UDC 519.161+519.852+519.687.1

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. Romanuke

Polish Naval Academy,
Śmidowicza Str., 69, Gdynia, 81-127, e-mail: romanukevadimv@gmail.com

The scheduling theory studying, in particular, minimization of the total weighted completion time, which refers to planning, organizing, and executing complex or multistep processes of assembling, manufacturing, building, dispatching, computing, etc., possesses both exact and heuristic approaches to the schedule computation. The computation time of the exact schedule approach grows immensely when the number of jobs is increased off just a few jobs (roughly, off 6 to 9, depending also on how jobs are divided into job parts). Therefore, a lot of heuristics are used to find the approximate schedule but to obtain it much faster. The heuristics' approximate schedule is not always executed in the exactly minimal total weighted completion time, but the loss is commonly not so great. Moreover, when the number of jobs is of order of hundreds, the scheduling problems become intractable by any exact schedule approaches, and so the heuristics remain the single way to find a schedule. Considering the preemptive scheduling problem by subsequent length-equal job importance growth, there are two ways to input the job release dates and the respective priority weights. On the one hand, the release dates can be given in ascending order; then the respective priority weights will be a set of, generally speaking, non-decreasing values. On the other hand, the release dates can be given in descending order; and then the respective priority weights will be a set of, generally speaking, non-increasing values. Having estimated the averaged time of obtaining the approximate schedule by both ascending and descending orders of inputting the job release dates, the heuristic's job order is revealed to be very significant. Its significance grows as the number of jobs increases. The influence of the heuristic's job order also grows as the number of job parts increases. The descending job order has the growing advantage for scheduling about 300 jobs and more. In particular, the descending job order's advantage in scheduling 100000 jobs divided in two parts each is almost 42 %. So, for total weighted completion time minimization in the preemptive scheduling problem by subsequent length-equal job importance growth, the job release dates are to be input in the descending order. However, the heuristic's job order gain in scheduling a lesser number of jobs (a few tens and up to 100, 200, 300) remains uncertain due to considerable fluctuations of the much shorter computation time.

Key words: scheduling theory, total weighted completion time minimization, single machine, preemption, heuristic, subsequent length-equal job importance growth, ascending/descending job order, computation time, relative advantage, heuristic's job order gain, computational speed.

1. Total weighted completion time minimization

Minimization of the total weighted completion time refers to planning, organizing, and executing complex or multistep processes of assembling, manufacturing, building, dispatching, computing, etc., that is studied by the scheduling theory [1, 2]. A number of jobs, each of which has its own importance valued as a weight, should be scheduled by respective release dates of the jobs so that the resulting schedule would be executed (or completed) as fast as possible. Thus, minimizing the total weighted completion time is the main criterion [2, 3], whether they consider a single machine to process jobs or multiple machines. Within a class of single-machine scheduling problems, there is a subclass of preemptive scheduling problems, wherein a job can be deliberately interrupted in favor of another job [3].

2. Scheduling by subsequent job importance growth

When a complex system is designed in multiple steps, where every subsequent step is executed by greater costs, a schedule for such a system can be obtained by setting subsequent jobs to greater priority weights. Those are preemptive scheduling problems by subsequent job importance growth. Scheduling by subsequent job importance growth is a common task in building (or assembling) hierarchical systems/objects whose build-ups above the basis are more complicated and expensive (see, e. g., [1, 2, 4, 5]). In particular, it can be loosely imagined as a cone-shaped body whose apex is at the bottom.

3. Approaches to scheduling

The scheduling problems can be solved exactly by using the Boolean linear programming model [6]. However, the computation time of the exact schedule approach grows immensely when the number of jobs is increased off just a few jobs (roughly, off 6 to 9, depending also on how jobs are divided into job parts) [1, 2]. Therefore, a lot of heuristics are used to find the approximate schedule but to obtain it much faster [1, 3, 7]. The heuristics' approximate schedule is not always executed in the exactly minimal total weighted completion time, but the loss is commonly not so great [3, 7]. Moreover, when the number of jobs is of order of hundreds, the scheduling problems become intractable by any exact schedule approaches, and so the heuristics remain the single way to find a schedule.

