DOI: 10.20535/kpi-sn.2019.5-6.178016
UDC 519.161+519.852+519.687.1

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

ACCURATE TOTAL WEIGHTED TARDINESS MINIMIZATION IN TIGHT-TARDY PROGRESSIVE SINGLE MACHINE SCHEDULING WITH PREEMPTIONS BY NO IDLE PERIODS

Background. The problem of minimization of total weighted tardiness can be solved either exactly by the corresponding models or heuristically. As of October 2019, nearly the best heuristic is one based on using remaining available and processing periods. The heuristic is extremely rapid compared to the exact solution models, but its accuracy can be both 100 % and intolerably low.

Objective. Issuing from the lack of knowledge in relationship between the heuristic and Boolean linear programming model provided for exact solutions, the goal is to study statistical difference between them for the preemptive single machine scheduling problem by no idle periods, in which processing periods are equal by progressively running release and due dates set tightly.

Methods. The relative gap of the heuristic is defined and then studied how it varies against increasing complexity of job scheduling problems. The complexity implies the number of jobs and the number of job processing periods. The computation times of the heuristic and the exact model are registered as well.

Results. The heuristic has successfully replaced the exact model no less than in 72 % of non-timeout instances, where it schedules with the same minimal total weighted tardiness (100 % accuracy). This rate is about 90 % to 97 % on average, although huge gaps may appear in the rest of cases. In the practice of fast-refreshable schedules, the Boolean linear programming model is indeed hardly tractable in scheduling no less than 14 two-parted jobs and no less than 10 three-parted jobs. Scheduling jobs divided into a greater number of parts each will have a significantly lower worst gap than scheduling jobs divided into a lesser number of parts. If a job is divisible, it is strongly recommended to divide the job into as great number of its parts as possible. If scheduling only 2 jobs is impossible, it is strongly recommended to artificially increase the number of jobs to be scheduled.

Conclusions. Total weighted tardiness minimization in tight-tardy progressive single machine scheduling with preemptions by no idle periods can be sufficiently accurate by the heuristic if no less than 7 jobs divided into no less than five parts each are scheduled (the “7/5” pattern). An exception from this rule is that the heuristic schedules just 2 jobs always at the 100 % accuracy, not depending on in how many parts the job is divided (the “2/any” exception). An intermediate between the “7/5” pattern and the “2/any” exception is that scheduling 3 jobs divided into either four or five parts is sufficiently accurate as well, where the inaccuracy does not exceed 0.7 %. In other cases the heuristic is either inapplicable or there is a high risk of obtaining intolerable gaps. The inapplicability does not directly imply a bad inaccuracy, but it implies an unpredictable accuracy drop. For example, 974 of 1000 instances of 3 two-parted jobs have been scheduled with the 100 % accuracy, but 26 instances have been scheduled with an average gap in 13.31 %, which is quite intolerable and thus inapplicable.

Keywords: job scheduling; preemptive single machine scheduling; exact model; heuristic; total weighted tardiness; heuristic’s accuracy; relative gap; top worst maximal relative gap.

Introduction

In job scheduling on a single resource/machine, a very important problem arises when some jobs are required to be completed till due dates. Due date is a kind of schedule expiration date, before which the job can be scheduled in any way favorable to the system. If a job is completed after its due date, an additional payment is imposed [1, 2]. The payment can be expressed as financially, as well as by reduction of availability. The purpose is to minimize the additional payments. More formally, it is referred to as minimization of tardiness.

The problem is related to minimization of total weighted completion time [2, 3]. Whereas the latter operates over associating each job with its processing time/period, release date, and priority weight, the tardiness problem includes due dates. Thus, if priority weights differ, the general problem is to minimize total weighted tardiness. A special requirement, by which no idle time intervals are allowed [1, 4, 5], can be attached. Additionally, preemptions can be allowed [6].

It is known that the preemptive single machine scheduling problem of minimizing total tardiness (when jobs have no weights) with arbitrary release and due dates by equal processing periods is polynomially solvable [7]. However, struggling to compute a schedule with the exactly minimal total weighted tardiness even for a few jobs may become very resource-consuming (implying processor clock speed, memory space, and time of computations) [2, 5]. As the number of jobs and the numbers of their processing periods increase, intractability of the problem dramatically grows [5, 8, 9].

Obviously, the minimal number of jobs is 2. Release and due dates are often given as integers. Setting priority weights at integers is always realizable. Therefore, a schedule ensuring the exactly minimal total weighted tardiness can be found with the respective integer linear programming problem. Models based on the branch-and-bound approach are commonly used for that [8, 10]. They resemble those intended to minimize the exact total weighted completion time [8, 9]. Along with models of obtaining an exact solution, there are a lot of heuristics allowing to find an approximate solution, which often coincides with the exact one and ensures thus the minimal total weighted tardiness also [4, 5, 7, 11]. The heuristics operate with the remaining available period [5], remaining slack [5, 11], and remaining processing period [8, 9]. However, the accuracy of the heuristics for minimizing total weighted tardiness is not as high as the accuracy of the rule of weighted shortest remaining processing period for total weighted completion time minimization [5, 11]. For instance, the recently substantiated heuristic, in which the decisive ratio is the priority weight divided by the maximum of a pair of the remaining processing period and remaining available period [5], may produce schedules of a few jobs whose total weighted tardiness is 50 % greater or even worse than the minimum. Nevertheless, the heuristics are extremely rapid compared to the exact solution models [8].

An open question is how close the exact and approximate solutions are for definite types of the scheduling problem. The closeness is meant as the average gap, although maximal gaps are considered as well. Another open question is what computational time is taken to find an approximate schedule by the heuristic compared to the computational time of the exact model. Finally, it would be very useful to learn whether "pathological" cases exist in which the gap is too big, and thus application of the heuristic for such cases is practically impossible. Therefore, real benefits of schedule approximation by the heuristic are to be estimated along with emphasizing cases in which it is inapplicable.

Problem statement

In a way, there are a lot of types of the preemptive single machine scheduling problem with arbitrary release and due dates by equal processing periods. A special class is that which has monotonously increasing both release and due dates. A subclass of this one is that where almost all the jobs are tardy as their due dates are tightly set after the respective release dates, although one job can always be completed without tardiness. Hence, issuing from the lack of knowledge in relationship between the mentioned heuristic and Boolean linear programming model provided for exact solutions, the goal is to study statistical difference between them for the preemptive single machine scheduling problem by no idle periods, in which processing periods are equal by progressively running release and due dates set tightly. The tight-tardy progressive single machine scheduling with preemptions by no idle periods is one of the hardest cases, which could serve as an "upper bound" for the statistical difference. When the tightness is relaxed, the difference will be expected to be less. To achieve the said goal, a computational study will be carried out with a purpose to see the inaccuracy of the heuristic. The inaccuracy will be studied how it varies against increasing complexity/size of job scheduling problems. The computation times are to be compared as well. The research result is expected to find a point, in which a hardly tractable exact model could be "linked" to a sufficiently accurate heuristic (a "lossless transfer" from the exactness to approximation). Otherwise, such a point may not exist (e. g., the exact solution is searched impracticably long, whereas the heuristic solution is still too inaccurate), but this should be shown and discussed anyway.

Exact solution by the Boolean linear programming model

First, consider an approach to find the exactly minimal total weighted tardiness. 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 priority weight w_n, a release date r_n, and a due date d_n, $n = \overline{1, N}$. Integer r_n is the time moment, at which job n becomes available for processing. So, in the case of equal processing periods,

$$\mathbf{H} = [H]_{1\times N} \in \mathbb{N}^N \ \text{(by } H = H_n \ \ \forall\, n = \overline{1, N}\text{)} \quad (1)$$

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,

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

is a vector of release dates, and

$$\mathbf{D} = [d_n]_{1\times N} \in \mathbb{N}^N \tag{4}$$

is a vector of due dates. Without loss of generality, to ensure the condition of "the proper start", which is

$$\exists n_1 \in \{\overline{1,\ N}\} \text{ such that } r_{n_1} = 1\,, \tag{5}$$

let vector (3) consist of a non-decreasing set of integers, where $r_1 = 1$. This is the first additional constraint to release dates (3). The second one comes from that there are no idle periods, i. e. condition

$$1 + \sum_{n=1}^{k} H_n \geq r_{k+1} \quad \forall\, k = \overline{1,\ N-1} \tag{6}$$

holds as well. Narrowing the problem to the already mentioned tight-tardy progressive single machine scheduling (with preemptions by no idle periods), the release dates are

$$r_n = n \quad \forall\, n = \overline{1,\ N} \tag{7}$$

and the due dates are

$$d_n = H + n - 1 \quad \forall\, n = \overline{1,\ N}\,. \tag{8}$$

Condition (8) is equivalent to condition

$$d_n = r_n + H - 1 \quad \forall\, n = \overline{1,\ N}\,. \tag{9}$$

In other words, condition (9) implies that one job herein can always be completed without tardiness.

