sciendo RIGA TECHNICAL UNIVERSITY

Applied Computer Systems

ISSN 2255-8691 (online)
ISSN 2255-8683 (print)
December 2019, vol. 24, no. 2, pp. 150–160
https://doi.org/10.2478/acss-2019-0019
https://content.sciendo.com

Minimal Total Weighted Tardiness in Tight-Tardy Single Machine Preemptive Idling-Free Scheduling

Vadim Romanuke*
Polish Naval Academy, Gdynia, Poland

***Abstract* – Two possibilities of obtaining the minimal total weighted tardiness in tight-tardy single machine preemptive idling-free scheduling are studied. The Boolean linear programming model, which allows obtaining the exactly minimal tardiness, becomes too time-consuming as either the number of jobs or numbers of job parts increase. Therefore, a heuristic based on remaining available and processing periods is used instead. The heuristic schedules 2 jobs always with the minimal tardiness. In scheduling 3 to 7 jobs, the risk of missing the minimal tardiness is just 1.5 % to 3.2 %. It is expected that scheduling 12 and more jobs has at the most the same risk or even lower. In scheduling 10 jobs without a timeout, the heuristic is almost 1 million times faster than the exact model. The exact model is still applicable for scheduling 3 to 5 jobs, where the averaged computation time varies from 0.1 s to 1.02 s. However, the maximal computation time for 6 jobs is close to 1 minute. Further increment of jobs may delay obtaining the minimal tardiness at least for a few minutes, but 7 jobs still can be scheduled at worst for 7 minutes. When scheduling 8 jobs and more, the exact model should be substituted with the heuristic.**

***Keywords* – Boolean linear programming model, heuristic, job scheduling, job preemptions, relative gap, remaining available period, remaining processing period, single machine scheduling, total weighted tardiness.**

I. Introduction

In practice, job scheduling strongly relates to planning, organising, and executing multi-step processes of assembling, manufacturing, building, dispatching, computing, etc. [1], [2]. Applications of the scheduling theory have a strong industrial impact [2], [3]. The optimal schedule allows shortening production and delivery time, reducing costs, relieving from overcharges/overloads, rationally distributing human and other resources [4].

The schedule has a few characteristics, among which total completion time and tardiness are the most important. The tardiness emerging from that every job is desired to be completed till a definite time moment is commonly unavoidable. The time moment is called a due date [3], [5]. Usually, a due date is tightly set after a release date, which implies a definite time moment when processing of the respective job can be started. The difference between due and release dates is not always such that it is sufficient for completing the job processing. Thus, tardiness emerges and it is taken into account until job processing is completed. While it is not completed, tardiness linearly increases [3], [5], [6]. Obviously, greater tardiness leads to greater additional payments.

When jobs are associated with their priority weights, the objective is to find such a schedule whose total weighted tardiness (TWT) would be minimal. Another three options are attached to this problem: the number of machines, on which the jobs are processed, allowance of preemptions and idle time periods [7]. Despite problems of single machine preemptive scheduling by no idling have been studied thoroughly, a few open practical questions still are not answered. The most important one is how to solve scheduling problems of big sizes [1], [2], [5], [8]. The matter is that the existing models allowing one to find an exact solution are too time-consuming, and, thus, a problem of scheduling even a few jobs can be intractable [3], [9]. On the other hand, a few existing heuristic approaches find approximate solutions very efficiently. The heuristic efficiency seems to be tending to 100 % as the number of jobs increases [4], [9]. Then the approximate solution coincides with the exact one, ensuring the same minimal TWT. However, scheduling a few jobs, a heuristic approach may give unacceptable inaccuracy (e.g., see [3], [6]), despite finding a schedule almost flash-likely (compared to the exact model). Therefore, it is necessary to know a point linking a hardly tractable exact model to an accurate heuristic. This point will direct to a transfer from the exact model to the heuristic.

II. Analysis of the Background

As of 2019, the Boolean linear programming model (BLPM) remains the source for exactly solving a wide variety of scheduling problems [7], [9]. The objective is to find such a set of Boolean decision variables whose weighted sum would be minimal. When preemptions are allowed, a job can be divided into equal processing periods (i.e., into job parts). The Boolean decision variable has three indices, indicating at the job number/tag, its part and time moment currently considered [6], [9]. To find an optimal solution, the branch-and-bound approach is effectively used [8], [9]. This is why the model becomes too time-consuming as either the number of jobs or numbers of job parts increase. For instance, scheduling even five jobs divided into four parts each invokes 400 decision variables, whereas increasing the number of jobs just by 1 invokes 576 variables.

A lot of heuristics help in coping with the intractability of the BLPM for scheduling larger numbers of jobs [6], [7], [9].

* Corresponding author's e-mail: romanukevadimv@gmail.com

The heuristics can be divided into two groups. The first group produces schedules in a few stages. At some stage a schedule is found ensuring a TWT, but at the next stage this schedule is partially or entirely changed (or, rather permuted), and a lesser TWT is obtained [8]. The second group of heuristics produces a schedule successively, adding a job in each time moment [9]. These are so-called online scheduling algorithms (OSAs) [6], [10]. It is clear that OSAs are the most effective method owing to which jobs already scheduled can be accomplished without regard to whether the schedule is completed or not.

One of the best OSAs is the approach based on using weighted reciprocals of remaining processing periods (RPPs). This OSA schedules a job, which has the greatest ratio of its priority weight to its RPP [9]. The weight-to-RPP heuristic is very accurate but it fits only to find schedules ensuring the minimum of total weighted completion time (TWCT). Nevertheless, the weight-to-RPP heuristic serves as a basis for scheduling with a minimal TWT. The remaining available period (RAP) is used for this along with RPP, where the corresponding OSA schedules a job, which has the greatest ratio of its priority weight to a maximum of its RPP and RAP [3]. The weight-to-RPP-or-RAP heuristic is not as accurate as the weight-to-RPP heuristic inasmuch as the TWT problem is more complicated as the TWCT problem. This dissimilarity strengthens when the number of tardy jobs (which cannot be completed anyhow before or on their due dates) is close to the number of all jobs.

III. Goal and Steps to Achieve It

As a breach exists in knowledge of how the BLPM could be efficiently substituted by the weight-to-RPP-or-RAP heuristic as the scheduling problem size increases, the goal is to study the heuristic's accuracy and the BLPM's tractability in order to "reconcile" them. The jobs scheduled are to be processed on a single machine. Preemptions are allowed, but idle time periods are not allowed. To achieve the said goal, the following items will be fulfilled. First, the job scheduling problem is stated and its specificity is described. Second, the BLPM and weight-to-RPP-or-RAP heuristic are stated. Then a computational study is carried out, in which the inaccuracy of the heuristics is examined how it varies versus the increasing complexity of the job scheduling problem. Finally, the research result is expected to answer the question of when the heuristic can efficiently substitute the exact model (i. e., when the lossless transfer to the heuristic can be done).

IV. Tight-Tardy Single Machine Preemptive Idling-Free Scheduling

Let it be required that N jobs, where $N \in \mathbb{N} \setminus \{1\}$, are to be scheduled, and job n consists of H_n processing periods, $n = \overline{1, N}$. Each processing period has the same duration, and so H_n is conventionally called the length of job n. The length is measured in integer units, and thus

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

is a vector of all job lengths to be processed on a single machine.

The jobs can have different importance, which is indicated with their priority weights. They can be easily set at natural values, so

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

is a vector of the priority weights.

Processing of every job can be started only since a certain time moment that is called a release date. Once again, the job release dates are conventionally considered as natural values, and so

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

is a vector of the release dates. If idle time periods are not allowed, release dates (3) can be set at monotonously increasing integers, naturally starting from 1:

$$r_n = n \quad \text{by} \quad n = \overline{1, N}\,. \tag{4}$$

Then, the minimal time interval is set at 1. Moreover, the schedule starts at the time moment that is 1. The second time moment of the schedule is 2, and so on. The schedule ends at the time moment that is

$$T = \sum_{n=1}^{N} H_n\,. \tag{5}$$

A vector of due dates

$$\mathbf{D} = \left[d_n\right]_{1\times N} \in \mathbb{N}^N \tag{6}$$

is reasonably set in a similar integer-value way of setting job lengths (1) and release dates (3). Theoretically, due dates (6) can be set at any integers, but in practice they are linked to the job release dates whichever they are. Vector (1) of job lengths also determines due dates (6). If a due date is tightly set after a release date with respect to the job length, tardiness is expected to be significantly great. This is a tight-tardy single machine preemptive idling-free scheduling. In this case, for example,

$$d_n = H_n + r_n - 1 \quad \text{by} \quad n = \overline{1, N}\,. \tag{7}$$

For the case with monotonously increasing release dates (4), due dates (7) are re-written as follows:

$$d_n = H_n + n - 1 \quad \text{by} \quad n = \overline{1, N}\,. \tag{8}$$

In particular, if job 1 is scheduled first with all its H_1 periods, it is completed without tardiness. In general, if job n is scheduled with all its H_n periods starting at time moment n, it is completed without tardiness as well. Then, however, the other jobs become tardy (they cannot be completed without tardiness). Hence, the tight-tardy single machine preemptive idling-free scheduling by (1), (4), and (8) is a class of hard scheduling problems, in which the inaccuracy of finding the minimal TWT has the strongest negative impact. This is almost the worst case, whose successful solution would positively serve just as the principle of minimax guaranteeing decreasing losses in the worst conditions (the maximum of unfavourable states) [11], [12].