Considering the preemptive scheduling problem by subsequent job importance growth, there are two ways to input the job release dates and the respective priority weights. On the one hand, the release dates can be given in ascending order; then the respective priority weights will be a set of, generally speaking, non-decreasing values (some weights may be equal). On the other hand, the release dates can be given in descending order; and then the respective priority weights will be a set of, generally speaking, non-increasing values.

4. Goal of article and stages to achieve it

In finding an approximate schedule by an heuristic, the goal is to study whether the order of inputting the job release dates results in different time of computations. The significance of the difference, if any, should be shown. For achieving the said goal, the four stages are to be fulfilled:

1. Considering a single machine to process jobs, to formally state the preemptive scheduling problem by subsequent job importance growth. The schedule should have no idle time intervals [2, 3].
2. To state a known heuristic for finding an approximate schedule. The heuristic should be close to a commonly best approach in approximating preemptive job schedules.
3. To estimate the averaged time of obtaining the approximate schedule by both ascending and descending orders of inputting the job release dates. For doing this, a model of generating the respective scheduling problems will be designed.
4. In finding an approximate schedule by the heuristic, to discuss and conclude on whether significant the order of inputting the job release dates is.

If the order really matters, it would be an optimization in using the heuristic for the subclass of the preemptive scheduling problem by subsequent job importance growth. The expected magnitude of the computation time difference must be estimated along with that.

5. The preemptive scheduling by subsequent job importance growth

The parameters of the scheduling problem are declared as follows. Let N be a number of jobs, $N \in \mathbb{N} \backslash \{1\}$, where job n is divided into H_n equal parts (i. e., job n has a processing period or time H_n), has a release date r_n, and a priority weight w_n, $n = \overline{1, N}$. So, in general,

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

is a vector of processing periods,

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

is a vector of priority weights, and

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

is a vector of release dates.

To simplify the research, components of vector of processing periods (1) will be made identical. This condition does not violate much the generalization. Priority weights in vector (2) are either non-decreasing for the release dates' ascending order or non-increasing for the release dates' descending order. Formally,

$$w_{l-1} \leqslant w_l \;\; \forall\, l = \overline{2, N} \;\; \text{but} \;\; \exists\, l_* \in \left\{\overline{2, N}\right\} \;\; \text{such that} \;\; w_{l_*-1} < w_{l_*} \tag{4}$$

for the ascending order, and

$$w_{l-1} \geqslant w_l \;\; \forall\, l = \overline{2, N} \;\; \text{but} \;\; \exists\, l_* \in \left\{\overline{2, N}\right\} \;\; \text{such that} \;\; w_{l_*-1} > w_{l_*} \tag{5}$$

for the descending order.

Another simplification is the release dates' order. Let

$$\mathbf{R} = [r_n]_{1 \times N} = [n]_{1 \times N} \;\; (\text{i. e.,} \;\; r_n = n \;\; \text{by} \;\; n = \overline{1, N}) \tag{6}$$

for the ascending order, and

$$\mathbf{R} = [r_n]_{1 \times N} = [N - n + 1]_{1 \times N} \;\; (\text{i. e.,} \;\; r_n = N - n + 1 \;\; \text{by} \;\; n = \overline{1, N}) \tag{7}$$

for the descending order. Thus, the subsequent job importance grows. This is why the case

$$H_n = 1 \;\; \forall\, n = \overline{1, N} \tag{8}$$

is excluded from consideration, inasmuch as then the scheduling problem would be trivial (would have a trivial solution).