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

$$\sum_{n=1}^{N} w_n \cdot \max\{0,\ \theta(n;\ H) - d_n\} \tag{10}$$

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

$$\theta(n;\ H) \in \{\overline{1,\ T}\} \text{ by } T = \sum_{n=1}^{N} H_n = N \cdot H\,. \tag{11}$$

This goal is equivalent to minimizing sum

$$\sum_{n=1}^{N} \sum_{h_n=1}^{H} \sum_{t=1}^{T} \lambda_{nh_n t} x_{nh_n t} \tag{12}$$

by the known Boolean linear programming model (applied for minimizing total weighted completion time also) [8, 9], where $x_{nh_n t}$ is the decision variable about assigning the h_n-th part of job n to time moment t: $x_{nh_n t} = 1$ if it is assigned; $x_{nh_n t} = 0$ otherwise. The triple-indexed weights are calculated as follows:

$$\lambda_{nh_n t} = 0 \tag{13}$$

by

$$r_n - 1 + h_n \leq t \leq T - H + h_n \quad \forall\, h_n = \overline{1,\ H-1} \tag{14}$$

and

$$\lambda_{nh_n t} = \alpha \tag{15}$$

by a sufficiently great positive integer α (similar to the meaning of infinity) when (14) is not true;

$$\lambda_{nHt} = 0 \tag{16}$$

by

$$r_n - 1 + H \leq t \leq d_n \tag{17}$$

and

$$\lambda_{nHt} = w_n(t - d_n) \tag{18}$$

by

$$d_n < t \leq T \tag{19}$$

and

$$\lambda_{nHt} = \alpha \tag{20}$$

when both (17) and (19) are not true. In (15) and (20), for instance,

$$\alpha = \sum_{n=1}^{N} \sum_{t=1}^{T} w_n t \tag{21}$$

can be used [8, 9]. So, sum (12) is defined on set

$$X = \{\{\{x_{nh_n t}\}_{n=1}^{N}\}_{h_n=1}^{H}\}_{t=1}^{T} \in \mathscr{X}\,,$$

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

$$X^* = \{\{\{x^*_{nh_n t}\}_{n=1}^{N}\}_{h_n=1}^{H}\}_{t=1}^{T} \in \mathscr{X} \tag{22}$$

on which minimum

$$\min_{X \in \mathscr{X}} \sum_{n=1}^{N} \sum_{h_n=1}^{H} \sum_{t=1}^{T} \lambda_{nh_n t} x_{nh_n t} \quad (23)$$

is achieved by constraints constituting set $\mathscr{X}$ (an integer binary lattice) [8, 9]:

$$x_{nh_n t} \in \{0, 1\} \text{ by } n = \overline{1, N} \text{ and } h_n = \overline{1, H} \text{ and } t = \overline{1, T}, \quad (24)$$

$$\sum_{t=1}^{T} x_{nh_n t} = 1 \text{ by } n = \overline{1, N} \text{ and } h_n = \overline{1, H}, \quad (25)$$

$$\sum_{n=1}^{N} \sum_{h_n=1}^{H} x_{nh_n t} = 1 \text{ by } t = \overline{1, T}, \quad (26)$$

$$\sum_{j=t+1}^{T} \sum_{h_n=1}^{H-1} x_{nh_n j} + Hx_{nHt} \leq H \text{ by } n = \overline{1, N} \text{ and } t = \overline{1, T-1}. \quad (27)$$

If (22) 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

$$\vartheta^*(N, H) = \sum_{n=1}^{N} \sum_{h_n=1}^{H} \sum_{t=1}^{T} \lambda_{nh_n t} x_{nh_n t}^* \quad (28)$$

is the exactly minimal total weighted tardiness for those N jobs. Obviously, a few optimal schedules ensuring the same minimum (28) can exist.

A heuristic based on remaining available and processing periods

The heuristic is an online scheduling algorithm, which returns a schedule stepwise as time t progresses. Let

$$\mathbf{Q} = [q_n]_{1\times N} = \mathbf{H} = [H]_{1\times N} \quad (29)$$

be a starting vector containing the remaining processing periods. Later on, elements of vector (29) 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 $t = \overline{1, T}$. For every set of available jobs

$$A(t) = \{i \in \{\overline{1, N}\} : r_i \leq t \text{ and } q_i > 0\} \subset \{\overline{1, N}\} \quad (30)$$

the remaining available time is calculated:

$$b_i = \max\{0, d_i - t + 1\} \quad \forall\, i \in A(t). \quad (31)$$

Paper [5] claims that the remaining slack

$$\zeta_i = \max\{0, b_i - q_i\} \quad \forall\, i \in A(t) \quad (32)$$

must be found also. Then a set of decisive ratios

$$\left\{ \frac{w_i}{q_i + \zeta_i} \right\}_{i \in A(t)} \quad (33)$$

is calculated. With remaining slack (32), however, it is easy to see that the ratio in (33) factually is

$$\frac{w_i}{q_i + \zeta_i} = \frac{w_i}{q_i + \max\{0, b_i - q_i\}} = \frac{w_i}{\max\{q_i, b_i\}} \quad (34)$$

because the denominator in the central fraction of statement (34) becomes equal to q_i by $b_i < q_i$ and it is b_i by $b_i \geq q_i$. Therefore, a set of decisive ratios

$$\frac{w_i}{\max\{q_i, b_i\}} \quad \forall\, i \in A(t) \quad (35)$$

is considered instead of (33). The maximal ratio is achieved at subset

$$A^*(t) = \arg\max_{i \in A(t)} \frac{w_i}{\max\{q_i, b_i\}}. \quad (36)$$

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

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

then

$$\tilde{s}_t = i^* \text{ by } q_{i^*}^{(\text{obs})} = q_{i^*} \text{ and } q_{i^*} = q_{i^*}^{(\text{obs})} - 1; \quad (37)$$

otherwise

$$A^*(t) = \{i_l^*\}_{l=1}^{L} \subset A(t) \subset \{\overline{1, N}\} \text{ by } L > 1, \quad (38)$$

whence

$$\tilde{s}_t = i_1^* \text{ by } q_{i_1^*}^{(\text{obs})} = q_{i_1^*} \text{ and } q_{i_1^*} = q_{i_1^*}^{(\text{obs})} - 1. \quad (39)$$

Assignment (39) executed by condition (38) for subset (36) implies that, in a case when there are two or more maximal decisive ratios, the earliest job is preferred to be scheduled. This is especially reasonable for the tight-tardy progressive single machine scheduling, in which the earlier releasable job has an earlier due date.

An approximately minimal total weighted tardiness is calculated successively for every $n = \overline{1, N}$ as follows: if

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

then job n is completed after moment $\tilde{\theta}(n; H)$. Finally, using formula (10),

$$\tilde{\vartheta}(N, H) = \sum_{n=1}^{N} w_n \cdot \max\{0, \tilde{\theta}(n; H) - d_n\} \quad (40)$$

is an approximately minimal total weighted tardiness that corresponds to schedule $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$. This schedule often coincides with the schedule produced by exact solution (22):

$$\vartheta^*(N, H) = \sum_{n=1}^{N} w_n \cdot \max\{0, \theta^*(n; H) - d_n\}, \quad (41)$$

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

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

Obviously, if amounts (40) and (41) 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. Moreover, the Boolean linear programming model by (1)—(27) may return more than one optimal schedule. The heuristic by (29)—(39) returns a single schedule. Eventually, if amounts (40) and (41) are different, i. e.

$$\tilde{\vartheta}(N, H) > \vartheta^*(N, H) \quad (42)$$

by whichever the respective schedules are, it only matters how far amount (40) is from amount (41).

Computational study

In computational studies, the relative error or gap of total weighted tardiness minimization in scheduling N jobs is

$$\varepsilon(N, H) = 100 \cdot \frac{\tilde{\vartheta}(N, H) - \vartheta^*(N, H)}{\vartheta^*(N, H)} \quad (43)$$

in percentage terms. Inasmuch as computation time $\tilde{\tau}(N, H)$ of the heuristic is always far less than computation time $\tau^*(N, H)$ of achieving minimum (23) by (24)—(27), then it is suitable to use a computation time ratio

$$\gamma(N, H) = \frac{\tau^*(N, H)}{\tilde{\tau}(N, H)}. \quad (44)$$

Consider the following generator of the scheduling problem instances. Priority weights (2) are

$$\mathbf{W} = [w_n]_{1\times N} = \psi(100 \cdot \Theta(1, N) + 1) \quad (45)$$

where operator $\Theta(1, N)$ returns a pseudorandom $1 \times N$ vector whose entries are drawn from the standard uniform distribution on the open interval $(0; 1)$, and function $\psi(\xi)$ returns the integer part of number ξ (e. g., see [8, 12]). Release dates (3) are taken as (7), and due dates are taken as (9). At first, we try to schedule up to 12 jobs. The minimal number of job parts is 2, whereas scheduling jobs of the single part, which herein is $\mathbf{S}^* = [n]_{1\times N}$, has no tardiness. A reasonable time period, through which a schedule is expected to be found, is of order of a minute. As the size of the scheduling problem grows, the exact model (1)—(27) may take more than 60 seconds to output a solution. This is called a timeout. So, denote the relative gap with excluding one-minute timeouts by $\varepsilon_{\le 60}(N, H)$. Additionally, denote the number of one-minute-timeout cases by $Q_{>60}(N, H)$ and denote the respective relative gap by $\varepsilon_{>60}(N, H)$. The number of one-minute-timeout cases, in which $\varepsilon_{>60}(N, H) \ge 0$, i. e. the exact solution is not worse than a heuristic solution, is denoted by $U_{\varepsilon>0}(N, H)$.

Fig. 1 shows gap (43) averaged over 1000 instances generated for each $N = \overline{2, 12}$ by when each job has only two processing periods (i. e., is divided into two identical parts). The "correct" gap $\varepsilon_{\le 60}(N, 2)$ is plotted on the same axes. There is no gap in scheduling only a pair of jobs. Scheduling 12 jobs is the most inaccurate on average, although the gap in 0.38 % seems to be tolerable. However, the instance with priority weights

$$\mathbf{W} = [w_n]_{1\times 5} = [16 \quad 36 \quad 56 \quad 6 \quad 100] \quad (46)$$

has the worst gap in 34.29 %, where the exact model produces an optimal schedule

$$\mathbf{S}^* = [s^*_t]_{1\times 10} = [1 \quad 1 \quad 3 \quad 3 \quad 5 \quad 5 \quad 2 \quad 2 \quad 4 \quad 4] \quad (47)$$

ensuring the exactly minimal total weighted tardiness value 210 as against the heuristic producing a schedule

$$\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 10} = [1 \quad 2 \quad 2 \quad 3 \quad 3 \quad 5 \quad 5 \quad 1 \quad 4 \quad 4] \quad (48)$$

whose total weighted tardiness is 282 (the second part of job 1 and the two parts of job 2 are like to have been just interchanged). The overall number of such bad gap instances is not as small as it would have seemed with the top average gap in 0.38 %: there are 72 instances (out of grand total 11000 ones) when the gap is no less than 10 %, and there are 203 instances when the gap is no less than 5 % (that still can be intolerable). There are no one-minute-timeout cases when scheduling less than 10 jobs. But starting off 10 jobs, a difference between gaps $\varepsilon(N, 2)$ and $\varepsilon_{\leq 60}(N, 2)$ becomes more apparent. The "incorrect" gap $\varepsilon_{>60}(N, 2)$ in Fig. 2 is shown just for such an interval of the number of jobs. It is expectedly decreasing as the number of one-minute-timeout cases for the interval is clearly increasing (see Fig. 3, where the number of "successful" timeouts $U_{\varepsilon>0}(N, 2)$ is shown as well). Obviously, 29 % of timeouts in scheduling 12 jobs mean that the exact model is too slow for this case. Meanwhile, the respective computation time ratio resembles an exponential increase (Fig. 4).

