UDC 519.161+519.852+519.687.1

JOB ORDER INPUT FOR EFFICIENT EXACT MINIMIZATION OF TOTAL TARDINESS IN TIGHT-TARDY PROGRESSIVE SINGLE MACHINE SCHEDULING WITH IDLING-FREE PREEMPTIONS

Romanuke V. V.

O. S. Popov Odesa National Academy of Telecommunications,
1 Kuznechna St., 65029, Ukraine, Odesa.
romanukevadimv@gmail.com

ПОРЯДОК ВВЕДЕННЯ ЗАВДАНЬ ДЛЯ ЕФЕКТИВНОЇ ТОЧНОЇ МІНІМІЗАЦІЇ ЗАГАЛЬНОГО ЗАПІЗНЮВАННЯ У ЩІЛЬНОМУ ПРОГРЕСУЮЧОМУ ОДНОМАШИННОМУ ПЛАНУВАННІ З ПЕРЕМИКАННЯМИ БЕЗ ПРОСТОЮ

Романюк В. В.

Одеська національна академія зв'язку ім. О. С. Попова,
65029, Україна, м. Одеса, вул. Кузнечна, 1.
romanukevadimv@gmail.com

ПОРЯДОК ВВЕДЕНИЯ ЗАДАНИЙ ДЛЯ ЭФФЕКТИВНОЙ ТОЧНОЙ МИНИМИЗАЦИИ ОБЩЕГО ЗАПАЗДЫВАНИЯ В ПЛОТНОМ ПРОГРЕССИРУЮЩЕМ ОДНОМАШИННОМ ПЛАНИРОВАНИИ С ПЕРЕКЛЮЧЕНИЯМИ БЕЗ ПРОСТОЯ

Романюк В. В.

Одесская национальная академия связи им. А. С. Попова,
65029, Украина, г. Одесса, ул. Кузнечная, 1.
romanukevadimv@gmail.com

Abstract. A schedule ensuring the exactly minimal total tardiness can be found with the respective integer linear programming problem. An open question is whether the exact schedule computation time changes if the job release dates are input into the model in reverse order. The goal is to ascertain whether the job order in tight-tardy progressive single machine scheduling with idling-free preemptions influences the speed of computing the exact solution. The Boolean linear programming model provided for finding schedules with the minimal total tardiness is used. To achieve the said goal, a computational study is carried out with the purpose of estimating the averaged computation time for both ascending and descending orders of job release dates. Instances of the job scheduling problem are generated so that schedules which can be obtained trivially, without the exact model, are excluded. As in the case of equal-length jobs, it has been ascertained that the job order really influences the speed of computing schedules whose total tardiness is minimal. Scheduling two to five jobs is executed on average faster by the descending job order input, where 1 to 3 % speed-up is expected. Further increment of the number of jobs to be scheduled cannot guarantee any speed-up even on average. This result is similar to that in the case of equal-length jobs, but there is no regularity in such an efficient job order input. Without any assurance for a single job scheduling problem, the efficient exact minimization of total tardiness by the descending job order input must be treated as on average only.

Key words: job scheduling; preemptive single machine scheduling; exact model; total tardiness; computation time; ascending job order; descending job order.

Анотація. Розклад, що забезпечує строго мінімальне загальне запізнювання, можна знайти за відповідною цілочисловою задачею лінійного програмування. Відкритим є питання про те, чи змінюється час обчислення точного розкладу, якщо дати запуску завдань вводяться у модель у зворотному порядку. Мета полягає у тому, щоб встановити, чи впливає на швидкість обчислення точного розв'язку порядок завдань у щільному прогресуючому одномашинному плануванні з перемиканнями без простою. Для пошуку розкладів з мінімальним загальним запізнюванням використовується модель бульового лінійного програмування. Для досягнення зазначеної мети

проводиться обчислювальне дослідження з метою оцінки усередненого часу обчислення як для висхідного порядку, так і для спадного порядку дат запуску завдань. Приклади задачі планування завдань генеруються так, що розклади, які можна отримати тривіально, без точної моделі, не розглядаються. Як і у випадку з рівноцінними завданнями, було встановлено, що порядок завдань дійсно впливає на швидкість обчислення розкладів, загальне запізнювання яких мінімальне. Планування від двох до п'яти завдань виконується у середньому швидше за спадним порядком завдань, де очікується прискорення від 1 до 3 %. Подальше збільшення кількості планованих завдань не може гарантувати жодного прискорення, навіть у середньому. Цей результат подібний до випадку з рівноцінними завданнями, але регулярності в такому ефективному порядку введення завдань немає. Без жодних гарантій щодо однієї задачі планування завдань, ефективна точна мінімізація загального запізнювання за спадним порядком завдань повинна трактуватися лише як у середньому.

Ключові слова: планування завдань; планування на одній машині з перемиканнями; точна модель; загальне запізнювання; час обчислення; висхідний порядок завдань; спадний порядок завдань.

Аннотация. Расписание, обеспечивающее строго минимальное общее запаздывание, можно найти по соответствующей целочисленной задаче линейного программирования. Открытым является вопрос о том, меняется ли время вычисления точного расписания, если даты запуска заданий вводятся в модель в обратном порядке. Цель состоит в том, чтобы установить, влияет ли на скорость вычисления точного решения порядок заданий в плотном прогрессирующем одномашинном планировании с переключениями без простоя. Для поиска расписаний с минимальным общим запаздыванием используется модель булевого линейного программирования. Для достижения указанной цели проводится вычислительное исследование с целью оценки усреднённого времени вычисления как для восходящего порядка, так и для нисходящего порядка дат запуска заданий. Примеры задачи планирования заданий генерируются так, что расписания, которые можно получить тривиально, без точной модели, не рассматриваются. Как и в случае с равноценными заданиями, было установлено, что порядок заданий действительно влияет на скорость вычисления расписаний, общее запаздывание которых минимально. Планирование от двух до пяти задач выполняется в среднем быстрее при нисходящем порядке заданий, где ожидается ускорение от 1 до 3 %. Дальнейшее увеличение количества планируемых заданий не может гарантировать никакого ускорения, даже в среднем. Этот результат подобен случаю с равноценными заданиями, но регулярности в таком эффективном порядке введения заданий нет. Без каких-либо гарантий касательно одной задачи планирования заданий, эффективная точная минимизация общего запаздывания при нисходящем порядке заданий должна трактоваться лишь как в среднем.

Ключевые слова: планирование заданий; планирование на одной машине с переключениями; точная модель; общее запаздывание; время вычисления; восходящий порядок заданий; нисходящий порядок заданий.

The idling-free preemptive scheduling. Tight-tardy progressive single machine scheduling [1] with idling-free job preemptions is a problem of great practical importance and impact. It serves as a means of minimizing both costs and time occupation/consumption. A schedule ensuring the exactly minimal total tardiness can be found with the respective integer linear programming problem involving the branch-and-bound approach [1, 2]. Owing to no weights are included, where release dates are set at non-repeating integers from 1 through the total number of jobs, and due dates are tightly set after the respective release dates (although sometimes a few jobs can be completed without tardiness), the exact model simplifies for such tight-tardy progressive single machine scheduling [1, 3]. An open question is whether the exact schedule computation time changes if the release dates are input into the model in reverse order [4]. The first attempt was made in article [5] which showed that a possibility exists to find schedules more efficiently by manipulating the job order in tight-tardy progressive single machine scheduling with idling-free preemptions of equal-length jobs. For instance, schedules of five jobs consisting of two processing periods each were found on average by 14.67 % faster for the descending job order. In another example of seven three-parted jobs, an optimal schedule was found on average in 69.51 seconds by the ascending job order, whereas the descending job order took just 36.52 seconds to find it, saving thus 32.99 seconds. In general, article [5] revealed that, in the case of equal-length jobs, scheduling a fewer jobs divided into a fewer job parts is

executed on average faster by the descending job order. As the number of jobs increases along with increasing the number of their processing periods, the ascending job order becomes more efficient. However, the computation time efficiency by both job orders tends to be irregular. So, now the question is whether similar conclusions and properties are still peculiar to the case when the jobs have different lengths (i. e., whose number of processing periods varies).

