DOI: 10.20535/kpi-sn.2019.3.164804
UDC 519.161+519.852+519.687.1

V.V. Romanuke*
Polish Naval Academy, Gdynia, Poland
*corresponding author: romanukevadimv@gmail.com

ACCURACY OF A HEURISTIC FOR TOTAL WEIGHTED COMPLETION TIME MINIMIZATION IN PREEMPTIVE SINGLE MACHINE SCHEDULING PROBLEM BY NO IDLE TIME INTERVALS

Background. A special case of the job scheduling process is that when jobs are processed on a single machine, preemptions are allowed, and there are no idle time intervals. Despite the exact solution models are always much slower than the heuristics, laws of heuristic's rapidness advantage and heuristic's solution closeness to the exact solution are unknown. Such laws would be useful to estimate real benefits of solution approximation.
Objective. Issuing from the lack of knowledge in relationship between heuristics and exact solution models, the goal is to study statistical difference between them for the preemptive single machine scheduling problem by no idle time intervals.
Methods. The two well-known approaches are invoked – the rule of weighted shortest remaining processing period as a heuristic and the Boolean linear programming model as an exact model. The relative error of the heuristic is defined and then studied how it varies against increasing complexity of job scheduling problems. The heuristic's rapidness gain is shown as well.
Results. The main issue with the heuristic's accuracy can arise at a few jobs to be scheduled. Additionally to this, if a sequence of jobs is divided into the fewest parts, the heuristic's accuracy becomes the lowest. The exception exists for the shortest sequences – when only two jobs are to be scheduled. As the number of jobs increases up off 6, the relative error expectedly decreases along with the dramatically growing heuristic's rapidness advantage. Therefore, scheduling a long sequence of jobs is preferable. The top relative error of the heuristic can exceed 6 % for three to five jobs to be scheduled, when they are divided into the fewest parts.
Conclusions. Starting off six jobs, the heuristic's accuracy averagely increases, by a fixed rate of randomness in processing periods, priority weights, and release dates, as the complexity of job scheduling problems increases. The rate of randomness influences inversely: if processing periods, priority weights, and release dates are more randomly scattered, the heuristic schedules more accurately. The exact approach is truly applicable for cases when three to five jobs are to be scheduled (in particular cases, when the number of job parts is constant and is 2, the upper number of jobs can be increased up to 10). For such cases, an approximate solution's real loss (given by the heuristic) is the average relative error not exceeding 1.2 % for job scheduling problems with low rate of the randomness. If such a loss is not admissible, the exact approach will work instead.

Keywords: job scheduling; preemptive single machine scheduling; exact model; heuristic; total weighted completion time; heuristic's accuracy; heuristic's rapidness advantage.

Introduction

Job scheduling is the process of allocating/assigning tasks to be processed on machines. A special case is that when jobs are processed on a single machine [1, 2]. The minimal number of jobs is 2, and each job has an arbitrary processing time/period, release date, and priority weight. Going with it a little bit wider, preemptions are allowed, which means that the processing of any job can be interrupted at any time and any number of times in favor of other jobs [3]. Another special condition/requirement is that there are no idle time intervals [1, 3, 4].

The preemptive single machine scheduling problem of minimizing total weighted completion time with arbitrary processing periods and release dates is an important NP-hard problem in the scheduling theory [1, 2]. Obtaining an exact solution to this problem becomes very resource-consuming (implying processor clock speed, memory space, and time) even for a few jobs [1, 5]. For a few tens of jobs, the problem becomes intractable even for the fastest and most powerful processors [1, 4, 6, 7].

Processing periods and release dates are often given as integers. This opens a way to solve the respective integer linear programming problem. Models based on the branch-and-bound approach are commonly used for that [6, 8]. Along with models of obtaining an exact solution, there are a lot of heuristics allowing to find an approximate solution [1, 2, 4, 7]. Computational studies claim that those heuristics are extremely rapid compared to the exact solution models. Besides, a heuristic approximate solution appears very close to the exact one [1, 4].

© The Author(s).
The article is distributed under the terms of the license CC BY 4.0.

Despite the exact solution models are always much slower than the heuristics, laws of heuristic's rapidness advantage and heuristic's solution closeness to the exact solution are unknown. Such laws would be useful to estimate real benefits of solution approximation. What is the actual difference between an exact approach and a heuristic? Are there any artifacts in the difference? Whether could we apply the exact model at all? If not (at least for a definite set of input parameters), then it is no sense to consider and develop such exact models for the mentioned job scheduling problem. These questions are open and waiting to be addressed.

Problem statement

Issuing from the lack of knowledge in relationship between heuristics and exact solution models, the goal is to study statistical difference between them for the preemptive single machine scheduling problem by no idle time intervals. The two well-known approaches will be invoked for this — the rule of weighted shortest remaining processing period as a heuristic and the Boolean linear programming model as an exact model [1, 6, 8]. The relative error of the heuristic will be defined and then studied how it varies against increasing complexity of job scheduling problems. The heuristic's rapidness gain is going to be shown as well. The research result should answer the questions of when the exact approach is truly applicable, and what an approximate solution's real loss (given by the heuristic) is.

Boolean linear programming model

Let N be a number of jobs, $N \in \mathbb{N} \setminus \{1\}$, where job n is divided into H_n equal parts (i. e., has a processing period H_n), has a release date r_n, and a priority weight w_n, $n = \overline{1,\ N}$. So,

$$\mathbf{H} = [H_n]_{1\times N} \in \mathbb{N}^N \tag{1}$$

is a vector of processing periods,

$$\mathbf{W} = [w_n]_{1\times N} \in \mathbb{N}^N \tag{2}$$

is a vector of priority weights, and

$$\mathbf{R} = [r_n]_{1\times N} \in \mathbb{N}^N \tag{3}$$

is a vector of release dates. However, whereas r_n is the time moment, at which job n becomes available for processing, condition of "the proper start", which is

$$\exists\, n_1 \in \{\overline{1,\ N}\} \text{ such that } r_{n_1} = 1, \tag{4}$$

holds. This is the first additional constraint to release dates (3). The second one comes from that there are no idle time intervals, i. e. having sorted components of $\mathbf{R}$ in ascending order to vector $\overline{\mathbf{R}} = [\overline{r}_n]_{1\times N}$, vector (1) becomes respectively sorted after $\overline{\mathbf{R}} = [\overline{r}_n]_{1\times N}$ to $\overline{\mathbf{H}} = [\overline{H}_n]_{1\times N}$ and condition

$$1 + \sum_{n=1}^{k} \overline{H}_n \geq \overline{r}_{k+1} \quad \forall\, k = \overline{1,\ N-1} \tag{5}$$

holds.