V. EXACTLY MINIMAL TWT

In the simplest terms, tardiness is a difference between the moment of the job completion and its due date, if the latter is surpassed by the job completion moment. The TWT takes into account priority weights (2). If job n is completed after time moment $\theta(n; H_n)$, which is

$$\theta(n; H_n) \in \{\overline{1, T}\} \tag{9}$$

by schedule's length (5), the TWT is

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

The schedule should be composed so that sum (10) would be minimal.

The exactly minimal TWT is found by the BLPM in the following way. Let x_{nh_nt} be a 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. This decision variable is weighted with a non-negative value λ_{nh_nt}:

$$\lambda_{nh_nt} = 0 \tag{11}$$

by

$$r_n - 1 + h_n \leqslant t \leqslant T - H_n + h_n \quad \forall h_n = \overline{1, H_n - 1} \tag{12}$$

and

$$\lambda_{nh_nt} = \alpha \tag{13}$$

by a sufficiently great positive integer α (similar to the meaning of infinity) when (12) is not true;

$$\lambda_{nH_nt} = 0 \tag{14}$$

by

$$r_n - 1 + H_n \leqslant t \leqslant d_n \tag{15}$$

and

$$\lambda_{nH_nt} = w_n(t - d_n) \tag{16}$$

by

$$d_n < t \leqslant T \tag{17}$$

and

$$\lambda_{nH_nt} = \alpha \tag{18}$$

when both (15) and (17) are not true; for instance,

$$\alpha = \sum_{n=1}^{N}\sum_{t=1}^{T} w_n t \tag{19}$$

can be used [9]. The goal is to find such a set

$$\left\{\left\{\left\{x^*_{nh_nt}\right\}_{n=1}^{N}\right\}_{h_n=1}^{H_n}\right\}_{t=1}^{T} \in X \tag{20}$$

of the decision variables, on which the sum

$$\vartheta^*(N) = \sum_{n=1}^{N}\sum_{h_n=1}^{H_n}\sum_{t=1}^{T} \lambda_{nh_nt} x^*_{nh_nt} \tag{21}$$

is minimal by constraints constituting a set X of all possible versions of the decision variables [9]:

$$x_{nh_nt} \in \{0, 1\}$$
$$\text{by } n = \overline{1, N} \text{ and } h_n = \overline{1, H_n} \text{ and } t = \overline{1, T}, \tag{22}$$

$$\sum_{t=1}^{T} x_{nh_nt} = 1 \text{ by } n = \overline{1, N} \text{ and } h_n = \overline{1, H_n}, \tag{23}$$

$$\sum_{n=1}^{N}\sum_{h_n=1}^{H_n} x_{nh_nt} = 1 \text{ by } t = \overline{1, T}, \tag{24}$$

$$\sum_{j=t+1}^{T}\sum_{h_n=1}^{H_n-1} x_{nh_nj} + H_n x_{nH_nt} \leqslant H_n$$
$$\text{by } n = \overline{1, N} \text{ and } t = \overline{1, T-1}. \tag{25}$$

Sum (21) is the exactly minimal TWT for those N jobs scheduled according to solution (20). The respective optimal job schedule is

$$\mathbf{S}^* = \left[s_t^*\right]_{1\times T} \text{ by } s_t^* \in \{\overline{1, N}\} \text{ for every } t = \overline{1, T}. \tag{26}$$

Using this schedule, TWT (21) can also be calculated by formula (10).

Obviously, a few optimal schedules (26) ensuring the same minimal TWT (21) on set X can exist. The eventual selection of a single schedule of them is a matter of separate research, although it is not as important as finding TWT (21). For instance, if there are 4 jobs to be scheduled by the minimal TWT and

$$\mathbf{H} = [3 \quad 5 \quad 2 \quad 2], \quad \mathbf{W} = [1 \quad 12 \quad 6 \quad 17], \tag{27}$$

then

$$\mathbf{R} = [1 \quad 2 \quad 3 \quad 4] \text{ and } \mathbf{D} = [3 \quad 6 \quad 4 \quad 5] \tag{28}$$

according to tight-tardy schedule concept (8) by monotonously increasing release dates (4). The BLPM by (9)–(25) finds an optimal schedule

$$\mathbf{S}^* = [1 \quad 2 \quad 3 \quad 4 \quad 4 \quad 3 \quad 2 \quad 2 \quad 2 \quad 2 \quad 1 \quad 1] \tag{29}$$

whose TWT using formula (10) is (note that jobs 1, 2, 3, 4 are completed after time moments 12, 10, 6, 5, respectively)

$$\vartheta^*(4) = \sum_{n=1}^{4} w_n \cdot \max\{0, \theta(n; H_n) - d_n\} =$$
$$= 1 \cdot \max\{0, 12 - 3\} + 12 \cdot \max\{0, 10 - 6\}$$
$$+ 6 \cdot \max\{0, 6 - 4\} + 17 \cdot \max\{0, 5 - 5\} =$$
$$= 1 \cdot 9 + 12 \cdot 4 + 6 \cdot 2 + 17 \cdot 0 = 69. \tag{30}$$

However, another optimal schedule

$$\mathbf{S}^* = [1 \quad 2 \quad 2 \quad 4 \quad 4 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 1 \quad 1] \qquad (31)$$

exists herein, in which completion moments of jobs 2 and 3 are changed into 8 and 10, respectively, but the TWT is the same:

$$\vartheta^*(4) = 1\cdot 9 + 12\cdot \max\{0, 8-6\} + 6\cdot \max\{0, 10-4\} + 17\cdot 0 = 1\cdot 9 + 12\cdot 2 + 6\cdot 6 = 69. \qquad (32)$$

Schedules (29) and (30) differ only at time moments 3, 6, 9, 10, wherein parts of jobs 2 and 3 are interchanged. Although preemptions are allowed, it is commonly supposed that their number should be as minimal as possible. The total number of job shifts (counted as the job tag changes) in schedule (29) is 6, whereas it is 5 in schedule (31). Therefore, the latter is a more acceptable schedule by an additional criterion of the job shift minimum. However, other additional criteria may be applied that can change the schedule preference. For this instance, already having optimal schedules (29) and (31) for (27), (28), schedule

$$\mathbf{S}^* = [1 \quad 2 \quad 4 \quad 4 \quad 3 \quad 3 \quad 2 \quad 2 \quad 2 \quad 2 \quad 1 \quad 1] \qquad (33)$$

is optimal also being easily obtained from (29) by just stitching up the two parts of job 3. The number of job shifts in schedule (33) is 5, and thus schedules (31) and (33) executable by the same minimal TWT become indistinguishable.