The goal and tasks to achieve it. The ascending job order implies inputting the release dates in ascending (i. e., starting from 1) order, and the descending job order implies inputting the release dates in descending (i. e., starting from the last job) order. The goal is to ascertain whether the job order in tight-tardy progressive single machine scheduling with idling-free preemptions influences the speed of computing the exact solution. Just as it was in [5], here the Boolean linear programming model provided for finding schedules with the minimal total tardiness will be used [6, 7]. To achieve the said goal, a computational study should be carried out with a purpose to estimate the averaged computation time for both ascending and descending orders. For this, a pattern of generating instances of the job scheduling problem will be suggested. Unlike article [5], this pattern will be supplemented with a method of generating random job lengths so that every instance would contain tardy jobs. Then the relative difference between the computation times is to be estimated and treated. The research result is expected to either reveal or disprove a possibility to manipulate the job order for obtaining schedules more efficiently, which will be a direct extension of the research result in article [5].

Minimal total tardiness by the varying number of processing periods. Every job is associated with its number of processing periods, release date, and due date. Let job n be of H_n processing periods, and r_n is its release date, d_n is its due date, where $n=\overline{1,\,N}$ by the total number of jobs N, $N\in\mathbb{N}\setminus\{1\}$. Integer r_n is the time moment, at which job n becomes available for processing. All the time moments and the processing periods are measured in the same time units, and thus they are synchronized.

The vector of processing periods (or job lengths)

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

does not have any specific constraints applied to it. Unlike vector (1), vector of release dates

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

is constrained depending on the job order input and the requirement of that job preemptions be idling-free. If the release dates are given in ascending order then

$$r_n=n\quad\forall n=\overline{1,\,N}\,. \tag{3}$$

If the release dates are given in descending order then

$$r_n=N-n+1\quad\forall n=\overline{1,\,N}\,. \tag{4}$$

The vector of due dates

$$\mathbf{D}=\left[d_n\right]_{1\times N}\in\mathbb{N}^N \tag{5}$$

is not purely random as the due dates are tightly set after the release dates, in whichever order they are given:

$$d_n=r_n+H_n-1+b_n\quad\forall n=\overline{1,\,N} \tag{6}$$

for ascending order and

$$d_n=r_n+H_n-1+b_{N-n+1}\quad\forall n=\overline{1,\,N} \tag{7}$$

for descending order, where b_n is a random due date shift generated as

$$b_n=\psi\left(H_n\cdot\zeta\right)\ \text{ for }\ n=\overline{1,\,N} \tag{8}$$

with a pseudorandom number ζ drawn from the standard normal distribution (with zero mean and unit variance), and function $\psi(\xi)$ returning the integer part of number ξ (e. g., see [4, 5]). Due date shifts (8) are generated until

$$d_n \geqslant 1 \quad \forall n = \overline{1,\ N}. \tag{9}$$

If

$$H_n \leqslant H_{n+1} \quad \forall n = \overline{1,\ N-1} \tag{10}$$

and, occasionally,

$$d_n \leqslant d_{n+1} \quad \forall n = \overline{1,\ N-1} \tag{11}$$

for the ascending job order input, then due date shifts (8) are re-generated as well. Symmetrically, if

$$H_n \geqslant H_{n+1} \quad \forall n = \overline{1,\ N-1} \tag{12}$$

and, occasionally,

$$d_n \geqslant d_{n+1} \quad \forall n = \overline{1,\ N-1} \tag{13}$$

for the descending job order input, then due date shifts (8) are generated once again until such shared monotonicity is broken. This is done so because in the case of when either both inequalities (10) and (11) or both inequalities (12) and (13) are true, a schedule ensuring the exactly minimal total tardiness is found trivially, without resorting to any algorithm or model: if (10) and (11) are true for the ascending job order input, an optimal schedule is composed by arranging jobs from the earliest one to the latest one; if (12) and (13) are true for the descending job order input, an optimal schedule is composed by arranging jobs from the latest one to the earliest one. In article [5], this fact was proved for the case of equal-length jobs. For the case of different job lengths, this is going to be proved below.

Once due date shifts (8) are given properly, due dates (6) set in the order corresponding to ascending order of the release dates (3) are

$$d_n = H_n + n - 1 + b_n \quad \forall n = \overline{1,\ N} \tag{14}$$

and due dates (7) set in the order corresponding to descending order of the release dates (4) are

$$d_n = N + H_n - n + b_{N-n+1} \quad \forall n = \overline{1,\ N}. \tag{15}$$

The length of the schedule is

$$T = \sum_{n=1}^{N} H_n. \tag{16}$$

The goal is to minimize the total tardiness through schedule's length (16), i. e. to schedule N jobs so that sum

$$\sum_{n=1}^{N} \max\left\{0, \theta\left(n; H_n\right) - d_n\right\} \tag{17}$$

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

$$\theta\left(n; H_n\right) \in \left\{\overline{1, T}\right\}.$$

This goal is equivalent to finding such decision variables which minimize sum [2, 6, 7]

$$\sum_{n=1}^{N} \sum_{h_n=1}^{H_n} \sum_{t=1}^{T} \lambda_{nh_nt} x_{nh_nt}, \tag{18}$$

where x_{nht} is the decision variable about assigning the h-th part of job n to time moment t: $x_{nht} = 1$ if it is assigned; $x_{nht} = 0$ otherwise. The triple-indexed weights (these ones are not the job priority weights)

$$\left\{\left\{\left\{\lambda_{nh_nt}\right\}_{t=1}^{T}\right\}_{h_n=1}^{H_n}\right\}_{n=1}^{N}$$

are calculated as follows:

$$\lambda_{nh_nt} = 0 \tag{19}$$

by

$$r_n - 1 + h_n \leqslant t \leqslant T - H_n + h_n \quad \forall h_n = \overline{1,\, H_n - 1} \tag{20}$$

and

$$\lambda_{nh_nt} = \alpha \tag{21}$$

by a sufficiently great positive integer α (similar to the meaning of infinity, i. e. it is an infinity "substitute" for real-practice calculations) when (20) is not true;

$$\lambda_{nH_nt} = 0 \tag{22}$$

by

$$r_n - 1 + H_n \leqslant t \leqslant d_n \tag{23}$$

and

$$\lambda_{nH_nt} = t - d_n \tag{24}$$

by

$$d_n < t \leqslant T \tag{25}$$

and

$$\lambda_{nH_nt} = \alpha \tag{26}$$

when both (23) and (25) are not true. In (21) and (26), for instance,

$$\alpha = \sum_{n=1}^{N} \sum_{t=1}^{T} t = \frac{N \cdot T \cdot (T+1)}{2} \tag{27}$$

can be used [1, 5, 6]. The decision variables constraints are as follows:

$$x_{nh_nt} \in \{0, 1\} \quad \text{by} \quad n = \overline{1,\, N} \quad \text{and} \quad h_n = \overline{1,\, H_n} \quad \text{and} \quad t = \overline{1,\, T}, \tag{28}$$

$$\sum_{t=1}^{T} x_{nh_nt} = 1 \quad \text{by} \quad n = \overline{1,\, N} \quad \text{and} \quad h = \overline{1,\, H_n}, \tag{29}$$

$$\sum_{n=1}^{N} \sum_{h=1}^{H_n} x_{nh_nt} = 1 \quad \text{by} \quad t = \overline{1,\, T}, \tag{30}$$

$$\sum_{j=t+1}^{T} \sum_{h_n=1}^{H_n - 1} x_{nh_nj} + H_n x_{nH_nt} \leqslant H_n \quad \text{by} \quad n = \overline{1,\, N} \quad \text{and} \quad t = \overline{1,\, T-1}. \tag{31}$$

An optimal job schedule

$$\mathbf{S}^* = \left[s_t^* \right]_{1 \times T} \quad \text{by} \quad s_t^* \in \{\overline{1,\, N}\} \quad \text{for every} \quad t = \overline{1,\, T} \tag{32}$$

is found by a set of the decision variables at which sum (18) is minimal, where

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

$$\text{and} \quad \theta^*(n; h_n) < \theta^*(n; h_n + 1) \quad \text{for} \quad h_n = \overline{1,\, H_n - 1}.$$

Thus, $\theta^*(n; H_n)$ is a moment after which job n is completed, and, according to sum (17),

$$\vartheta^*(N) = \sum_{n=1}^{N} \max\{0, \theta^*(n; H_n) - d_n\} \tag{33}$$

is the exactly minimal total tardiness for those N jobs. Generally speaking, the problem of minimizing sum (18) by (19) – (27) and constraints (28) – (31) can have multiple solutions (multiple sets of the optimal decision variables), so multiple optimal schedules ensuring the same minimal total tardiness (33) can exist.