In scheduling 11 two-parted jobs, 88 one-minute-timeout cases came out of 1000. Therefore, we study scheduling three-parted jobs only for up to 10 jobs. Fig. 5 shows gap $\varepsilon(N, 3)$ which drops since 8 jobs because of timeouts (averaging over 1000 instances becomes incorrect since this moment). The "correct" gap $\varepsilon_{\leq 60}(N, 3)$ and "incorrect" gap $\varepsilon_{>60}(N, 3)$ are both shown on the same plot (Fig. 6). Note that the "correct" gap for 10 jobs $\varepsilon_{\leq 60}(10, 3)$ is about 1.6 % (which is far greater than the "correct" gap in 0.38 % for 12 jobs in Fig. 1) but it is averaged over just 86 cases (see Fig. 7) as against those 910 cases without timeouts for 12 jobs (Fig. 3). This is why the respective computation time ratio herein does not resemble an exponential increase (Fig. 8) having a drop at $N = 10$ (the size of the

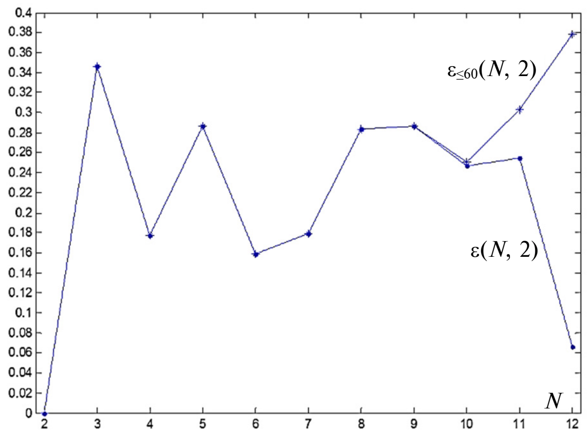

Fig. 1. Gap (43) averaged over 1000 instances generated for each $N = \overline{2, 12}$ by $H = 2$ (no timeouts until 10 jobs)

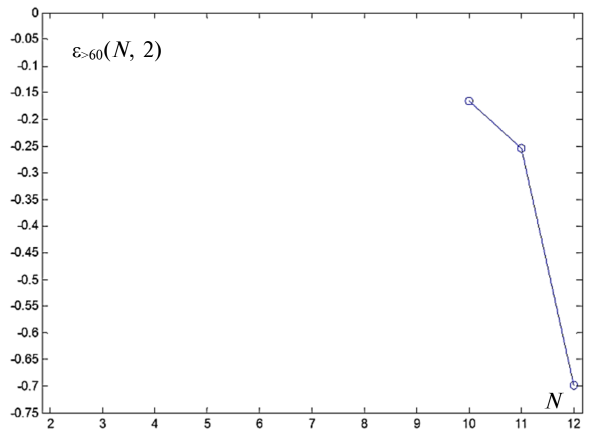

Fig. 2. A decreasing gap averaged over one-minute-timeout cases came out for each $N = \overline{10, 12}$ by $H = 2$

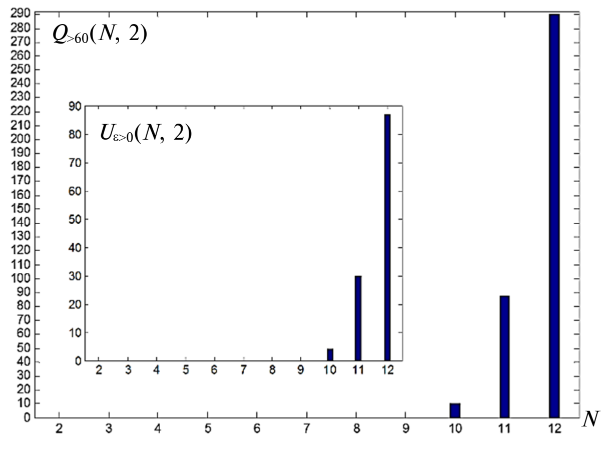

Fig. 3. The number of one-minute-timeout cases came out for each $N = \overline{10, 12}$ by $H = 2$ with the number of "successsul" timeouts

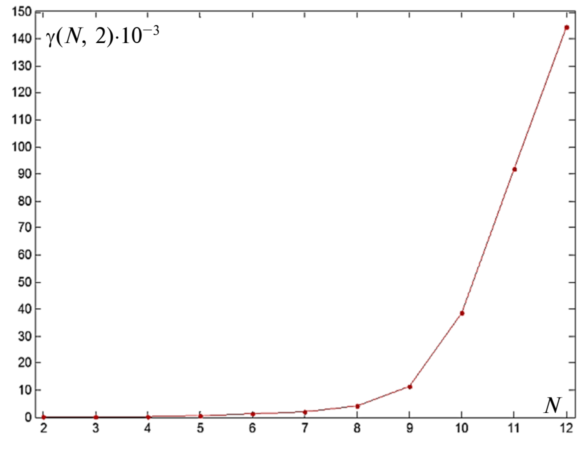

Fig. 4. Computation time ratio (44) averaged over 1000 instances generated for each $N = \overline{2, 12}$ by $H = 2$

scheduling problem has grown and the heuristic takes more time to find a solution whereas the exact model, in cases without timeouts, still takes no more than 60 seconds). Not considering scheduling of 11 and 12 jobs, ratio $\gamma(N, 3)$ is apparently greater than $\gamma(N, 2)$. Another noticeable fact is that the worst gap here is 18.05 %, and it is produced by the instance with priority weights

$$\mathbf{W} = [w_n]_{1\times 4} = [13 \quad 20 \quad 15 \quad 59] \qquad (49)$$

optimal schedule for which

$$\mathbf{S}^* = [s_t^*]_{1\times 12}$$

$$= [1 \quad 1 \quad 1 \quad 4 \quad 4 \quad 4 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3] \qquad (50)$$

with $\vartheta^*(4, 3) = 205$ is “torn” into an approximate schedule

$$\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 12}$$

$$= [1 \quad 2 \quad 2 \quad 2 \quad 4 \quad 4 \quad 4 \quad 1 \quad 1 \quad 3 \quad 3 \quad 3] \qquad (51)$$

with $\tilde{\vartheta}(4, 3) = 242$ similarly to the instance with priority weights (46) and schedules (47) and (48) (once again, the last two parts of job 1 and the three parts of job 2 are like to have been just interchanged). The overall number of bad gap instances now is smaller: there are only 23 instances (out of grand total 9000 ones) when the gap is no less than 10 %, and there are 54 instances when the gap is no less than 5 %.

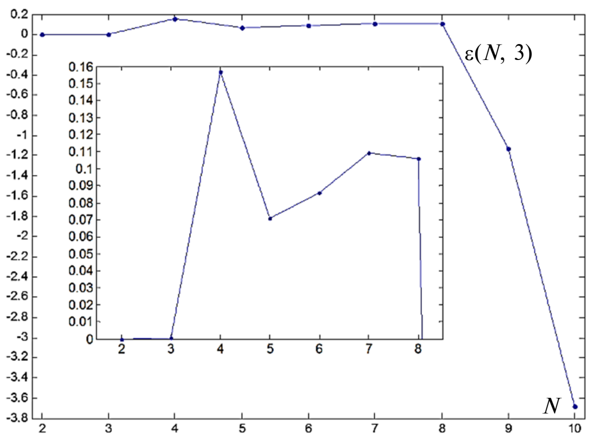

Fig. 5. Gap (43) averaged over 1000 instances generated for each $N = \overline{2, 10}$ by $H = 3$ (no timeouts until 8 jobs)

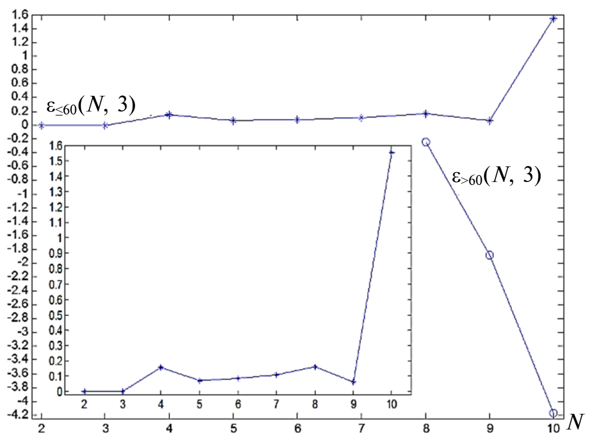

Fig. 6. A timeout-exclusion gap and a gap averaged over one-minute-timeout cases came out for each $N = \overline{8, 10}$ by $H = 3$

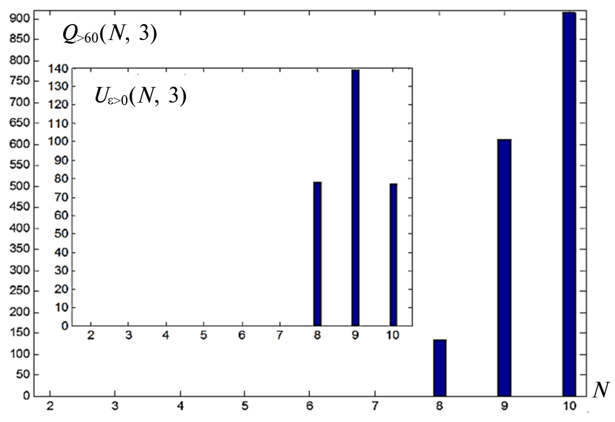

Fig. 7. The number of one-minute-timeout cases came out for each $N = \overline{8, 10}$ by $H = 3$ with the number of “successful” timeouts