The goal is to minimize the total weighted completion time, i. e. to schedule the jobs so that sum

$$\sum_{n=1}^{N} w_n \theta(n;\, H_n) \tag{6}$$

would be minimal, where job n is completed after moment $\theta(n;\, H_n)$, which is

$$\theta(n;\, H_n) \in \{\overline{1,\ T}\} \text{ by } T = \sum_{n=1}^{N} H_n\,. \tag{7}$$

This goal is equivalent to minimizing sum

$$\sum_{n=1}^{N} \sum_{h_n=1}^{H_n} \sum_{t=1}^{T} \lambda_{nh_nt} x_{nh_nt} \tag{8}$$

by the known Boolean linear programming model [6, 8], where x_{nh_nt} is the decision variable about assigning the h_n-th part of job n to time moment t: $x_{nh_nt} = 1$ if it is assigned; $x_{nh_nt} = 0$ otherwise. The triple-indexed weights are calculated as follows:

$$\lambda_{nh_nt} = 0 \tag{9}$$

by

$$r_n - 1 + h_n \leq t \leq T - H_n + h_n \quad \forall\, h_n = \overline{1,\ H_n - 1} \tag{10}$$

and

$$\lambda_{nh_nt} = \alpha \tag{11}$$

by sufficiently great positive integer α (similar to the meaning of infinity) when (10) is not true;

$$\lambda_{nH_nt} = w_n t \tag{12}$$

by

$$r_n - 1 + H_n \leq t \leq T \tag{13}$$

and

$$\lambda_{nH_nt} = \alpha \tag{14}$$

when (13) is not true; for instance,

$$\alpha = \sum_{n=1}^{N}\sum_{t=1}^{T} w_n t \tag{15}$$

can be used. So, sum (8) is defined on set

$$X = \left\{\left\{\{x_{nh_nt}\}_{n=1}^{N}\right\}_{h_n=1}^{H_n}\right\}_{t=1}^{T} \in \mathscr{X},$$

where $\mathscr{X}$ is a set of all possible versions of the decision variables' set. The goal is to find such a set

$$X^* = \left\{\left\{\{x^*_{nh_nt}\}_{n=1}^{N}\right\}_{h_n=1}^{H_n}\right\}_{t=1}^{T} \in \mathscr{X} \tag{16}$$

on which minimum

$$\min_{X\in\mathscr{X}} \sum_{n=1}^{N}\sum_{h_n=1}^{H_n}\sum_{t=1}^{T} \lambda_{nh_nt} x_{nh_nt} \tag{17}$$

is achieved by constraints constituting set $\mathscr{X}$ (an integer binary lattice):

$$\begin{aligned} &x_{nh_nt} \in \{0, 1\} \text{ by } n = \overline{1, N} \\ &\text{and } h_n = \overline{1, H_n} \text{ and } t = \overline{1, T}, \end{aligned} \tag{18}$$

$$\sum_{t=1}^{T} x_{nh_nt} = 1 \text{ by } n = \overline{1, N} \text{ and } h_n = \overline{1, H_n}, \tag{19}$$

$$\sum_{n=1}^{N}\sum_{h_n=1}^{H_n} x_{nh_nt} = 1 \text{ by } t = \overline{1, T}, \tag{20}$$

$$\begin{aligned} &\sum_{j=t+1}^{T}\sum_{h_n=1}^{H_n-1} x_{nh_nj} + H_n x_{nH_nt} \le H_n \\ &\text{by } n = \overline{1, N} \text{ and } t = \overline{1, T-1}. \end{aligned} \tag{21}$$

If (16) is a solution of the problem, it is the optimal job schedule $\mathbf{S}^* = [s_t^*]_{1\times T}$ by $s_t^* \in \{\overline{1, N}\}$ for every $t = \overline{1, T}$. Then

$$\rho^*(N) = \sum_{n=1}^{N}\sum_{h_n=1}^{H_n}\sum_{t=1}^{T} \lambda_{nh_nt} x^*_{nh_nt} \tag{22}$$

is the exact total weighted completion time for those N jobs.

A heuristic approach

The heuristic is an online scheduling algorithm, which applies the rule of weighted shortest remaining processing period [1]. Let

$$\mathbf{Q} = [q_n]_{1\times N} = \mathbf{H} = [H_n]_{1\times N} \tag{23}$$

be a starting vector containing the remaining processing periods. Later on, elements of vector (23) will be decreased as time t progresses. Denote by $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$ the whole set of jobs scheduled by the algorithm, where $\tilde{s}_t \in \{\overline{1, N}\}$ for every $t = \overline{1, T}$. A set of available jobs

$$A(t) = \{i \in \{\overline{1, N}\} : r_i \le t \text{ and } q_i > 0\} \subset \{\overline{1, N}\} \tag{24}$$

gives a set of ratios

$$\left\{\frac{w_i}{q_i}\right\}_{i\in A(t)}, \tag{25}$$

whence the maximal ratio is achieved at subset

$$A^*(t) = \arg\max_{i\in A(t)} \frac{w_i}{q_i}. \tag{26}$$

If $|A^*(t)| = 1$, where

$$A^*(t) = \{i^*\} \subset A(t) \subset \{\overline{1, N}\},$$

then

$$\tilde{s}_t = i^* \text{ by } q_{i^*}^{(\text{obs})} = q_{i^*} \text{ and } q_{i^*} = q_{i^*}^{(\text{obs})} - 1; \tag{27}$$

otherwise

$$A^{**}(t) = \arg\max_{i^*\in A^*(t)} w_{i^*} \subset A^*(t) \subset A(t)$$

and

$$A^{**}(t) = \{i_l^{**}\}_{l=1}^{L} \subset A^*(t) \subset A(t) \subset \{\overline{1, N}\},$$

whence

$$\tilde{s}_t = i_1^{**} \text{ by } q_{i_1^{**}}^{(\text{obs})} = q_{i_1^{**}} \text{ and } q_{i_1^{**}} = q_{i_1^{**}}^{(\text{obs})} - 1. \tag{28}$$

Then an approximate total weighted completion time is calculated successively for every $n = \overline{1, N}$ as follows: if

$$\tilde{s}_{\tilde{\theta}(n;\, h_n)} = n \quad \forall\, h_n = \overline{1, H_n},$$

then job n is completed after moment $\tilde{\theta}(n; H_n)$.