As it has been mentioned above, the stated model for calculating schedules with the exactly minimal total tardiness is needless when the job lengths, release dates, and due dates are

given in non-descending order (or in non-ascending order). This has been called the shared monotonicity. For instance, a job scheduling problem with

$$\mathbf{H}=\left[H_n\right]_{1\times 4}=\left[3\quad 4\quad 4\quad 6\right],$$
$$\mathbf{R}=\left[r_n\right]_{1\times 4}=\left[1\quad 2\quad 3\quad 4\right],$$
$$\mathbf{D}=\left[d_n\right]_{1\times 4}=\left[3\quad 3\quad 3\quad 5\right]$$

has an obvious optimal schedule

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 17}=\left[1\quad 1\quad 1\quad 2\quad 2\quad 2\quad 2\quad 3\quad 3\quad 3\quad 3\quad 4\quad 4\quad 4\quad 4\quad 4\quad 4\right],$$

which can be easily figured out by common sense. Its total tardiness

$$\vartheta^*(4)=\sum_{n=1}^{4}\max\left\{0,\theta^*(n;H_n)-d_n\right\}=$$
$$=\max\left\{0,3-3\right\}+\max\left\{0,7-3\right\}+$$
$$+\max\left\{0,11-3\right\}+\max\left\{0,17-5\right\}=24$$

is then trivially calculated. The "reverse" instance, whose inputs

$$\mathbf{H}=\left[H_n\right]_{1\times 4}=\left[6\quad 4\quad 4\quad 3\right],$$
$$\mathbf{R}=\left[r_n\right]_{1\times 4}=\left[4\quad 3\quad 2\quad 1\right],$$
$$\mathbf{D}=\left[d_n\right]_{1\times 4}=\left[5\quad 3\quad 3\quad 3\right]$$

are clearly seen to be given in non-ascending order, is built analogously:

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 17}=\left[4\quad 4\quad 4\quad 3\quad 3\quad 3\quad 3\quad 2\quad 2\quad 2\quad 2\quad 1\quad 1\quad 1\quad 1\quad 1\quad 1\right],$$

where the total tardiness is surely the same. Obviously, a great deal of such examples can be stated. The following theorem rigorously describes a class of tight-tardy progressive single machine scheduling problems with idling-free preemptions which do not need the approach with minimizing sum (18) by (19) – (27) and constraints (28) – (31).

Theorem 1 (the ascending job order input). A single machine scheduling problem with idling-free preemptions of jobs whose processing periods satisfy inequalities (10) by release dates (2) as (3) and due dates (5) by inequalities (11) has an optimal schedule

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times T}\quad\text{by}\quad s_t^*=n\quad\forall t=\overline{\sum_{j=1}^{n-1}H_j+1,\sum_{j=1}^{n}H_j}\quad\text{for}\quad n=\overline{1,N}\tag{34}$$

whose total tardiness

$$\vartheta_{1\ldots N}=\sum_{n=1}^{N}\max\left\{0,\sum_{j=1}^{n}H_j-d_n\right\}\tag{35}$$

is minimal.

Proof. Suppose that amount (35) can be decreased by interchanging some different jobs m and p in schedule (34), where $m<p$. The interchange implies that the job is moved either forward or backward as a comprehensive whole, with all its processing periods standing in a row in schedule (34). Firstly, let

$$\sum_{j=1}^{m}H_j-d_m\geqslant 0\quad\text{and}\quad\sum_{j=1}^{p}H_j-d_p\geqslant 0\tag{36}$$

for these jobs. The collective tardiness of jobs m and p in schedule (34) is

$$\max\left\{0,\sum_{j=1}^{m}H_j-d_m\right\}+\max\left\{0,\sum_{j=1}^{p}H_j-d_p\right\}=$$

$$= \sum_{j=1}^{m} H_j - d_m + \sum_{j=1}^{p} H_j - d_p . \tag{37}$$

Then, in the new schedule, job p is completed at moment

$$\sum_{j=1}^{m-1} H_j + H_p . \tag{38}$$

Jobs from $(m+1)$-th one through $(p-1)$-th one follow moment (38). So, job m is completed at moment

$$\sum_{j=1}^{m-1} H_j + H_p + \sum_{j=m+1}^{p-1} H_j + H_m =$$

$$= \sum_{j=1}^{m-1} H_j + H_m + \sum_{j=m+1}^{p-1} H_j + H_p = \sum_{j=1}^{p} H_j . \tag{39}$$

As job m is completed later than in schedule (34), then its tardiness is greater than that in (34):

$$\max\left\{0, \sum_{j=1}^{p} H_j - d_m\right\} > \max\left\{0, \sum_{j=1}^{m} H_j - d_m\right\} \geqslant 0 . \tag{40}$$

If the tardiness of job p, scheduled now earlier than in schedule (34), is

$$\sum_{j=1}^{m-1} H_j + H_p - d_p \geqslant 0 \tag{41}$$

then the collective tardiness of jobs m and p in the new schedule is not less than the collective tardiness of these jobs in schedule (34):

$$\max\left\{0, \sum_{j=1}^{p} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{m-1} H_j + H_p - d_p\right\} =$$

$$= \sum_{j=1}^{p} H_j - d_m + \sum_{j=1}^{m-1} H_j + H_p - d_p = \tag{42}$$

$$= \sum_{j=1}^{m-1} H_j + H_p - d_m + \sum_{j=1}^{p} H_j - d_p \geqslant \sum_{j=1}^{m} H_j - d_m + \sum_{j=1}^{p} H_j - d_p ,$$

where property $H_p \ldots H_m$ from (10) is used along with (36), (40), and (41). Otherwise, if

$$\sum_{j=1}^{m-1} H_j + H_p - d_p < 0 \tag{43}$$

then the collective tardiness of jobs m and p in the new schedule is greater than (37):

$$\max\left\{0, \sum_{j=1}^{p} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{m-1} H_j + H_p - d_p\right\} =$$

$$= \sum_{j=1}^{p} H_j - d_m = \sum_{j=1}^{p-1} H_j + H_p - d_m >$$

$$> \sum_{j=1}^{p-1} H_j + H_m - d_m + \sum_{j=1}^{m-1} H_j + H_p - d_p =$$

$$= \sum_{j=1}^{m} H_j - d_m + \sum_{j=1}^{p} H_j - d_p, \tag{44}$$

where property $H_p \geqslant H_m$ from (10) is used along with (36), (40), and (43). Secondly, let

$$\sum_{j=1}^{m} H_j - d_m < 0 \quad \text{and} \quad \sum_{j=1}^{p} H_j - d_p \geqslant 0. \tag{45}$$