VI. Approximate Solution

The weight-to-RPP-or-RAP heuristic is indirectly described in [3]. This is an OSA, which builds a schedule successively, moment by moment. After each next time moment, a job part is added to a current schedule, for all $t = \overline{1, T}$. Let us denote RPPs at the start by

$$\mathbf{Q} = [q_n]_{1\times N} = \mathbf{H} = [H]_{1\times N}. \qquad (34)$$

As time t progresses, vector $\mathbf{Q}$ is changed: one element in $\mathbf{Q}$ is decreased at every time moment until this vector contains N zeros. For every set of available jobs

$$A(t) = \left\{ i \in \{\overline{1, N}\} : r_i \leqslant t \text{ and } q_i > 0 \right\} \subset \{\overline{1, N}\} \qquad (35)$$

a set of RAPs is calculated:

$$b_i = \max\{0, d_i - t + 1\} \quad \forall i \in A(t). \qquad (36)$$

Along with RAP (36) and vector $\mathbf{Q}$ of RPPs, paper [3] claims that the remaining slack

$$\zeta_i = \max\{0, b_i - q_i\} \quad \forall i \in A(t) \qquad (37)$$

must also be found, whereupon a set of decisive ratios

$$\left\{ \frac{w_i}{q_i + \zeta_i} \right\}_{i\in A(t)} \qquad (38)$$

is calculated. With remaining slack (37), however, it is easy to see that the ratio in (38) factually is

$$\frac{w_i}{q_i + \zeta_i} = \frac{w_i}{q_i + \max\{0, b_i - q_i\}} = \frac{w_i}{\max\{q_i, b_i\}} \qquad (39)$$

because the denominator in the central fraction of statement (39) becomes equal to q_i by $b_i < q_i$ and it is b_i by $b_i \geqslant q_i$. Therefore, remaining slack (37) is useless here, and instead of (38) a set of decisive ratios

$$\frac{w_i}{\max\{q_i, b_i\}} \quad \forall i \in A(t) \qquad (40)$$

is considered. The maximal decisive ratio is achieved at subset

$$A^*(t) = \arg\max_{i\in A(t)} \frac{w_i}{\max\{q_i, b_i\}}. \qquad (41)$$

Let us 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 $t = \overline{1, T}$. If $|A^*(t)| = 1$, where

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

then

$$\tilde{s}_t = i^* \text{ by } q_{i^*}^{(\text{obs})} = q_{i^*} \text{ and } q_{i^*} = q_{i^*}^{(\text{obs})} - 1; \qquad (42)$$

otherwise the earliest-releasable job is preferred to be scheduled: if subset (41) is

$$A^*(t) = \{i_l^*\}_{l=1}^{L} \subset A(t) \subset \{\overline{1, N}\} \text{ by } L > 1, \qquad (43)$$

then

$$\tilde{s}_t = i_1^* \text{ by } q_{i_1^*}^{(\text{obs})} = q_{i_1^*} \text{ and } q_{i_1^*} = q_{i_1^*}^{(\text{obs})} - 1. \qquad (44)$$

An approximately minimal TWT is calculated successively for every $n = \overline{1, N}$ using (34)–(36), (40)–(44) 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 time moment $\tilde{\theta}(n; H_n)$. Finally,

$$\tilde{\vartheta}(N) = \sum_{n=1}^{N} w_n \cdot \max\{0, \tilde{\theta}(n; H_n) - d_n\} \qquad (45)$$

is an approximately minimal TWT corresponding to schedule $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$. It is worth noting that this schedule often coincides with the schedule (or, rather one of the schedules) produced by exact solution (20). Nevertheless, the weight-to-RPP-or-RAP heuristic, unlike the BLPM, always returns a single schedule. In the example with (27), (28), the heuristic returns schedule (31).

VII. Relative Gaps

In optimal schedule (26), if $\theta^*(n; H_n)$ is a time moment after which job n is completed, i.e.

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

then exactly minimal TWT is

$$\vartheta^*(N) = \sum_{n=1}^{N} w_n \cdot \max\left\{0,\, \theta^*(n; H_n) - d_n\right\}. \tag{46}$$

Even if schedules $\tilde{\mathbf{S}} = \left[\tilde{s}_t\right]_{1\times T}$ and $\mathbf{S}^* = \left[s^*_t\right]_{1\times T}$ do not coincide, TWTs (45) and (46) can be equal. If they are different, then, obviously,

$$\tilde{\vartheta}(N) > \vartheta^*(N). \tag{47}$$

To compare the heuristic TWT to the exactly minimal TWT, the relative error percentage or the so-called gap is calculated:

$$\varepsilon^*(N) = 100 \cdot \frac{\tilde{\vartheta}(N) - \vartheta^*(N)}{\vartheta^*(N)}. \tag{48}$$

It is obvious that, theoretically, gap (48) is always non-negative. However, if searching for a solution by the BLPM is stopped prematurely, its resulting TWT may be not minimal, and then the factual ("premature") gap may be negative. If Ω is a time period (e.g., in minutes) given to the BLPM to find the minimum, then let $\vartheta^*_\Omega(N)$ be a TWT after the period elapses (this is called the timeout). The respective timeout gap is

$$\varepsilon_{>\Omega}(N) = 100 \cdot \frac{\tilde{\vartheta}(N) - \vartheta^*_\Omega(N)}{\vartheta^*_\Omega(N)}. \tag{49}$$

Note that the case when

$$\vartheta^*_\Omega(N) = \vartheta^*(N) \tag{50}$$

is not excluded. If the BLPM is stopped due to the solution is found, then let $\vartheta^*_{\leqslant\Omega}(N)$ be the non-timeout minimal TWT. The respective non-timeout ("regular") gap is

$$\varepsilon^*_{\leqslant\Omega}(N) = 100 \cdot \frac{\tilde{\vartheta}(N) - \vartheta^*_{\leqslant\Omega}(N)}{\vartheta^*_{\leqslant\Omega}(N)} \tag{51}$$

where, obviously, always

$$\vartheta^*_{\leqslant\Omega}(N) = \vartheta^*(N)$$

and

$$\varepsilon^*_{\leqslant\Omega}(N) = \varepsilon^*(N).$$

An aggregate of premature gap (49) and regular gap (51) can also be defined as a "common" gap, which is

$$\varepsilon(N) = 100 \cdot \frac{\tilde{\vartheta}(N) - \vartheta(N)}{\vartheta(N)}, \tag{52}$$

where $\vartheta(N)$ is a TWT returned by the BLPM regardless of whether it a timeout or a non-timeout value. In common gap (52), it is either

$$\vartheta(N) = \vartheta^*_\Omega(N)$$

or

$$\vartheta(N) = \vartheta^*_{\leqslant\Omega}(N).$$

Hereinafter, when corresponding gaps (49), (51), and (52) are averaged, they will be referred to simply as premature gap (49), regular gap (51), and common gap (52), respectively.

VIII. Generation of Random Instances

To generate random instances of the tight-tardy single machine preemptive idling-free scheduling by (4) and (8), two independent generators for job lengths (1) and priority weights (2) are constructed. Let job lengths (1) be

$$\mathbf{H} = \left[H_n\right]_{1\times N} = \psi\left(4 \cdot \Theta_{\mathbf{H}}(1, N) + 2\right), \tag{53}$$

where operator $\Theta_{\mathbf{H}}(1, N)$ 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 [9], [11]). Statement (53) implies that job lengths are randomly generated within an integer interval from 2 to 5. Let priority weights (2) be

$$\mathbf{W} = \left[w_n\right]_{1\times N} = \psi\left(100 \cdot \Theta_{\mathbf{W}}(1, N) + 1\right), \tag{54}$$

where operator $\Theta_{\mathbf{W}}(1, N)$ runs and returns outputs identically to operator $\Theta_{\mathbf{H}}(1, N)$ but they are independent of each other. Statement (54) implies that priority weights are randomly generated within an integer interval from 1 to 100.

IX. Computation Time Ratio

Before starting the computational study, it is important to know how the computation times of the BLPM and heuristic are compared. Inasmuch as computation time $\tilde{\tau}(N)$ of the heuristic is always far less than computation time $\tau^*(N)$ of achieving minimum (21) by (22)–(25), then it is suitable to use a computation time ratio (CTR)

$$\gamma^*(N) = \frac{\tau^*(N)}{1000 \cdot \tilde{\tau}(N)}. \tag{55}$$

CTR (55) is purely theoretical because the minimum is not always achievable in a given time period. Therefore, CTRs corresponding to premature gap (49), regular gap (51), and common gap (52) should be defined based on timeout computation time $\tau^*_{\Omega}(N)=\Omega$, regular computation time $\tau^*_{\leqslant\Omega}(N)$, and common computation time $\tau(N)$, respectively:

$$\gamma_{>\Omega}(N)=\frac{\Omega}{1000\cdot\tilde{\tau}(N)}, \tag{56}$$

$$\gamma^*_{\leqslant\Omega}(N)=\frac{\tau^*_{\leqslant\Omega}(N)}{1000\cdot\tilde{\tau}(N)}, \tag{57}$$

$$\gamma(N)=\frac{\tau(N)}{1000\cdot\tilde{\tau}(N)}. \tag{58}$$

Again, when corresponding CTRs (56), (57), and (58) are averaged, they will be referred to simply as premature CTR (56), regular CTR (57), and common CTR (58), respectively.

