STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **9**, December 2021, pp 820–837.
Published online in International Academic Press (www.IAPress.org)

Infinity Substitute in Finding Exact Minimum of Total Weighted Tardiness in Tight-Tardy Progressive 1-machine Scheduling by Idling-free Preemptions

Vadim Romanuke*

Faculty of Mechanical and Electrical Engineering, Polish Naval Academy, Gdynia, Poland

Abstract A job schedule ensuring the exact minimum of total weighted tardiness can be found with the respective integer linear programming problem model, in which the infinity showing that the respective states are either forbidden or impossible is substituted with a sufficiently great positive integer. An open question is whether the substitute can be selected so that the computation time would be decreased. Thus, it is ascertained that, whichever job lengths and its priority weights are, substituting the infinity with just "a sufficiently great positive integer" is never recommended. To decrease the computation time on average, it is strongly recommended to select the infinity substitute as multiple maximum over finite decision variable weights in the exact model. It is sufficient to take 2 to 5 such maxima as the infinity substitute. However, the shortened computation time is not guaranteed for solving a single or few scheduling problems. It is only an expected benefit, which builds up as a few hundred scheduling problems are solved at least.

Keywords Job scheduling, Preemptive 1-machine scheduling, Total weighted tardiness, Exact model, Infinity substitute, Computation time

AMS 2010 subject classifications 90C10, 90B35, 68M20

DOI:10.19139/soic-2310-5070-1256

1. The infinity in exact minimization of total weighted tardiness

Total weighted tardiness is a measure which indicates a cumulative lag in a job schedule executed on a single or multiple machines [13, 7]. When it is scheduled on a single machine (1-machine), the model of exact minimization of total weighted tardiness is rendered to solving an integer linear programming problem (ILPP) involving the branch-and-bound approach [1, 11]. One of the most practically valuable scheduling problems is the tight-tardy progressive 1-machine scheduling by idling-free preemptions [11, 12]. In this problem, 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 a few jobs still can be completed without tardiness. Besides, each job may have its own priority (importance) given by its weight. Periods of idle time are impossible (not allowed) because the machine must keep functioning by natural economic reasons. Theoretically, the ILPP model to find the exact minimum of total weighted tardiness contains infinities which are intended to show that the respective states are either forbidden or impossible. In real computations, the infinity is substituted with a sufficiently great positive integer [11, 1, 9]. A particularly open question is whether the substitute influences the computation time of exact schedules (ones ensuring the exact minimum of total weighted tardiness), i. e. whether it is possible to select the infinity substitute so that the computation time would be decreased [5].

The case in which all the jobs have equal length by no weight priorities was studied by Romanuke [10]. Having considered nine versions of the infinity substitute, it was ascertained that the increment of the infinity substitute

*Correspondence to: Vadim Romanuke (Email: romanukevadimv@gmail.com). Faculty of Mechanical and Electrical Engineering, Polish Naval Academy, 69 Śmidowicza Street, Gdynia, Poland, 81-127.

ISSN 2310-5070 (online) ISSN 2311-004X (print)

in the ILPP model may have bad influence on the computation time of exact (optimal) schedules. At least, the greater value of the infinity substitute could not produce an optimal schedule faster. Therefore, in order to decrease the computation time, the paper [10] strongly recommended to select the infinity substitute as less as possible for equal-length job scheduling without weights.

2. The goal and tasks

The research is set up on finding the exact minimum of total weighted tardiness in the tight-tardy progressive 1-machine scheduling by idling-free preemptions using the ILPP model. The goal is to ascertain how increment of the infinity substitute in this model influences the computation time of exact schedules. Factually, this research will be an extension of the paper [10]. For achieving the goal, the following stages are to be stated and fulfilled: the ILPP model, a pattern of generating instances of the job scheduling problem, and a computational study. Whether the influence appears to be significant or not, recommendations on selecting the infinity substitute to decrease the computation time will be discussed. The conclusions will be linked to those ones in the paper [10].

3. Finding exact minimum of total weighted tardiness

If N jobs are to be scheduled, where $N \in \mathbb{N}\setminus\{1\}$ and job n consists of H_n unit-duration processing periods ($n = \overline{1,\ N}$), then the vector of job lengths $[H_n]_{1\times N}$ is linked to the respective vectors of priority weights $[w_n]_{1\times N}$, release dates $[r_n]_{1\times N}$ (r_n is a certain time moment at which processing of job n can be started), and due date $[d_n]_{1\times N}$. Conventionally, all these vectors are of natural numbers [13, 7, 1, 3, 6].

As idle time periods are not allowed, the release dates can be set at monotonously increasing integers, naturally starting from 1:

$$r_n = n \ \text{ for } \ n = \overline{1,\ N}. \tag{1}$$

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

$$T = \sum_{n=1}^{N} H_n. \tag{2}$$

Theoretically, the due dates can be set at any integers, but in practice they are linked to the job release dates whichever they are. Job lengths determine the due dates also. In the case of the tight-tardy progressive scheduling, for example,

$$d_n = H_n + r_n - 1 + b_n \ \text{ for } \ n = \overline{1,\ N} \tag{3}$$

by a random due date shift [12]

$$b_n = \psi\left(H\cdot\zeta\right) \ \text{ for } \ n = \overline{1,\ N} \tag{4}$$

with a pseudorandom number ζ drawn from the standard normal distribution (with zero mean and unit variance), and function $\psi\left(\xi\right)$ returning the integer part of number ξ [9, 12]. For the case with monotonously increasing release dates (1), due dates (3) are re-written as follows:

$$d_n = H_n + n - 1 + b_n \ \text{ for } \ n = \overline{1,\ N}. \tag{5}$$

The tight-tardy 1-machine preemptive idling-free scheduling by (1) and (5) with (4) is a class of hard scheduling problems, in which the inaccuracy of finding the total weighted tardiness minimum has the strongest negative

impact [11, 6]. 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 unfavorable states) [11, 2, 8].

In the simplest term, the 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 [11]. If job n is completed after time moment $\theta\left(n;\ H_n\right)$, which is [4]

$$\theta\left(n;\ H_n\right) \in \left\{\overline{1,\ T}\right\} \tag{6}$$

by length (2) of the schedule, the total weighted tardiness is

$$\sum_{n=1}^{N} w_n \cdot \max\left\{0,\ \theta\left(n;\ H_n\right) - d_n\right\}. \tag{7}$$

The schedule should be composed so that sum (7) would be minimal.

The exact minimum of total weighted tardiness is found by the ILPP model in the following way [11, 1, 9, 10]. 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 nonnegative value λ_{nh_nt} (not to be confused with a job priority weight):

$$\lambda_{nh_nt} = 0 \tag{8}$$

by

$$r_n - 1 + h_n \leqslant t \leqslant T - H_n + h_n \ \ \forall\, h_n = \overline{1,\ H_n - 1} \tag{9}$$

and

$$\lambda_{nh_nt} = \alpha \tag{10}$$

by a sufficiently great positive integer α (similar to the meaning of infinity) when (9) is not true;

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

by

$$r_n - 1 + H_n \leqslant t \leqslant d_n \tag{12}$$

and

$$\lambda_{nH_nt} = w_n\left(t - d_n\right) \tag{13}$$

by

$$d_n < t \leqslant T \tag{14}$$

and

$$\lambda_{nH_nt} = \alpha \tag{15}$$

when both (12) and (14) are not true. The goal is to find such a set

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

of the decision variables, on which the sum

$$\vartheta^*(N)=\sum_{n=1}^{N}\sum_{h_n=1}^{H_n}\sum_{t=1}^{T}\lambda_{nh_nt}x^*_{nh_nt} \tag{17}$$

is minimal by constraints constituting a set X of all possible versions of the decision variables [1, 11]:

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

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

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

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

Sum (17) is the exact minimum of total weighted tardiness for those N jobs scheduled according to solution (16). The respective optimal job schedule is

$$\mathbf{S}^*=[s^*_t]_{1\times T}\ \text{ by }\ s^*_t\in\left\{\overline{1,\ N}\right\}\ \text{ for every }\ t=\overline{1,\ T}. \tag{22}$$