The collective tardiness of jobs m and p in schedule (34) is

$$\max\left\{0, \sum_{j=1}^{m} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{p} H_j - d_p\right\} =$$

$$= \sum_{j=1}^{p} H_j - d_p. \tag{46}$$

If the tardiness of job p, scheduled now earlier than in schedule (34), satisfies inequality (41), then the collective tardiness of jobs m and p in the new schedule is not less than the collective tardiness of these jobs in schedule (34):

$$\max\left\{0, \sum_{j=1}^{p} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{m-1} H_j + H_p - d_p\right\} =$$

$$= \sum_{j=1}^{p} H_j - d_m + \max\left\{0, \sum_{j=1}^{m-1} H_j + H_p - d_p\right\} \geqslant$$

$$\geqslant \sum_{j=1}^{p} H_j - d_m \geqslant \sum_{j=1}^{p} H_j - d_p, \tag{47}$$

where property $d_m \text{„} d_p$ from (11) is used along with (41) and (45). Otherwise, if the tardiness of job p in the new schedule satisfies inequality (43), then once again the collective tardiness of jobs m and p in the new schedule is not less than the collective tardiness of these jobs in schedule (34):

$$\max\left\{0, \sum_{j=1}^{p} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{m-1} H_j + H_p - d_p\right\} =$$

$$= \sum_{j=1}^{p} H_j - d_m \geqslant \sum_{j=1}^{p} H_j - d_p, \tag{48}$$

where property $d_m \text{„} d_p$ from (11) is used along with (43) and (45). Finally, let

$$\sum_{j=1}^{m} H_j - d_m \geqslant 0 \quad \text{and} \quad \sum_{j=1}^{p} H_j - d_p < 0. \tag{49}$$

The collective tardiness of jobs m and p in schedule (34) is

$$\max\left\{0, \sum_{j=1}^{m} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{p} H_j - d_p\right\} =$$

$$= \sum_{j=1}^{m} H_j - d_m. \tag{50}$$

Owing to $m < p$,

$$\sum_{j=1}^{m-1} H_j + H_p - d_p < \sum_{j=1}^{p} H_j - d_p < 0,$$

so inequality (43) is true again. Then the collective tardiness of jobs m and p in the new schedule is greater than (50):

$$\max\left\{0, \sum_{j=1}^{p} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{m-1} H_j + H_p - d_p\right\} =$$

$$= \sum_{j=1}^{p} H_j - d_m > \sum_{j=1}^{m} H_j - d_m. \qquad (51)$$

Therefore, in each of the cases (36), (45), (49), total tardiness (35) is not decreased. If, occasionally,

$$\sum_{j=1}^{m} H_j - d_m < 0 \quad \text{and} \quad \sum_{j=1}^{p} H_j - d_p < 0, \qquad (52)$$

then the collective tardiness of jobs m and p in schedule (34) is zero, and thus it cannot be decreased. Interchanging a pair of the completing parts of two jobs leads to the same conclusions by straightforwardly using (36) – (52). In general, these statements imply that moving an earlier job forward cannot decrease total tardiness (35). Consequently, schedule (34) is optimal and thus total tardiness (35) is minimal. The theorem has been proved.

It is worth to note that the conditions of Theorem 1 hold for any number of jobs. The "flipped" case (i. e., the descending job order input) is easily proved by using the obvious symmetry in reasoning.

Theorem 2 (the descending job order input). A single machine scheduling problem with idling-free preemptions of jobs whose processing periods satisfy inequalities (12) by release dates (2) as (4) and due dates (5) by inequalities (13) has an optimal schedule

$$\mathbf{S}^* = \left[s_t^*\right]_{1\times T} \quad \text{by} \quad s_t^* = N - n + 1 \quad \forall t = \overline{\sum_{j=1}^{n-1} H_{N-j+1} + 1,\ \sum_{j=1}^{n} H_{N-j+1}} \quad \text{for} \quad n = \overline{1,\ N} \qquad (53)$$

whose total tardiness is (35).

In a partial case of Theorem 1, when

$$d_n = H_n + n - 1 \quad \forall n = \overline{1,\ N} \qquad (54)$$

(i. e., $b_n = 0 \quad \forall n = \overline{1,\ N}$) by (10), due dates (54) themselves are given in ascending order:

$$d_{n+1} - d_n = H_{n+1} + n + 1 - 1 - \left(H_n + n - 1\right) =$$

$$= H_{n+1} - H_n + 1 > 0,$$

so

$$d_{n+1} > d_n \quad \forall n = \overline{1,\ N-1}.$$

Then total tardiness (35) is

$$\vartheta_{1\ldots N} = \sum_{n=1}^{N} \max\left\{0, \sum_{j=1}^{n} H_j - d_n\right\} =$$

$$= \sum_{n=1}^{N} \max\left\{0, \sum_{j=1}^{n} H_j - H_n - n + 1\right\} =$$

$$= \sum_{n=1}^{N} \max\left\{0, \sum_{j=1}^{n-1} H_j - n + 1\right\} =$$

$$= \sum_{n=1}^{N} \left(\sum_{j=1}^{n-1} H_j - n + 1 \right) = \sum_{j=1}^{N-1} (N-j) H_j - \frac{(N-1)N}{2}. \quad (55)$$

Unlike scheduling equal-length jobs, where the optimal schedules for this partial case can be derived (see Theorem 2 in [5]) from the respective partial case of schedule (34), which is

$$\mathbf{S}^* = \left[s_t^* \right]_{1\times(N\cdot H)} \text{ by } s_t^* = n \quad \forall t = \overline{(n-1)H+1,\, nH} \text{ by } H_n = H \text{ for } n = \overline{1,\, N}, \quad (56)$$

partial case (54) by different job lengths satisfying inequalities (10) for

$$H_n = H + n - 1 \quad \forall n = \overline{1,\, N} \quad (57)$$

has an opposite property.

Theorem 3 (the single schedule in ascending order). A single machine scheduling problem with idling-free preemptions of jobs whose processing periods are (57) by release dates (3) and due dates (54) has the single optimal schedule (34).

Proof. Conditions of this theorem are a partial case (strict inequalities) of conditions of Theorem 1, so schedule (34) is optimal. Therefore, it is about to prove that no other optimal schedules exist here. Consider interchanging some different jobs m and p in schedule (34), where $1 < m < p$ (as previously, the interchange implies that the job is moved either forward or backward as a comprehensive whole). The collective tardiness of jobs m and p in optimal schedule (34) is

$$\max\left\{0, \sum_{j=1}^{m} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{p} H_j - d_p\right\} =$$

$$= \max\left\{0, \sum_{j=1}^{m} (H + j - 1) - (H + m - 1 + m - 1)\right\} +$$

$$+ \max\left\{0, \sum_{j=1}^{p} (H + j - 1) - (H + p - 1 + p - 1)\right\} =$$

$$= \max\left\{0, \sum_{j=1}^{m-1} H + \sum_{j=1}^{m} (j-1) - 2(m-1)\right\} +$$

$$+ \max\left\{0, \sum_{j=1}^{p-1} H + \sum_{j=1}^{p} (j-1) - 2(p-1)\right\} =$$

$$= \max\left\{0, (m-1)H + \frac{m}{2}(m-1) - 2(m-1)\right\} +$$

$$+ \max\left\{0, (p-1)H + \frac{p}{2}(p-1) - 2(p-1)\right\} =$$

$$= \max\left\{0, (m-1)\left(H + \frac{m}{2} - 2\right)\right\} +$$

$$+ \max\left\{0, (p-1)\left(H + \frac{p}{2} - 2\right)\right\} =$$

$$= (m-1)\left(H + \frac{m}{2} - 2\right) + (p-1)\left(H + \frac{p}{2} - 2\right). \quad (58)$$

The collective tardiness of jobs m and p after the interchange is

$$\max\left\{0, \sum_{j=1}^{p} H_j - d_m\right\} + \max\left\{0, \sum_{j=1}^{m-1} H_j + H_p - d_p\right\} =$$

$$= \max\left\{0, \sum_{j=1}^{p} (H + j - 1) - (H + m - 1 + m - 1)\right\} +$$

$$+ \max\left\{0, \sum_{j=1}^{m-1} (H + j - 1) + H + p - 1 - (H + p - 1 + p - 1)\right\} =$$

$$= \max\left\{0, (p-1)H + \frac{p}{2}(p-1) - 2(m-1)\right\} +$$

$$+ \max\left\{0, (m-1)H + \frac{m-1}{2}(m-2) - (p-1)\right\} =$$

$$= (p-1)\left(H + \frac{p}{2}\right) - 2(m-1) +$$

$$+ \max\left\{0, (m-1)\left(H + \frac{m-2}{2}\right) - (p-1)\right\}. \quad (59)$$