X. Computational Study

Let 400 instances be generated for each $N=\overline{2,10}$ by job lengths (53), priority weights (54), release dates (4), and due dates (8). A reasonable time period, through which a schedule is expected to be found, is 30 minutes. Thus, 3600 job scheduling problems have been generated, each of which has its own TWT, the respective gap (Fig. 1) and CTR. Timeouts have been registered only in scheduling 8, 9, and 10 jobs (Fig. 2). The number of timeouts abruptly increases after 9 jobs. At 10 jobs, 38.5 % of the generated instances (154 of 400) are timeouts (the premature gap is mostly negative).

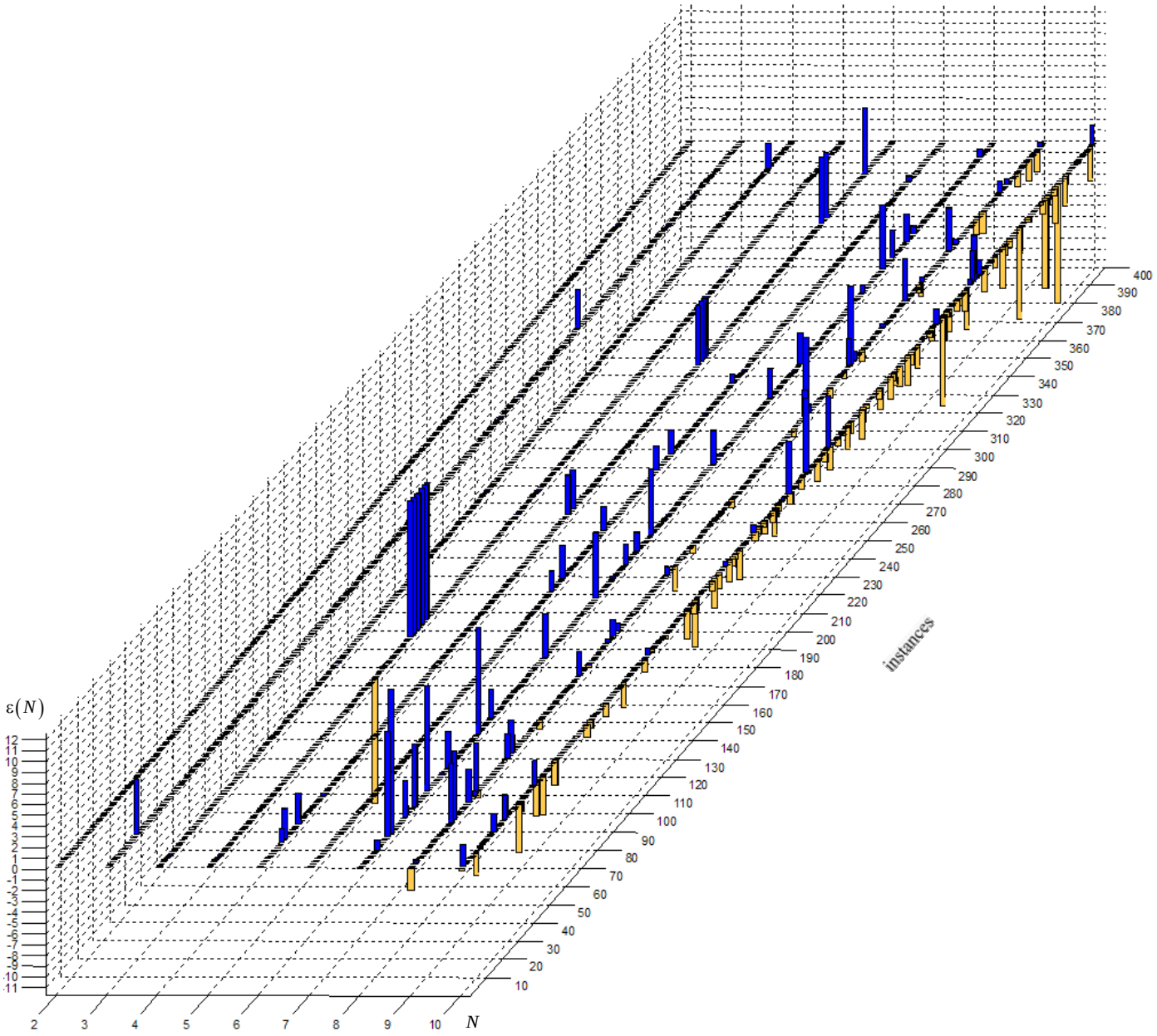

Fig. 1. Common gap (52), where negative gaps are barred with a lighter colour. The heuristic has returned each of 400 schedules of 2 jobs with the respective minimal TWT. While scheduling 3 and 4 jobs, the heuristic "erred" only two times (the gaps in 5.13 % and 3.63 %) and once (the gap in 2.33 %), respectively. In scheduling 5 jobs, seven instances have been generated in a row (128, 130, 132, 133, 135, 137, 138) with the gap in 12.57 %. It is a pseudorandomness artefact.

Regular gap (51) is shown in Fig. 3. This saw-like polyline has a positive linear trend. Thus, it is plausible that the peak at 8 jobs may be not the highest one if to extend the study over 11 jobs and more. The regular gap in 0.12 % in scheduling 10 jobs is statistically valid [12] because 246 of 400 instances are not timeouts (61.5 % instances have regular gaps).

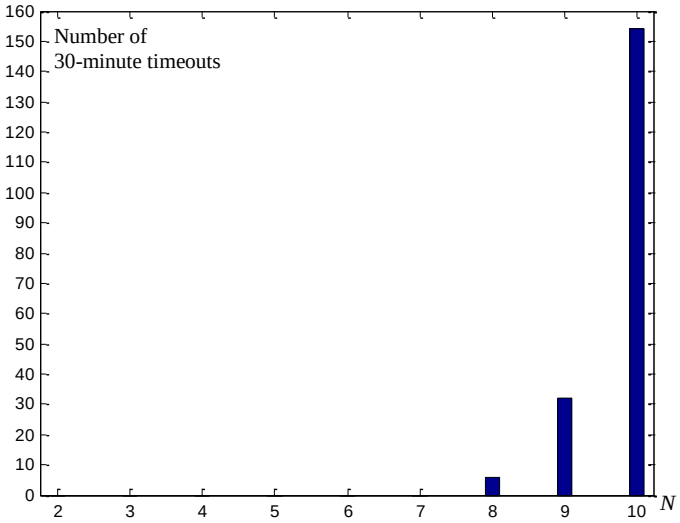

Fig. 2. The number of 30-minute timeouts distributed among 3600 generated instances. Despite the huge number of timeouts at 10 jobs, it is 38.5 % of the generated instances (154 of 400), so it does not break the statistical validity.

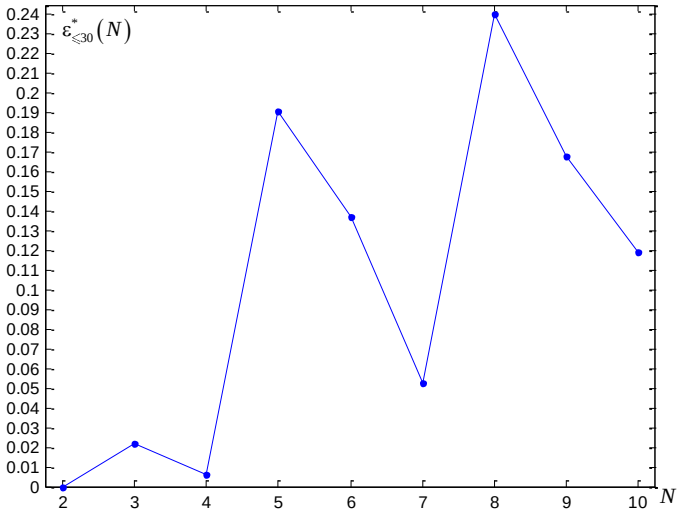

Fig. 3. Regular gap (51) averaged over 400 instances for the number of jobs from 2 to 7, and averaged over 394, 368, and 246 instances of scheduling 8, 9, and 10 jobs, respectively. The peaks may indicate that the number of instances per number of jobs is not sufficiently great. On the other hand, the regular gap itself is very small (although it is not insignificant here) and thus such peaks are unavoidable. After the two sharp peaks, a peak at 11 jobs is likely.

Because of the abruptly increasing number of timeouts, premature gap (49), defined only at scheduling 8, 9, and 10 jobs, is decreasing (Fig. 4). In each of the six timeout instances at scheduling 8 jobs the premature gap is 0, although it does not mean that the minimal TWT is found. Eventually, the premature gap drops so that it seems to be "more exact" than the exact BLPM by almost 1 %. This drop is distinctly seen in common gap (52) shown in Fig. 5. This type of gap is conditionally useful being "distorted" by timeouts.