Finally,

$$\tilde{\rho}(N) = \sum_{n=1}^{N} w_n \tilde{\theta}(n;\, H_n) \qquad (29)$$

is an approximately minimal total weighted completion time that corresponds to the nearly optimal job schedule $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$. This schedule often coincides with the job schedule produced by exact solution (16):

$$\rho^*(N) = \sum_{n=1}^{N} w_n\, \theta^*(n;\, H_n)\,, \qquad (30)$$

where $\theta^*(n;\, H_n)$ is a moment after which job n is completed, i. e.

$$s^*_{\theta^*(n;\, h_n)} = n \quad \forall\, h_n = \overline{1,\, H_n}\,.$$

Obviously, if amounts (29) and (30) are equal, a difference between schedules $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$ and $\mathbf{S}^* = [s^*_t]_{1\times T}$ does not matter. Difference between total weighted completion times (30) and (29) is a matter of scrupulous interest.

Relative error of the heuristic

Generally,

$$\tilde{\rho}(N) \geq \rho^*(N) \qquad (31)$$

but computation time $\tilde{\tau}(N)$ of the heuristic is always far less than computation time $\tau^*(N)$ of achieving minimum (17) by (18)–(21):

$$\tilde{\tau}(N) < \tau^*(N)\,. \qquad (32)$$

Therefore, relative error

$$\varepsilon(N) = \frac{\tilde{\rho}(N) - \rho^*(N)}{\rho^*(N)} \qquad (33)$$

and computation time ratio

$$\gamma(N) = \frac{\tau^*(N)}{\tilde{\tau}(N)} \qquad (34)$$

make sense to be considered for cases when inequality (31) is strict.

Counterexamples of the heuristic job scheduling

Consider a problem of scheduling five jobs by the following parameters:

$$\mathbf{H} = [H_n]_{1\times 5} = [2 \quad 1 \quad 3 \quad 2 \quad 1]\,,$$

$$\mathbf{W} = [w_n]_{1\times 5} = [2 \quad 2 \quad 1 \quad 4 \quad 1]\,,$$

$$\mathbf{R} = [r_n]_{1\times 5} = [1 \quad 2 \quad 2 \quad 2 \quad 1]\,.$$

The heuristic schedule

$$\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 9} = [1 \quad 4 \quad 4 \quad 1 \quad 2 \quad 5 \quad 3 \quad 3 \quad 3]$$

is obtained rapidly (in about 100 microseconds), and its total weighted completion time is

$$\tilde{\rho}(5) = \sum_{n=1}^{5} w_n \tilde{\theta}(n;\, H_n)$$
$$= 2\cdot 4 + 2\cdot 5 + 1\cdot 9 + 4\cdot 3 + 1\cdot 6 = 45. \qquad (35)$$

The exactly optimal schedule

$$\mathbf{S}^* = [s^*_t]_{1\times 9} = [5 \quad 4 \quad 4 \quad 2 \quad 1 \quad 1 \quad 3 \quad 3 \quad 3]$$

is obtained a way slower (in about 80 times), but its total weighted completion time is

$$\rho^*(5) = \sum_{n=1}^{5} w_n\, \theta^*(n;\, H_n)$$
$$= 2\cdot 6 + 2\cdot 4 + 1\cdot 9 + 4\cdot 3 + 1\cdot 1 = 42. \qquad (36)$$

So, here for total weighted completion times (35) and (36) the relative error

$$\varepsilon(5) = \frac{\tilde{\rho}(5) - \rho^*(5)}{\rho^*(5)} = \frac{45 - 42}{42} = \frac{1}{14} \qquad (37)$$

is pretty noticeable, although computation time ratio (34) is quite great. Nonetheless, when it is not critical to spend up to 10 milliseconds for an exact solution, the heuristic (23)–(29) for this example would be worse than the exact solution by Boolean linear programming model (1)–(22).

If we modify the scheduling problem just in the third job processing period to

$$\mathbf{H} = [H_n]_{1\times 5} = [2 \quad 1 \quad 1 \quad 2 \quad 1]\,,$$

then the relative error becomes even greater than (37):

$$\varepsilon(5) = \frac{\tilde{\rho}(5) - \rho^*(5)}{\rho^*(5)} = \frac{43 - 40}{40} = 0.075 > \frac{1}{14}\,. \qquad (38)$$

Relative error (38), i. e. the error in 7.5 %, may be critical in some cases.

Another counterexample with five jobs is peculiar in that all the job processing periods are equal:

$$\mathbf{H} = [H_n]_{1\times 5} = [2 \quad 2 \quad 2 \quad 2 \quad 2]\,,$$

$$\mathbf{W} = [w_n]_{1\times 5} = [6 \quad 1 \quad 12 \quad 2 \quad 24],$$

$$\mathbf{R} = [r_n]_{1\times 5} = [3 \quad 1 \quad 4 \quad 2 \quad 5].$$

The heuristic schedule

$$\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 10} = [2 \quad 4 \quad 1 \quad 3 \quad 5 \quad 5 \quad 3 \quad 1 \quad 4 \quad 2]$$

differs from the exactly optimal schedule

$$\mathbf{S}^* = [s_t^*]_{1\times 10} = [2 \quad 2 \quad 1 \quad 1 \quad 5 \quad 5 \quad 3 \quad 3 \quad 4 \quad 4]$$

in four positions again, but this time the relative error is

$$\varepsilon(5) = \frac{\tilde{\rho}(5) - \rho^*(5)}{\rho^*(5)} = \frac{304 - 286}{286} = \frac{9}{143} \approx 0.0629. \qquad (39)$$

Almost the same relative error exists in the example with seven jobs:

$$\mathbf{H} = [H_n]_{1\times 7} = [2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2],$$

$$\mathbf{W} = [w_n]_{1\times 7} = [6 \quad 6 \quad 2 \quad 1 \quad 6 \quad 3 \quad 3],$$

$$\mathbf{R} = [r_n]_{1\times 7} = [5 \quad 7 \quad 2 \quad 1 \quad 6 \quad 4 \quad 3].$$

Here the heuristic schedule $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 14}$ differs from the exactly optimal schedule $\mathbf{S}^* = [s_t^*]_{1\times 14}$ in 10 positions, and the relative error remains significant:

$$\varepsilon(7) = \frac{\tilde{\rho}(7) - \rho^*(7)}{\rho^*(7)} = \frac{236 - 222}{222} = \frac{7}{111} \approx 0.0631. \qquad (40)$$

Just like in the previous case, we have an error in 6.3 % once again. Despite relative error (40) for seven jobs stands against the same value (39) for five ones, a greater number of jobs does not necessarily make the respective relative error less significant.