It is easy to see that the last term in (59) can be both nonnegative and negative. If

$$(m-1)\left(H + \frac{m-2}{2}\right) - (p-1) \geqslant 0 \quad (60)$$

then the collective tardiness of jobs m and p after the interchange is

$$(p-1)\left(H + \frac{p}{2}\right) - 2(m-1) +$$

$$+ (m-1)\left(H + \frac{m-2}{2}\right) - (p-1) =$$

$$= (p-1)\left(H + \frac{p}{2} - 1\right) + (m-1)\left(H + \frac{m-2}{2} - 2\right). \quad (61)$$

The difference between collective tardiness (58) of jobs m and p in optimal schedule (34) and collective tardiness (61) of these jobs after the interchange by (60) is

$$(m-1)\left(H + \frac{m}{2} - 2\right) + (p-1)\left(H + \frac{p}{2} - 2\right) -$$

$$- (p-1)\left(H + \frac{p}{2} - 1\right) - (m-1)\left(H + \frac{m-2}{2} - 2\right) =$$

$$= (m-1) - (p-1) = m - p < 0. \quad (62)$$

Inequality (62) implies that the interchange by (60) only increases collective tardiness (58). Otherwise, if

$$(m-1)\left(H + \frac{m-2}{2}\right) - (p-1) < 0 \quad (63)$$

then the collective tardiness of jobs m and p after the interchange is just

$$(p-1)\left(H + \frac{p}{2}\right) - 2(m-1). \quad (64)$$

The difference between collective tardiness (58) of jobs m and p in optimal schedule (34) and collective tardiness (64) of these jobs after the interchange by (63) is

$$(m-1)\left(H+\frac{m}{2}-2\right)+(p-1)\left(H+\frac{p}{2}-2\right)-$$
$$-(p-1)\left(H+\frac{p}{2}\right)+2(m-1)=$$
$$=(m-1)\left(H+\frac{m}{2}\right)-2(p-1). \quad (65)$$

The difference between the left term of inequality (63) and the last term in (65) is

$$(m-1)\left(H+\frac{m-2}{2}\right)-(p-1)-$$
$$-(m-1)\left(H+\frac{m}{2}\right)+2(p-1)=$$
$$=-(m-1)+(p-1)=p-m>0,$$

so difference (65) between collective tardiness (58) and collective tardiness (64) is negative. This implies that the interchange by (63) only increases the collective tardiness (58). Interchanging a pair of the completing parts of two jobs leads to the same conclusions by straightforwardly using (58) – (65). In general, these statements imply that moving an earlier job forward only increases total tardiness (35). Consequently, optimal schedule (34) is single. The theorem has been proved.

It is worth to note that the conditions of Theorem 3 hold for any number of jobs and for any natural H. The "flipped" case (i. e., for the single schedule in descending order) is easily proved by using the obvious symmetry in reasoning.

Theorem 4 (the single schedule in descending order). A single machine scheduling problem with idling-free preemptions of jobs whose processing periods are

$$H_n=H+N-n \quad \forall n=\overline{1,N} \quad (66)$$

by release dates (4) and due dates

$$d_n=N+H_n-n \quad \forall n=\overline{1,N} \quad (67)$$

has the single optimal schedule (53).

In fact, due dates (54) are

$$d_n=H+2n-2 \quad \forall n=\overline{1,N}$$

after taking into account job lengths (57), and due dates (67) are

$$d_n=2N+H-2n \quad \forall n=\overline{1,N}$$

after taking into account job lengths (66). Schedule (34) for the case of the single schedule in ascending order is simplified as

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times T} \text{ by } s_t^*=n \quad \forall t=\overline{(n-1)\left(H+\frac{n-2}{2}\right)+1,\, n\left(H+\frac{n-1}{2}\right)} \text{ for } n=\overline{1,N} \quad (68)$$

whose total tardiness (35) simplified as

$$\vartheta_{1\ldots N}=\sum_{n=1}^{N}\max\left\{0,\, n\left(H+\frac{n-1}{2}\right)-(H+2n-2)\right\}=$$
$$=\sum_{n=1}^{N}\max\left\{0,\,(n-1)\left(H+\frac{n}{2}-2\right)\right\}=$$
$$=\sum_{n=2}^{N}\max\left\{0,\,(n-1)\left(H+\frac{n}{2}-2\right)\right\}=$$
$$=\sum_{n=2}^{N}(n-1)\left(H+\frac{n}{2}-2\right). \quad (69)$$

Schedule (53) for the case of the single schedule in descending order is simplified as

$$\mathbf{S}^* = \left[s_t^* \right]_{1\times T} \text{ by } s_t^* = N - n + 1 \quad \forall t = \overline{(n-1)\left(H + \frac{n-2}{2} \right) + 1,\, n\left(H + \frac{n-1}{2} \right)} \text{ for } n = \overline{1,\, N}. \tag{70}$$

For instance, a job scheduling problem with

$$\mathbf{H} = \left[H_n \right]_{1\times 5} = \left[3 \quad 4 \quad 5 \quad 6 \quad 7 \right],$$
$$\mathbf{R} = \left[r_n \right]_{1\times 5} = \left[1 \quad 2 \quad 3 \quad 4 \quad 5 \right],$$
$$\mathbf{D} = \left[d_n \right]_{1\times 5} = \left[3 \quad 5 \quad 7 \quad 9 \quad 11 \right],$$

where $H = 3$, has the optimal schedule

$$\mathbf{S}^* = \left[s_t^* \right]_{1\times 25} = \left[1 \quad 1 \quad 1 \quad 2 \quad 2 \quad 2 \quad 2 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \right.$$
$$\left. 4 \quad 4 \quad 4 \quad 4 \quad 4 \quad 4 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \quad 5 \right]$$

which can be easily built by (68). Its total tardiness

$$\vartheta_{1\ldots 5} = \sum_{n=2}^{5} (n-1)\left(H + \frac{n}{2} - 2 \right) =$$
$$= \sum_{n=2}^{5} (n-1)\left(1 + \frac{n}{2} \right) = 30$$

is then trivially calculated by (69). The "reverse" instance, whose inputs

$$\mathbf{H} = \left[H_n \right]_{1\times 5} = \left[7 \quad 6 \quad 5 \quad 4 \quad 3 \right],$$
$$\mathbf{R} = \left[r_n \right]_{1\times 5} = \left[5 \quad 4 \quad 3 \quad 2 \quad 1 \right],$$
$$\mathbf{D} = \left[d_n \right]_{1\times 5} = \left[11 \quad 9 \quad 7 \quad 5 \quad 3 \right]$$

are clearly seen to be given in descending order, is built analogously by (70):

$$\mathbf{S}^* = \left[s_t^* \right]_{1\times 25} = \left[5 \quad 5 \quad 5 \quad 4 \quad 4 \quad 4 \quad 4 \quad 3 \quad 3 \quad 3 \quad 3 \quad 3 \right.$$
$$\left. 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 2 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \right],$$

where the total tardiness is surely the same.

Obviously, all those "naive" cases described by Theorems 1 – 4 are excluded from the computational study. However, the exclusion does not exclude cases in which the schedule trivially coincides with one of optimal schedules (34), (53), (68), (70), although not obeying the conditions of these theorems (see Fig. 4 in [5]). Such coincidences are just occasional.