Obviously, the total weighted tardiness by sum (17) can be calculated as well by formula (7) using N job completion moments of T components in schedule (22):

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

by

$$\theta^*(n;\ h_n)\in\left\{\overline{1,\ T}\right\}$$

and

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

Integer α theoretically meant as the infinity in formulae (10) and (15) can be given using nonzero and finite values λ_{nh_nt} (by $n=\overline{1,\ N}$ and $h_n=\overline{1,\ H_n}$ and $t=\overline{1,\ T}$). In fact, these are decision variable weights (13) valid by inequality (14). Thus, the following infinite substitutions can be made:

$$\alpha=\sum_{n=1}^{N}\sum_{t=d_n+1}^{T}\lambda_{nH_nt}, \tag{23}$$

$$\alpha=1+\max_{n=\overline{1,\ N}}\max_{t=\overline{d_n+1,\ T}}\lambda_{nH_nt}, \tag{24}$$

$$\alpha=k\cdot\max_{n=\overline{1,\ N}}\max_{t=\overline{d_n+1,\ T}}\lambda_{nH_nt}\ \text{ by }\ k\in\mathbb{N}\setminus\{1\}. \tag{25}$$

Additionally, integer α can be set at just an arbitrarily great natural number. The nine versions of the infinity substitute used by Romanuke [10] are presented in Table 1.

Table 1. Nine versions of the infinity substitute

The tag to α	1	2	3	4	5	6	7	8	9
The real value of α (formula)	(24)	(25) by $k = 2$	(25) by $k = 3$	(25) by $k = 4$	(25) by $k = 5$	(23)	$\alpha = 10^4$	$\alpha = 10^5$	$\alpha = 10^6$
Mnemonics	max	2max	3max	4max	5max	sum	–	–	–

4. A pattern of generating instances of the job scheduling problem

Instances of the job scheduling problem should be generated such that the schedule ensuring the exact minimum of total weighted tardiness could not be found trivially [10]. For this, three independent generators for job lengths, priority weights, and due date shifts (4) are constructed. The vector of job lengths is [11]

$$[H_n]_{1\times N} = \psi\left(4 \cdot \Theta_{\mathbf{H}}\left(1,\ N\right) + 2\right) \tag{26}$$

where operator $\Theta_{\mathbf{H}}\left(1,\ N\right)$ returns a pseudorandom $1 \times N$ vector whose entries are drawn from the standard uniform distribution on the open interval $(0;\ 1)$. Statement (26) implies that job lengths are randomly generated within an integer interval from 2 to 5. The vector of priority weights is [11]

$$[w_n]_{1\times N} = \psi\left(100 \cdot \Theta_{\mathbf{W}}\left(1,\ N\right) + 1\right) \tag{27}$$

where operator $\Theta_{\mathbf{W}}\left(1,\ N\right)$ runs and returns outputs identically to operator $\Theta_{\mathbf{H}}\left(1,\ N\right)$ but they are independent of each other. Statement (27) implies that priority weights are randomly generated within an integer interval from 1 to 100. Due date shifts are generated by (4), but if

$$H_n \leqslant H_{n+1} \ \text{ and } \ d_n \leqslant d_{n+1} \ \text{ and } \ w_n \geqslant w_{n+1} \ \ \forall\, n = \overline{1,\ N-1} \tag{28}$$

it is re-generated. Besides, due date shift (4) for job n is re-generated until $d_n \geqslant 1$.

Using this pattern, a series of 100 instances of the job scheduling problem for every version of the infinity substitute (Table 1) is generated for each $N = \overline{2,\ 10}$. The computation time of optimal schedule (22) by ILPP model (6)–(21) is registered for every generated instance. If the computation time drags on beyond half an hour (i. e., more than 1800 seconds), this is technically called a timeout [1, 11, 10] and the computation is stopped, whichever the current result is (whether the exact minimum is achieved or not yet).

5. Computational study

At a fixed number of jobs N, for a job scheduling problem instance tagged by an integer c, denote the schedule computation time by $\delta\left(N,\ c,\ \alpha\right)$ by integer α as an infinity substitute tagged according to Table 1. Value $\delta\left(N,\ c,\ \alpha\right)$ implies computation time spent on just searching a solution of the respective ILPP, i. e. on exploring nodes by the branch-and-bound algorithm. If the total number of the instances is denoted by C, then the maximum of the computation time is

$$\delta_{\max}\left(N,\ \alpha\right) = \max_{c=\overline{1,\ C}} \delta\left(N,\ c,\ \alpha\right). \tag{29}$$

Along with maximum (29), the averaged computation time

$$\delta\left(N,\ \alpha\right) = \frac{1}{C} \cdot \sum_{c=1}^{C} \delta\left(N,\ c,\ \alpha\right) \tag{30}$$

and the minimum of the computation time

$$\delta_{\min}(N,\ \alpha) = \min_{c=\overline{1,\ C}} \delta(N,\ c,\ \alpha) \tag{31}$$

are to be considered as well.

Set at $C = 100$, maximum (29) versus the nine tags to α (Table 1) is shown in Figure 1 for each $N = \overline{2,\ 10}$. It is worth noting that scheduling 8 to 10 jobs (the bottom subplot row), as it was expected [11, 12], has been followed by half-hourly timeouts. Thus, the values of polylines $\delta_{\max}(8,\ \alpha)$, $\delta_{\max}(9,\ \alpha)$, and $\delta_{\max}(10,\ \alpha)$ range from 1800.057 to 1800.419 seconds.