Those four counterexamples might serve as the basics for a few classes of the job scheduling problem, wherein the heuristic works poorly. Hence, it is common to learn some statistics of the heuristic's accuracy. It will help in revealing "weak places" of the heuristic.

An analysis of heuristic's advantage

First of all, if every job consists of a single part, total weighted completion times (30) and (29) are the same, although the heuristic's and the exact model's schedules may differ (in particular, jobs with identical priority weights may be permuted). It is very easy to check this for up to 100 jobs and even more owing to that finding solution (16) by the Boolean linear programming model is relatively very rapid for the case when the number of jobs is equal to the grand total of processing periods T in (7). Henceforward, learning statistics of the heuristic's accuracy should be started off the cases wherein, along with (1),

$$H_n \geq 1 \quad \forall n = \overline{1, N} \text{ but } \exists m \in \{\overline{1, N}\} \text{ such that } H_m > 1. \qquad (41)$$

For obtaining credible results of a statistical analysis, a few series of job scheduling problems should be generated with randomizers. The generators for processing periods (1), priority weights (2), and release dates (3) are constructed as follows. Processing periods (1) are

$$\mathbf{H} = [H_n]_{1\times N} = \psi(|\beta_{\mathbf{H}}(N)\Omega(1, N)| + 1) \qquad (42)$$

where function $\Omega(1, N)$ returns a pseudorandom $1\times N$ vector [9, 10] whose entries are drawn from the standard normal distribution (with zero mean and unit variance), function $\psi(\xi)$ returns the integer part of number ξ (e. g., see [11]), $\beta_{\mathbf{H}}(N)$ is a positive factor depending on the number of jobs. Priority weights (2) are

$$\mathbf{W} = [w_n]_{1\times N} = \psi(\eta\Theta(1, N) + 1) \qquad (43)$$

where function $\Theta(1, N)$ returns a pseudorandom $1\times N$ vector whose entries are drawn from the standard uniform distribution on the open interval $(0; 1)$, η is a positive factor. Release dates (3) are taken as

$$\mathbf{R} = [r_n]_{1\times N} = \psi(|\beta_{\mathbf{R}}(N)\Omega(1, N)| + 1) \qquad (44)$$

where $\beta_{\mathbf{R}}(N)$ is a positive factor depending on the number of jobs, if only conditions (4) and (5) hold. If one of them does not hold, vector (44) is re-generated until conditions (4) and (5) hold.

As N increases, growth of factors $\beta_{\mathbf{H}}(N)$ and $\beta_{\mathbf{R}}(N)$ must be decreasing. Therefore, they can be

$$\beta_{\mathbf{H}}(N) = 5 - \frac{8}{N} \text{ and } \beta_{\mathbf{R}}(N) = 3 - \frac{9}{2N} \text{ for } N = \overline{2, 10}. \qquad (45)$$

Fig. 1 prompts that factor η can be set at 5. A result of using such factors for the number of jobs not exceeding 10 is in Fig. 2.

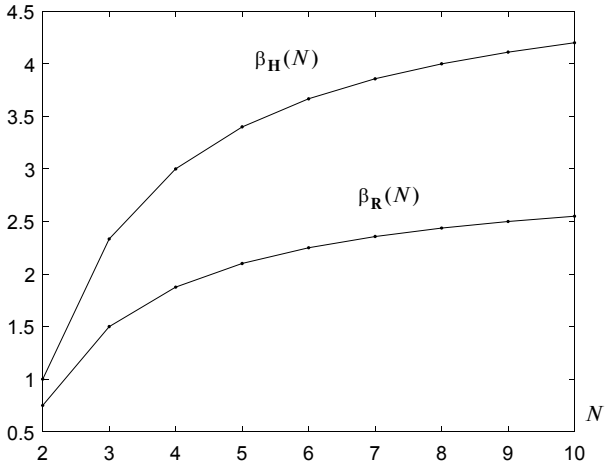

Fig. 1. Factors (45) for the generators in (42) and (44)

```
2   1
2   1
2   1

1   1   1
5   1   1
2   1   3

3   3   1   1
4   1   4   1
3   1   2   1

2   5   2   5   3
1   3   2   5   1
3   1   1   2   2

5   2   2   7   2   4
5   2   1   3   5   1
1   1   4   1   3   2

1   3   5   5   8   3   2
5   2   5   4   4   1   1
1   1   4   1   3   3   5

4   5   1   7   11  14  4   6
4   1   1   4   1   3   2   1
3   4   1   2   1   3   3   1

10  3   5   8   7   1   3   1   2
2   1   5   5   3   5   1   5   1
1   4   2   1   1   6   2   5   2

2   2   7   3   3   2   4   4   8   6
5   5   3   4   3   3   2   1   4   1
2   1   5   1   4   2   3   4   4   1
```

Fig. 2. A series of processing periods (1), priority weights (2), and release dates (3), constructed by (42), (43), (44), and by (45) and $\eta = 5$ for $N = \overline{2,10}$