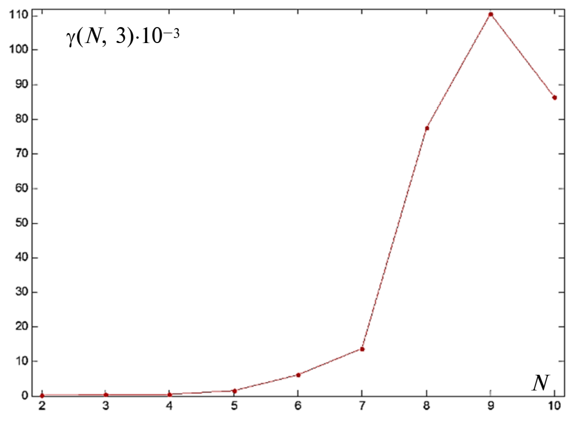

Fig. 8. Computation time ratio (44) averaged over 1000 instances generated for each $N = \overline{2, 10}$ by $H = 3$

In scheduling 9 three-parted jobs, 612 one-minute-timeout cases came out of 1000. Therefore, we study scheduling four-parted jobs only for up to 8 jobs. Fig. 9 shows gap $\varepsilon(N, 4)$ which drops since 7 jobs because of timeouts. The "correct" gap $\varepsilon_{\leq 60}(N, 4)$ and "incorrect" gap $\varepsilon_{>60}(N, 4)$ both shown on the same plot (Fig. 10) resemble those ones in Fig. 6, especially that unexpected jump at the maximal number of jobs. Although $\varepsilon_{\leq 60}(8, 4) \approx 0.36$ % versus $\varepsilon_{\leq 60}(10, 3) \approx 1.6$ %, the jump when scheduling four jobs is stronger because

$$\frac{\varepsilon_{\leq 60}(10, 3)}{\varepsilon_{\leq 60}(9, 3)} \approx 24.8422 \text{ and } \frac{\varepsilon_{\leq 60}(8, 4)}{\varepsilon_{\leq 60}(7, 4)} \approx 119.65,$$

which is pretty weird itself. Indeed, Fig. 11 showing the number of one-minute-timeout cases does not contain a huge jump. As in the previous case, the respective computation time ratio herein does not resemble an exponential increase (Fig. 12) having a drop at $N = 8$. Meanwhile, the worst gap here in 9.28 % is caught in scheduling 6 jobs. The overall number of bad gap instances is small: there are only 3 instances (out of grand total 7000 ones) when the gap is no less than 5 %, and there are 50 instances when the gap is no less than 1 % (that can be interpreted as a tolerable value). Furthermore, only 89 instances have been scheduled with the gap no less than 0.1 %.

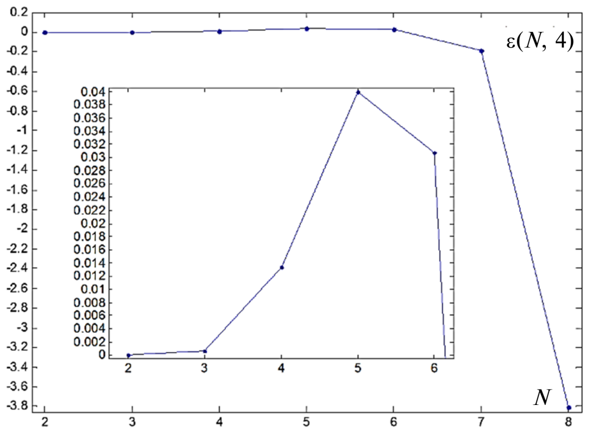

Fig. 9. Gap (43) averaged over 1000 instances generated for each $N = \overline{2, 8}$ by $H = 4$ (no timeouts until 7 jobs)

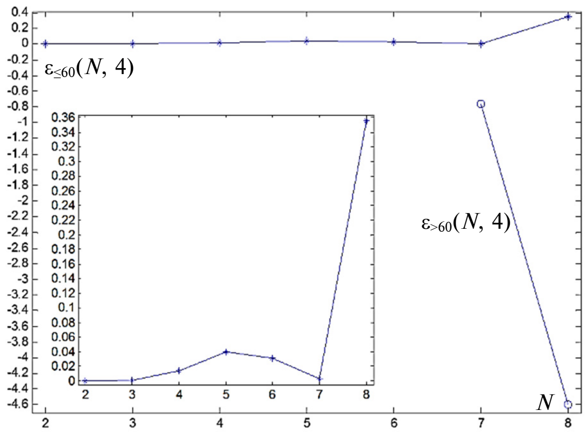

Fig. 10. A timeout-exclusion gap and a gap averaged over one-minute-timeout cases came out for $N = 7$ and $N = 8$ by $H = 4$

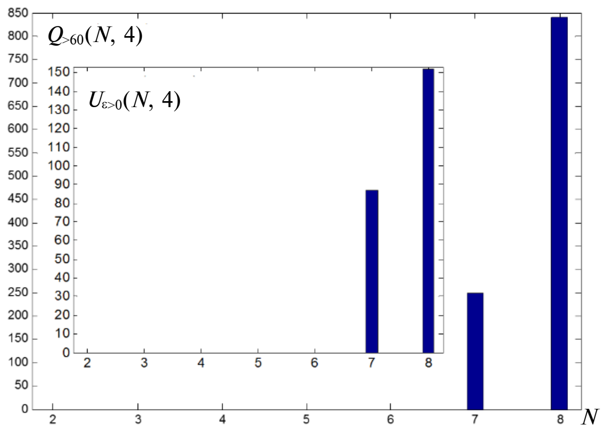

Fig. 11. The number of one-minute-timeout cases came out for $N = 7$ and $N = 8$ by $H = 4$ with the number of "successful" timeouts

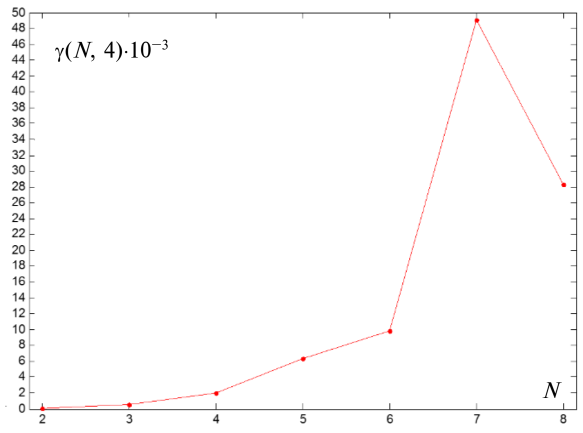

Fig. 12. Computation time ratio (44) averaged over 1000 instances generated for each $N = \overline{2, 8}$ by $H = 4$

Consequently, we continue to study scheduling for an increased number of job parts. Fig. 13 shows gap $\varepsilon(N, 5)$ for five-parted jobs which drops since 6 jobs because of timeouts. Once again, the "correct" gap $\varepsilon_{\leq 60}(N, 5)$ and "incorrect" gap $\varepsilon_{>60}(N, 5)$ both shown on the same plot (Fig. 14) resemble those ones in the previous cases (for three- and four-parted jobs). However, now there is an unexpected drop at 7 jobs, whereas all 1000 instances at 8 jobs are timeouts (Fig. 15). The respective computation time ratio (Fig. 16) therein does not resemble that in Fig. 12. Meanwhile, the worst gap here drops to 7.12 % caught in scheduling 6 jobs once again. The overall number of bad gap instances is small similarly to the previous case: there are only 4 instances when the gap is no less than 5 %, and there are 32 instances when the gap is no less than 1 %. Furthermore, only 64 instances have been scheduled with the gap no less than 0.1 % (that is a quite tolerable value).

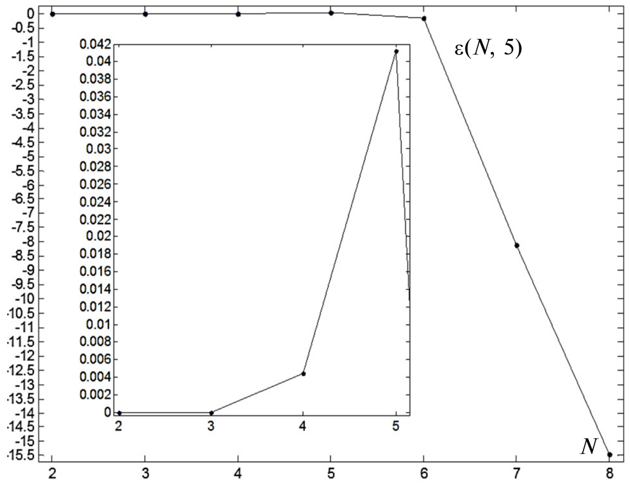

Fig. 13. Gap (43) averaged over 1000 instances generated for each $N = \overline{2, 8}$ by $H = 5$ (no timeouts until 6 jobs)

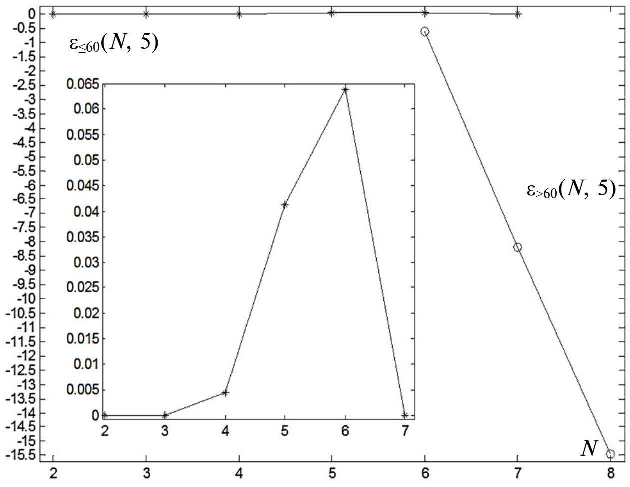

Fig. 14. A timeout-exclusion gap and a gap averaged over one-minute-timeout cases came out for each $N = \overline{6, 8}$ by $H = 5$