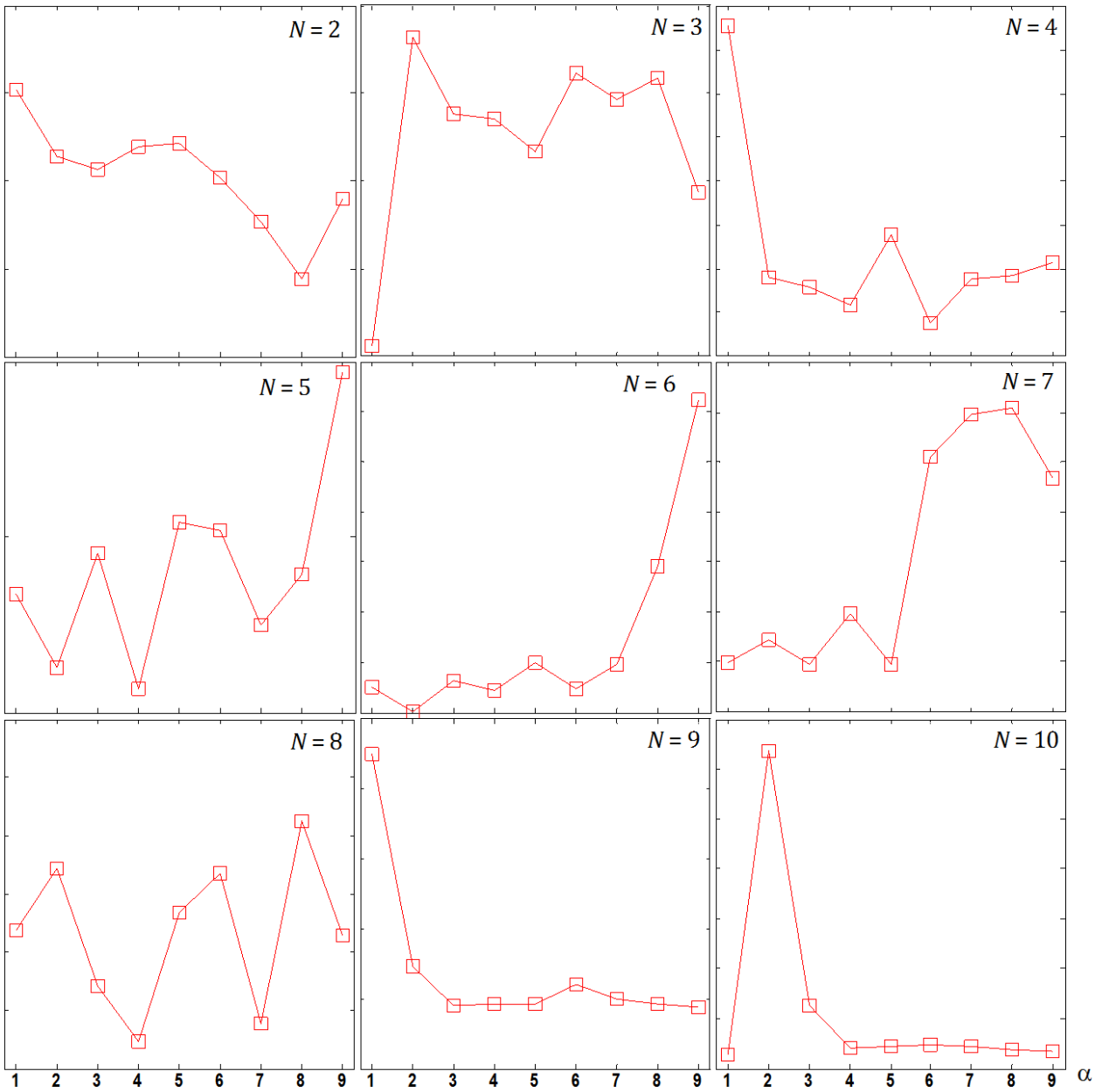

Figure 1. Maximum (29) of the computation time plotted versus the nine tags to α by increasing the number of jobs

Averaged computation time (30) is shown in Figure 2, where similarity of the polylines in each subplot row is easily seen. Moreover, the polylines standing for scheduling 5 to 10 jobs (the middle and bottom row polylines) are generally similar as well. If to consider scheduling just 2 to 4 jobs, the "max" infinity substitute appears to be the worst. However, if 5 to 10 jobs are scheduled, the worst choice turns out to be opposite. In fact, this is "a sufficiently great positive integer" ($\alpha = 10^6$), which was not recommended by Romanuke [10], unless this integer was close to an infinity substitute by "max", "2max", ..., "5max", "sum".

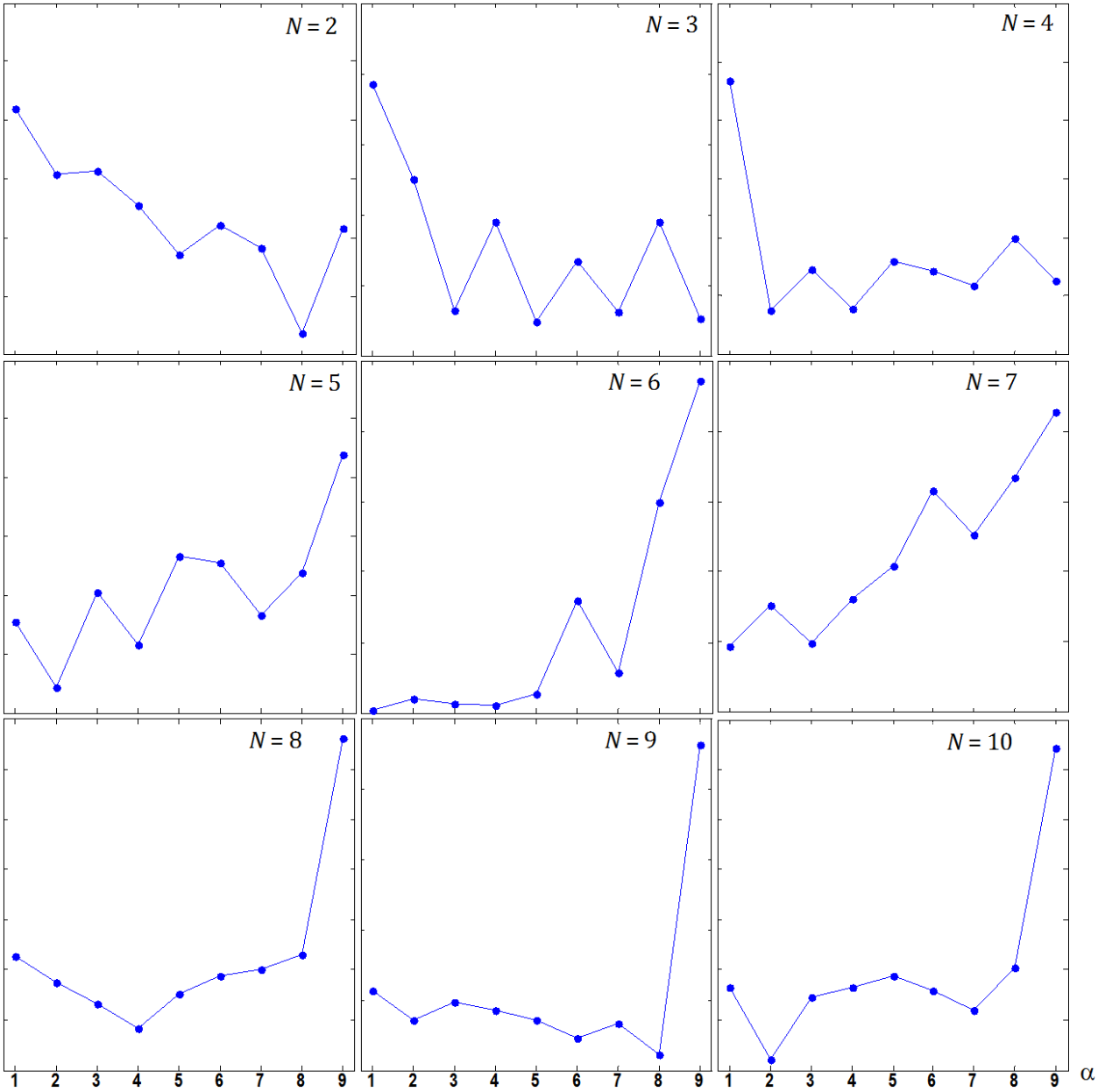

Figure 2. Averaged computation time (30) plotted versus the nine tags to α by increasing the number of jobs

Minimum (31) versus the nine tags to α is shown in Figure 3, but the useful information is poorer here. This is so because the minimum polylines appear to be almost randomly fluctuating, apart from polyline $\delta_{\min}(8,\ \alpha)$

ranging from 2.144 to 2.671 seconds. As a consequence of it, any inferences made from these minimum polylines are hardly to be considered reliable. The exception by $\delta_{\min}(8,\ \alpha)$ is not substantially valuable.