Less intensive job sequences are generated by dividing factors (45) by 1.75 and 1.25, 2 and 1.5, 3 and 2, respectively. A series of four instances of processing periods (1), priority weights (2), and release dates (3), constructed by such intensity decrements, are shown in Fig. 3.

```
1   1   1   1   1   2   1   2   1   2
4   3   2   2   1   1   1   3   1   1
1   1   1   1   1   1   2   2   2   2

1   1   3   2   3   1   2   2   1   1
2   1   4   5   5   3   1   4   2   1
2   1   1   1   2   1   2   3   1   2

1   1   2   2   1   3   2   2   1   1
3   2   2   5   3   2   5   4   4   1
1   1   1   1   2   1   1   2   1   1

2   1   1   1   2   3   1   3   1   2
4   1   1   4   2   4   2   5   3   1
1   2   1   1   1   2   1   1   1   2
-----------------------------------
4   1   4   2   1   2   2   6   1   3
4   1   2   1   5   5   2   2   3   1
2   2   1   2   4   1   1   2   2   1

4   2   1   4   1   1   1   1   1   7
2   2   4   2   1   1   2   1   1   1
4   1   2   1   1   1   1   1   1   1

1   2   2   1   4   2   5   2   1   2
1   3   3   5   4   1   2   1   1   1
1   1   1   1   1   1   2   3   1   1

4   4   2   4   2   2   1   1   2   3
4   4   4   3   1   2   5   3   5   1
1   1   1   1   3   2   1   2   2   2
-----------------------------------
3   2   1   2   1   2   1   1   5   2
1   4   4   1   2   5   3   4   1   1
1   2   4   4   5   1   2   4   3   4

1   2   1   3   2   1   3   3   1   3
4   1   3   2   3   4   5   3   2   1
1   1   1   1   1   2   2   1   2   2

4   2   2   1   1   1   3   2   4   1
4   3   5   4   5   5   5   1   1   1
2   1   2   1   2   4   1   2   1   4

1   3   2   1   3   7   3   1   2   2
3   1   3   3   3   1   2   4   3   1
1   4   1   3   1   1   3   1   4   1
```

Fig. 3. A series of four instances of processing periods (1), priority weights (2), and release dates (3), constructed by dividing factors (45) by 3 and 2, 2 and 1.5, 1.75 and 1.25, respectively from top to below (separated with dashes), at $\eta = 5$ for $N = 10$

Apart from the mentioned parameters for generating job scheduling problems, a very interesting case is when we have an identical number of job parts, each job has its unique release date, and the later job has a greater weight. For instance,

$$H_n = 2 \quad \forall\, n = \overline{1,\, N} \tag{46}$$

by

$$\begin{gathered} w_n \le w_{n+1} \;\; \forall\, n = \overline{1,\, N-1} \\ \text{but } \exists\, m \in \{\overline{1,\, N-1}\} \text{ such that } w_m < w_{m+1}, \end{gathered} \tag{47}$$

and

$$\mathbf{R} = [r_n]_{1\times N} = [n]_{1\times N}. \tag{48}$$

Such job scheduling problems are to be generated by

$$\eta \in \{5, 10, 15, 20, 25\} \tag{49}$$

for the priority weights with properties (47).

Fig. 4 reveals an average of relative error (33) for nine types of job scheduling problems generated by the following parameters:

1) factors (45) are divided by 3 and 2, respectively, and $\eta = 5$;

2) factors (45) are divided by 2 and 1.5, respectively, and $\eta = 5$;

3) factors (45) are divided by 1.75 and 1.25, respectively, and $\eta = 5$;

4) factors (45) are used as they are, and $\eta = 5$;

5) the case with (46)–(48) by $N = \overline{2, 10}$ and $\eta = 5$;

6) the case with (46)–(48) by $N = \overline{2, 10}$ and $\eta = 10$;

7) the case with (46)–(48) by $N = \overline{2, 10}$ and $\eta = 15$;

8) the case with (46)–(48) by $N = \overline{2, 10}$ and $\eta = 20$;

9) the case with (46)–(48) by $N = \overline{2, 10}$ and $\eta = 25$.

As it is clearly seen, randomly generated job scheduling problems by (42)–(44) are solved more accurately by the heuristic. The average relative error does not exceed 0.2 %. On the contrary, job scheduling problems with a way lesser randomness, where only priority weights are generated random by (43), are not always solved accurately enough. For such cases, the average relative error exceeds 1.2 % which may be a significant loss. In particular cases, the relative error easily jumps beyond 6 %. Indeed, just for a few instances with three jobs to be scheduled, the following results are very "discreditable" for the heuristic:

$$\mathbf{W} = [w_n]_{1\times 3} = [1 \quad 2 \quad 4] \text{ by } \eta = 5,$$
$$\varepsilon(3) = \frac{\tilde{\rho}(3) - \rho^*(3)}{\rho^*(3)} = \frac{32 - 30}{30} = \frac{1}{15} \approx 0.0667; \tag{50}$$

$$\mathbf{W} = [w_n]_{1\times 3} = [2 \quad 4 \quad 9] \text{ by } \eta = 10,$$
$$\varepsilon(3) = \frac{\tilde{\rho}(3) - \rho^*(3)}{\rho^*(3)} = \frac{68 - 64}{64} = \frac{1}{16} \approx 0.0625; \tag{51}$$

$$\mathbf{W} = [w_n]_{1\times 3} = [2 \quad 4 \quad 8] \text{ by } \eta = 15,$$
$$\varepsilon(3) = \frac{\tilde{\rho}(3) - \rho^*(3)}{\rho^*(3)} = \frac{64 - 60}{60} = \frac{1}{15} \approx 0.0667; \tag{52}$$

$$\mathbf{W} = [w_n]_{1\times 3} = [4 \quad 8 \quad 17] \text{ by } \eta = 20,$$
$$\varepsilon(3) = \frac{\tilde{\rho}(3) - \rho^*(3)}{\rho^*(3)} = \frac{132 - 124}{124} = \frac{2}{31} \approx 0.0645; \tag{53}$$

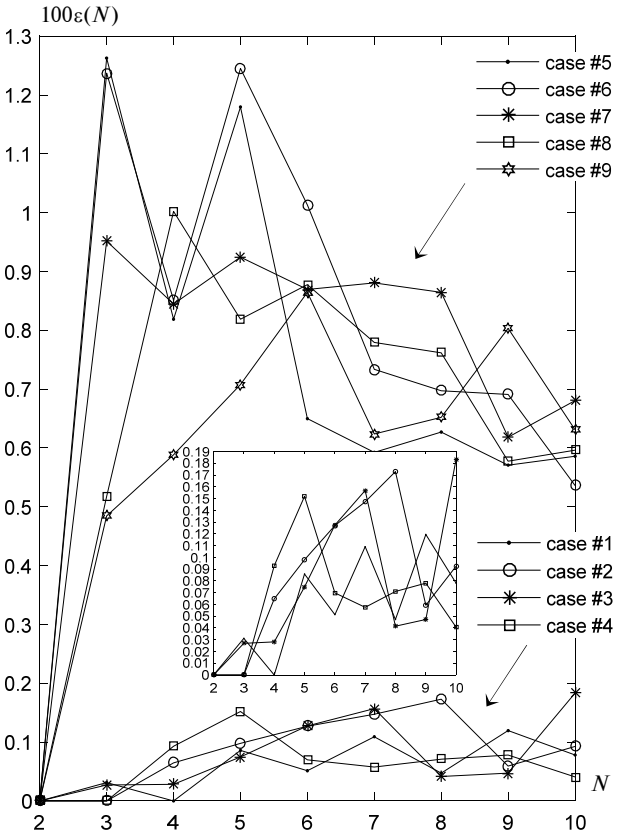

Fig. 4. An average of relative error (33) given in percentages for the nine types of job scheduling problems; despite examples (50)–(57) are particular occurrences of such a great relative error, the likelihood of such cases cannot be excluded