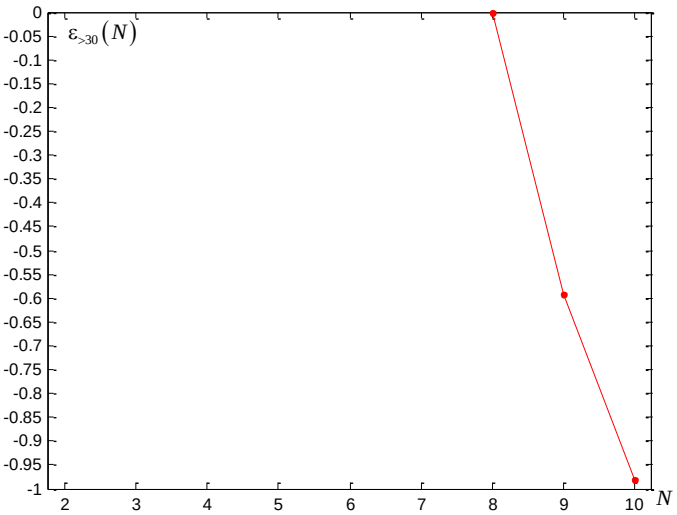

Fig. 4. Premature gap (49) averaged only 6, 32, and 154 instances of scheduling 8, 9, and 10 jobs, respectively. The drop is seemingly almost linear.

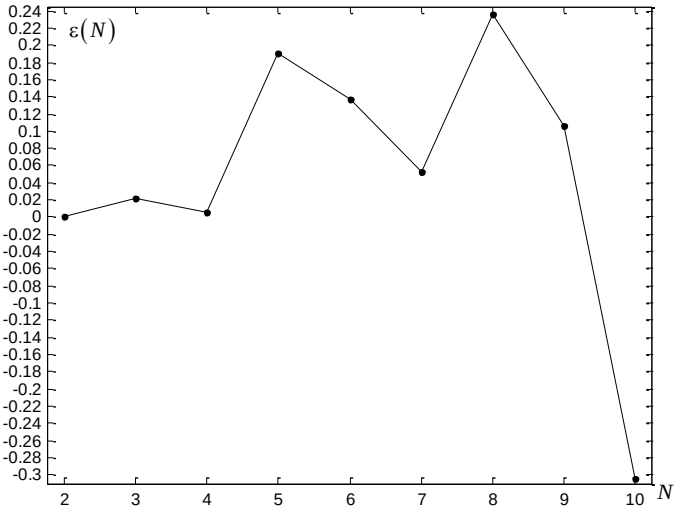

Fig. 5. Common gap (52) averaged over 400 instances for each number of jobs. It coincides with the regular gap from 2 to 7 jobs. Then, from 8 to 10 jobs, timeouts start influencing and the common gap drops similarly as in Fig. 4.

Whichever the averaged gap is, the worst cases must be studied as well. Thus, the maximal gap denoted by $\varepsilon_{\max}(N)$ is shown in Fig. 6. It has three distinctive peaks seen in Fig. 3 and, less prominently, in Fig. 5. Now, this saw-like polyline still has a positive linear trend, but, if to approximate it with a parabolic curve, the top peak at 5 jobs (a result of the pseudorandomness artefact) may be the highest one if to extend the study over 11 jobs and more. The peak at 8 jobs is 9.82 %, and the maximal gap seems to slowly decrease since then. However, the gap in 6.92 % at 10 jobs may be intolerable in a lot of application domains.

In addition to the worst cases, it is necessary 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. It is a fraction of cases when the exact BLPM is factually needless. Let us denote this ratio by $\rho_{\varepsilon=0}(N)$. Figure 7 shows the ratio, which is no less than 0.9289 (the minimum is at 8 jobs). Consequently, the BLPM is needless in no less than 92.89 % of the cases generated by (53) and (54).

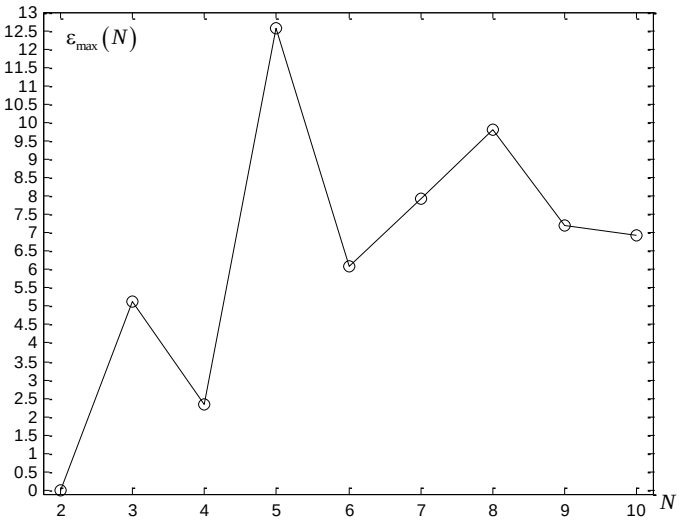

Fig. 6. The common gap maximum taken over 400 instances for each number of jobs. The top peak at 5 jobs is a result of the pseudorandomness artefact viewed and mentioned above (Fig. 1). The gap slowly decreases since 8 jobs.

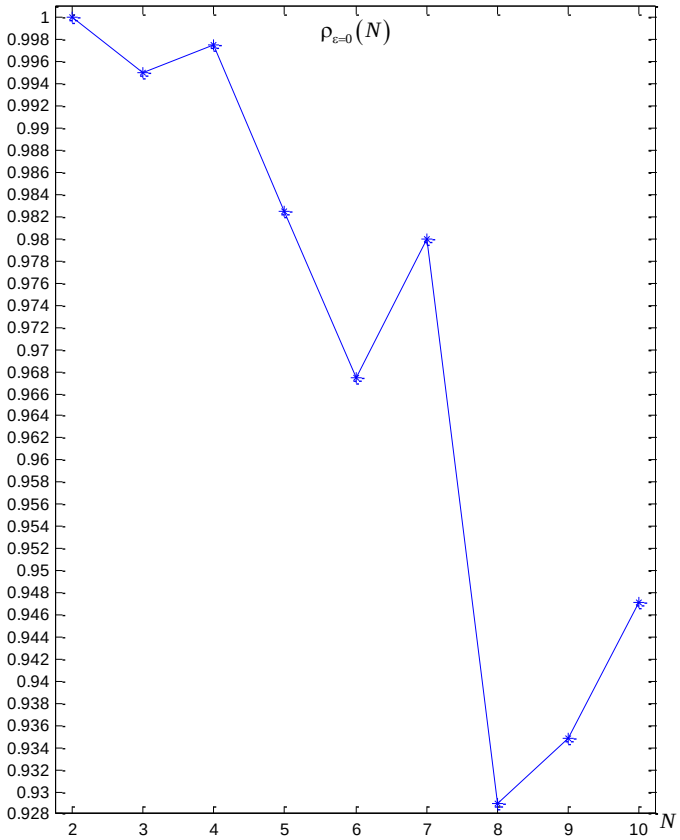

Fig. 7. The ratio of non-timeout instances, in which the heuristic gives the minimal TWT, to the total number of non-timeout instances. It is strictly 1 in scheduling just 2 jobs, independent of the number of processing periods they are divided. The ratio is decreasing till 8 jobs. The minimum of the ratio is at 8 jobs, where 92.89 % of the non-timeout instances have been scheduled by the heuristic as optimally as the BLPM has scheduled them. Thus, the heuristic can efficiently substitute the BLPM at least in 92.89 % of the cases.

A very strong argument for the heuristic is the CTR by (55). Regular and common CTRs (57) and (58) shown in Fig. 8 resemble an exponential growth. In scheduling 10 jobs without a timeout, the heuristic is almost 1 million times faster than the BLPM. If timeouts are admitted, the heuristic acquires roughly the same rapidness in scheduling 9 jobs. Premature CTR (56) here is even more persuasive (3 to almost 4 million times) but it is so huge because the BLPM is prematurely stopped.