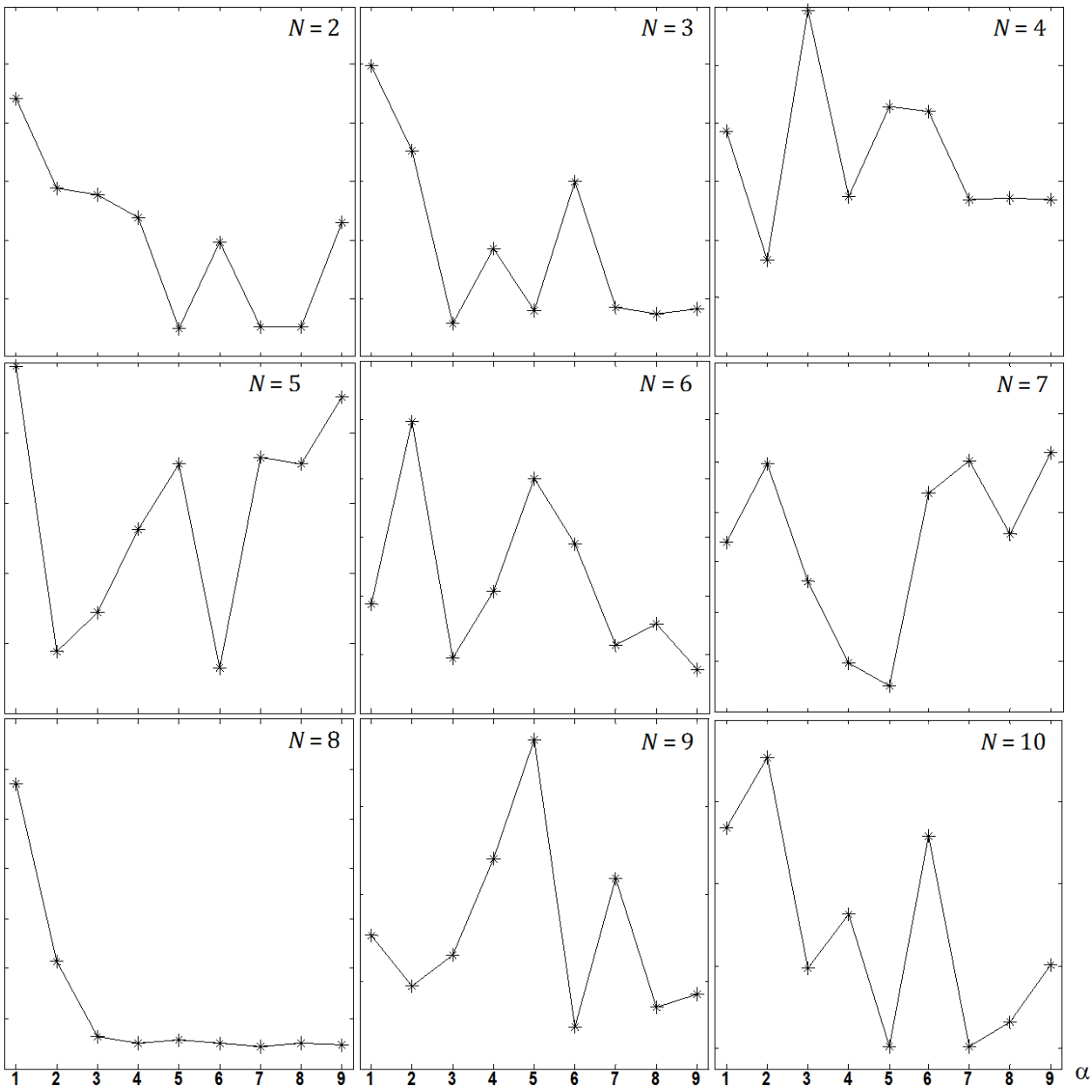

Figure 3. Minimum (31) of the computation time plotted versus the nine tags to α by increasing the number of jobs

Obviously, it is disputable that the plotted polylines can help in unambiguously ascertaining how increment of the infinity substitute influences the computation time. However, the plotted data might be interpreted and visualized in a more convenient way by just tracing the global minima and maxima of polylines (29) – (31). It is clear that the minima are the best versions of the infinity substitute, and the maxima are the worst ones. They are determined at every pair of the number of jobs and α. The minima and maxima can be marked, e. g., by a big plus and blockage sign, respectively. In addition to this, the pairs of N and α closest to the minima and maxima can be marked as

well by the similar signs using squiggles. Thus, the best and worst pairs with respect to maximum (29) are marked in Table 2 using the described presentation style. Unfortunately, a certain pattern cannot be seen in this table. The global minima and maxima (along with the "smaller" minima and maxima which are the closest to the global ones) are scattered unsystematically. Regardless of this fact, it should be noted that the column corresponding to the "4max" infinity substitute has two global and two "smaller" minima and does not have any maxima. Besides, "3max", "sum", and $\alpha = 10^4$ have only one "smaller" maximum. The other five versions of the infinity substitute have at least one global maximum (the worst version) for some N.

Table 2. The best and worst versions of the infinity substitute with respect to maximum (29) of the computation time (Figure 1)

	The tag to α								
N	1	2	3	4	5	6	7	8	9
2	⊖				≈		≂	✚	
3	✚	⊖				≈			≂
4	⊖			≂	≈	✚			
5		≂		✚	≈				⊖
6		✚		≂				≈	⊖
7			✚		≂		≈	⊖	
8		≈		✚			≂	⊖	
9	⊖	≈	≂						✚
10	✚	⊖	≈						≂

With respect to averaged computation time (30), the best and worst versions of the infinity substitute are not scattered that bad (Table 3). A certain pattern is still not clearly seen but the advantage of the abovementioned "3max", "4max", "sum", and $\alpha = 10^4$ is confirmed. Moreover, the "4max" infinity substitute has one global minima and three "smaller" ones. The worst version of the substitute is either "max" (for scheduling 2 to 4 jobs) or "the sufficiently great positive integer" $\alpha = 10^6$ (for scheduling more than 4 jobs).

The best and worst versions of the infinity substitute with respect to minimum (31) of the computation time (Table 4) are badly scattered again, although "the sufficiently great positive integers" appear better in this table than "max" and "2max". The "sum" infinity substitute has two global minima and no maxima.

To extract additional information from the polylines, maxima (29) are normalized and averaged as

$$\tilde{\delta}_{\max}(\alpha) = \frac{1}{9} \cdot \sum_{N=2}^{10} \frac{\delta_{\max}(N,\ \alpha)}{\max\limits_{\beta=\overline{1,\ 9}} \delta_{\max}(N,\ \beta)}. \tag{32}$$

Averaged computation times (30) and minima (31) are stranded into single polylines in the same way:

$$\tilde{\delta}(\alpha) = \frac{1}{9} \cdot \sum_{N=2}^{10} \frac{\delta(N,\ \alpha)}{\max\limits_{\beta=\overline{1,\ 9}} \delta(N,\ \beta)} \tag{33}$$

Table 3. The best and worst versions of the infinity substitute with respect to averaged computation time (30) (Figure 2)

	The tag to α								
N	1	2	3	4	5	6	7	8	9
2	⊖		≈		∓			✚	
3	⊖	≈			✚				∓
4	⊖	✚		∓				≈	
5		✚		∓	≈				⊖
6	✚			∓				≈	⊖
7	✚		∓					≈	⊖
8			∓	✚				≈	⊖
9						∓		✚	⊖
10		✚					∓	≈	⊖

Table 4. The best and worst versions of the infinity substitute with respect to minimum (31) of the computation time (Figure 3)

	The tag to α								
N	1	2	3	4	5	6	7	8	9
2	⊖	≈			✚			∓	
3	⊖	≈	✚					∓	
4		✚	⊖		≈		∓		
5	⊖	∓				✚			≈
6		⊖	∓		≈				✚
7				∓	✚		≈		⊖
8	⊖	≈					✚		∓
9				≈	⊖	✚		∓	
10	≈	⊖			∓		✚		

and

$$\tilde{\delta}_{\min}(\alpha) = \frac{1}{9} \cdot \sum_{N=2}^{10} \frac{\delta_{\min}(N,\ \alpha)}{\max\limits_{\beta=\overline{1,\ 9}} \delta_{\min}(N,\ \beta)}. \tag{34}$$

Averages (32) – (34) presented in Figure 4 still have some contradictions, but now it is quite obvious that selecting