$$\mathbf{W} = [w_n]_{1\times 3} = [5 \quad 10 \quad 21] \text{ by } \eta = 25,$$
$$\varepsilon(3) = \frac{\tilde{\rho}(3) - \rho^*(3)}{\rho^*(3)} = \frac{164 - 154}{154} = \frac{5}{77} \approx 0.0649. \quad (54)$$

Statistically, scheduling three and five jobs is the most "vulnerable" when using the heuristic. Despite scheduling five jobs by the heuristic fails to be accurate rarely, its "vulnerability" is still impressive by $\eta = 25$, e. g.

$$\mathbf{W} = [w_n]_{1\times 5} = [3 \quad 6 \quad 11 \quad 12 \quad 24],$$
$$\varepsilon(5) = \frac{\tilde{\rho}(5) - \rho^*(5)}{\rho^*(5)} = \frac{375 - 350}{350} = \frac{1}{14} \approx 0.0714; \quad (55)$$

$$\mathbf{W} = [w_n]_{1\times 5} = [3 \quad 7 \quad 13 \quad 13 \quad 23],$$
$$\varepsilon(5) = \frac{\tilde{\rho}(5) - \rho^*(5)}{\rho^*(5)} = \frac{394 - 370}{370} = \frac{12}{185} \approx 0.0649; \quad (56)$$

$$\mathbf{W} = [w_n]_{1\times 5} = [5 \quad 10 \quad 19 \quad 19 \quad 22],$$
$$\varepsilon(5) = \frac{\tilde{\rho}(5) - \rho^*(5)}{\rho^*(5)} = \frac{500 - 470}{470} = \frac{3}{47} \approx 0.0638. \quad (57)$$

Nevertheless, scheduling a greater number of jobs (7, 8, 9, 10, and so on) appears more accurate (then the maximal relative ratio does not exceed 5 %).

Fig. 5 reveals that the heuristic's rapidness gain by computation time ratio (34) grows dramatically: since $N > 5$ it is hardly comparable to the solution procedure by the Boolean linear programming model. The heuristic is averagely at least 100 times faster. Three jobs are scheduled on average 580 times faster, whereas five jobs are scheduled beyond 10000 times faster. Besides, the higher randomness in processing periods (42), priority weights (43), and release dates (44) facilitates in the faster solving. However, the heuristic's rapidness gain for three jobs is not so perfect ever: there are cases when it drops to about 100 down to 10.

Fig. 4 hints at that the constant processing period (46) and unique monotonously increasing release dates (48) are the "weak places" of the heuristic. When the constant processing period is increased to

$$H_n = 3, \; H_n = 4, \; H_n = 5 \; (n = \overline{1, N}), \quad (58)$$

the "weakness" gradually disappears (Fig. 6). Such an outcome truly does depend on (49), but the dependence is weak itself.

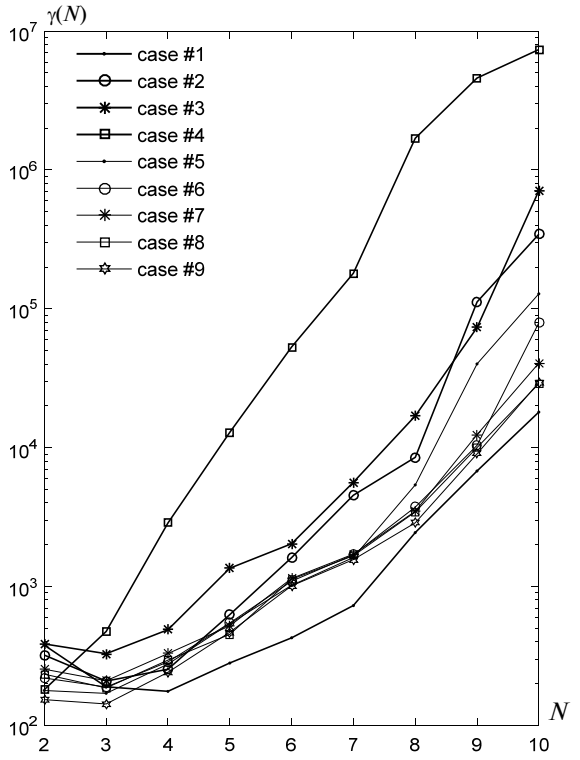

Fig. 5. An average of computation time ratio (34) for the nine types of job scheduling problems; apart from formally confirming inequality (32), this example shows the heuristic's advantage in its rapidness

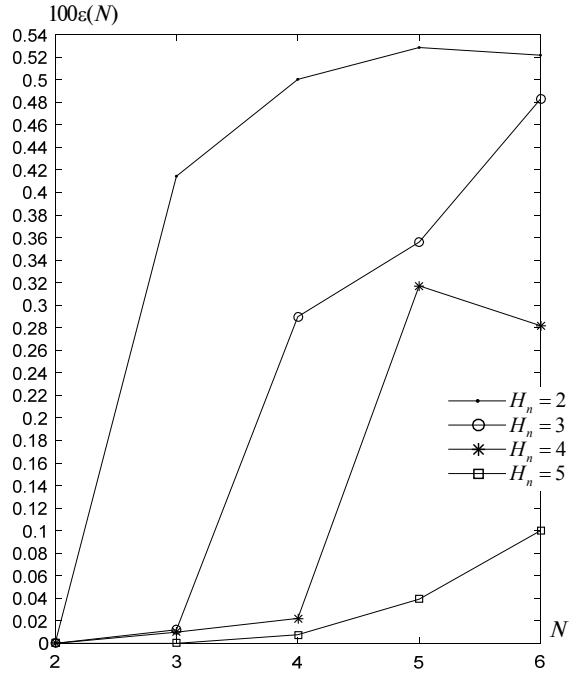

Fig. 6. An average of relative error (33) given in percentages for $\eta = 25$ in priority weights (43) by a constant processing period (46), (58), and release dates (48)

It is worth to note that whereas the "weakest" job scheduling problem is of three jobs, it has also the least rapidness gain (see Fig. 5) by computation time ratio (34). But at the higher randomness in processing periods (42), priority weights (43), and release dates (44) the slowest heuristic is expected for job scheduling problems with two jobs.