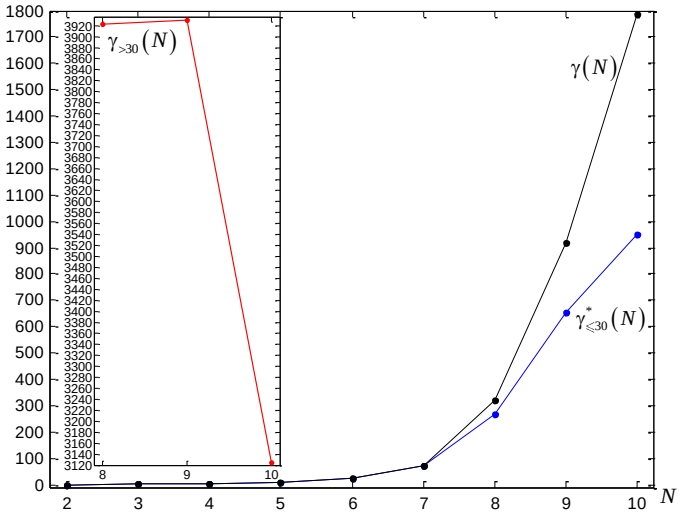

Fig. 8. The exponentially increasing CTRs (57) and (58). CTR (56) put within the same axes is "broken" because of the abrupt intensification of timeouts.

If 11 jobs are scheduled, it turns out that the 30-minute time period is too short for the timeout. Indeed, after increasing the timeout period to 120 minutes (2 hours) for scheduling 11 jobs, the timeouts have been registered in 47 of 100 instances. The number of 120-minute timeouts in scheduling 12 jobs expectedly increases: the BLPM has been prematurely stopped in 81 of 100 instances. Henceforward, the BLPM becomes practically intractable since scheduling more than 10 jobs whose lengths vary from 2 to 5.

A promising fact is that the common gap maximum taken over 100 instances for 11 jobs is 6.44 %, which is slightly less than that for 10 jobs (see Fig. 6). Furthermore, for 12 jobs this maximum is already 2.49 %, which may be tolerable in some application domains. It is noticeable that neither the gap maximum in 6.44 % nor the gap maximum in 2.49 % belongs to the regular gaps. Moreover, among those 53 non-timeout instances of 11 jobs, there are only two identical instances (it is another pseudorandomness artefact similar to that in Fig. 1) whose regular gap is 2.16 %. The remaining 51 non-timeout instances are scheduled by the heuristic with the exactly minimal TWT. Nevertheless, there are 12 pseudorandomness artefacts among those 51 instances (see their job lengths and weights in Fig. 9), so just 39 of them are unique. Among those 19 non-timeout instances of 12 jobs, there are 13 unique instances (Fig. 10), but they all are scheduled by the heuristic with the exactly minimal TWT. Consequently, the ratio of non-timeout instances, in which the heuristic gives the minimal TWT, to the total number of non-timeout instances is promisingly greater than that at 10 jobs (see Fig. 7). It is 96.23 % in scheduling 11 jobs (as 49 out of 51 non-timeout instances, without considering the artefacts, are scheduled with the exactly minimal TWT). It is remarkable that in scheduling 12 jobs, where all 19 non-timeout instances are scheduled with the exactly minimal TWT, this ratio is 100 %. Thus, the minimum at 8 jobs (Fig. 7) is expected to be global.

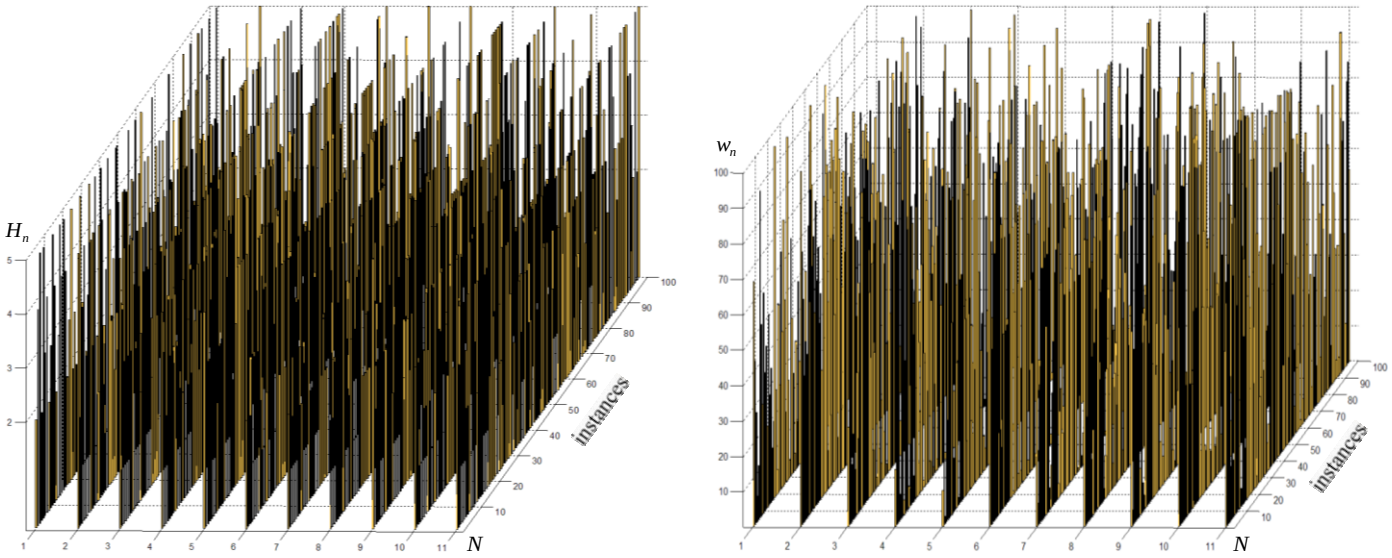

Fig. 9. Job lengths and weights generated for 100 instances of scheduling 11 jobs, where 67 instances are unique. The 53 non-timeout instances are barred with a lighter colour. The 39 non-timeout instances are unique. Such a violation of the pseudorandomness is a consequence of parallelising generations (53) and (54).

It is worth noting that the pseudorandomness artefacts revealed and shown in Fig. 1 (although not clearly seen in Figs. 9 and 10) are caused by the same peculiarities of parallelisation on CPU cores, which have caused so many repetitions of instances in scheduling 11 and 12 jobs. All these artefacts are a consequence of simultaneously generating a few series of integers, either (53) or (54). Surely, this effect may seem negative as those repetitions decrease statistical validity. However, repetitions of schedules are not excluded in common practice. For example, scheduling arrivals and departures at airports (where tardiness often occurs) is tried to be as constant as possible. Similar examples can be stated for railway stations, but the cost of tardiness for them is lesser. Therefore, the pseudorandomness artefacts have been stored and subsequently exposed. They are part of reality and so they should be in the model of computations.

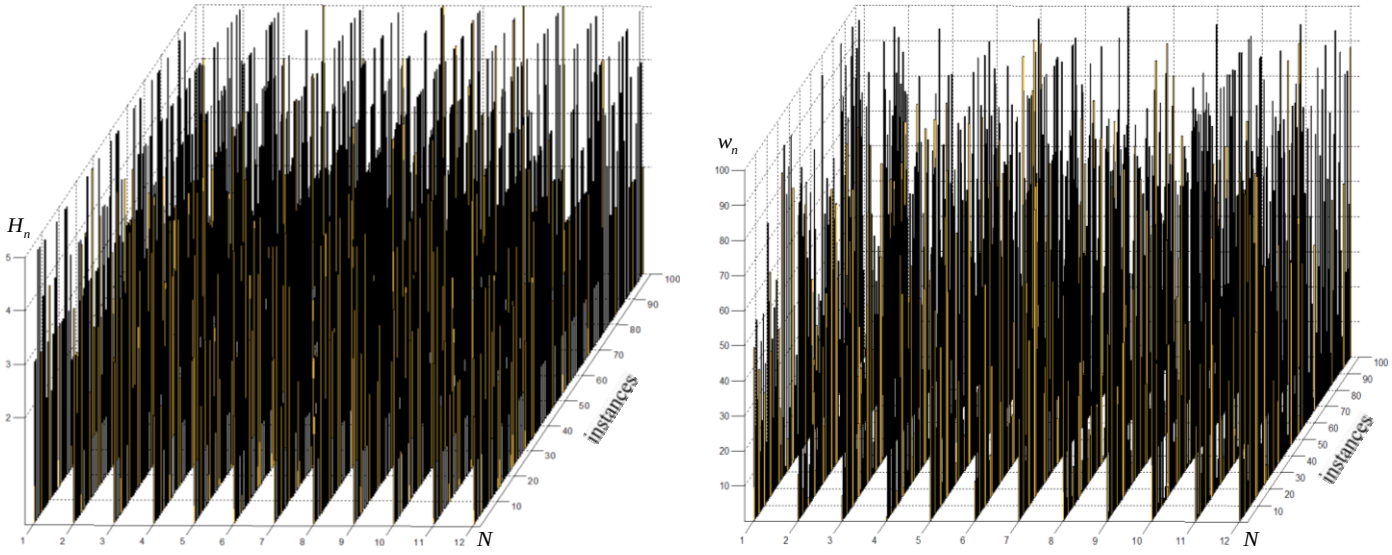

Fig. 10. Job lengths and weights generated for 100 instances of scheduling 12 jobs, where 58 instances are unique (even less than that in the case of scheduling 11 jobs). The 19 non-timeout instances are barred with a lighter colour. The 13 non-timeout instances are unique. Despite one job longer series of integers generated by (53) and (54) is returned concurrently on each of two CPU cores (first, a pair of vectors of job lengths is generated, and then a pair of vectors of priority weights is generated on the cores), the pseudorandomness here has been violated severer than that in generating 100 instances of 11 jobs. It is an artefact.