A pattern of generating instances of the job scheduling problem. Different job lengths are randomly generated [6]:

$$H_n = \psi(4\upsilon + 2) \text{ for } n = \overline{1,\, N} \tag{71}$$

with a pseudorandom number υ drawn from the standard uniform distribution on the open interval $(0;\, 1)$. When job lengths (71) and due date shifts (8) by some N are generated for the ascending job order input so that inequality (9) holds and at least one of the inequalities in (10) and (11) is violated, then an ascending order schedule by job lengths $\{H_n\}_{n=1}^{N}$, release dates (3), and due dates (14) is computed by minimizing sum (18) by (19) – (27) and constraints (28) – (31). Alternatively, a descending order schedule by job lengths

$$\{H_n\}_{n=1}^{N} \text{ after } H_j^{(\text{obs})} = H_j \quad \forall j = \overline{1,\, N} \text{ and } H_n = H_{N-n+1}^{(\text{obs})} \text{ for } n = \overline{1,\, N}, \tag{72}$$

release dates (4), and due dates (15) is computed as well, where job lengths (72), release dates (4), and due dates (15) are obtained by just reversing (i. e., flipping the left and right) job lengths $\{H_n\}_{n=1}^{N}$, release dates (3), and due dates (14) for the ascending job order input.

At a fixed number of jobs N and for a job scheduling problem instance tagged by an integer c, denote the schedule computation times by ascending order and descending order by

$\delta_{Asc}(N,c)$ and $\delta_{Desc}(N,c)$ in seconds, respectively. Each of these amounts implies computation time spent on just searching the solution to the problem of minimizing sum (18) by (19) – (27) and constraints (28) – (31), i. e. on exploring nodes by the branch-and-bound algorithm. At that, the time spent on forming constraints (28) – (31) is not counted in $\delta_{Asc}(N,c)$ and $\delta_{Desc}(N,c)$. Therefore, these amounts in article [5] were called inner computation times. However, article [5] showed that difference between inner computation times and outer computation times (which count also time spent on forming the constraints) is negligible.

If the total number of the instances is C, then the averaged inner computation times are

$$\delta_{Asc}(N)=\frac{1}{C}\sum_{c=1}^{C}\delta_{Asc}(N,c) \tag{73}$$

and

$$\delta_{Desc}(N)=\frac{1}{C}\sum_{c=1}^{C}\delta_{Desc}(N,c). \tag{74}$$

In percentage terms, the relative difference between inner computation times (73) and (74) is

$$\mu(N)=100\cdot\frac{\delta_{Asc}(N)-\delta_{Desc}(N)}{\delta_{Asc}(N)}. \tag{75}$$

Relative difference (75) will be estimated by $N=\overline{2,8}$ for $C=250$.

Computational study. The computational study is executed on CPU Intel Core i5-7200U@2.50 GHz using MATLAB R2018a. Relative difference (75) between inner computation times (73) and (74) is shown in Fig. 1, where the horizontal zero level line is imposed. This line allows seeing where the schedule is computed faster by the respective job order input.

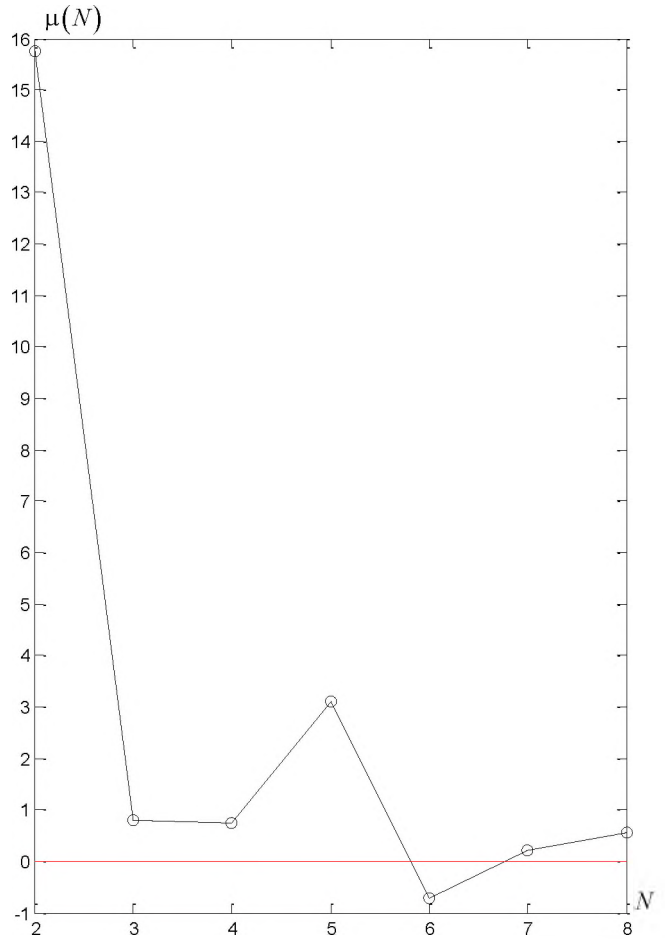

Figure 1 – Relative difference (75) between inner computation times (73) and (74)

Fig. 2 shows the percentage of instances at each number of jobs, which have coincided with either optimal schedule (34) or (53) for the ascending job order input and descending job order input, respectively, although not obeying the conditions of Theorem 1 and Theorem 2. Comparing Fig. 2 to Fig. 4 in [5], it is clear that the number of such occasional coincidences is lesser for the case of when the jobs have different lengths.

However, the number of re-generations of job scheduling problem instances caused by simultaneously true inequalities (10) and (11) or (12) and (13), or just by violating inequalities (9), is still huge (Fig. 3). Despite this number decreases as the number of jobs increases, it takes roughly a half of all instances to be re-generated at even scheduling eight jobs (where the ratio in Fig. 3 is roughly equal to 1).

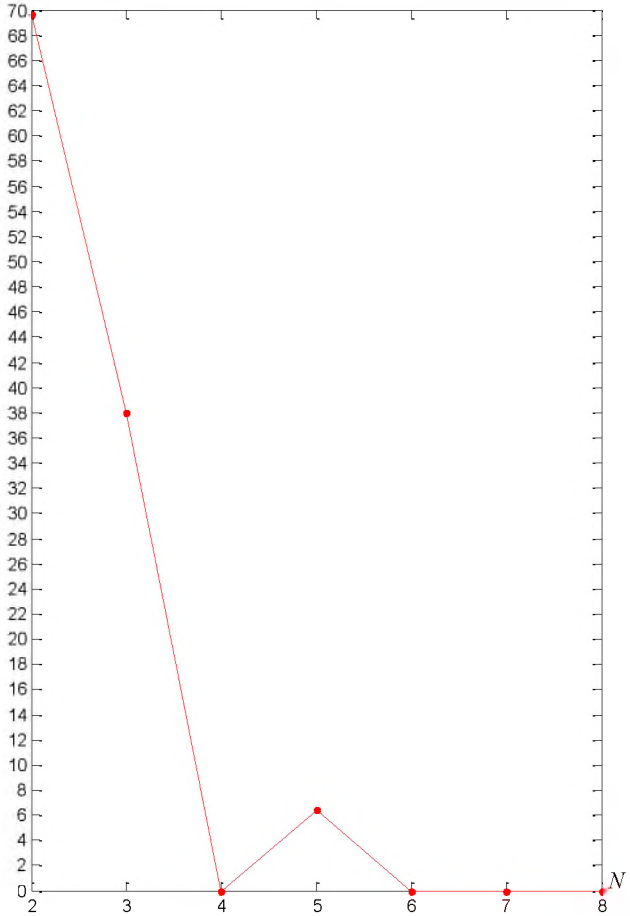

Figure 2 – The percentage of instances, which have coincided with either optimal schedule (34) or (53) for the ascending job order input and descending job order input, respectively, although not obeying the conditions of Theorem 1 and Theorem 2