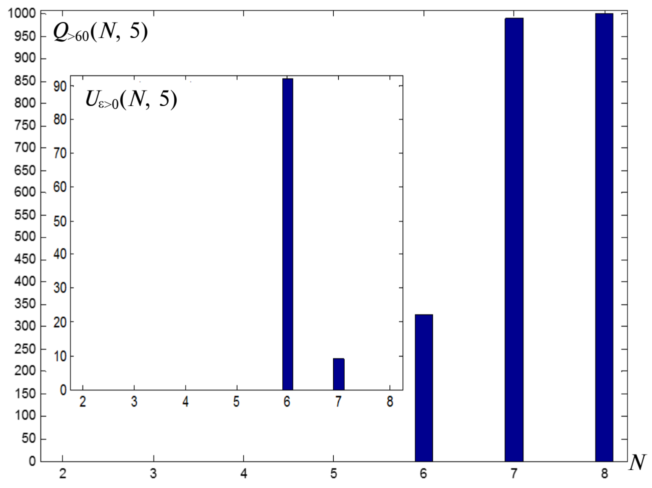

Fig. 15. The number of one-minute-timeout cases came out for each $N = \overline{6, 8}$ by $H = 5$ with the number of "successful" timeouts

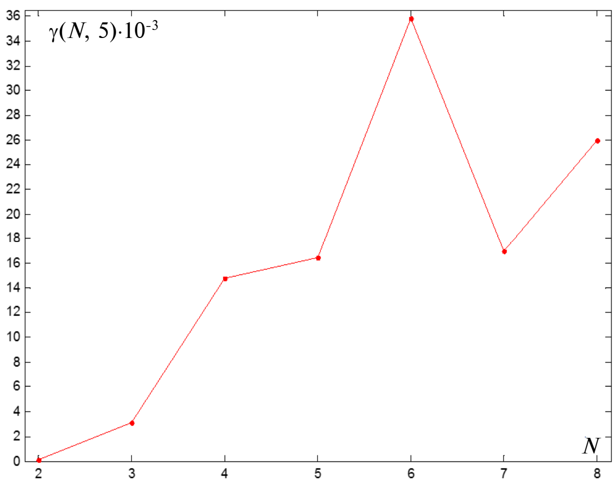

Fig. 16. Computation time ratio (44) averaged over 1000 instances generated for each $N = \overline{2, 8}$ by $H = 5$

As it is easily seen, the worst scheduling cases, when the gap achieves practically intolerable values, cannot be excluded or prevented. In such cases, the heuristic by (29)—(39) does not help and the exact model remains the only means to find a schedule. Knowing the worst gap values thus is fundamentally needful. Maximal relative gap denoted by $\hat{\varepsilon}(N, H)$ is shown for the studied schedulings in Fig. 17. This is a qualitative and very important supplement to Figs. 1—16. Despite $\hat{\varepsilon}(12, 2) > 19.23$ % for the instance with priority weights

$$\mathbf{W} = [w_n]_{1\times 12} = [10 \quad 13 \quad 1 \quad 27 \quad 62 \quad 12 \quad 92 \quad 40 \quad 31 \quad 3 \quad 92 \quad 85] \qquad (52)$$

optimal schedule for which

$$\mathbf{S}^* = [s_t^*]_{1\times 24} = [1 \quad 1 \quad 2 \quad 2 \quad 5 \quad 5 \quad 7 \quad 7 \quad 8 \quad 8 \quad 11 \quad 11 \quad 12 \quad 12 \quad 9 \quad 9 \quad 4 \quad 4 \quad 6 \quad 6 \quad 10 \quad 10 \quad 3 \quad 3] \qquad (53)$$

with $\vartheta^*(12, 2) = 884$ is “torn” into an approximate schedule

$$\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 24} = [1 \quad 1 \quad 2 \quad 4 \quad 5 \quad 5 \quad 7 \quad 7 \quad 4 \quad 8 \quad 11 \quad 11 \quad 12 \quad 12 \quad 8 \quad 9 \quad 9 \quad 2 \quad 6 \quad 6 \quad 10 \quad 10 \quad 3 \quad 3] \qquad (54)$$

with $\tilde{\vartheta}(12, 2) = 1054$, it is clearly seen that the greater number of processing periods lessens the worst gap values. The top worst maximal relative gap $\hat{\varepsilon}(5, 2)$ is obtained by the instance with (46) whose optimal schedule (47) with $\vartheta^*(5, 2) = 210$ is “torn” into an approximate schedule (48) with $\tilde{\vartheta}(5, 2) = 282$. Fig. 17 also allows to suppose that the top worst gaps are concentrated over scheduling 3 to 6 jobs.

It is clear that the study of the predominant one-minute-timeout cases should be supplemented also. For this, let the timeout threshold be elongated to 10 minutes. So, denote the relative gap with excluding 10-minute timeouts by $\varepsilon_{\leq 600}(N, H)$. The number of 10-minute-timeout cases is denoted by $Q_{>600}(N, H)$ and the respective relative gap is denoted by $\varepsilon_{>600}(N, H)$. The denotation of the number of 10-minute-timeout cases, in which $\varepsilon_{>600}(N, H) \geq 0$, i. e. the exact solution is not worse than a heuristic solution, is remained the same — it is $U_{\varepsilon>0}(N, H)$. Unlike the initial study, for the supplementary study, we take 200 instances (instead of 1000). This is forced by the two factors. Firstly, the increased span for the solving (from 1 minute to 10 minutes), considering the number of jobs at which timeouts are very likely, may require a way longer computation time than that for the initial study. Secondly, the amount of 1000 instances itself is some overstated for the reliable averaging and therefore can be reduced.

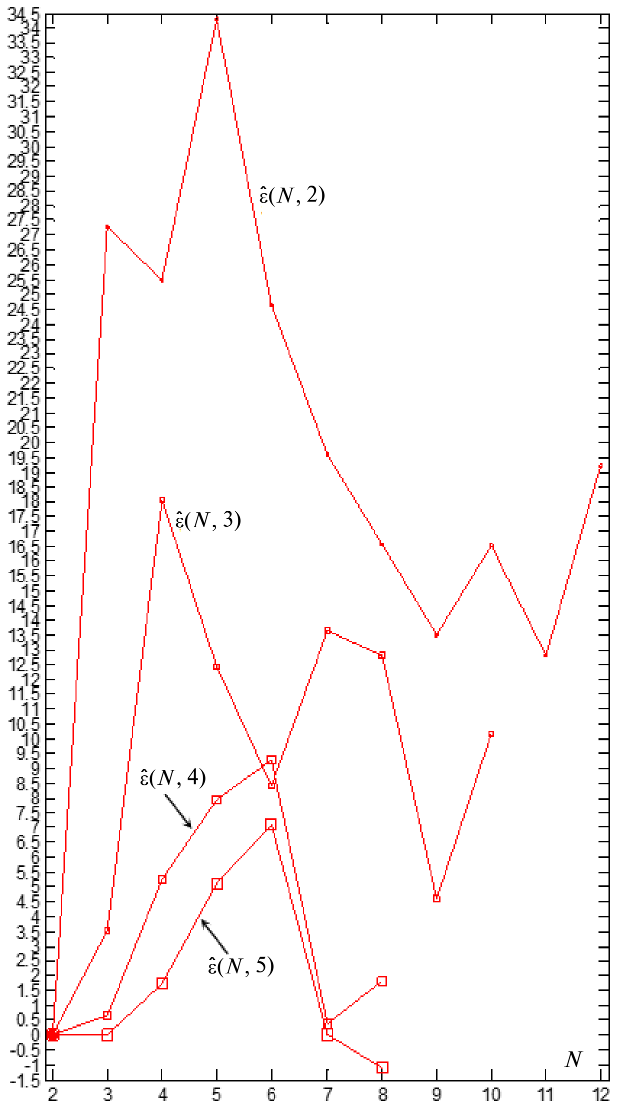

Fig. 17. Maximal relative gaps (over 1000 instances generated for each number of jobs by the respective number of job parts), where a bigger square marker corresponds to a greater number of processing periods

For each $N = \overline{10, 16}$ by $H = 2$ Fig. 18 shows the “correct” gap averaged over only cases from which 10-minute timeouts are excluded. This can be imagined as a natural extension of the “correct” gap in Fig. 1. Obviously, the peak at 15 jobs is casual. The respective gap (43) and the “incorrect” gap are shown in Fig. 19, where we see that the 10-minute timeouts start here at 12 jobs. The number of 10-minute-timeout cases (Fig. 20) is immensely increasing, and only 34 instances of scheduling 16 two-parted jobs have been solved in no more than 600 seconds. The respective computation time ratio (Fig. 21) starts roughly resembling an S-shaped curve. As it is clearly seen, in scheduling 16 two-parted jobs, the heuristic by (29)—(39) is about one million times (!) faster than the Boolean linear programming model by (1)—(27).

In scheduling three-parted jobs the “correct” gap averaged over only cases from which 10-minute timeouts are excluded has a decreasing trend (Fig. 22). Now at 10 jobs we can see only a casual peak, rather than that huge jump up at about 1.6 % in Fig. 6. Consequently, that value of $\varepsilon_{\leq 60}(10, 3)$ in Fig. 6 is a computational artifact, although it has been revealed under the same generator of the scheduling problem instances, where priority weights (2) are pseudorandomly generated by (45). The respective gap (43) and the “incorrect” gap are shown in Fig. 23, where we see that the 10-minute timeouts start at 9 three-parted jobs. The number of 10-minute-timeout cases (Fig. 24) is more immensely increasing, and now only 12 instances

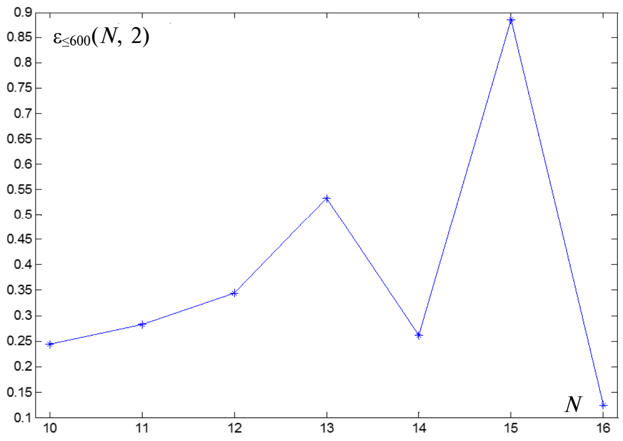

Fig. 18. A gap averaged over only timeout-exclusion cases out of 200 instances generated for each $N = \overline{10, 16}$ by $H = 2$ (the 10-minute timeouts start at 12 jobs)