Considering a single machine to process jobs, the goal is to minimize the total weighted completion time, i. e. to schedule the jobs so that sum

$$\sum_{n=1}^{N} w_n \theta\,(n;\, H_n) \tag{9}$$

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

$$\theta(n;\, H_n) \in \left\{\overline{1,\, T}\right\} \tag{10}$$

by

$$T = \sum_{n=1}^{N} H_n. \tag{11}$$

The resulting schedule is a set of job tags/numbers $\mathbf{S} = [s_t]_{1\times T}$ along the grand total of job parts (11), where $s_t \in \left\{\overline{1,\, N}\right\}$ for every $t = \overline{1,\, T}$. The grand total (11) can be measured in time units as well.

6. The heuristic

A heuristic known to be close to a commonly best approach in approximating preemptive job schedules is an online scheduling algorithm, which applies the rule of weighted shortest remaining processing period [3]. Let

$$\mathbf{Q} = [q_n]_{1\times N} = \mathbf{H} = [H_n]_{1\times N} \tag{12}$$

be a starting vector containing the remaining processing periods. Later on, elements of vector (12) will be decreased as time t progresses. Denote by $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$ the whole set of jobs scheduled by the algorithm, where $\tilde{s}_t \in \left\{\overline{1, N}\right\}$ for every $t = \overline{1,\, T}$. It is a heuristic's approximate schedule. A set of available jobs

$$A(t) = \left\{i \in \left\{\overline{1, N}\right\} :\ r_i \leqslant t \ \text{ and }\ q_i > 0\right\} \subset \left\{\overline{1, N}\right\} \tag{13}$$

gives a set of ratios

$$\left\{\frac{w_i}{q_i}\right\}_{i\in A(t)}, \tag{14}$$

whence the maximal ratio is achieved at subset

$$A^*(t) = \arg\max_{i\in A(t)} \frac{w_i}{q_i}. \tag{15}$$

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

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

then

$$\tilde{s}_t = i^* \ \text{ by }\ q_{i^*}^{(\text{obs})} = q_{i^*} \ \text{ and }\ q_{i^*} = q_{i^*}^{(\text{obs})} - 1; \tag{16}$$

otherwise, if $|A^*(t)| > 1$, then a set

$$A^{**}(t) = \arg\max_{i^*\in A^*(t)} w_{i^*} \subset A^*(t) \subset A(t) \tag{17}$$

is found, where

$$A^{**}(t) = \{i_l^{**}\}_{l=1}^{L} \subset A^*(t) \subset A(t) \subset \left\{\overline{1, N}\right\}, \tag{18}$$

whence

$$\tilde{s}_t = i_1^{**} \ \text{ by }\ q_{i_1^{**}}^{(\text{obs})} = q_{i_1^{**}} \ \text{ and }\ q_{i_1^{**}} = q_{i_1^{**}}^{(\text{obs})} - 1. \tag{19}$$

Then an approximate total weighted completion time is calculated successively for every $n = \overline{1, N}$ using the moments at which each job is completed. If

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

in a schedule $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$, then job n is completed after moment $\tilde{\theta}\,(n;\, H_n) \in \left\{\overline{1,\, T}\right\}$. Finally,

$$\tilde{\rho}\,(N) = \sum_{n=1}^{N} w_n \tilde{\theta}\,(n;\, H_n) \tag{20}$$

is an approximately minimal total weighted completion time that corresponds to the nearly optimal job schedule $\tilde{\mathbf{S}} = [\tilde{s}_t]_{1\times T}$.

7. A model of generating the respective scheduling problems

Obviously, the minimal number of job parts is 2. The minimal number of jobs is 2 also. Besides, let

$$H_n = k \;\; \forall\, n = \overline{1, N} \;\text{ by }\; k = \overline{2, 18} \;\text{ and }\; N = \overline{2, 1000}. \tag{21}$$

Priority weights are generated as follows [8]:

$$w_n = \psi\,(N\zeta + 1) \;\; \forall\, n = \overline{1, N} \tag{22}$$

by either (4) or (5), where ζ is a pseudorandom number drawn from the standard uniform distribution on the open interval (0; 1), and function $\psi\,(\xi)$ returns the integer part of number ξ. Thus, the respective scheduling problem is going to be generated for each k and N according to (21): the ascending order's problems are generated by (6) and (22) by (4); the descending order's problems are generated by (7) and (22) by (5). Each problem will be repeated for 100 times to ensure good enough statistical confidence of the results.