The timeout instances normally are of jobs having greater processing periods. An example is the instance of 11 jobs scheduled by BLPM in less than 17 seconds. Job lengths in this instance are

$$\mathbf{H} = [2 \quad 2 \quad 2 \quad 2 \quad 5 \quad 2 \quad 5 \quad 3 \quad 2 \quad 2 \quad 2], \qquad (59)$$

its priority weights are

$$\mathbf{W} = [23 \quad 55 \quad 82 \quad 6 \quad 41 \quad 49 \quad 12 \quad 97 \quad 53 \quad 35 \quad 100], \qquad (60)$$

and its due dates are

$$\mathbf{D} = [2 \quad 3 \quad 4 \quad 5 \quad 9 \quad 7 \quad 11 \quad 10 \quad 10 \quad 11 \quad 12], \qquad (61)$$

where the release dates are successively set at 1 to 11 according to (4). An optimal schedule for (59)–(61) is

$$\mathbf{S}^* = \left[s_t^* \right]_{1\times 29} = [1 \;\; 2 \;\; 2 \;\; 3 \;\; 3 \;\; 6 \;\; 6 \;\; 8 \;\; 8 \;\; 8 \;\; 11 \;\; 11 \;\; 9 \;\; 9 \;\; 1 \;\; 10 \;\; 10 \;\; 5 \;\; 5 \;\; 5 \;\; 5 \;\; 5 \;\; 4 \;\; 4 \;\; 7 \;\; 7 \;\; 7 \;\; 7 \;\; 7] \quad (62)$$

and its length (5) is 29. TWT of schedule (62) is 1666, whereas TWT of the 47 timeout instances of 11 jobs varies from 3389 to 7954. The timeout instance of 11 jobs with the minimal TWT is:

$$\mathbf{H} = [5 \;\; 4 \;\; 2 \;\; 2 \;\; 2 \;\; 4 \;\; 2 \;\; 3 \;\; 3 \;\; 2 \;\; 4], \quad (63)$$

$$\mathbf{W} = [28 \;\; 75 \;\; 37 \;\; 25 \;\; 45 \;\; 29 \;\; 94 \;\; 48 \;\; 57 \;\; 50 \;\; 40], \quad (64)$$

$$\mathbf{D} = [5 \;\; 5 \;\; 4 \;\; 5 \;\; 6 \;\; 9 \;\; 8 \;\; 10 \;\; 11 \;\; 11 \;\; 14]. \quad (65)$$

Eventually, the BLPM fails to find an optimal schedule for (63)–(65) as the schedule

$$\tilde{\mathbf{S}} = \left[\tilde{s}_t \right]_{1\times 33} = [1 \;\; 2 \;\; 2 \;\; 2 \;\; 2 \;\; 5 \;\; 7 \;\; 7 \;\; 5 \;\; 10 \;\; 10 \;\; 9 \;\; 9 \;\; 9 \;\; 3 \;\; 3 \;\; 8 \;\; 8 \;\; 8 \;\; 4$$
$$4 \;\; 11 \;\; 11 \;\; 11 \;\; 11 \;\; 6 \;\; 6 \;\; 6 \;\; 6 \;\; 1 \;\; 1 \;\; 1 \;\; 1] \quad (66)$$

found by the heuristic has lesser TWT: it is 3386, which is slightly better (by 0.0886 %) than the BLPM schedule found in 2 hours. Nonetheless, it is unknown whether schedule (66) is optimal. It is remarkable that the length of schedule (66) is just about 13.8 % greater that the length of schedule (62), whereas TWT of schedule (66) is more than twice worse.

XI. Instances of Stronger Discrepancy

All the timeout instances at 8, 9, and 10 jobs are scheduled by the heuristic with the same TWT returned by the BLPM. A stronger discrepancy between the heuristic and BLPM appears at 11 jobs, although for them the timeout period is four times greater. Thus, 42 of 47 timeout instances have schedules whose TWTs do not coincide with TWTs returned by the heuristic for these instances. Only in two instances of those 42 ones the BLPM returns lesser TWTs (by 6.44 % and 0.25 %). For 12 jobs, these things are not any better: 63 of 81 timeout instances have schedules whose TWTs do not coincide with the heuristic TWTs; occasionally, in seven instances of those 63 ones the BLPM returns lesser TWTs (by up to 2.49 %).

The timeout instance of 11 jobs with the maximal TWT (which is 7954 as mentioned above) is scheduled by the heuristic in 0.58 milliseconds. This instance is:

$$\mathbf{H} = [3 \;\; 3 \;\; 5 \;\; 4 \;\; 5 \;\; 5 \;\; 4 \;\; 5 \;\; 5 \;\; 2 \;\; 2], \quad (67)$$

$$\mathbf{W} = [92 \;\; 27 \;\; 51 \;\; 52 \;\; 93 \;\; 100 \;\; 78 \;\; 79 \;\; 42 \;\; 34 \;\; 83], \quad (68)$$

$$\mathbf{D} = [3 \;\; 4 \;\; 7 \;\; 7 \;\; 9 \;\; 10 \;\; 10 \;\; 12 \;\; 13 \;\; 11 \;\; 12]. \quad (69)$$

Eventually, the BLPM fails to find an optimal schedule for (67)–(69) as the schedule

$$\tilde{\mathbf{S}} = \left[\tilde{s}_t \right]_{1\times 43} = [1 \;\; 1 \;\; 1 \;\; 4 \;\; 5 \;\; 5 \;\; 5 \;\; 5 \;\; 5 \;\; 6 \;\; 11 \;\; 11 \;\; 6 \;\; 6 \;\; 6 \;\; 6 \;\; 7 \;\; 7 \;\; 7 \;\; 7$$
$$4 \;\; 4 \;\; 4 \;\; 10 \;\; 10 \;\; 8 \;\; 8 \;\; 8 \;\; 8 \;\; 8 \;\; 3 \;\; 3 \;\; 3 \;\; 3 \;\; 3 \;\; 2 \;\; 2 \;\; 2 \;\; 9 \;\; 9 \;\; 9 \;\; 9 \;\; 9] \quad (70)$$

found by the heuristic has lesser TWT: it is 7716, which is 2.99 % better than the BLPM schedule found in 2 hours. The strongest discrepancy at 11 jobs has been registered in 5.93 %, where the heuristic finds schedule

$$\tilde{\mathbf{S}} = \left[\tilde{s}_t \right]_{1\times 42} = [1 \;\; 2 \;\; 2 \;\; 4 \;\; 4 \;\; 3 \;\; 3 \;\; 3 \;\; 3 \;\; 3 \;\; 10 \;\; 10 \;\; 10 \;\; 7 \;\; 7 \;\; 7 \;\; 7 \;\; 5 \;\; 5 \;\; 5$$
$$5 \;\; 8 \;\; 8 \;\; 8 \;\; 8 \;\; 1 \;\; 1 \;\; 1 \;\; 9 \;\; 9 \;\; 9 \;\; 9 \;\; 6 \;\; 6 \;\; 6 \;\; 6 \;\; 6 \;\; 11 \;\; 11 \;\; 11 \;\; 11 \;\; 11]. \quad (71)$$

TWT of schedule (71) found in 0.48 milliseconds is 6933, whereas the BLPM in 2 hours returns a schedule whose TWT is 7370. If the timeout period was 30 minutes, the resulting TWT and the discrepancy would be likely far worse. Moreover, the strongest discrepancy at 12 jobs is almost twice greater: it is 11.11 %, where the heuristic finds a schedule whose TWT is 8227, whereas the BLPM in 2 hours returns a schedule whose TWT is 9255. Therefore, the timeout period in 2 hours is short for scheduling more than 10 jobs. This is another evidence of practical intractability of the BLPM for scheduling more than 10 jobs whose lengths vary from 2 to 5. Obviously, scheduling (longer) jobs having more processing periods cannot be even thought of. The heuristic then remains the means to do that.