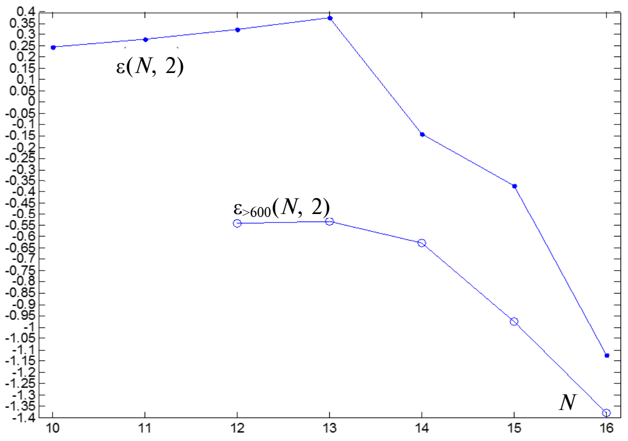

Fig. 19. Gap (43) averaged over 200 instances generated for each $N = \overline{10, 16}$ by $H = 2$ and a gap averaged over 10-minute-timeout cases came out for each $N = \overline{12, 16}$

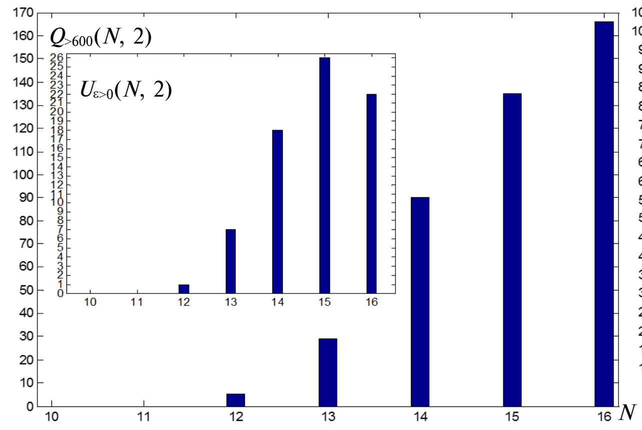

Fig. 20. The number of 10-minute-timeout cases came out for each $N = \overline{12, 16}$ by $H = 2$ with the number of “successful” timeouts

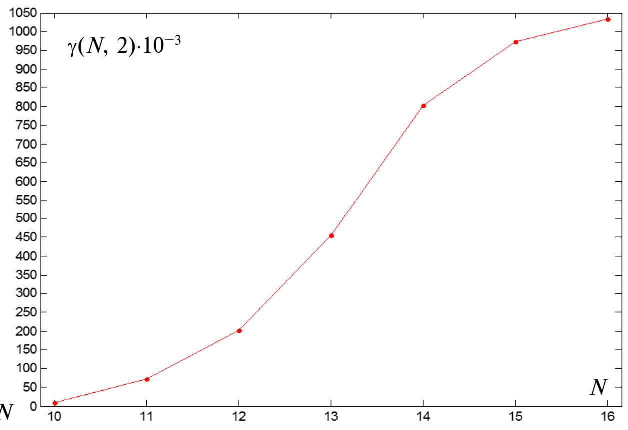

Fig. 21. Computation time ratio (44) averaged over 200 instances generated for each $N = \overline{10, 16}$ by $H = 2$

of scheduling 16 three-parted jobs have been solved in no more than 600 seconds. This is why the respective computation time ratio (Fig. 25) has been "contorted" again (similarly to Figs. 8, 12, 16). Nevertheless, in scheduling 12 three-parted jobs, the heuristic is about half a million times faster than the Boolean linear programming model. This ratio in scheduling 12 two-parted jobs (Fig. 21) is about 2.5 times less.

Maximal relative gap for the supplementary study (Fig. 18—25) is shown in Fig. 26. These two polylines confirm the previously inferred suspicion about the worst gap values dropping as the number of processing periods increases. Indeed, the huge value of $\hat{\varepsilon}(15, 2)$ is a computational artifact issued from the instance with priority weights

$$\mathbf{W} = [w_n]_{1\times 15} = [16 \;\; 60 \;\; 31 \;\; 1 \;\; 23 \;\; 62 \;\; 50 \;\; 16 \;\; 35 \;\; 100 \;\; 98 \;\; 13 \;\; 35 \;\; 77 \;\; 81] \qquad (55)$$

optimal schedule for which

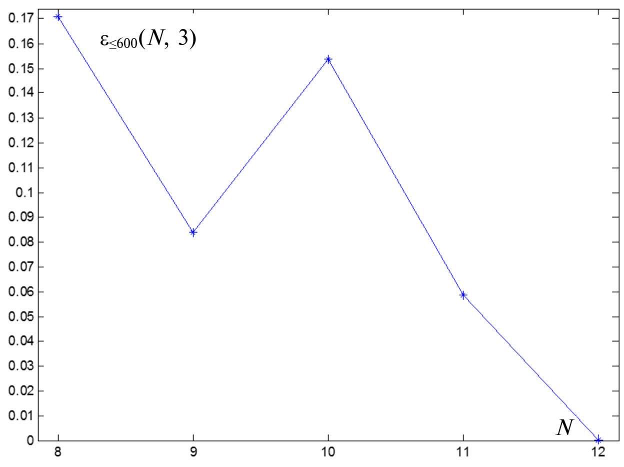

Fig. 22. A gap averaged over only timeout-exclusion cases out of 200 instances generated for each $N = \overline{8, 12}$ by $H = 3$ (the 10-minute timeouts start at 9 jobs)

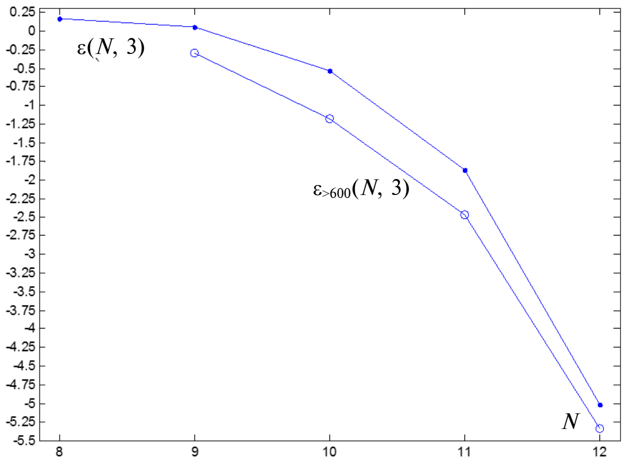

Fig. 23. Gap (43) averaged over 200 instances generated for each $N = \overline{8, 12}$ by $H = 3$ and a gap averaged over 10-minute-timeout cases came out for each $N = \overline{9, 12}$

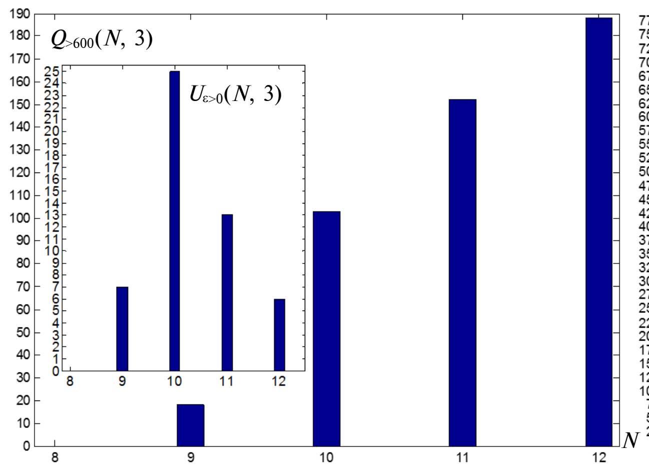

Fig. 24. The number of 10-minute-timeout cases came out for each $N = \overline{9, 12}$ by $H = 3$ with the number of "successful" timeouts

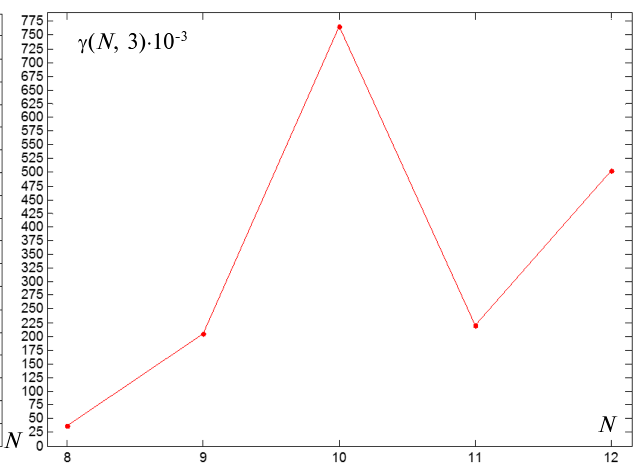

Fig. 25. Computation time ratio (44) averaged over 200 instances generated for each $N = \overline{8, 12}$ by $H = 3$

$$\mathbf{S}^* = [s_t^*]_{1\times 30}$$
$$= [1\ \ 2\ \ 2\ \ 3\ \ 3\ \ 6\ \ 6\ \ 7\ \ 7\ \ 10\ \ 10$$
$$11\ \ 11\ \ 14\ \ 14\ \ 15\ \ 15\ \ 13\ \ 13\ \ 9$$
$$9\ \ 1\ \ 5\ \ 5\ \ 8\ \ 8\ \ 12\ \ 12\ \ 4\ \ 4] \qquad (56)$$

with $\vartheta^*(15, 2) = 2046$ is "torn" into an approximate schedule

$$\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 30}$$
$$= [1\ \ 2\ \ 2\ \ 1\ \ 3\ \ 3\ \ 6\ \ 6\ \ 7\ \ 7$$
$$10\ \ 10\ \ 11\ \ 11\ \ 15\ \ 15\ \ 14\ \ 14\ \ 9\ \ 9$$
$$13\ \ 13\ \ 5\ \ 5\ \ 8\ \ 8\ \ 12\ \ 12\ \ 4\ \ 4] \qquad (57)$$

with $\tilde{\vartheta}(15, 2) = 2319$. The frequency of such huge-gap computational artifacts is not significant for considering them as statistically regular.