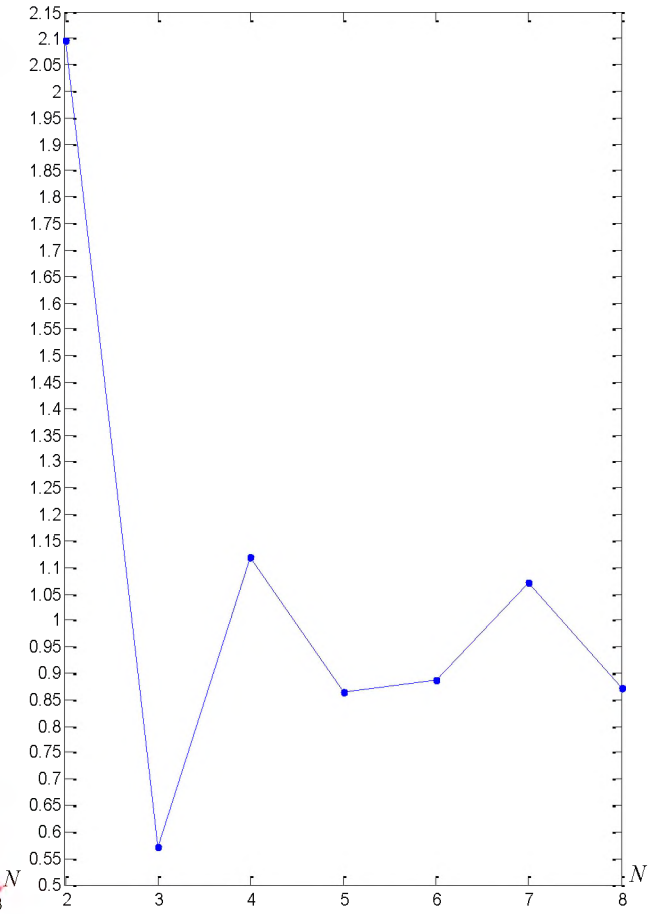

Figure 3 – The ratio of the number of re-generations of due date shifts (8), which are caused by simultaneously true inequalities (10) and (11) or (12) and (13), or just by violating inequalities (9), to the total number of instances to be generated at a fixed number of jobs

In general, the trends of polylines in Fig. 1 – 3 are pretty similar. Repetitions of such a study (i. e., estimations by $N = 2, 8$ for $C = 250$) produce polylines whose trends are roughly the same. Surely, the meaning and impact of Fig. 1 are far more important that those of Fig. 2 and 3, so the obtained estimation of the percentage of relative difference (75) is to be discussed in a more specific way.

Discussion. Scheduling just two jobs is a trivial problem which is successfully solved by heuristics [1]. Therefore, the possibility to obtain an optimal schedule of two jobs by almost 15.7 % faster with the descending job order input has no practical impact. A more interesting case is when three jobs are scheduled (not always the heuristics to minimize total tardiness

achieve the minimum), where it is expected to obtain an optimal schedule by almost 1 % faster with the descending job order input. For instance, the problem with

$$\mathbf{H}=[4\ \ 3\ \ 6],\ \mathbf{R}=[1\ \ 2\ \ 3],\ \mathbf{D}=[4\ \ 6\ \ 3], \tag{76}$$

whose optimal schedule (by the ascending job order input)

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 13}=[1\ \ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3] \tag{77}$$

with $\vartheta^*(3)=11$ is obtained in 691 milliseconds, is solved almost thrice faster (!) by the descending job order input (in 231 milliseconds) as

$$\mathbf{H}=[6\ \ 3\ \ 4],\ \mathbf{R}=[3\ \ 2\ \ 1],\ \mathbf{D}=[3\ \ 6\ \ 4], \tag{78}$$

whence an optimal schedule has a view of

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 13}=[3\ \ 3\ \ 3\ \ 3\ \ 2\ \ 2\ \ 2\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1]. \tag{79}$$

Schedules (77) and (79), although not obeying the conditions of Theorem 1 and Theorem 2, have just coincided with schedules (34) and (53). In another example with $\vartheta^*(4)=22$, an optimal schedule

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 18}=[1\ \ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 4\ \ 4\ \ 4\ \ 4\ \ 4\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3] \tag{80}$$

of four jobs by

$$\mathbf{H}=[4\ \ 3\ \ 6\ \ 5],\ \mathbf{R}=[1\ \ 2\ \ 3\ \ 4],\ \mathbf{D}=[4\ \ 6\ \ 3\ \ 6] \tag{81}$$

is found in 1308 milliseconds, whereas the same schedule in a view of

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 18}=[4\ \ 4\ \ 4\ \ 4\ \ 3\ \ 3\ \ 3\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 2\ \ 2\ \ 2] \tag{82}$$

for

$$\mathbf{H}=[5\ \ 6\ \ 3\ \ 4],\ \mathbf{R}=[4\ \ 3\ \ 2\ \ 1],\ \mathbf{D}=[6\ \ 3\ \ 6\ \ 4] \tag{83}$$

is found in 693 milliseconds, i. e. by almost 88.7 % faster. So, why is the descending job order input only 1 % to 3 % faster in scheduling a few jobs? The matter is that the relative difference is an average, so there are also examples with negative influence of the descending job order input. Thus, the problem with

$$\mathbf{H}=[6\ \ 3\ \ 5],\ \mathbf{R}=[1\ \ 2\ \ 3],\ \mathbf{D}=[8\ \ 6\ \ 3], \tag{84}$$

whose optimal schedule (by the ascending job order input)

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 14}=[1\ \ 2\ \ 2\ \ 2\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3] \tag{85}$$

with $\vartheta^*(3)=12$ is obtained in 269 milliseconds, is solved by 65.6 % slower by the descending job order input (in 445 milliseconds) as

$$\mathbf{H}=[5\ \ 3\ \ 6],\ \mathbf{R}=[3\ \ 2\ \ 1],\ \mathbf{D}=[3\ \ 6\ \ 8], \tag{86}$$

and

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 14}=[3\ \ 2\ \ 2\ \ 2\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 1\ \ 1\ \ 1\ \ 1\ \ 1]. \tag{87}$$

It is worth to note that problems with (76) and (84) differ in lengths of the first and third jobs and in the due date of the first job. Eventually, problem with (84) has job lengths less appearing like sorted in ascending order (they appear more as sorted in descending order). Moreover, due dates in (84) are indeed sorted in descending order. Another counterexample is built on the base of the problem with (81): the problem with

$$\mathbf{H}=[6\ \ 3\ \ 5\ \ 6],\ \mathbf{R}=[1\ \ 2\ \ 3\ \ 4],\ \mathbf{D}=[8\ \ 6\ \ 3\ \ 9] \tag{88}$$

whose optimal schedule (by the ascending job order input)

$$\mathbf{S}^*=\left[s_t^*\right]_{1\times 20}=[1\ \ 1\ \ 2\ \ 2\ \ 2\ \ 1\ \ 1\ \ 1\ \ 1\ \ 3\ \ 3\ \ 3\ \ 3\ \ 3\ \ 4\ \ 4\ \ 4\ \ 4\ \ 4\ \ 4] \tag{89}$$

with $\vartheta^*(4)=23$ is obtained in 3.063 seconds, is solved by 20.8 % slower by the descending job order input (in 3.7 seconds) as

$$\mathbf{H}=\begin{bmatrix}6 & 5 & 3 & 6\end{bmatrix},\ \mathbf{R}=\begin{bmatrix}4 & 3 & 2 & 1\end{bmatrix},\ \mathbf{D}=\begin{bmatrix}9 & 3 & 6 & 8\end{bmatrix} \tag{90}$$

and

$$\mathbf{S}^{*}=\left[s_t^{*}\right]_{1\times 20}=\begin{bmatrix}4 & 4 & 3 & 3 & 3 & 4 & 4 & 4 & 4 & 2 & 2 & 2 & 2 & 2 & 1 & 1 & 1 & 1 & 1 & 1\end{bmatrix}. \tag{91}$$

Once again, having both the problems with (81) and (88) as the ascending job order input, it is worth to note that the problem with (81) is more like to be presented "in ascending order" than the problem with (88). The same concerns the problems with (83) and (90): the job lengths and due dates in (83) are more like to be presented "in descending order" than those in (90). Therefore, the positive average relative difference between computation times for the ascending and descending job order inputs in scheduling three to five jobs may be violated if the job order input appears more to be like the opposite order (ascending order as descending and vice versa).