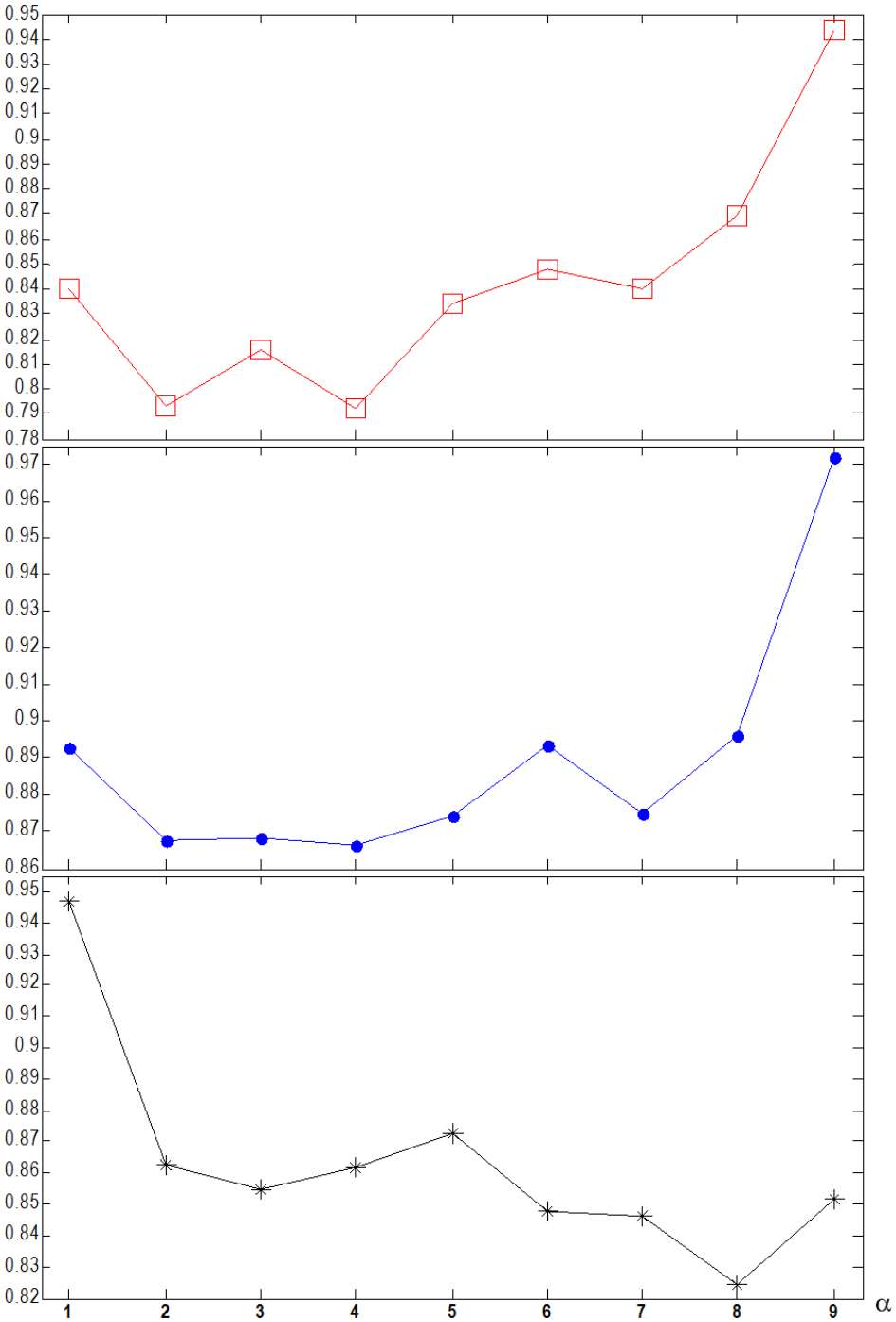

Figure 4. Averages by (32) – (34) of computation time polylines in Figures 1 – 3

the infinity substitute as “2max”, “3max”, “4max” is more beneficial. The disclosure of all generated instances in Figure 5 partially confirms that setting $\alpha \geqslant 10^4$ is not beneficial (except for $N = 2$, occurrences of green squares

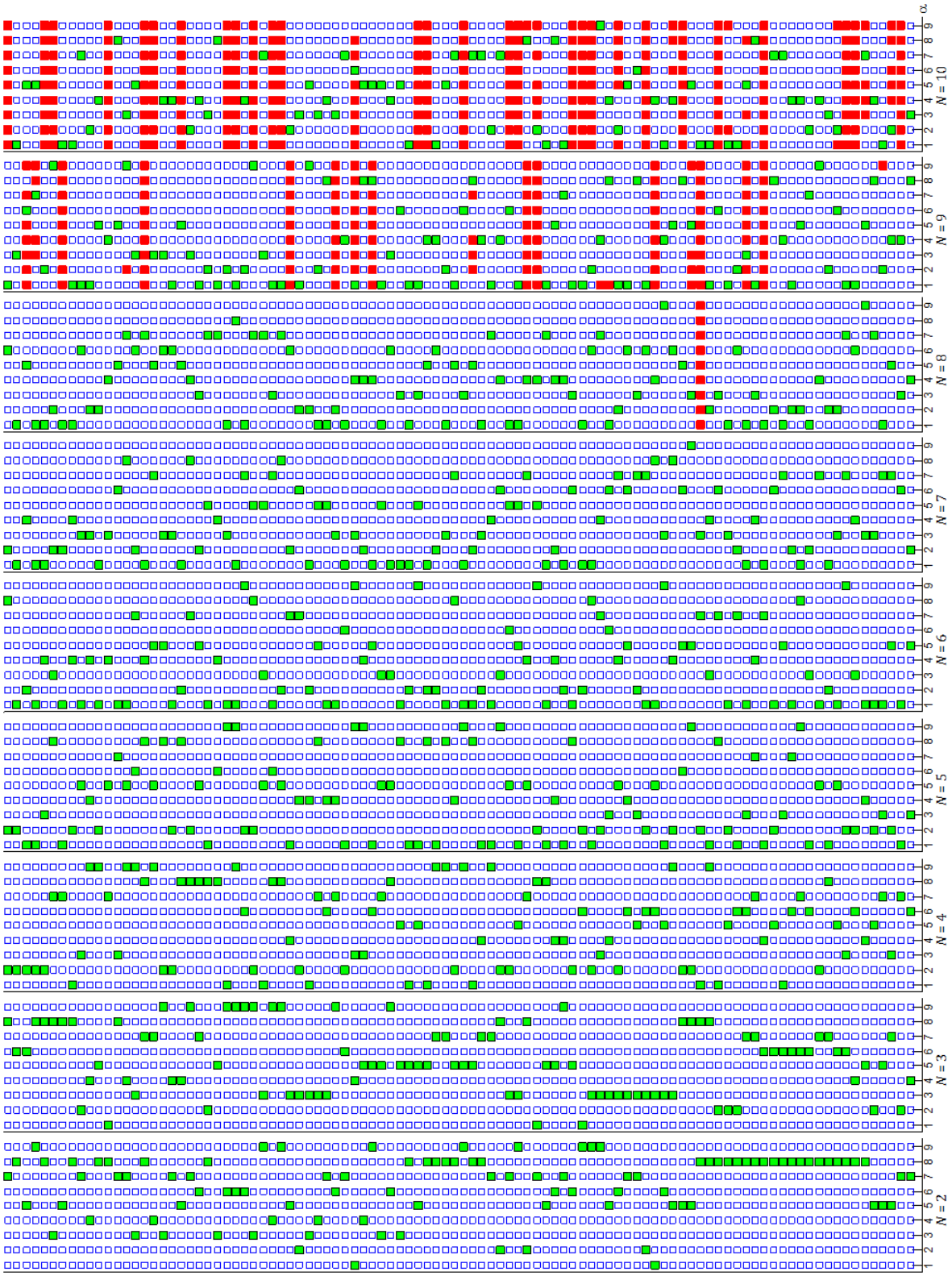

Figure 5. The disclosure of all generated instances, where minima of the computation time are highlighted with green, and the instances followed by half-hourly timeouts (occurred only at scheduling 8 jobs and more) are highlighted with red

are indeed rarer at $\alpha \geqslant 10^4$), although the column of green squares at $N = 2$ and $\alpha = 10^5$ seems to disprove it. Nevertheless, the columns of $N = 5$ to $N = 8$ in Figure 5 correspond to the inference from the paper by Romanuke [10] claiming that the greater value of the infinity substitute cannot produce an optimal schedule faster (i. e., "max", "5max", and "sum" are locally beneficial here, although "max" is even more preferable). The same can be roughly inferred from the columns of $N = 9$ and $N = 10$, although the timeouts here are becoming severer.