In the end, it is very important to learn a ratio of non-timeout instances, in which the heuristic gives the minimal total weighted tardiness, to the total number of non-timeout instances (i. e., a fraction or percentage of cases when the exact model is factually needless). Denote this ratio by $\rho_{\varepsilon=0}(N, H)$. Fig. 27, in which the supplementary study results are drawn with a thicker line, and a bigger square marker corresponds to a greater number of processing periods, shows that the heuristic has successfully replaced the exact model no less than in 72 % of non-timeout instances. As the number of jobs increases, the fluctuations in those polylines become severer, that is explained with the decreasing number of non-timeout instances causing less statistical reliability of the ratio. In scheduling just 2 jobs, whichever way they are parted, the heuristic always gives the minimal total weighted tardiness.

Any further extensions and supplements of the computational study are worthless as they would require raising the timeout threshold up to half an hour and even to a few hours, and that is leading to

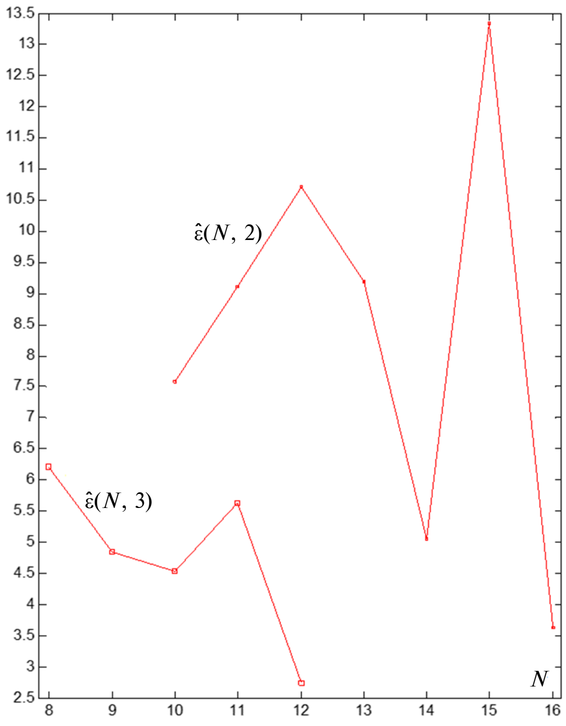

Fig. 26. Maximal relative gaps (over 200 instances generated for each number of jobs by the respective number of job parts)

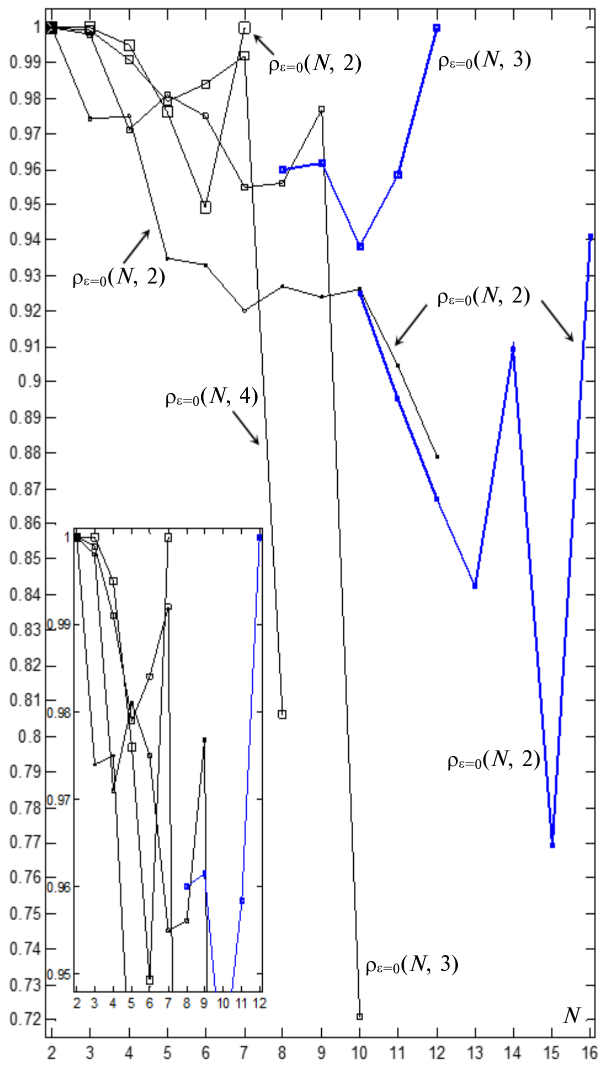

Fig. 27. The ratio of non-timeout instances, in which the heuristic gives the minimal total weighted tardiness, to the total number of non-timeout instances

what we call “intractability” of the exact model. For example, if there is an international airport, its schedules of departures and arrivals are frequently corrected (e. g., by reason of specific meteorological conditions, delays, flight cancellations, etc.), and thus recalculating an optimal schedule for even a few minutes may be critical. This is not a direct task of the air traffic controller. The special software should fast-refreshably trace the optimality of the schedules. Otherwise profitability of the airline will drop, the rate of flights will be reduced, and subsequently a lot of the concerned passengers will suffer losses.

Discussion

Turning back to the expected research result, where is the point of the “lossless transfer” from the exactness to approximation? Could the hardly tractable Boolean linear programming model by (1)—(27) eventually be “linked” to the sufficiently accurate heuristic by (29)—(39)? Based on Figs. 1—3, 5—7, 9—11, 13—15, 17—20, 22—24, and 26, the answer to the first question is negative — there is no the “lossless transfer” point. On average, the heuristic produces schedules with the same minimal total weighted tardiness just as the exact model does at pretty high rate (see Fig. 27). However, “pathological” cases like those with priority weights (46), (49), (52), (55), whose corresponding optimal schedules (47), (50), (53), (56) differ from respective heuristic schedules (48), (51), (54), (57) not so much, do exist. Unfortunately, they cannot be predicted or systemized. This is so because the way in which the heuristic “tears” the optimal schedule is unclear so far. For example, the instance with priority weights

$$\mathbf{W} = [w_n]_{1\times 8} = [28 \quad 85 \quad 81 \quad 67 \quad 35 \quad 45 \quad 70 \quad 86] \tag{58}$$

of three-parted jobs has heuristic schedule

$$\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times 24} = [1 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3 \quad 8 \quad 8 \quad 8 \quad 7 \quad 7 \quad 7 \quad 4 \quad 4 \quad 4 \quad 6 \quad 6 \quad 6 \quad 1 \quad 1 \quad 5 \quad 5 \quad 5] \tag{59}$$

which looks like a “torn” one. Nevertheless, schedule (59) is optimal for (58) by

$$\tilde{\vartheta}(8, 3) = \vartheta^{*}(8, 3) = 2706.$$

The similar seeming “tear”-in-schedule example is that with priority weights (55), heuristic schedule (57) for which is not optimal.

The Boolean linear programming model by (1)—(27) is indeed hardly tractable in scheduling no less than 14 two-parted jobs (see Fig. 20) and no less than 10 three-parted jobs (see Fig. 24). Whereas the top worst maximal gap for three-parted jobs is not greater than 6 % (for no less than 10 three-parted jobs), it may be beyond 10 % for two-parted jobs (see those peaks in Fig. 26). Therefore, scheduling three-parted jobs by the heuristic herein is preferable to scheduling two-parted jobs. Moreover, owing to Fig. 17, the following can be generalized for the heuristic:

1. Scheduling jobs divided into a greater number of parts each will have a significantly lower worst gap than scheduling jobs divided into a lesser number of parts.
2. If a job is divisible, it is strongly recommended to divide the job into as great number of its parts as possible (thus, allowing more preemptions, job shifts, job “tears”, etc.).
3. If scheduling only 2 jobs is impossible, it is strongly recommended to artificially increase the number of jobs to be scheduled (along with superdividing each job).

Hence, the listed three items make a “link” to the sufficiently accurate heuristic by (29)—(39) plausible. Besides, scheduling a fewer number of jobs (up to 12) divided into only two parts each by the heuristic is inapplicable. The high risk of obtaining a huge heuristic gap exists for three-parted jobs as well. On the other hand, scheduling either 3 or no less than 7 jobs divided into five parts each ensures the most accurate heuristic schedules (very close to the minimal total weighted tardiness). Specifically, scheduling 3 five-parted jobs have had no gaps through 1000 instances (see Fig. 14).

Conclusions

The heuristic based on remaining available and processing periods is an extremely rapid and simple technique of scheduling with minimizing total weighted tardiness: it has taken on average between 0.13 to 1.2 milliseconds to complete a schedule for the generated instances. In the worst cases, the heuristic’s computation time has varied between 1.1 to 44 milliseconds. It is obvious that the computation time (both for the heuristic and the Boolean linear programming model) stretches out as the job scheduling problem complexity/size increases. And even with this stretch the computation time ratio is still gigantic: it can reach beyond 10^6 in scheduling multiple jobs having two processing periods each, although it drops down to 10^4 when the number of processing periods (or job parts) increases. Meanwhile, the risk of obtaining a huge heuristic gap is higher at the computation time ratio maxima.

Total weighted tardiness minimization in tight-tardy progressive single machine scheduling with preemptions by no idle periods can be sufficiently accurate by the heuristic if no less than 7 jobs divided into no less than five parts each are scheduled. This is the main conclusion (let it be called the "7/5" pattern). An exception from this rule is that the heuristic schedules just 2 jobs always at the 100 % accuracy, not depending on in how many parts the job is divided (let it be called the "2/any" exception). An intermediate between the "7/5" pattern and the "2/any" exception is that scheduling 3 jobs divided into either four or five parts is sufficiently accurate as well, where the inaccuracy does not exceed 0.7 %. These three cases (the "7/5" pattern, "2/any" exception, and intermediate) constitute a domain where the heuristic is fully applicable and should entirely replace the exact approach. In other cases (which can be thought of as the complement of the domain) the heuristic is either inapplicable or there is a high risk of obtaining intolerable gaps. In particular, the complement includes any number of jobs greater than 2 divided into either two or three parts each (wherein the heuristic is inapplicable). The adjacent cases (like 3 three-parted jobs or 6 five-parted jobs) are risky of 3.5 % to 7 % gap. Nevertheless, the inapplicability does not mean that the heuristic schedules, e. g., 3 two-parted jobs badly inaccurately because 974 of 1000 instances have been scheduled with the minimal total weighted tardiness (i. e., with the 100 % accuracy); however, 26 instances have been unpredictably scheduled with an average gap in 13.31 %, which is quite intolerable and thus inapplicable.