Discussion

The graphical results in Figs. 4–6 are quite credible owing to good enough averaging (it has been used 100 repetition cycles of generating a one job scheduling problem). If the generations were repeated all over again, similar graphics would be obtained, although the peaks in Fig. 4 and polylines in Fig. 6 might then be slightly displaced. The displacement would likely be perceptible but it would not break the general tendency.

Obviously, the statistical analysis carried out by generating both random job scheduling problems and problems wherein the later job has a greater weight by a constant processing period might have been continued: a class of randomizers could be widened, and the constant processing period could be increased up to a few tens. However, even the factually obtained results visualized in Figures 4 and 6 allow to confidentially claim that the main issue with the heuristic's accuracy can arise at a few jobs to be scheduled. Additionally to this, if a sequence of jobs is divided into the fewest parts, the heuristic's accuracy becomes the lowest. The exception exists for the shortest sequences – when only two jobs are to be scheduled. As the number of jobs increases up off 6, the relative error expectedly decreases along with the dramatically growing computation time ratio (34). Therefore, scheduling a long sequence of jobs is preferable, although it does not mean that we can just "glue" together a few short scheduling problems into a longer one.

Conclusions

In approximately solving the preemptive single machine scheduling problem by no idle time intervals, the top relative error of the heuristic can exceed 6 % for three to five jobs to be scheduled, when they are divided into the fewest parts. As a triviality, the heuristic schedules two jobs exactly, with 100 % accuracy. Another triviality is that, whichever the number of single-part jobs is, by just excluding case (41), the heuristic schedules them exactly as well.

Starting off six jobs, the heuristic's accuracy averagely increases, by a fixed rate of randomness in processing periods, priority weights, and release dates, as the complexity of job scheduling problems increases. The term "complexity" here can be roughly treated as a computational complexity [12]. Therefore, the heuristic's advantage grows as the job scheduling problem becomes more complicated. The rate of randomness influences inversely: if processing periods, priority weights, and release dates are more randomly scattered, the heuristic schedules more accurately.

The heuristic's rapidness is hardly comparable to that of the exact model even for a few jobs. It has been revealed that the heuristic's rapidness gain by computation time ratio (34) grows dramatically as the number of jobs increases. Meanwhile, the complexity of job scheduling problems is not so influential on the heuristic's rapidness gain.

Hence, the exact approach is truly applicable for cases when three to five jobs are to be scheduled (in particular cases, when the number of job parts is constant and is 2, the upper number of jobs can be increased up to 10). For such cases, an approximate solution's real loss (given by the heuristic) is the average relative error not exceeding 1.2 % for job scheduling problems with low rate of the randomness. If such a loss is not admissible, the exact approach will work instead. Thus, the further research may be focused on possibilities to speed up the computations by the exact approach.

References

[1] H. Belouadah *et al.*, "Scheduling with release dates on a single machine to minimize total weighted completion time", *Discrete Appl. Math.*, vol. 36, iss. 3, pp. 213–231, 1992. doi: 10.1016/0166-218X(92)90255-9

[2] M.L. Pinedo, *Scheduling: Theory, Algorithms, and Systems.* Springer International Publishing, 2016, 670 p. doi: 10.1007/978-3-319-26580-3

[3] M.L. Pinedo, *Planning and Scheduling in Manufacturing and Services.* Springer-Verlag New York, 2009, 536 p. doi: 10.1007/978-1-4419-0910-7

[4] P. Brucker, *Scheduling Algorithms.* Springer-Verlag Berlin Heidelberg, 2007, 371 p. doi: 10.1007/978-3-540-69516-5

[5] W. Tian and Y. Zhao, "Energy efficiency by minimizing total busy time of offline parallel scheduling in cloud computing", in *Optimized Cloud Resource Management and Scheduling*, W. Tian and Y. Zhao, eds. Morgan Kaufmann, 2015, pp. 135–157. doi: 10.1016/B978-0-12-801476-9.00007-0

[6] W.-Y. Ku and J. C. Beck, "Mixed Integer Programming models for job shop scheduling: A computational analysis", *Computers & Operations Research*, vol. 73, pp. 165–173, 2016. doi: 10.1016/j.cor.2016.04.006

[7] A.C. Nearchou, "An efficient meta-heuristic for the single machine common due date scheduling problem", in *Intelligent Production Machines and Systems*, D.T. Pham *et al.*, eds. Elsevier Science Ltd, 2006, pp. 431–435. doi: 10.1016/B978-008045157-2/50077-8

[8] L.P. Fávero and P. Belfiore, "Integer Programming", in *Data Science for Business and Decision Making*, L.P. Fávero and P. Belfiore, eds. Academic Press, 2019, pp. 887–918. doi: 10.1016/B978-0-12-811216-8.00019-7

[9] V.V. Romanuke, "Acyclic-and-asymmetric payoff triplet refinement of pure strategy efficient Nash equilibria in trimatrix games by maximinimin and superoptimality", *Naukovi Visti NTUU KPI*, no. 4, pp. 38–53, 2018. doi: 10.20535/1810-0546.2018.4.131696

[10] R.L. Bowman, "Evaluating pseudo-random number generators", in *Chaos and Fractals*, C. Pickover, Ed. Elsevier Science, 1998, pp. 133–142. doi: 10.1016/B978-044450002-1/50023-0

[11] V.V. Romanuke, "Optimal pixel-to-shift standard deviations ratio for training 2-layer perceptron on shifted 60×80 images with pixel distortion in classifying shifting-distorted objects", *Appl. Comp. Syst.*, vol. 19, pp. 61–70, 2016. doi: 10.1515/acss-2016-0008

[12] M. Mnich and R. van Bevern, "Parameterized complexity of machine scheduling: 15 open problems", *Computers & Operations Research*, vol. 100, pp. 254–261, 2018. doi: 10.1016/j.cor.2018.07.020

В.В. Романюк

ТОЧНІСТЬ ЕВРИСТИКИ ДЛЯ МІНІМІЗАЦІЇ ЗАГАЛЬНОГО ЗВАЖЕНОГО ЧАСУ ЗАВЕРШЕННЯ В ОДНОМАШИННІЙ ЗАДАЧІ ПЛАНУВАННЯ З ПЕРЕМИКАННЯМИ БЕЗ ІНТЕРВАЛІВ ПРОСТОЮ

Проблематика. Особливий випадок процесу планування завдань полягає в тому, що завдання обробляються на одному комп'ютері, дозволені перемикання і немає інтервалів простою. Незважаючи на те що моделі точного розв'язання завжди набагато повільніші, ніж евристики, закони переваги евристик у швидкості та близькість евристичного розв'язку до точного розв'язку невідомі. Такі закони були б корисні для оцінки реальних переваг апроксимації розв'язку.