XII. Discussion

The polyline in Fig. 7 seems to be very promising. Indeed, if the heuristic returns the exact solution in at least 92.89 % of the cases, where 3 to 10 jobs are scheduled (2 jobs are always scheduled with the minimal TWT), it is a quite high rate. However, the polyline in Fig. 6 disappoints: the heuristic fails to schedule 5 jobs with the minimal TWT in about 2 cases out of 100 (more precisely, $\rho_{\varepsilon=0}(5) = 0.9825$ and so $1-\rho_{\varepsilon=0}(5) = 0.0175$), but the gap in more than 12 % is quite intolerable. Furthermore, even the least non-zero gap at 4 jobs (Fig. 6), which is 2.33 %, is not tolerable everywhere (e. g., it may rather be at an airport, but not at a railway station).

When more than 7 jobs are scheduled, the risk of a heuristic fail increases (Fig. 7) with the simultaneously increasing risk of a BLPM fail (Fig. 2). At the same time, the gap maximum does not decrease much (Fig. 6) being still intolerable at 10 jobs. Hence, the risk of obtaining a fast and inexact schedule of 3 to 10 jobs cannot be removed. This risk achieves its top at 7.11 % (Fig. 7) when 30-minute timeouts start. Scheduling 11 jobs is less risky (no more than 4 % risk of the heuristic fails), though. Moreover, the expectance of obtaining a fast and exact schedule of 12 jobs is higher. Empirically, based on the 13 unique non-timeout instances, it approximates to 100 % (the zero risk). Nevertheless, it cannot be proved theoretically, these are just estimations. As the job length varies from 2 to 5, which is not much for up to 10 jobs (because in this case instances generated for a fixed number of jobs roughly "resemble" each other), even studying a few instances can be considered as not a small statistical sample [12]. So, the estimations are statistically reliable.

In some application domains like airports, for which the BLPM is too slow, the risk of obtaining a fast but inexact schedule is acceptable. "Risky" schedules yielding lesser profits because of non-minimised tardiness are certainly undesirable, but waiting for a schedule during 30 minutes for any airport is inadmissibly long. In other, less intensive domains, such as railway stations, river ports, building, manufacturing, etc., where schedules are stable, minimal TWT can be ensured by the BLPM.

In the case of equal-length jobs, the scheduling problem of minimising TWT by (4) and (8) is solved trivially. An optimal schedule in this case is composed successively starting from the first job, i.e., it is job 1 (as a whole, with all its processing periods), job 2 (as a whole), job 3 (as a whole), and so on. It is easy to show that such a schedule is not single. Thus, neither the BLPM nor the heuristic is needed for the case of equal-length jobs. This is why the case is not considered during the study.

XIII. Conclusion

Based on the carried out experiments, the minimal TWT in tight-tardy single machine preemptive idling-free scheduling is achieved by the weight-to-RPP-or-RAP heuristic in about 92 % of the cases and more. The risk of a significant gap is not excluded. The risk slightly decreases as the complexity of the job scheduling problem is increased. At the same time, the BLPM can lose its practical tractability at scheduling 8 jobs, at which the gap maximum is still intolerably great. Consequently, the heuristic can efficiently substitute the BLPM in scheduling 3 to 7 jobs with the risk of 1.5 % to 3.2 %. It is expected that scheduling 12 and more jobs has at the most the same risk or even lower.

Inasmuch as the studied tight-tardy scheduling is the worst case, the losses caused by non-minimal TWT will be lesser for other cases, where the tight tardiness is relaxed. The risk estimations may be roughly the same, but they cannot be predicted certainly. Anyway, where possible, it is strongly recommended to schedule just 2 jobs, which is almost instantaneously executed by the heuristic always with minimum TWT. Otherwise, if there are multiple jobs consisting of multiple processing periods, it is recommended to artificially divide jobs so that the resulting number of jobs is 12 or greater. Then the heuristic will return amounts of TWT, which are either sufficiently close to the minimum or achieve the minimum.

The BLPM is applicable for scheduling 3 to 5 jobs, where the averaged computation time varies from 0.1 s to 1.02 s. The maximal computation time for 6 jobs is close to 1 minute. Further increment of jobs may delay obtaining the minimal TWT at least for a few minutes, but 7 jobs can still be scheduled at worst for 7 minutes. When scheduling 8 jobs and more, the BLPM should be substituted with the heuristic.

References

[1] A. S. Uyar, E. Ozcan, and N. Urquhart, Eds. *Automated Scheduling and Planning: From Theory to Practice*. Springer-Verlag Berlin Heidelberg, 2013. https://doi.org/10.1007/978-3-642-39304-4

[2] J. M. Framinan, R. Leisten, and R. R. García, *Manufacturing Scheduling Systems: An Integrated View on Models, Methods and Tools*. Springer-Verlag London, 2014. https://doi.org/10.1007/978-1-4471-6272-8

[3] F. Jaramillo and M. Erkoc, "Minimizing Total Weighted Tardiness and Overtime Costs for Single Machine Preemptive Scheduling," *Computers & Industrial Engineering*, vol. 107, pp. 109–119, May 2017. https://doi.org/10.1016/j.cie.2017.03.012

[4] B. Yang, J. Geunes, and W. J. O'Brien, "A Heuristic Approach for Minimizing Weighted Tardiness and Overtime Costs in Single Resource Scheduling," *Computers and Operations Research*, vol. 31, pp. 1273–1301, Jul. 2004. https://doi.org/10.1016/S0305-0548(03)00080-7

[5] J. M. van den Akker, G. Diepen, and J. A. Hoogeveen, "Minimizing Total Weighted Tardiness on a Single Machine With Release Dates and Equal-Length Jobs," *Journal of Scheduling*, vol. 13, iss. 6, pp. 561–576, Dec. 2010. https://doi.org/10.1007/s10951-010-0181-1

[6] R. Panneerselvam, "Simple Heuristic to Minimize Total Tardiness in a Single Machine Scheduling Problem," *The International Journal of Advanced Manufacturing Technology*, vol. 30, iss. 7–8, pp. 722–726, Oct. 2006. https://doi.org/10.1007/s00170-005-0102-1

[7] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*. Springer Inter. Publishing, 2016. https://doi.org/10.1007/978-3-319-26580-3

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

[9] 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 Science News*, no. 3, pp. 52–62, 2019. https://doi.org/10.20535/kpi-sn.2019.3.164804

[10] S. Haruhiko and S. Hiroaki, *Online Scheduling in Manufacturing: A Cumulative Delay Approach*. Springer-Verlag London, 2013. https://doi.org/10.1007/978-1-4471-4561-5

[11] V. V. Romanuke, "Decision Making Criteria Hybridization for Finding Optimal Decisions' Subset Regarding Changes of the Decision Function," *Journal of Uncertain Systems*, vol. 12, no. 4, pp. 279–291, 2018.

[12] J. O. Berger, Ed. *Statistical Decision Theory and Bayesian Analysis*. New York: Springer, 1985. https://doi.org/10.1007/978-1-4757-4286-2

Vadim V. Romanuke was born in 1979. He graduated from the Technological University of Podillya in 2001. The higher education was received in 2001. In 2006, he received the Degree of Candidate of Technical Sciences in Mathematical Modelling and Computational Methods. The degree of Doctor of Technical Sciences in Mathematical Modelling and Computational Methods was received in 2014. In 2016, Vadim Romanuke received the academic status of Full Professor.

He is a Professor of the Faculty of Mechanical and Electrical Engineering at the Polish Naval Academy. His current research interests concern decision making, game theory, statistical approximation, and control engineering based on statistical correspondence. He has 347 published scientific articles, one monograph, one tutorial, methodical guidelines in functional analysis, mathematical and computer modelling, guidelines for the development of Master Thesis, and guidelines for conflict-controlled systems. Since 2019, Vadim Romanuke has been participating as a scientific collaborator in two budget grant works concerning automation in navigation.

Address for correspondence: 69 Śmidowicza Street, Gdynia, Poland, 81–127.
E-mail: romanukevadimv@gmail.com
ORCID iD: https://orcid.org/0000-0003-3543-3087