The vector of processing periods formed by (71) does not have any features of either the ascending or descending order. The only "carrier" of the job order input feature (apart from the release dates) is the due dates. As the number of jobs increases, this feature may disappear as then the definite order (defined by the release dates) in the due dates shatters. Consequently, the speed of computing the exact solution at scheduling six jobs or more cannot be reliably increased even on average. Nevertheless, scheduling a lesser number of jobs can be sped up significantly if this routine is repeatedly continued. For example, an optimal schedule

$$\mathbf{S}^{*}=\left[s_t^{*}\right]_{1\times 17}=\begin{bmatrix}1 & 1 & 1 & 3 & 3 & 5 & 5 & 5 & 5 & 4 & 4 & 4 & 4 & 2 & 2 & 2 & 2\end{bmatrix} \tag{92}$$

of five jobs by

$$\mathbf{H}=\begin{bmatrix}3 & 4 & 2 & 4 & 4\end{bmatrix},\ \mathbf{R}=\begin{bmatrix}1 & 2 & 3 & 4 & 5\end{bmatrix},\ \mathbf{D}=\begin{bmatrix}6 & 9 & 6 & 7 & 9\end{bmatrix} \tag{93}$$

with $\vartheta^{*}(5)=14$ is found in 822 milliseconds, whereas the same schedule in a view of

$$\mathbf{S}^{*}=\left[s_t^{*}\right]_{1\times 17}=\begin{bmatrix}5 & 5 & 3 & 3 & 2 & 5 & 2 & 2 & 2 & 4 & 4 & 4 & 4 & 1 & 1 & 1 & 1\end{bmatrix}, \tag{94}$$

which is not derived straightforwardly from schedule (92) by just flipping the left and right, for

$$\mathbf{H}=\begin{bmatrix}4 & 4 & 2 & 4 & 3\end{bmatrix},\ \mathbf{R}=\begin{bmatrix}5 & 4 & 3 & 2 & 1\end{bmatrix},\ \mathbf{D}=\begin{bmatrix}9 & 7 & 6 & 9 & 6\end{bmatrix} \tag{95}$$

is found in 472 milliseconds, i. e. by almost 74.3 % faster. Here the difference is 350 milliseconds. Thus, if this routine is repeated for 1000 times, the descending job order input by (95) saves 350 seconds; 10000 repetitions save more than 58 minutes. Obviously, the saved computation time grows as the job processing periods are increased: e. g., if the job lengths in the example with (93), (95) are twice increased, an optimal schedule by the ascending job order input is found in 36.25 seconds, whereas the descending job order input takes 30.24 seconds. Then, after just 1000 repetitions, the saved time is more than 100 minutes (!), although here the descending job order input is only 19.87 % faster.

Conclusions. As in the case of equal-length jobs, it has been ascertained that the job order in tight-tardy progressive single machine scheduling with idling-free preemptions really influences the speed of computing schedules whose total tardiness is minimal. Based on the pattern of generating instances of the job scheduling problem, in which for avoiding trivial schedules both job lengths and due dates are neither simultaneously given in non-descending order, nor are simultaneously given in non-ascending order, it has been revealed that scheduling two to five jobs is executed on average faster by the descending job order input. Further increment of the number of jobs to be scheduled cannot guarantee any speed-up even on average. The reason is the due dates in either ascending or descending job order input start losing their feature of "approximately being sorted" in ascending or descending order for a greater number of jobs. The "disappearing" order is explained with that the due dates are formed by adding due date shifts which are random values distributed normally (for a shorter sequence of job lengths, the due dates definite order defined by the release dates is less "shattered").

The average speed-up is about 1 to 3 % by the descending job order input at scheduling two to five jobs, but not every instance will have such a positive impact. Contrary to this, the real speed-up for a single instance can achieve 20 to 80 % and more. However, it is balanced by

negative impact of other instances (where the definite order is less "prominent" or just appears more to be like the opposite order), so the final result of 1 to 3 % speed-up is expected. This result is similar to that of article [5], where the descending job order computation time can be shorter up to 10 % and more in scheduling two to six jobs divided into two to four or even five parts each, but there is no regularity in such an efficient job order input. Without any assurance for a single job scheduling problem, the efficient exact minimization of total tardiness by the descending job order input must be treated as on average only.

The research should be furthered by studying the case when different priority weights are considered for exactly minimizing total weighted tardiness. An eventual comparison and arrangement of the respective results for the three classes of job scheduling problems (total tardiness by equal job lengths [5], total tardiness by different job lengths, total weighted tardiness) should be made.

REFERENCES:

1. Romanuke V. V. "Accurate total weighted tardiness minimization in tight-tardy progressive single machine scheduling with preemptions by no idle periods." KPI Science News, no. 5 – 6, 2019, pp. 26 – 42.

2. Ku W.-Y., Beck J. C. "Mixed Integer Programming models for job shop scheduling: A computational analysis." Computers & Operations Research, vol. 73, 2016, pp. 165 – 173.

3. Pinedo M. L. Planning and Scheduling in Manufacturing and Services. Springer-Verlag New York, 2009: 536 p.

4. Romanuke V. V. "A heuristic's job order gain for total weighted completion time minimization in the preemptive scheduling problem by subsequent length-equal job importance growth." Visnyk of the Lviv University. Series Appl. Math. and Informatics, iss. 27, 2019, pp. 108 – 117.

5. Romanuke V. V. "Efficient exact minimization of total tardiness in tight-tardy progressive single machine scheduling with idling-free preemptions of equal-length jobs." KPI Science News, no. 1, 2020, pp. 27 – 39.

6. Romanuke V. V. "Minimal total weighted tardiness in tight-tardy single machine preemptive idling-free scheduling." Applied Computer Systems, vol. 24, no. 2, 2019, pp. 150 – 160.

7. Batsyn M., Goldengorin B., Pardalos P., Sukhov P. "Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time." Optimization Methods & Software, vol. 29, no. 5, 2014, pp. 955 – 963.

ЛІТЕРАТУРА:

1. Romanuke V. V. Accurate total weighted tardiness minimization in tight-tardy progressive single machine scheduling with preemptions by no idle periods / V. V. Romanuke // KPI Science News. – 2019. – No. 5–6. – P. 26 – 42.

2. Ku W.-Y. Mixed Integer Programming models for job shop scheduling: A computational analysis / W.-Y. Ku, J. C. Beck // Computers & Operations Research. – 2016. – Vol. 73. – P. 165 – 173.

3. Pinedo M. L. Planning and Scheduling in Manufacturing and Services. – Springer-Verlag New York, 2009. – 536 p.

4. Romanuke V. V. A heuristic's job order gain for total weighted completion time minimization in the preemptive scheduling problem by subsequent length-equal job importance growth / V. V. Romanuke // Visnyk of the Lviv University. Series Appl. Math. and Informatics. – 2019. – Issue 27. – P. 108 – 117.

5. Romanuke V. V. Efficient exact minimization of total tardiness in tight-tardy progressive single machine scheduling with idling-free preemptions of equal-length jobs / V. V. Romanuke // KPI Science News. – 2020. – No. 1. – P. 27 – 39.

6. Romanuke V. V. Minimal total weighted tardiness in tight-tardy single machine preemptive idling-free scheduling / V. V. Romanuke // Applied Computer Systems. – 2019. – Vol. 24, No. 2. – P. 150 – 160.

7. Batsyn M. Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time / M. Batsyn, B. Goldengorin, P. Pardalos, P. Sukhov // Optimization Methods & Software. – 2014. – Vol. 29, No. 5. – P. 955 – 963.

DOI 10.33243/2518-7139-2020-1-1-19-36