8. Averaged time of computations

Let $\tau_{As}\,(k,\; N)$ be an averaged time of obtaining the heuristic's schedule by the ascending job order for definite k and N. The averaging is executed over those 100 repetitions. Denote an averaged time of obtaining the heuristic's schedule by the descending job order by $\tau_{Des}\,(k,\; N)$ likewise. Then

$$\beta\,(k,\; N) = 100 \cdot \frac{\tau_{As}\,(k,\; N) - \tau_{Des}\,(k,\; N)}{\tau_{Des}\,(k,\; N)} \tag{23}$$

is a percentage of a relative advantage of the descending order, if value (23) is positive, over the ascending order. Clearly, if value (23) is negative, this is a percentage of a relative advantage of the ascending order over the descending order.

An ensemble of percentages (23) in 17 preemptive scheduling problems by subsequent length-equal job importance growth generated according to (21) and (22) is presented in fig. 1. Some obvious computational speed artifacts are indicated with ellipses. In addition, as it is seen, here are a lot of artifacts for up to 100 jobs. Therefore, the polylines in fig. 1 are shown in detail for 100 to 1000 jobs in fig. 2 within a range of 14 %.

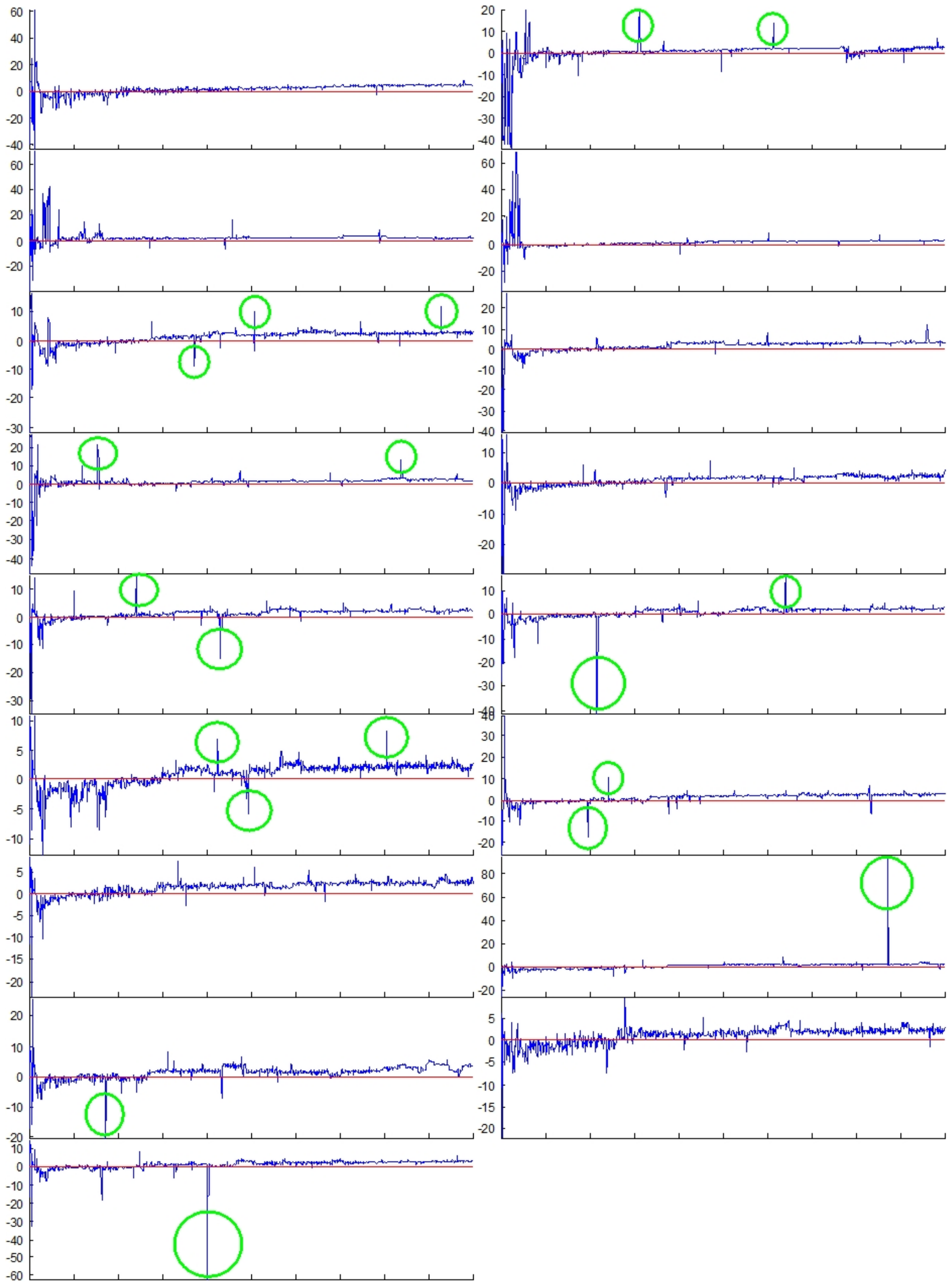

Fig. 1. Percentages (23) by $k = \overline{2, 18}$ (left to right downward) versus $N = \overline{2, 1000}$ with the horizontal zero level