This research has confirmed that, despite the computation time gain may become significantly lesser, it is better to schedule big-sized job problems. As the size grows, the inaccuracy drops, but the drop is expected to be even deeper for tardy-relaxed problems owing to that the research has been the "upper bound". Whether it will be corrected or no when the processing periods are unequal is a question requireing a further research.

References

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

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

[3] 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

[4] 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

[5] F. Jaramillo and M. Erkoc, "Minimizing total weighted tardiness and overtime costs for single machine preemptive scheduling", *Comp. Industr. Eng.*, vol. 107, pp. 109–119, 2017. doi: 10.1016/j.cie.2017.03.012

[6] R. L. Graham *et al.*, "Optimization and approximation in deterministic sequencing and scheduling: A survey", *Ann. Discrete Math.*, vol. 5, pp. 287–326, 1979. doi: 10.1016/S0167-5060(08)70356-X

[7] Z. J. Tian *et al.*, "An $O(n^2)$ algorithm for scheduling equal-length preemptive jobs on a single machine to minimize total tardiness", *J. Scheduling*, vol. 9, iss. 4, pp. 343–364, 2006. doi: 10.1007/s10951-006-7039-6

[8] V.V. Romanuke, "Accuracy of a heuristic for total weighted completion time minimization in preemptive single machine scheduling problem by no idle time intervals", *KPI Sci. News*, no. 3, pp. 52–62, 2019. doi: 10.20535/kpi-sn.2019.3.164804

[9] V.V. Romanuke, "The exact minimization of total weighted completion time in the preemptive scheduling problem by subsequent length-equal job importance growth", *Bulletin of V. Karazin Kharkiv National University. Series "Mathematical Modelling. Information Technology. Automated Control Systems"*, iss. 40, pp. 60–66, 2018.

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

[11] R. Panneerselvam, "Simple heuristic to minimize total tardiness in a single machine scheduling problem", *Int. J. Adv. Manuf. Technol.*, vol. 30, iss. 7-8, pp. 722–726, 2006. doi: 10.1007/s00170-005-0102-1

[12] 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

В.В. Романюк

ТОЧНА МІНІМІЗАЦІЯ ЗАГАЛЬНОГО ЗВАЖЕНОГО ЗАПІЗНЮВАННЯ У ЩІЛЬНОМУ ПРОГРЕСУЮЧОМУ ОДНОМАШИННОМУ ПЛАНУВАННІ З ПЕРЕМИКАННЯМИ БЕЗ ПЕРІОДІВ ПРОСТОЮ

Проблематика. Задача мінімізації загального зваженого запізнювання може бути розв’язана або точно за відповідними моделями, або евристично. На момент жовтня 2019 р. близькою до найкращої є евристика на основі використання залишкового наявного ресурсу та залишкового періоду до обробки. Ця евристика є надзвичайно швидкою порівняно з моделями точного розв’язку, але її точність може бути як 100 %, так і недопустимо низькою.

Мета дослідження. З огляду на відсутність знань щодо взаємозв’язку між цією евристикою та моделлю булевого лінійного програмування для точних розв’язків метою є вивчення статистичної різниці між ними для одномашинної задачі планування з перемиканнями без періодів простою, у якій періоди обробки є рівними, а послідовно прогресуючі моменти запуску та прийому виконання задані щільно.

Методика реалізації. Визначається відносна похибка евристики, а потім вивчається, як вона змінюється залежно від зростаючої складності задач планування завдань. Під складністю розуміється кількість завдань разом із кількістю періодів обробки одного завдання. Тривалості обчислень евристики і точної моделі фіксуються також.

Результати дослідження. Евристика успішно замінила точну модель не менш ніж у 72 % окремих прикладів без перевищення ліміту часу, де вона розплановує завдання з тим самим мінімальним загальним зваженим запізнюванням (100 %-ва точність). У середньому цей рівень коливається від 90 до 97 %, хоча у решті випадків можуть з’являтися величезні похибки. У практиці розкладів із вимогами швидкого оновлення модель булевого лінійного програмування насправді є ледь здійсненною, коли йдеться про планування розкладів не менш ніж 14 завдань, де кожне завдання складається з двох частин, та не менш ніж 10 завдань, де кожне завдання складається з трьох частин. Планування завдань, де кожне завдання розділене на більшу кількість частин, матиме значно меншу найгіршу похибку, ніж планування завдань, де кожне завдання розділене на меншу кількість частин. Якщо завдання ще можна розділити, то наполегливо рекомендується ділити завдання на якомога більшу кількість частин. Якщо планування лише 2-х завдань є неможливим, то наполегливо рекомендується штучно збільшити кількість завдань для складання їх розкладу.

Висновки. Мінімізація загального зваженого запізнювання у щільному прогресуючому одномашинному плануванні з перемиканнями без періодів простою може бути достатньо точною за допомогою евристики, якщо складається розклад для не менш ніж 7-х завдань, де кожне розділене не менш ніж на п’ять частин (схема “7/5”). Винятком з цього правила є те, що лише 2 завдання евристика розплановує завжди з точністю 100 %, незалежно від того, на скільки частин завдання розділене (виняток “2/any”). Певною проміжною ланкою між схемою “7/5” та винятком “2/any” є те, що планування 3-х завдань, де кожне розділене на чотири або п’ять частин, є також достатньо точним, де неточність не перевищує 0,7 %. В інших випадках або евристика незастосовна, або існує великий ризик отримання недопустимих похибок. Незастосовність тут не означає одразу сильну неточність, а мається на увазі непередбачуване існування серйозного спаду точності. Наприклад, 974 з 1000 окремих прикладів, що складалися з виключно 3-х завдань, розділених на дві частини, були розплановані з точністю 100 %, але 26 прикладів були розплановані із середньою похибкою в 13,31 %, що є вкрай недопустимим і, відповідно, незастосовним.

Ключові слова: планування завдань; планування на одній машині з перемиканнями; точна модель; евристика; загальне зважене запізнювання; точність евристики; відносна похибка; верх найгіршої максимальної відносної похибки.

В.В. Романюк

ТОЧНАЯ МИНИМИЗАЦИЯ ОБЩЕГО ВЗВЕШЕННОГО ЗАПАЗДЫВАНИЯ В ПЛОТНОМ ПРОГРЕССИРУЮЩЕМ ОДНОМАШИННОМ ПЛАНИРОВАНИИ С ПЕРЕКЛЮЧЕНИЯМИ БЕЗ ПЕРИОДОВ ПРОСТОЯ

Проблематика. Задача минимизации общего взвешенного запаздывания может быть решена или точно по соответствующим моделям, или эвристически. На момент октября 2019 г. близкой к наилучшей является эвристика на основе использования остаточного имеющегося ресурса и остаточного периода к обработке. Эта эвристика является чрезвычайно быстрой в сравнении с моделями точного решения, но ее точность может быть как 100 %, так и недопустимо низкой.

Цель исследования. Исходя из отсутствия знаний о взаимосвязи между данной эвристикой и моделью булевого линейного программирования для точных решений, целью является изучение статистической разницы между ними для одномашинной задачи планирования с переключениями без периодов простоя, в которой периоды обработки являются равными, а последовательно прогрессирующие моменты запуска и приема выполнения заданы плотно.

Методика реализации. Определяется относительная погрешность эвристики, а затем изучается, как она меняется в зависимости от возрастающей сложности задач планирования заданий. Эта сложность подразумевает количество заданий вместе с количеством периодов обработки одного задания. Длительности вычислений эвристики и точной модели регистрируются также.

Результаты исследования. Эвристика успешно заменила точную модель не менее чем в 72 % отдельных примеров без превышения лимита времени, где она планирует задания с тем самым минимальным общим взвешенным запаздыванием (100 %-ная точность). В среднем этот уровень колеблется от 90 до 97 %, хотя в остальных случаях могут появляться огромные погрешности. В практике расписаний с требованиями быстрого обновления модель булевого линейного программирования на самом деле едва осуществима, когда речь идет о планировании расписаний не менее чем 14 заданий, где каждое задание состоит из двух частей, и не менее чем 10 заданий, где каждое задание состоит из трех частей. Планирование заданий, где каждое задание разделено на большее количество частей, будет иметь значительно меньшую наихудшую погрешность, чем планирование заданий, где каждое задание разделено на меньшее количество частей. Если задание еще можно разделить, то настойчиво рекомендуется делить задание на по возможности большее количество частей. Если планирование лишь 2-х заданий является невозможным, то настойчиво рекомендуется искусственно увеличить количество заданий для составления их расписания.

Выводы. Минимизация общего взвешенного запаздывания в плотном прогрессирующем одномашинном планировании с переключениями без периодов простоя может быть достаточно точной с помощью эвристики, если составляется расписание для не менее чем 7-ми заданий, где каждое разделено не менее чем на пять частей (схема “7/5”). Исключением из этого правила является то, что лишь 2 задания эвристика планирует всегда с точностью 100 %, вне зависимости от того, на сколько частей задание разделено (исключение “2/any”). Определенным промежуточным звеном между схемой “7/5” и исключением “2/any” является то, что планирование 3-х заданий, где каждое разделено на четыре или пять частей, также является достаточно точным, где неточность не превышает 0,7 %. В других случаях или эвристика неприменима, или существует большой риск получения недопустимых погрешностей. Неприменимость здесь не означает сразу сильную неточность, а имеется в виду не поддающееся прогнозированию существование серьезного спада точности. Например, 974 из 1000 отдельных примеров, которые состояли из исключительно 3-х заданий, разделенных на две части, были распланированы с точностью 100 %, но 26 примеров были распланированы со средней погрешностью в 13,31 %, что крайне недопустимо и, соответственно, неприменимо.

Ключевые слова: планирование заданий; планирование на одной машине с переключениями; точная модель; эвристика; общее взвешенное запаздывание; точность эвристики; относительная погрешность; верх наихудшей максимальной относительной погрешности.

Рекомендована Радою
факультету прикладної математики
КПІ ім. Ігоря Сікорського

Надійшла до редакції
12 вересня 2019 року

Прийнята до публікації
31 жовтня 2019 року