Мета дослідження. Виходячи з відсутності знань щодо взаємозв'язку між евристиками та моделями точного розв'язку, метою є вивчення статистичної різниці між ними для одномашинної задачі планування з перемиканнями без інтервалів простою.

Методика реалізації. Запропоновано два відомих підходи – правило зваженого найкоротшого залишкового періоду обробки як евристики і моделі булевого лінійного програмування як точної моделі. Визначається відносна похибка евристики, а потім вивчається, як вона змінюється залежно від зростаючої складності задач планування завдань. Також показується виграш евристики у швидкості.

Результати дослідження. Основна проблема з точністю евристики може виникнути за незначної кількості завдань, які мають бути розплановані. Крім того, якщо послідовність завдань розділена на малу кількість частин, то точність евристики стає найменшою. Існує виняток для найкоротших послідовностей – коли потрібно розпланувати лише два завдання. Щойно кількість завдань збільшується від 6, відносна похибка очікувано зменшується разом із різко зростаючою перевагою евристики у швидкості. Отже, планування довгої послідовності завдань є кращим. Найвища відносна похибка евристики може перевищувати 6 % для випадку планування від трьох до п'яти завдань, коли вони розділені на малу кількість частин.

Висновки. Починаючи з шести завдань, точність евристики в середньому зростає за фіксованого рівня випадковості в періодах обробки, вагах пріоритетів і датах запусків, щойно зростає складність задач планування завдань. Рівень випадковості впливає у зворотний бік: якщо періоди обробки, ваги пріоритетів і дати запусків розсіяні більш випадково, то евристика розпланує більш точно. Точний підхід є дійсно застосовним у випадках, коли необхідно розпланувати від трьох до п'яти завдань (в окремих випадках, коли кількість частин завдання є сталою і дорівнює 2, верхнє число цих завдань можна збільшити до 10). У таких випадках реальна втрата за наближеним розв'язком (за евристикою) є тією середньою відносною похибкою, що не перевищує 1,2 % для задач планування завдань із низьким рівнем випадковості. Якщо така втрата неприпустима, то замість неї буде працювати точний підхід.

Ключові слова: планування завдань; планування на одній машині з перемиканнями; точна модель; евристика; загальний зважений час завершення; точність евристики; перевага евристики у швидкості.

В.В. Романюк

ТОЧНОСТЬ ЭВРИСТИКИ ДЛЯ МИНИМИЗАЦИИ ОБЩЕГО ВЗВЕШЕННОГО ВРЕМЕНИ ЗАВЕРШЕНИЯ В ОДНОМАШИННОЙ ЗАДАЧЕ ПЛАНИРОВАНИЯ С ПЕРЕКЛЮЧЕНИЯМИ БЕЗ ИНТЕРВАЛОВ ПРОСТОЯ

Проблематика. Особый случай процесса планирования заданий заключается в том, что задания обрабатываются на одном компьютере, разрешены переключения и нет интервалов простоя. Несмотря на то что модели точного решения всегда гораздо медленнее, чем эвристики, законы преимущества эвристик в скорости и близость эвристического решения к точному решению неизвестны. Такие законы были бы полезны для оценки реальных преимуществ аппроксимации решения.

Цель исследования. Исходя из отсутствия знаний о взаимосвязи между эвристиками и моделями точного решения, целью является изучение статистической разницы между ними для одномашинной задачи планирования с переключениями без интервалов простоя.

Методика реализации. Предложены два известных подхода – правило взвешенного кратчайшего остаточного периода обработки как эвристики и модели булевого линейного программирования как точной модели. Определяется относительная погрешность эвристики, а затем изучается, как она меняется в зависимости от возрастающей сложности задач планирования заданий. Также показывается выигрыш эвристики в скорости.

Результаты исследования. Основная проблема с точностью эвристики может возникнуть при небольшом количестве заданий, которые должны быть распланированы. Кроме того, если последовательность заданий разделена на малое количество частей, то точность эвристики становится наименьшей. Существует исключение для коротких последовательностей – когда нужно распланировать только два задания. Как только количество заданий увеличивается от 6, относительная погрешность ожидаемо уменьшается вместе с резко возрастающим преимуществом эвристики в скорости. Следовательно, планирование длинной последовательности заданий предпочтительно. Самая высокая относительная погрешность эвристики может превышать 6 % для случая планирования от трех до пяти заданий, когда они разделены на малое количество частей.

Выводы. Начиная с шести заданий, точность эвристики в среднем растет при фиксированном уровне случайности в периодах обработки, весах приоритетов и датах запусков, как только возрастает сложность задач планирования заданий. Уровень случайности влияет в обратную сторону: если периоды обработки, веса приоритетов и даты запусков рассеяны более случайно, то эвристика планирует более точно. Точный подход действительно применимым в случаях, когда необходимо распланировать от трех до пяти заданий (в отдельных случаях, когда количество частей задания является постоянным и равно 2, верхнее число этих заданий можно увеличить до 10). В таких случаях реальная потеря по приближенному решению (по эвристике) является этой средней относительной погрешностью, не превышающей 1,2 % для задач планирования заданий с низким уровнем случайности. Если такая потеря недопустима, то вместо нее будет работать точный подход.

Ключевые слова: планирование заданий; планирование на одной машине с переключениями; точная модель; эвристика; общее взвешенное время завершения; точность эвристики; преимущество эвристики в скорости.

Рекомендована Радою
факультету прикладної математики
КПІ ім. Ігоря Сікорського

Надійшла до редакції
20 квітня 2019 року

Прийнята до публікації
20 червня 2019 року