An important question is whether the obtained results will be roughly the same if to repeat the generation of job scheduling problem instances. Therefore, a second series of 50 instances of the job scheduling problem for every version of the infinity substitute (Table 1) is generated for each $N = \overline{2,\ 10}$. Set at $C = 50$, maximum (29) for the second series is shown in Figure 6, whose polylines are hardly comparable with those ones in Figure 1. The

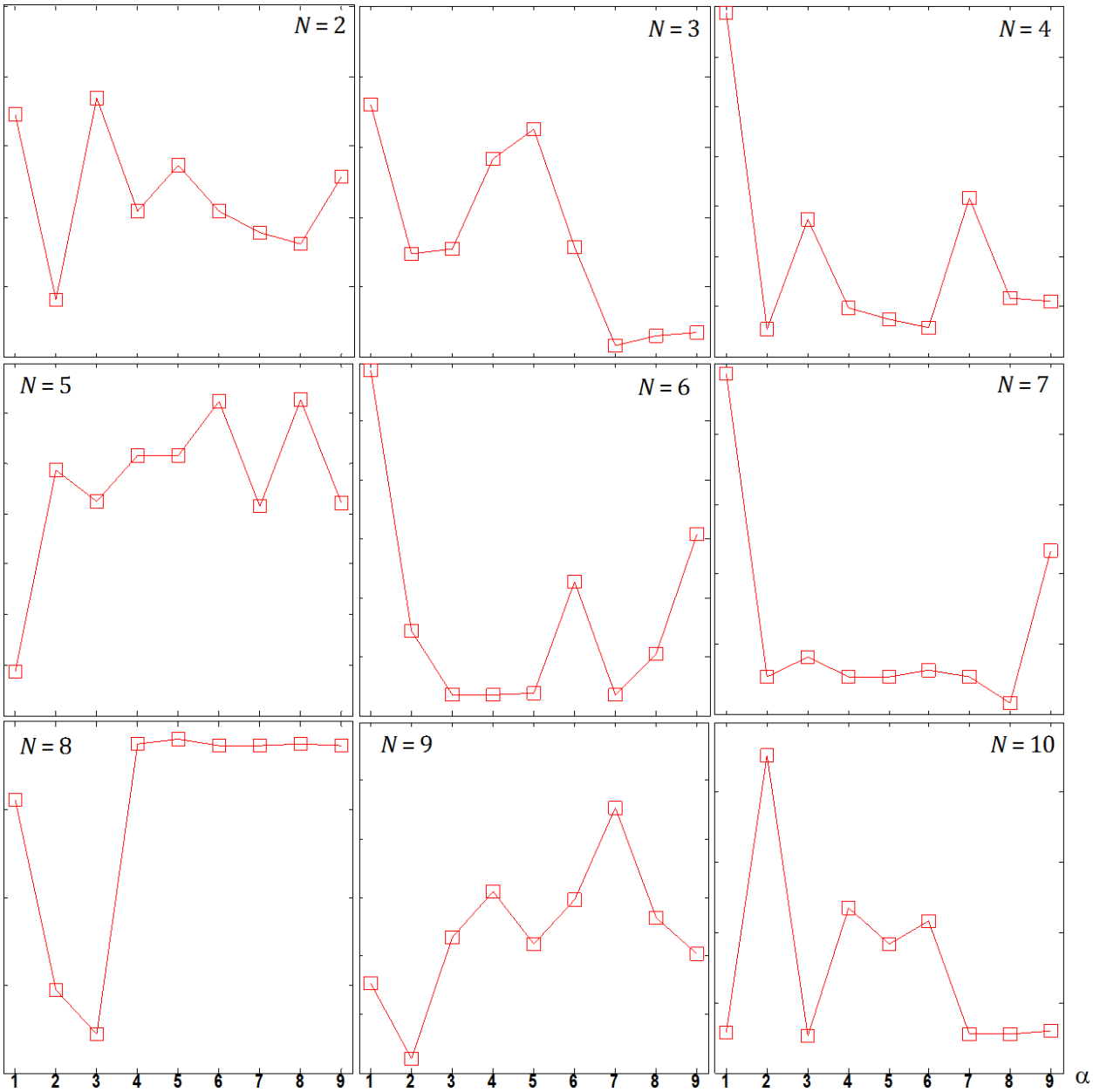

Figure 6. Maximum (29) of the computation time for the second series of 50 instances per N

timeouts are still registered at scheduling 9 and 10 jobs. Averaged computation time (30) for the second series is shown in Figure 7, whose polylines resemble those ones in Figure 2 (excluding 2, 3, and 5 jobs). Minimum (31)

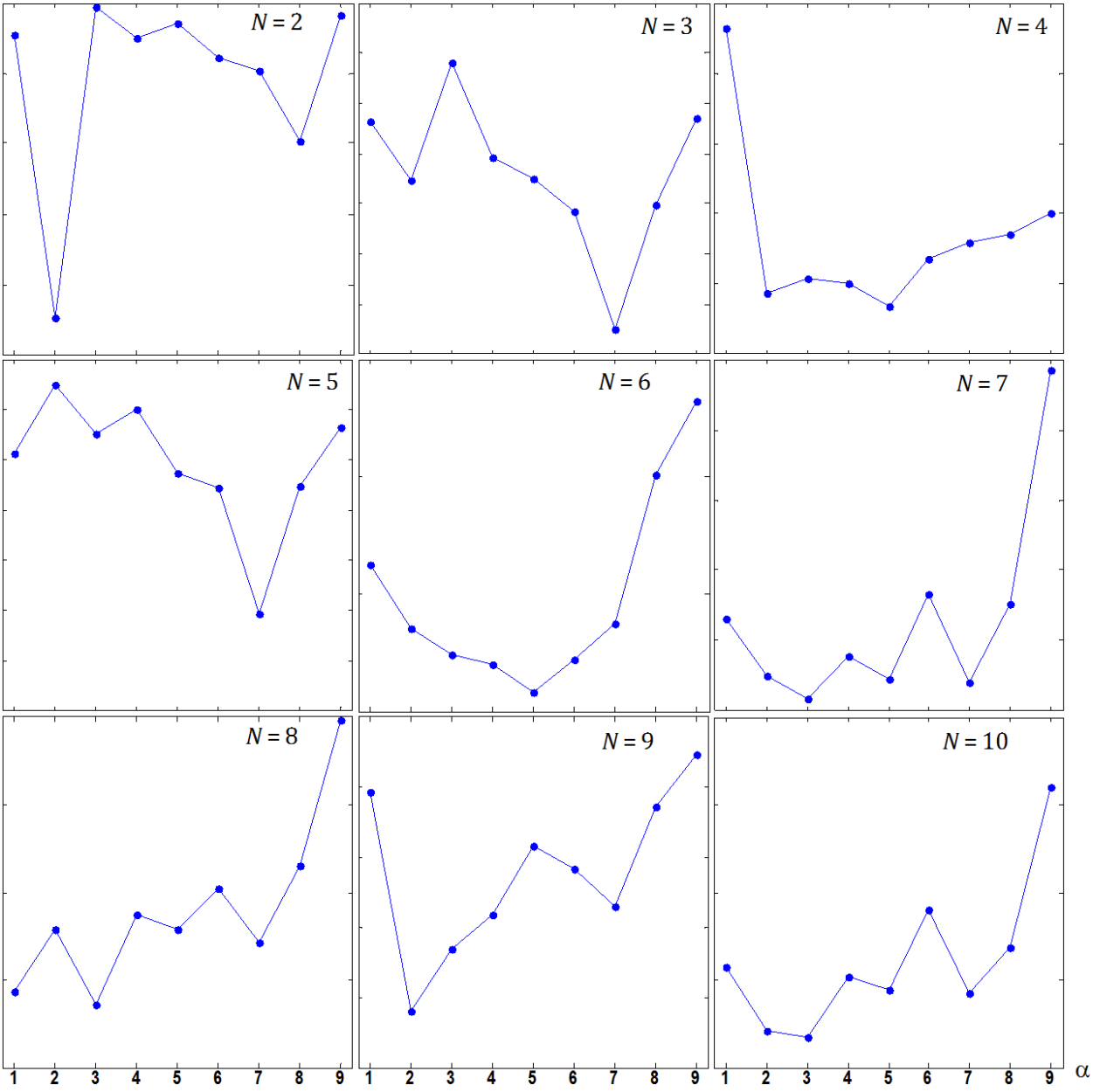

Figure 7. Averaged computation time (30) for the second series of 50 instances per N

for the second series is shown in Figure 8, but its polylines are of little significance (the similarity of $\delta_{\min}(8,\ \alpha)$ to that in Figure 3 is just spurious), as well as the disclosure in Figure 9 (no pattern is seen, and half-hourly timeouts are registered at scheduling only 9 and 10 jobs). However, the averages by (32)–(34) for this series (Figure 10) have a strong resemblance to those ones in Figure 4, especially "total" average $\tilde{\delta}(\alpha)$. In particular, setting $\alpha \geqslant 10^5$ is not beneficial by Figure 10, as well as substituting the infinity with "max". Selecting the infinity substitute as "2max", "3max", "4max" is more beneficial, but "2max" has an obvious benefit (considering only

Figure 10). Consequently, the obtained results can be declared repeatable implying an approximate repeatability of their averages.