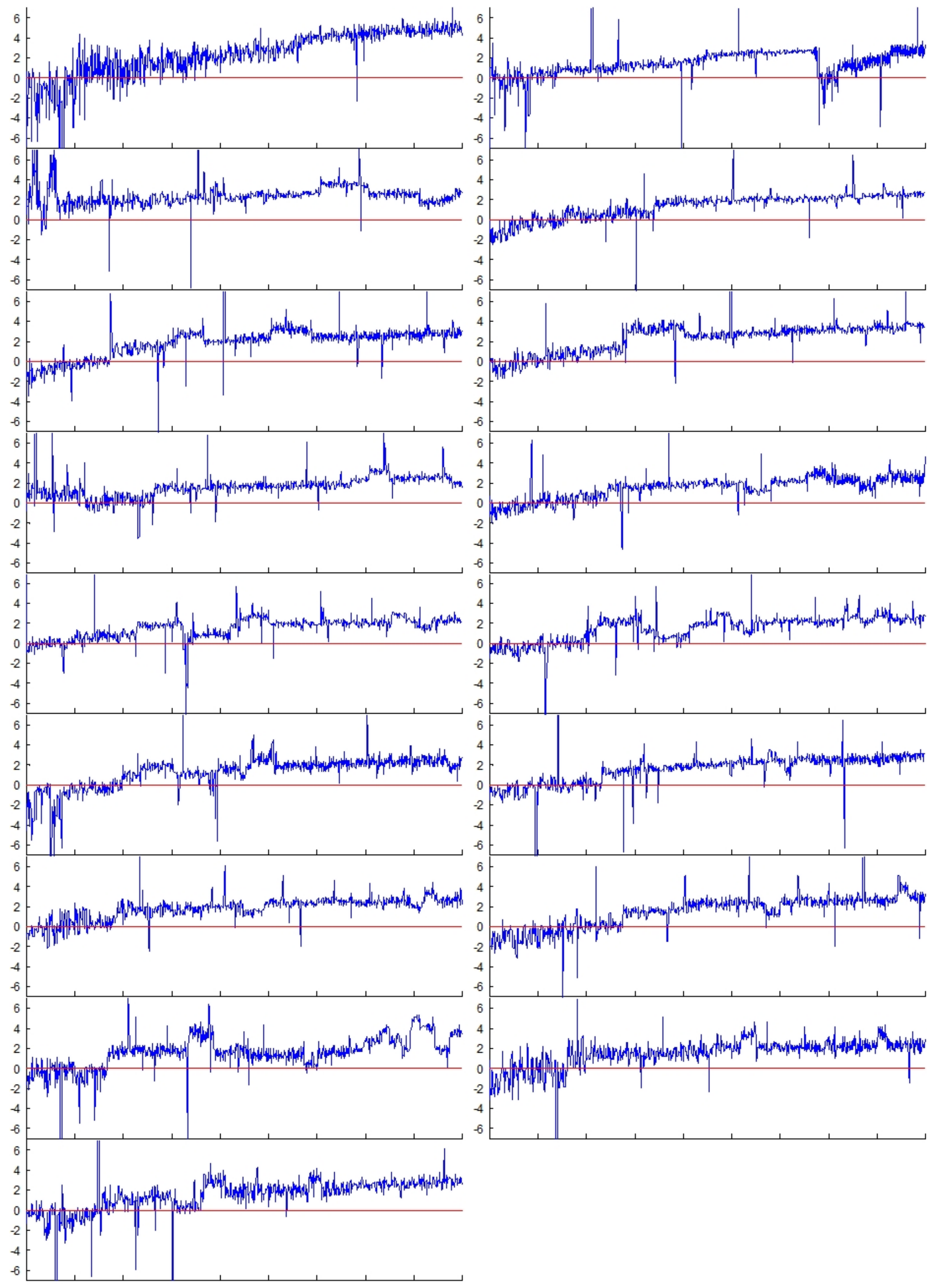

Fig. 2. Percentages (23) taken off fig. 1 versus $N = \overline{100, 1000}$ by ignoring the artifacts

It is also clearly seen that the heuristic's schedule by the descending job order is obtained faster starting off scheduling 300 jobs. The relative advantage is about 2 %. This trend is not expected to decrease by scheduling more than 1000 jobs. Nevertheless, the ascending job order has an expected advantage by scheduling between 100 and 300 jobs, although this advantage is weaker than that of the descending job order.

The real time (in seconds) of obtaining the heuristic's schedule by the ascending/descending job order is shown in fig. 3 for $N \in \{5 \cdot 10^3,\ 10^4,\ 10^5\}$ by $k = 2$ (left column) and for $N \in \{5 \cdot 10^3,\ 10^4,\ 2 \cdot 10^4\}$ by $k = 5$ (right column). These graphs totally confirm the mentioned trend. The advantage of the descending order increases as the number of jobs increases. This holds as well for the greater number of job parts, although then the advantage is apparently less. In particular, the descending job order's advantage in scheduling 100000 jobs divided in two parts each is almost 42 %.

