \rceil\}$, $p_i^{med} = p_i$, $p_i^{max} = \lceil p_i + p_i/6 \rceil$. For each group of instances there were, in accordance with (7), the values of (minimum and maximum average) parameters Δ established, which are presented in Table 1. The resistance coefficient of pTSGPU algorithm is $S(pTSGPU, \Omega) = 3.41\,\%$, whereas pFzTSGPU algorithm – $S(pFzTSGPU, \Omega) = 3.45\,\%$. Resistances of both algorithms differ very little from each other. It results among others, from the fact that fuzzy values of tasks termination moments (after defuzzification) are little different from the tasks set termination moments for deterministic data.

Table 1. Resistance coefficients of algorithms to data perturbances.

Instance			*pTSGPU*			*pFzTSGPU*		
Name	$n \times m$	*Flex*	Δ_{min}	Δ_{aprd}	Δ_{max}	Δ_{min}	Δ_{aprd}	Δ_{max}
Mk01	10×6	2.09	0.315	0.618	0.861	0.268	0.528	0.775
Mk02	10×6	4.10	0.052	0.292	0.468	0.138	0.298	0.441
Mk03	15×8	3.01	0.086	0.271	0.598	0.165	0.396	0.554
Mk04	15×8	1.91	−0.039	0.350	0.625	0.210	0.396	0.584
Mk05	15×4	1.71	−0.037	0.006	0.069	−0.053	0.012	0.075
Mk06	10×15	3.27	0.725	0.860	1.052	0.506	0.942	1.164
Mk07	20×5	2.83	0.080	0.283	0.418	0.026	0.283	0.490
Mk08	20×10	1.43	0.103	0.161	0.232	0.120	0.230	0.350
Mk09	20×10	2.53	0.159	0.451	0.865	0.047	0.216	0.360
Mk10	20×15	2.98	−0.087	0.011	0.314	0.008	0.148	0.286

Representation of Trapezoidal Fuzzy Numbers. If p_i is the time of execution of i-th operation, then the fuzzy time of its execution $\widehat{p}_i = (p_i^{min}, p_i^{med1}, p_i^{med2}, p_i^{max})$, where $p_i^{min} = \max\{1, \lceil p_i - 3p_i/4 \rceil\}$, $p_i^{med1} = \lceil p_i - p_i/5 \rceil$, $p_i^{med2} = \lceil p_i + p_i/5 \rceil$ and $p_i^{max} = \lceil p_i + 3p_i/4 \rceil$ The disturbed data were generated in the same way as in the case of experiments concerning triangular fuzzy numbers. For each example of deterministic data (including the disturbed data) with the use of NEH [9] algorithm, there was a solution determined. The solution was used when calculating the resistance coefficient (8). One of the purposes of the carried out numerical experiments was to study the influence of defuzzification

function on the resistance of the obtained solutions. Following calculations were done: $S(A_d)$ – resistance of the deterministic algorithm, $S_\Psi(A_f)$ – resistance of fuzzy algorithm with defuzzification function $\Psi \in \{LOM, MOM, COA\}$. In this case, solutions to all three versions of the fuzzy algorithm (average over all test instances: $S_{LOM}(A_f) = 7.64$, $S_{COA}(A_f) = S_{COA}(A_f) = 8.37$) are more resistant to data disturbances than solutions designated by deterministic algorithm ($S(A_d) = 9.07$). The most resistant are the solutions, when as 'the last of maximum' defuzzification function is used.

6 Remarks and Conclusions

In the paper there was a job shop problem with uncertain times of operations execution represented by fuzzy numbers presented. To its solution there was parallel tabu search algorithm for the GPU (in deterministic versions and for fuzzy numbers) implemented. In the design of the algorithm there were blocks eliminating properties used. For each deterministic instance there was 100 random examples of disturbed data generated. On their basis there were resistance coefficients of algorithms to input disturbance generated. There was also the influence of defuzzification functions on the resistance of the algorithm examined.

References

1. Balin, S.: Parallel machine scheduling with fuzzy processing times using a robust genetic algorithm and simulation. Inf. Sci. **161**, 3551–3569 (2011)
2. Bożejko, W., Wodecki, M.: On the theoretical properties of swap multimoves. Oper. Res. Lett. **35**(2), 227–231 (2007)
3. Bożejko, W., Uchroski, M., Wodecki, M.: The new golf neighborhood for the flexible job shop problem. In: Proceedings of the ICCS 2010, Procedia Computer Science 1, pp. 289–296. Elsevier (2010)
4. Bożejko, W., Uchroski, M., Wodecki, M.: Solving the flexible job shop problem on Multi-GPU. In: Proceedings of the ICCS 2012, Procedia Computer Science 9, pp. 378–394. Elsevier (2012)
5. Bożejko, W., Uchroski, M., Wodecki, M.: Block approach to the cyclic flow shop scheduling. Comput. Ind. Eng. **81**, 158–166 (2015)
6. Brandimarte, P.: Routing and scheduling in a flexible job shop by tabu search. Ann. Oper. Res. **41**, 157–183 (1993)
7. Dubois, D., Prade, H.: Theorie des Possibilites. Applications a la representation des connaissances en informatique. Masson, Paris (1988)
8. Hodgson, T.J., King, R.E., Stanfield, P.M.: Ready-time scheduling with stochastic service time. Oper. Res. **45**(5), 779–783 (1997)
9. Nawaz, M., Enscore, E.E., Ham, I.: A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. OMEGA Int. J. Manag. Sci. **11**, 91–95 (1983)
10. Prade, H.: Uzing fuzzy set theory in a scheduling problem. Fuzzy Sets Syst. **2**, 153–165 (1979)
11. Sowiski, R., Hapke, M.: Scheduling under Fuzziness. Physica-Verlag, New York (2000)