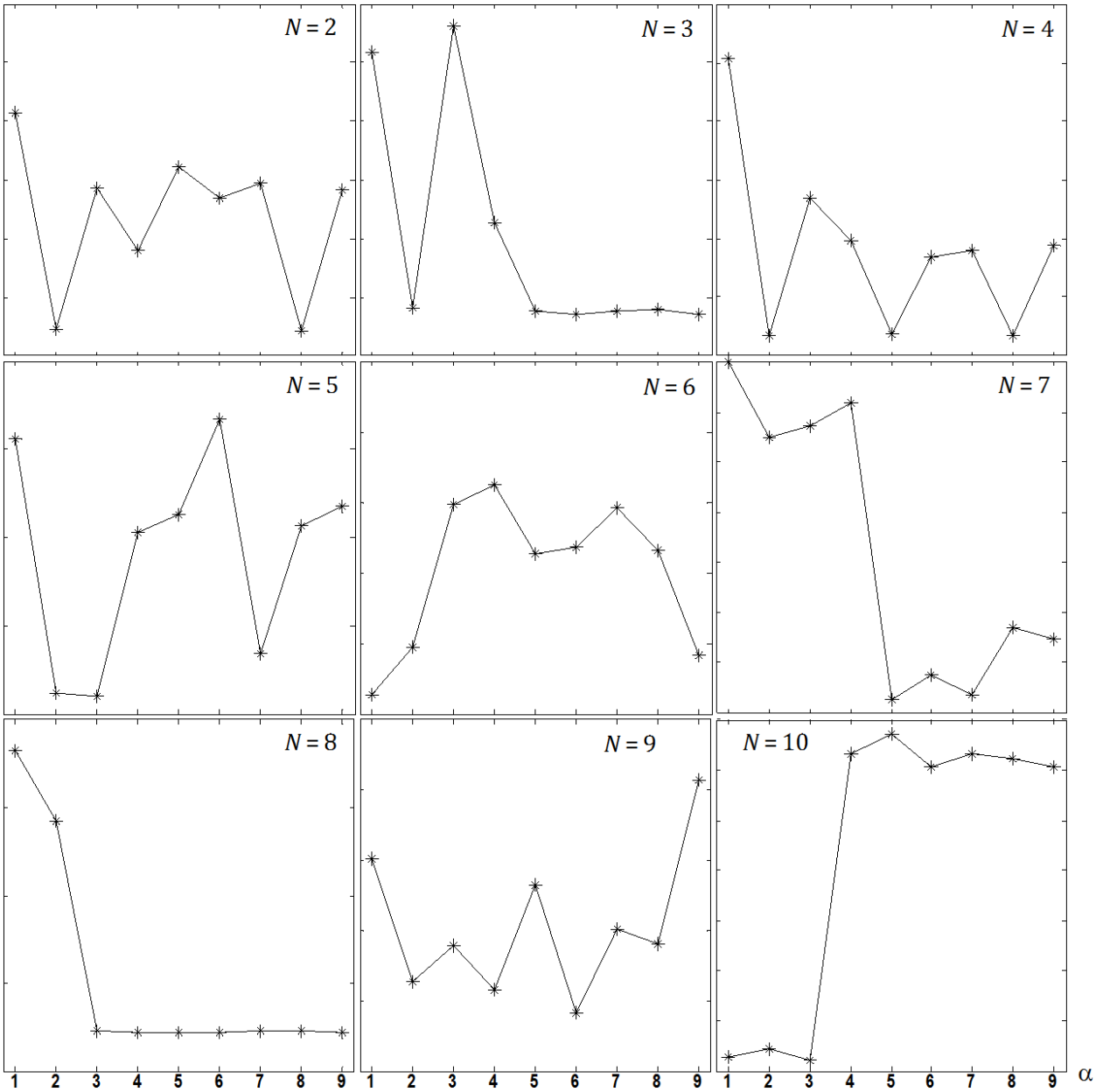

Figure 8. Minimum (31) of the computation time for the second series of 50 instances per N

6. Discussion

Owing to the approximate repeatability of the results obtained by generating job scheduling problem instances, the respective inferences from Figures 1 – 9 and Tables 2 – 4 will be statistically valid. Although it is hard to find strong regularities in the subplots of Figures 1 – 3 and 6 – 8, the infinity substitutes by “max”, “2max”, ..., “5max”,