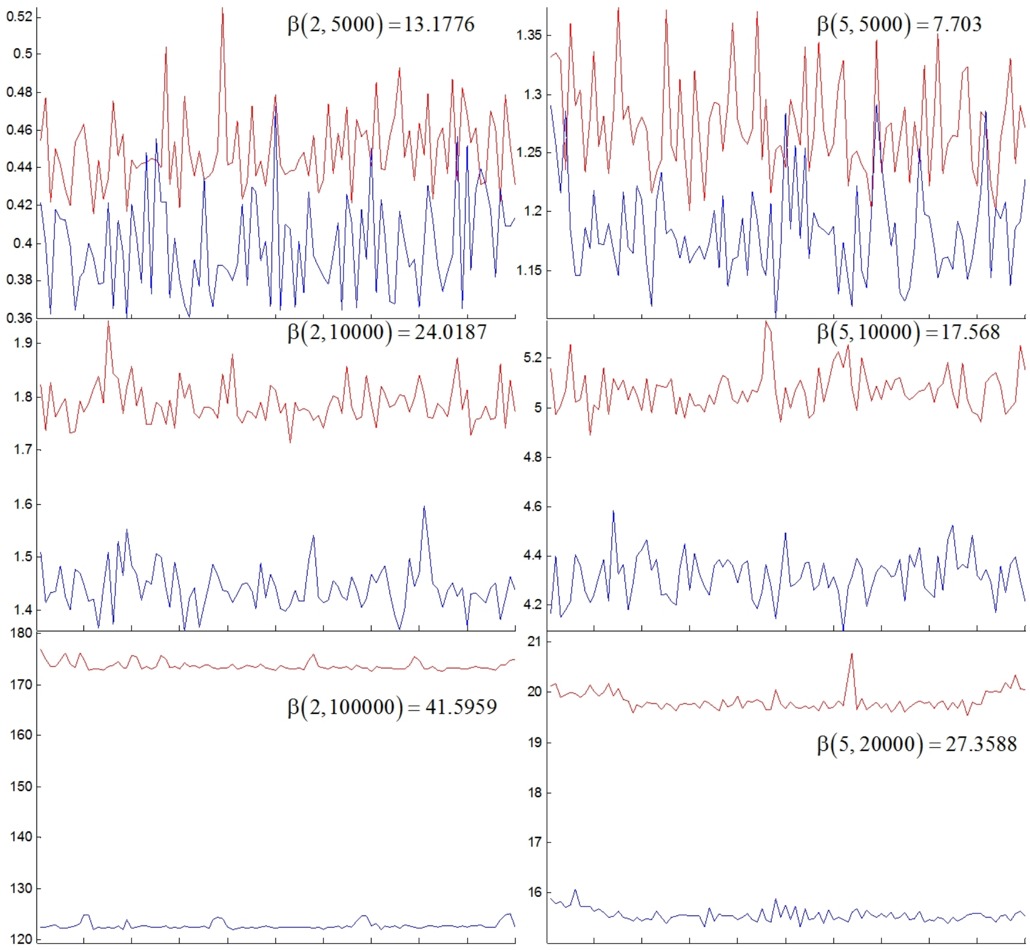

Fig. 3. The real time (in seconds) of obtaining the heuristic's schedule along a series of 100 repetitions

A single schedule in this case is obtained by no shorter than in two minutes, whereas the ascending order approach takes up to three minutes. In scheduling 20000 jobs divided into five parts each, the descending job order's advantage is a little greater than that for 10000 jobs divided in two parts, but the expected computation time difference is about 4 seconds (8 times greater).

References

1. *Brucker P.* Scheduling Algorithms / P. Brucker. – Berlin, Heidelberg: Springer-Verlag, 2007. – 371 p.
2. *Pinedo M. L.* Scheduling: Theory, Algorithms, and Systems / M. L. Pinedo. – Springer Int. Publ., 2016. – 670 p.
3. *Belouadah H.* Scheduling with release dates on a single machine to minimize total weighted completion time / H. Belouadah, M. E. Posner, C. N. Potts // Discrete Applied Mathematics. – 1992. – Vol. 36, Iss. 3. – P. 213–231.
4. *Mandzyuk I. A.* Rheometric research of polypropylene Licocene PP2602 melts / I. A. Mandzyuk, V. V. Romanuke // Archives of Materials Science and Engineering. – 2011. – Vol. 50, Iss. 1. – P. 31–35.
5. *Romanuke V. V.* Appropriate number and allocation of ReLUs in convolutional neural networks / V. V. Romanuke // Research Bulletin of NTUU "Kyiv Polytechnic Institute". – 2017. – No. 1. – P. 69–78.
6. *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.
7. *Tian W.* Optimized Cloud Resource Management and Scheduling / W. Tian, Y. Zhao. – Morgan Kaufmann, 2015. – P. 135–157.
8. *Romanuke V. V.* Acyclic-and-asymmetric payoff triplet refinement of pure strategy efficient Nash equilibria in trimatrix games by maximinimin and superoptimality / V. V. Romanuke // KPI Science News. – 2018. – No. 4. – P. 38–53.