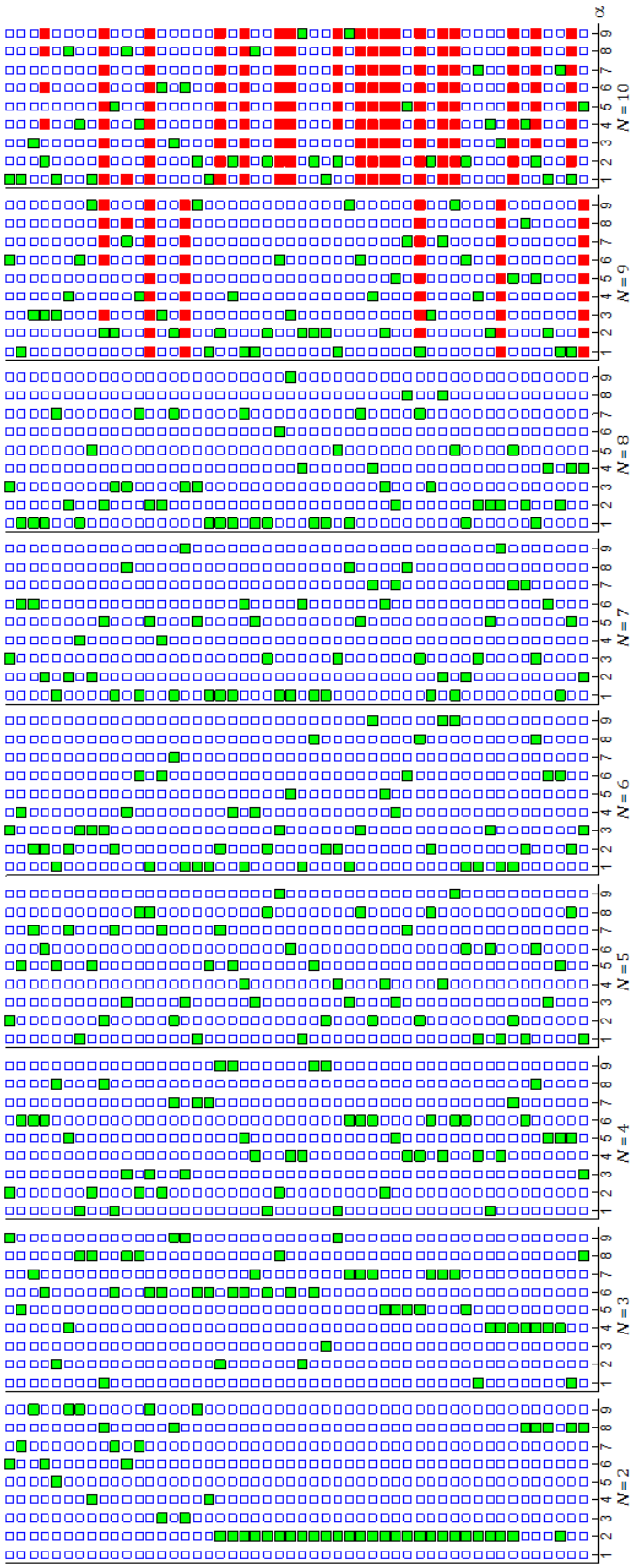

Figure 9. The disclosure of all generated instances in the second series

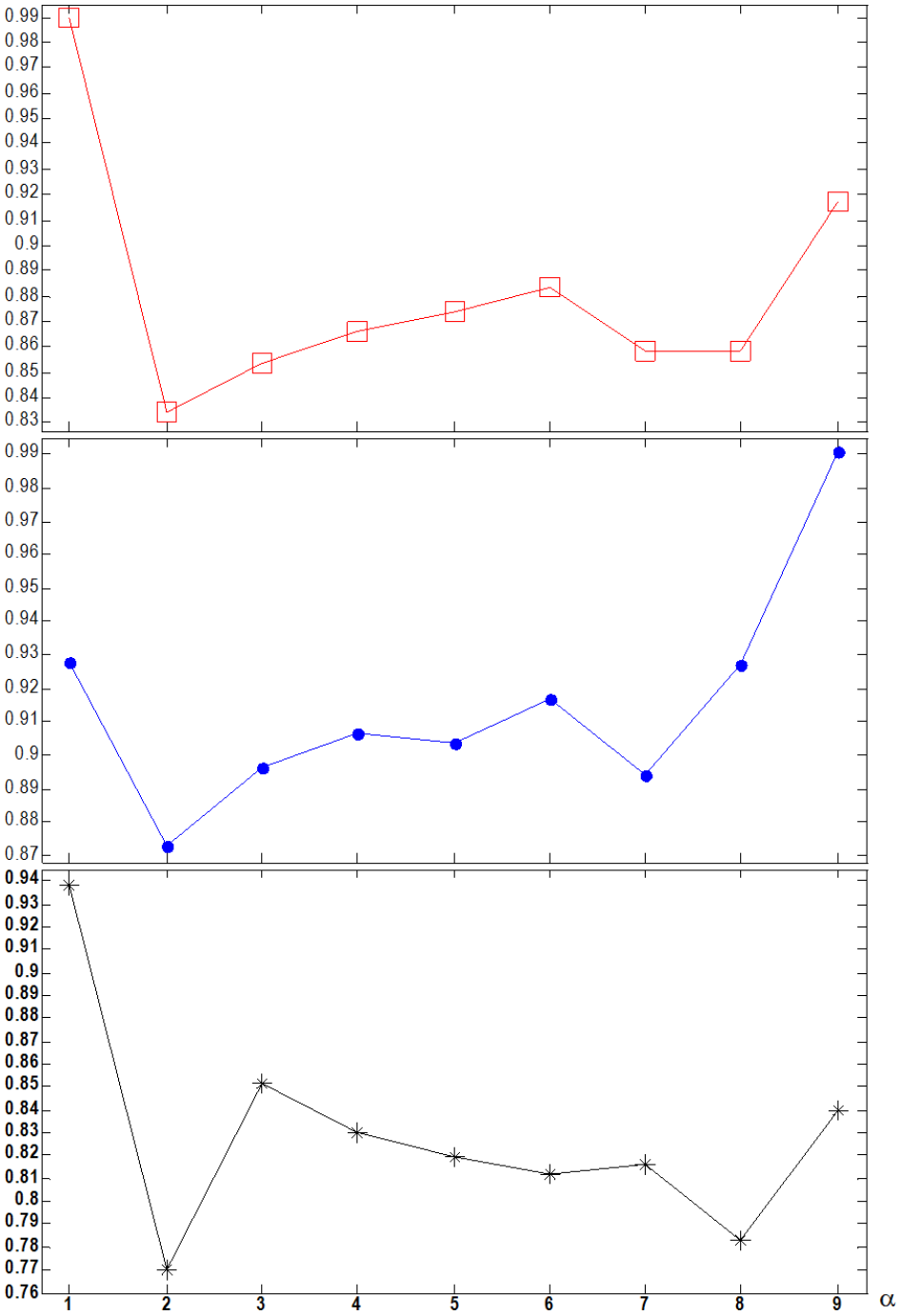

Figure 10. Averages by (32) C (34) of computation time polylines in Figures 6 – 8

"sum" are visibly more beneficial than setting $\alpha \geqslant 10^4$. Indeed, the total computation time spent for scheduling 900 instances (for all N) and averaged over the infinity substitutes by "max", "2max", ..., "5max", "sum" is 7135.93761 seconds less than that spent for scheduling those instances and averaged over three versions of "the sufficiently great positive integer" (i. e., $\alpha \geqslant 10^4$). This is almost 2-hour computation time saving. For the second series, the

difference is 3634.829857 seconds (more than an hour is saved). Nevertheless, "max" is not as good as it is in scheduling equal-length jobs [10]. Although it may be a computational artifact, "2max" is expected to be more beneficial. This is the only exclusion from the "as-less-as-possible infinity substitute" rule revealed and strongly recommended by Romanuke [10]. In all other respects, the global inference from the paper [10] is confirmed for weighted jobs of various lengths: in tight-tardy progressive 1-machine scheduling by idling-free preemptions, the exact minimum of total weighted tardiness cannot be found faster with the greater value of the infinity substitute.

7. Conclusion

The increment of the infinity substitute in the ILPP model may have significantly negative influence on the computation time of exact schedules, whichever job lengths and weights are. To decrease the computation time on average, it is strongly recommended to select the infinity substitute as multiple maximum over finite decision variable weights in the ILPP model. It is sufficient to take 2 to 5 such maxima as the infinity substitute. The two other possible substitutes, which are the single maximum increased by 1 and the sum of all those decision variable weights, are not denied but they are less beneficial than the multiple maxima. Unlike the case in which all the jobs have equal length by no weight priorities [10], the "max" infinity substitute is not the best in the general case. Meanwhile, substituting the infinity with just "a sufficiently great positive integer" is never recommended. It is worth noting that a benefit (shortened computation time) from an appropriate selection of the infinity substitute is not guaranteed for solving a single or few scheduling problems. It is only an expected benefit, which builds up as a few hundred scheduling problems are solved at least.

Acknowledgement

The work was technically supported by the Faculty of Mechanical and Electrical Engineering at the Polish Naval Academy, Gdynia, Poland.

REFERENCES

1. M. Batsyn, B. Goldengorin, P. Pardalos, and P. Sukhov, *Online heuristic for the preemptive single machine scheduling problem of minimizing the total weighted completion time*, Optimization Methods & Software, vol. 29, iss. 5, pp. 955–963, 2014.
2. J. O. Berger, *Statistical Decision Theory and Bayesian Analysis*, Springer, New York, NY, 1985.
3. P. Brucker, *Scheduling Algorithms*, Springer-Verlag Berlin Heidelberg, 2007.
4. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. Rinnooy Kan, *Optimization and approximation in deterministic sequencing and scheduling: A survey*, Annals of Discrete Mathematics, no. 5, pp. 287–326, 1979.
5. F. Jaramillo, B. Keles, and M. Erkoc, *Modeling single machine preemptive scheduling problems for computational efficiency*, Annals of Operations Research, no. 285, pp. 197–222, 2020.
6. M. L. Pinedo, *Planning and Scheduling in Manufacturing and Services*, Springer-Verlag New York, 2009.
7. M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Springer Inter. Publishing, 2016.
8. 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.
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.
10. V. V. Romanuke, *Infinity substitute in exactly minimizing total tardiness in tight-tardy progressive 1-machine scheduling by idling-free preemptions of equal-length jobs*, Bulletin of V. Karazin Kharkiv National University. Mathematical Modelling. Information Technology. Automated Control Systems, iss. 44, pp. 94–101, 2019.
11. V. V. Romanuke, *Minimal total weighted tardiness in tight-tardy single machine preemptive idling-free scheduling*, Applied Computer Systems, vol. 24, no. 2, pp. 150–160, 2019.
12. V. V. Romanuke, *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, pp. 27–39, 2020.
13. A. S. Uyar, E. Ozcan, and N. Urquhart, *Automated Scheduling and Planning: From Theory to Practice*, Springer-Verlag Berlin Heidelberg, 2013.