Article: received 27.08.2019
revised 21.10.2019
printing adoption 23.10.2019

ВИГРАШ ПОРЯДКУ ЗАВДАНЬ ЗА ОДНІЄЮ ЕВРИСТИКОЮ ДЛЯ МІНІМІЗАЦІЇ ЗАГАЛЬНОГО ЗВАЖЕНОГО ЧАСУ ЗАВЕРШЕННЯ У ЗАДАЧІ ПЛАНУВАННЯ З ПЕРЕМИКАННЯМИ ЗІ ЗРОСТАННЯМ ЗНАЧУЩОСТІ НАСТУПНИХ ЗАВДАНЬ ОДНАКОВОГО ОБ'ЄМУ

В. Романюк

Військово-морська Академія Польщі,
вул. Шмідовича, 69, м. Гдиня, 81-127, e-mail: romanukevadimv@gmail.com

Теорія розкладів, яка вивчає, зокрема, мінімізацію загального зваженого часу завершення, що належить до планування, організації та виконання комплексних або багатоетапних процесів компонування, виробництва, будівництва, диспетчеризації, обчислень тощо, володіє і точними, і евристичними підходами до обчислення розкладів. Обчислювальний час підходу з точним розкладом непомірно зростає, коли кількість завдань збільшують від лише декількох одиниць (приблизно від 6 до 9, залежно також від того, як завдання розбиті на частини). Тому використовують низку евристик для того, щоб знайти наближений розклад й отримати його якомога швидше. Наближений розклад за евристиками не завжди виконується за точно мінімальний загальний зважений час завершення, але втрата зазвичай є невеликою. Коли кількість завдань становить порядок сотень, задачі планування стають нерозв'язними за будь-якими підходами до точних розкладів, і тому евристики залишаються єдиним способом визначення розкладу. Розглядаючи задачу планування з перемиканнями зі зростанням значущості наступних завдань однакового об'єму, існує два шляхи подання моментів відпуску завдань і відповідних ваг пріоритетів. З одного боку, моменти відпуску можуть бути подані у порядку зростання; тоді відповідні ваги пріоритетів будуть множиною, взагалі кажучи, неспадних значень. З іншого – моменти відпуску можуть бути подані у порядку спадання; тоді вже відповідні ваги пріоритетів будуть множиною, загалом незростаючих значень. Оцінивши середній час отримання наближеного розкладу за зростаючим і спадаючим порядками подання моментів відпуску завдань, виявляється, що порядок завдань у визначеній евристиці вельми значущий. Його значущість зростає за зростаючої кількості завдань. Вплив порядку завдань у визначеній евристиці також зростає зі зростанням кількості частин завдання. Спадаючий порядок завдань має зростаючу перевагу при плануванні близько 300 завдань і більше. Зокрема, перевага спадаючого порядку завдань при плануванні 100000 завдань, кожне з яких розділене на дві частини, становить майже 42 %. Отже, для мінімізації загального зваженого часу завершення у задачі планування з перемиканнями зі зростанням значущості наступних завдань однакового об'єму моменти відпуску завдань треба подавати у спадаючому порядку. Однак виграш порядку завдань у визначеній евристиці при плануванні меншої кількості завдань (від декількох десятків до 100, 200, 300) залишається невизначеним внаслідок суттєвих флуктуацій значно меншого часу обчислень.

Ключові слова: теорія розкладів, загальний зважений час завершення, один комп'ютер, перемикання, евристика, зростання значущості наступних завдань однакового об'єму, зростаючий/спадаючий порядок завдань, час обчислень, відносна перевага, виграш порядку завдань у визначеній евристиці, обчислювальна швидкість.